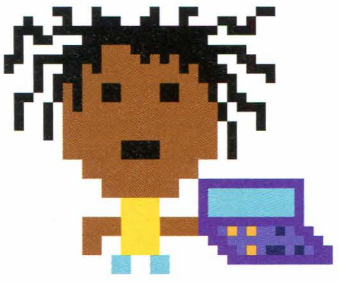
ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ

ИЛЛЮСТРИРОВАННОЕ РУКОВОДСТВО ПО ЯЗЫКАМ SCRATCH И PYTHON

ПРОГРАММИРОВАНИЕ

ДЛЯ ДЕТЕЙ

Перевод с английского
Станислава Ломакина



Москва
«Манн, Иванов и Фербер»
2015

Издано с разрешения Dorling Kindersley Limited

На русском языке публикуется впервые

Авторы: Кэрол Вордерман, Джон Вудкок, Шон Макаманус, Крейг Стили, Клэр Куигли, Дэниел Маккаферти

Научный редактор: Денис Голиков, scratch teacher, автор сайта scratch4russia.com

Программирование для детей / К. Вордерман, Дж. Вудкок, Ш. Макаманус и др. ; пер. с англ. С. Ломакина. — М. : Манн, Иванов и Фербер, 2015. — 224 с. : ил.

ISBN 978-5-00057-472-0

Всем, что делает компьютер, управляют строки программного кода, введенные с клавиатуры. Компьютерный код похож на иностранный язык, и этот язык может освоить каждый!

Это уникальное руководство научит создавать игры и анимацию с помощью двух разных языков программирования — Scratch и Python. Интересные, веселые проекты, краткие понятные инструкции, яркие цветные иллюстрации помогут тебе без труда разобраться в основах программирования и написать коды для твоих первых игр и анимации. А еще ты узнаешь, как устроен компьютер и для чего вообще стоит учиться программированию.

Издание предназначено для широкого круга читателей.

ISBN 978-5-00057-472-0

Оригинальное название: Computer Coding for Kids

Издание для досуга

Кэрол **Вордерман,** Джон **Вудкок,**
Шон **Макаманус** и др.

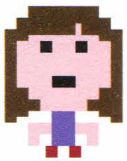
Программирование для детей

Главный редактор *Артем Степанов*
Руководитель редакции *Анастасия Кренева*
Ответственный редактор *Анастасия Кузнецова*
Литературный редактор *Ольга Збарская*
Арт-директор *Алексей Богомолов*
Верстка *Надежда Кудрякова*
Корректоры *Евгения Смирнова, Наталья Витько, Елена Пинчукова*

Подписано в печать 09.02.2015.
Формат 84×108 $^{1}/_{16}$. Гарнитура Myriad.
Бумага мелованная. Печать офсетная.
Усл. печ. л. 19. Тираж 4000 экз. Заказ № 1487.

ООО «Манн, Иванов и Фербер»
mann-ivanov-ferber.ru
facebook.com/mifdetstvo
vk.com/mifdetstvo

Отпечатано в типографии TBB, a.s., Словакия



КЭРОЛ ВОРДЕРМАН — одна из самых популярных британских телеведущих, известная своими познаниями в математике, обладатель степени магистра технических наук колледжа Сидни Сассекс Кембриджского университета и кавалер ордена Британской империи. Кэрол увлечена программированием и считает, что каждый ребенок должен иметь возможность освоить этот полезный навык. Выступала ведущей множества телешоу, посвященных естественным и техническим наукам, например Tomorrow's World и How 2, а также церемоний вручения наград «Гордость Великобритании» — на телеканалах BBC, ITV и Channel 4. Также она в течение 26 лет была соведущей интеллектуальной математической телеигры Countdown на канале Channel 4, стала вторым по популярности автором нон-фикшн в начале 2000-х, консультировала премьер-министра Дэвида Кэмерона по вопросам, связанным с перспективами математического образования в Великобритании, — и все это время Кэрол посвящала себя поиску простых и занимательных способов обучения математике, естественным и техническим наукам.

ДЖОН ВУДКОК окончил физический факультет Оксфордского университета и получил степень доктора наук по вычислительной астрофизике от Лондонского университета. Начал заниматься программированием с восьми лет, работал на самых разных компьютерах, от одночиповых микроконтроллеров до суперкомпьютеров мирового класса. Среди его многочисленных проектов — космические симуляторы, исследования в области высоких технологий и «разумные» роботы, сделанные из подручного хлама. Джон увлекается обучением людей естественным и техническим наукам: читает лекции о космосе и организует школьные кружки программирования. Также он как соавтор и консультант участвовал в создании множества научно-технических книг.

ШОН МАКМАНУС научился программировать, когда ему было девять лет. Его первым языком программирования стал Logo. Сейчас Шон — опытный автор и журналист в области технологий. Среди других его книг — «Программирование на Scratch: шаг за шагом», «Веб-дизайн: шаг за шагом» и «Raspberry Pi для "чайников"». На его сайте www.sean.co.uk можно найти игры и учебные материалы по Scratch.

КРЕЙГ СТИЛИ — специалист по обучению информатике. Проект-менеджер в CoderDojo Scotland, сети бесплатных кружков программирования для молодежи. Ранее Крейг работал в Шотландском агентстве академической аттестации, Научном центре Глазго и Университете Глазго. Первым компьютером Крейга был ZX Spectrum.

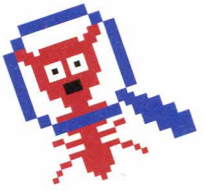
КЛЭР КУИГЛИ изучала информатику в Университете Глазго, получила степени бакалавра и доктора наук. Работала в компьютерной лаборатории Университета Кембриджа над проектом, посвященным развитию навыков алгоритмического мышления у учеников младших классов. Также Клэр — наставник в CoderDojo Scotland, сети кружков программирования для молодежи.

ДЭНИЕЛ МАККАФЕРТИ окончил Университет Стратклайда по специальности «информатика». После выпуска разрабатывал программное обеспечение для некоторых из крупнейших в мире инвестиционных банков. В свободное время Дэниел преподает в CoderDojo Scotland, сети кружков программирования для молодежи.

Содержание

4 УСТРОЙСТВО КОМПЬЮТЕРОВ

5 ПРОГРАММИРОВАНИЕ В РЕАЛЬНОМ МИРЕ



Введение

Совсем недавно компьютерное программирование казалось таинственным ремеслом, уделом специалистов. Мысль о том, что программирование может быть увлекательным занятием для каждого, большинству людей и в голову не приходила. Но мир изменился. Интернет, электронная почта, социальные сети, смартфоны и мобильные приложения, ураганом влетев в нашу жизнь, преобразили ее всего за несколько лет.

Компьютеры занимают очень важное место в современном мире, и мы принимаем это как должное. Вместо звонков по телефону мы посылаем текстовые сообщения или используем социальные сети. Мы охотно поглощаем любые плоды компьютеризации — от интернет-шопинга и развлечений до новостей и игр. Однако мы можем не только использовать эти технологии — научившись программировать, мы можем развивать их, создавать собственные произведения цифрового искусства.

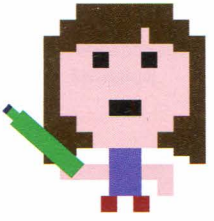
Всем, что делает компьютер, управляют строки программного кода, введенные с клавиатуры. Компьютерный код похож на иностранный язык, но язык этот может освоить каждый и довольно быстро. Многие считают, что программирование — одно из наиболее важных в XX веке умений.

Учиться программировать очень интересно, ведь ты можешь получить результаты сразу же, и не важно, сколько еще материала предстоит изучить. Более того, создание игр и программ — такое увлекательное занятие, что очень скоро покажется, будто оно почти не требует усилий. Это отличная возможность для творчества, наверное, первая область науки, совмещающая искусство, логику, сочинительство и бизнес.

Кроме того, умение программировать очень пригодится в жизни. Оно развивает логику и интеллект, которые важны в самых разных областях — от науки и инженерного дела до медицины и юриспруденции. Количество вакансий, где нужно умение программировать, будет со временем только расти, причем хороших программистов не хватает уже сейчас. Научись программированию — и цифровой мир будет открыт для тебя!

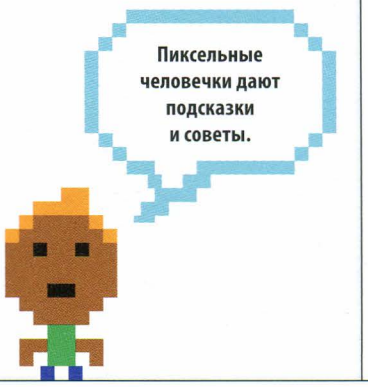
Carol Vorderman

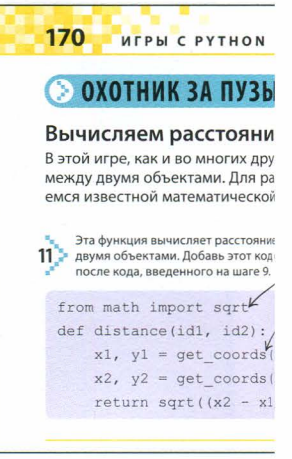
КЭРОЛ ВОРДЕРМАН



Как устроена эта книга

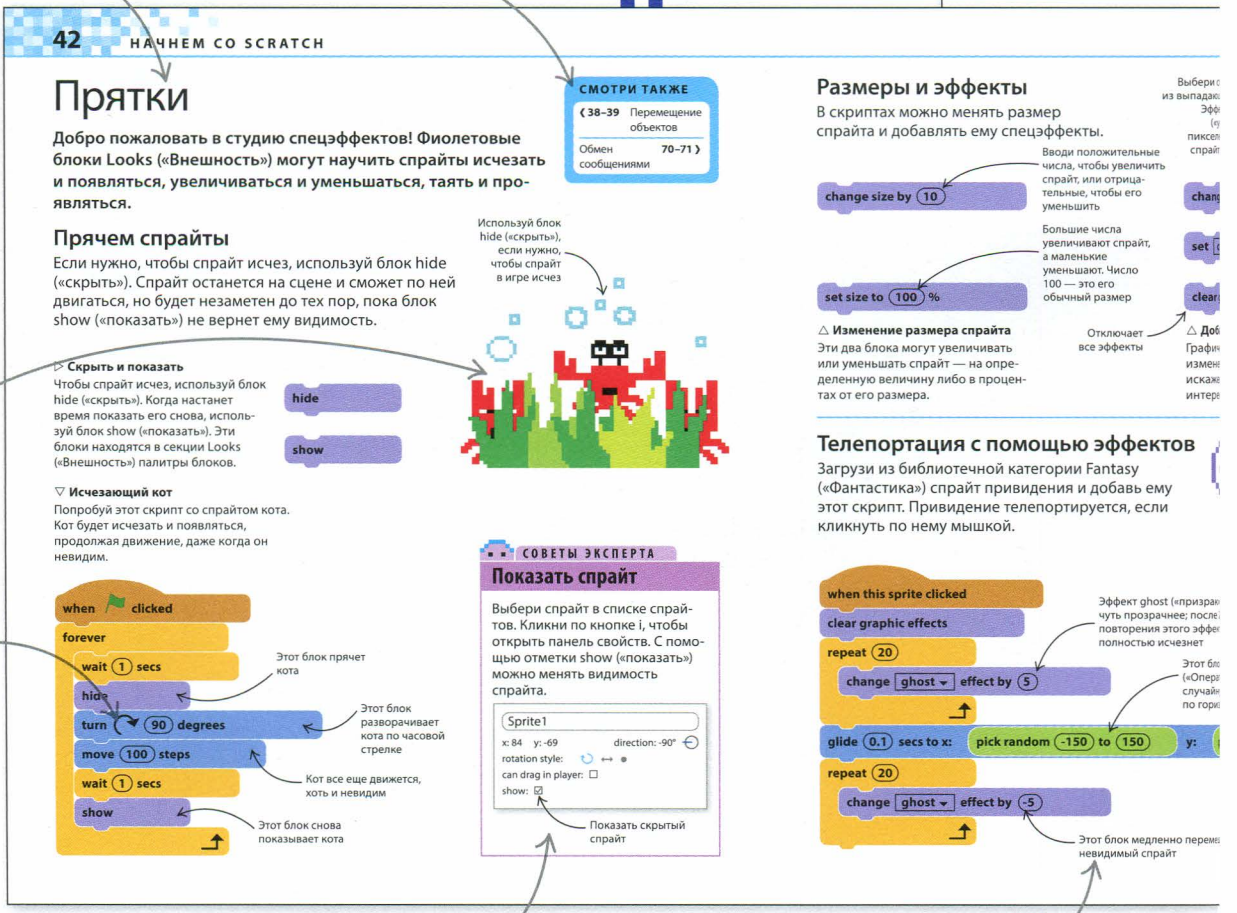
Эта книга знакомит с основными понятиями, которые нужно усвоить, чтобы научиться программировать. Еще ты найдешь здесь увлекательные проекты, воплощающие идеи программирования на практике. Чтобы информацию было легче понять и усвоить, она разбита на небольшие блоки.





Каждая тема рассматривается подробно, с примерами и упражнениями

В блоках «Смотри также» перечислены другие относящиеся к теме разделы



Цветные рисунки поясняют различные понятия программирования

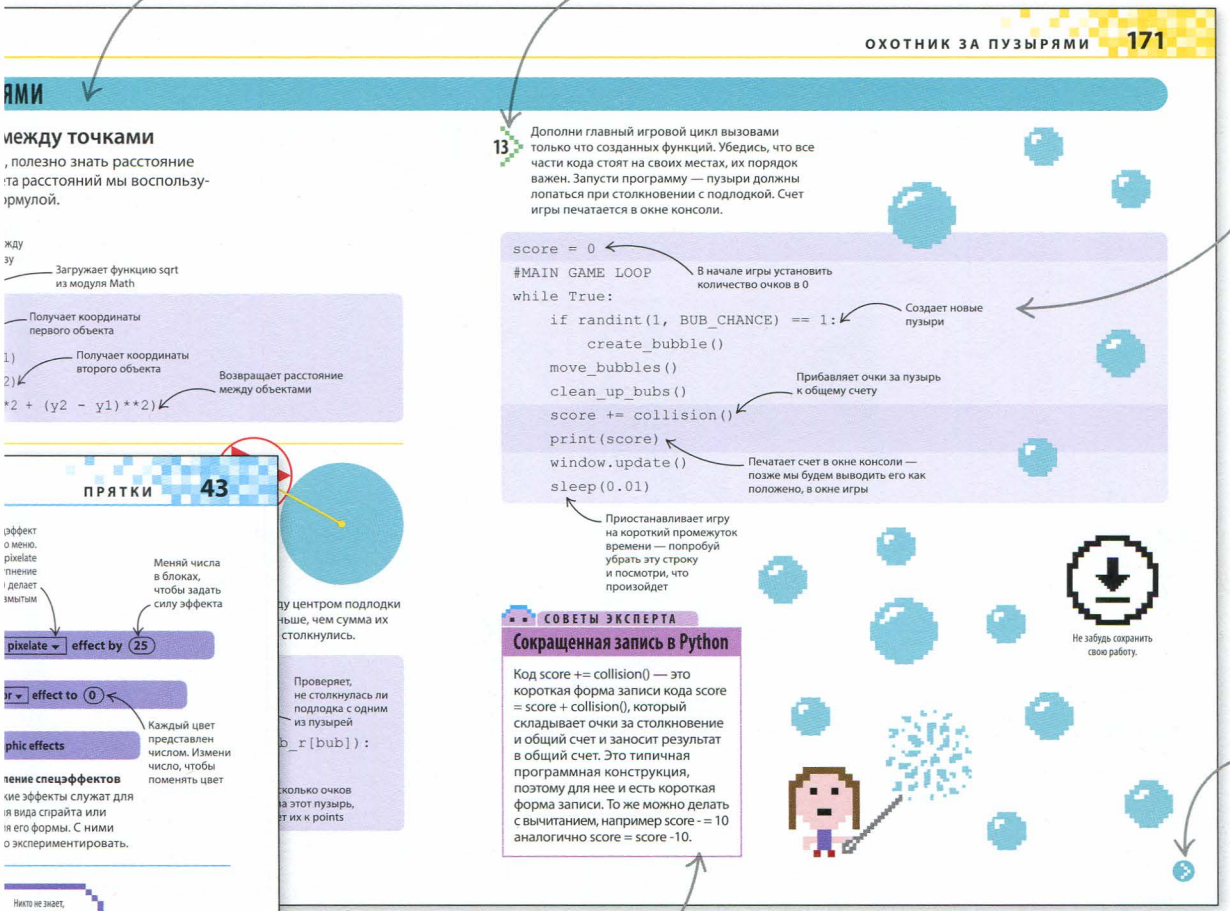
Программный код и скрипты объясняются построчно

Инструкции указывают, что надо нажать, перетащить или выделить

Подписи поясняют каждый шаг

Семь проектов для развития навыков программирования. Страницы проектов выделены синей полосой

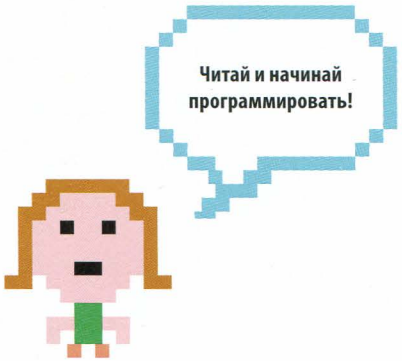
Простые пошаговые инструкции проведут тебя по каждому из проектов



Пометки с указанием на строку, чтобы ничего не перепутать

Эта иконка указывает, что продолжение проекта на следующей странице

В рамочках — дополнительная информация: подсказки, определения, важные для запоминания вещи



СОВЕТЫ ЭКСПЕРТА

Сохранение

Иконка сохранения появляется на разворотах с проектами. Она напоминает тебе сохранять выполненную работу, чтобы ничего не потерялось в случае проблем с компьютером.

Сохраняй работу почаще.

Что такое программирование?

Что такое компьютерная программа?

СМОТРИ ТАКЖЕ

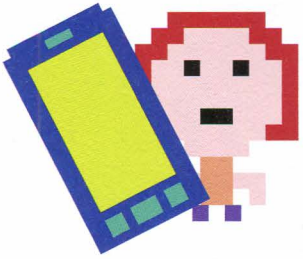
Думай как компьютер **16–17 ❯**

Как стать программистом **18–19 ❯**

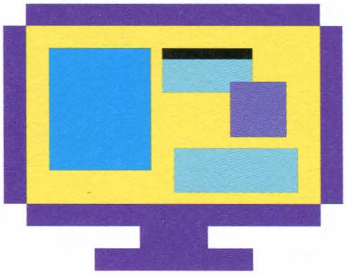
Компьютерная программа — это набор инструкций, следуя которым компьютер выполняет поставленную задачу. Программировать — значит писать для компьютера пошаговые инструкции, объясняющие, что и как ему нужно делать.

Компьютерные программы — повсюду

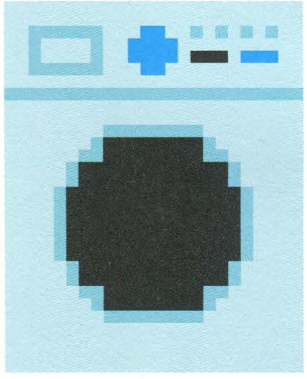
Нас окружают компьютерные программы. Множеством приборов и мобильных устройств, которыми мы пользуемся изо дня в день, управляют программы. То есть эти устройства следуют пошаговым инструкциям, которые написаны программистами.



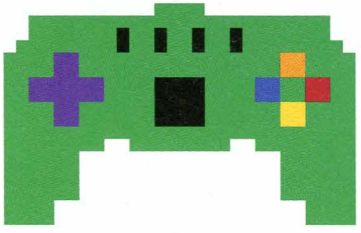
◁ **Мобильные телефоны**
Программы позволяют сделать звонок или отправить текстовое сообщение. При поиске контакта по имени программа сама находит нужный телефонный номер.



△ **Компьютерные приложения**
Все, что мы делаем с помощью компьютера, от просмотра сети интернет до составления документов и проигрывания музыки, возможно благодаря коду, написанному программистами.

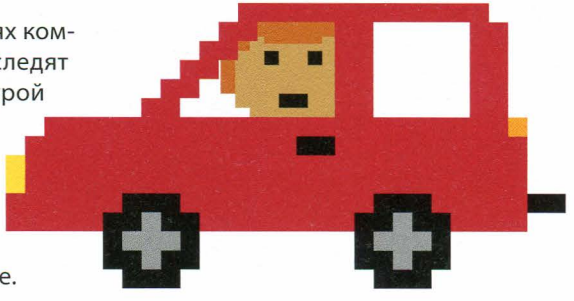


△ **Стиральные машины**
Стиральные машины программируются на разные режимы стирки. Компьютерный код следит за температурой воды и временем стирки.



◁ **Игры**
Игровая приставка — это всего лишь специальный тип компьютера. Все игры, сделанные для приставок, — это программы. Графика, звук и управление в играх расписаны в программном коде.

▷ **Автомобили**
В некоторых автомобилях компьютерные программы следят за скоростью, температурой воздуха и количеством топлива в баке. Программы могут даже помочь вовремя притормозить, чтобы поездка была безопаснее.

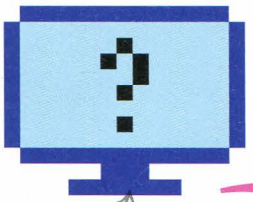


Как работают компьютерные программы

Компьютеры могут казаться очень умными, но это всего лишь напичканные электроникой ящики, которые умеют очень быстро и точно выполнять инструкции. Мы, разумные существа, можем давать компьютерам задачи, описывая их в виде программ, то есть пошаговых инструкций.

1 Компьютеры не умеют думать

Сам по себе компьютер ни на что не способен. Задача программиста — давать ему инструкции.



Без инструкций компьютер совсем бестолковый

2 Напиши программу

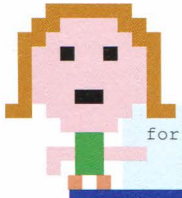
Ты можешь объяснить компьютеру, что делать, написав очень подробные инструкции, которые называются программой. Каждая инструкция должна быть достаточно простой, чтобы компьютер ее понял. Если инструкция написана неверно, компьютер поведет себя не так, как ты хочешь.

Это компьютерная программа, выполняющая обратный отсчет перед запуском

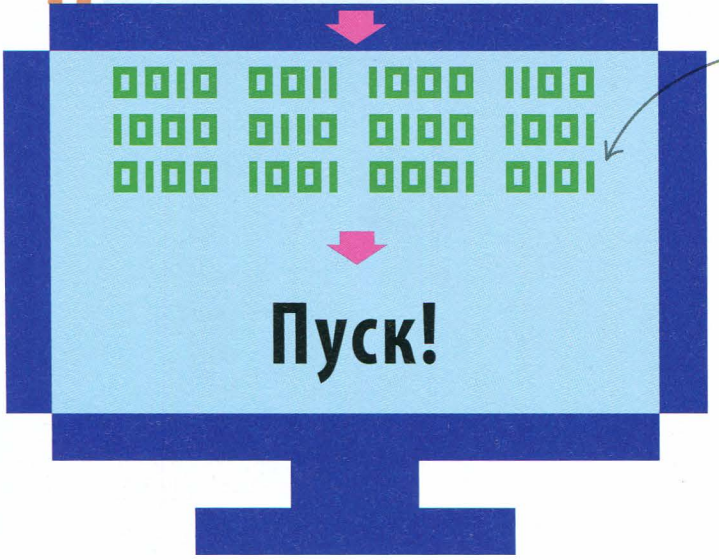
```
for count in range(10, 0, -1):
    print("Counting down", count)
```

3 Языки программирования

Компьютер может выполнять инструкции лишь на тех языках, которые ему понятны. Программистам приходится выбирать, какой язык лучше подходит для решения конкретной задачи.



```
for count in range(10, 0, -1):
    print("Counting down", count)
```



В итоге все программы становятся двоичным кодом — простейшим компьютерным языком, состоящим из нулей и единиц

СЛЕНГ

Софт и железо

Железо (или аппаратные средства) — это части компьютера, которые ты можешь увидеть или потрогать (провода, схемы, клавиатура, экран и так далее). Софтом, или программными средствами, называют программы, которые выполняются на компьютере и управляют его работой. Аппаратные и программные средства работают сообща, чтобы компьютер мог выполнять полезные задачи.

Думай как компьютер

Программист должен научиться думать как компьютер. Все задачи нужно разбивать на небольшие подзадачи, которым легко следовать и которые невозможно понять неправильно.

СМОТРИ ТАКЖЕ

❮ **14–15** Что такое компьютерная программа?

Как стать программистом **18–19** ❯

Думай как робот

Представь себе кафе с официантом-роботом. Поскольку у робота бесхитростный компьютерный мозг, ему нужно объяснить, как доставлять блюда с кухни посетителям, ожидающим в обеденном зале. Сперва эту проблему нужно разбить на простые, понятные компьютеру подзадачи.

СЛЕНГ

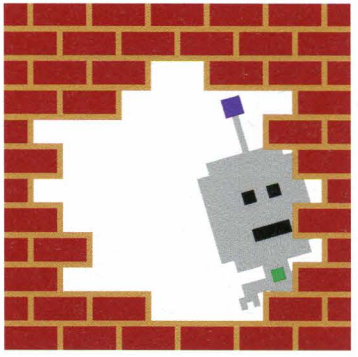
Алгоритм

Алгоритм — это набор простых инструкций, объясняющих, как выполнить задачу. Программа — это алгоритм, переведенный на понятный компьютеру язык.

1 **Программа 1 для робота-официанта**

Следуя этой программе, робот берет еду с тарелки, вламывается из кухни в обеденный зал прямо сквозь стену и кладет еду на пол. Этот алгоритм составлен недостаточно подробно.

1. Взять еду
2. Идти из кухни к столу посетителя
3. Положить еду

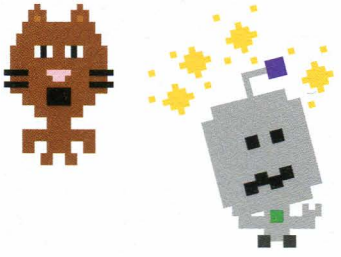


◁ **Беда!**
Инструкции слишком неоднозначны: мы забыли объяснить роботу, что идти нужно через дверь. Людям это показалось бы очевидным, но компьютеры не умеют думать сами.

2 **Программа 2 для робота-официанта**

На этот раз мы сказали роботу, чтобы он шел через кухонную дверь. Так он и делает, но затем натыкается на местного кота, падает и разбивает тарелку.

1. Взять тарелку с едой
2. Идти из кухни к столу посетителя таким образом:
   - идти к кухонной двери
   - идти от двери к столу посетителя
3. Поставить тарелку на стол перед посетителем

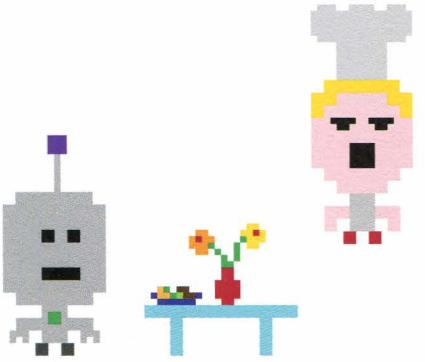


△ **Все еще не идеально**
Робот не знает, как быть с препятствиями (вроде кошек). Чтобы он мог безопасно перемещаться, программа должна содержать еще более подробные инструкции.

3 **Программа 3 для робота-официанта**

Следуя этому варианту программы, робот благополучно доставляет блюдо посетителю, огибая препятствия. Однако, поставив тарелку на стол, робот застывает на месте, а на кухне тем временем скапливается приготовленная еда.

1. Взять тарелку с едой и держать ее ровно
2. Идти из кухни к столу посетителя таким образом:
    - идти к кухонной двери
        - если замечено препятствие, обойти его
    - идти от двери к столу посетителя
        - если замечено препятствие, обойти его
3. Поставить тарелку на стол перед посетителем

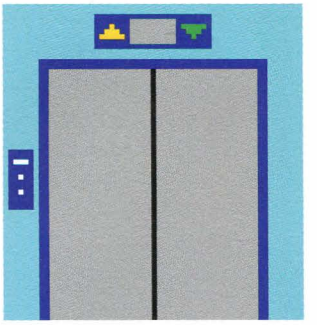


△ **Неужели получилось?**

Наконец-то робот смог благополучно доставить еду. Но мы забыли сказать ему, что потом нужно вернуться на кухню за новой тарелкой.

Пример из жизни

Может, наш робот-официант и выдуманный, однако алгоритмы такого типа встречаются повсюду. Например, управляемый компьютером лифт имеет дело со схожими проблемами. Вверх или вниз ему следует двигаться? На какой следующий этаж ехать?



1. Ждать до закрытия дверей
2. Ждать нажатия кнопки
    - Если нажата кнопка этажа выше текущего:
        - начать движение вверх
    - Если нажата кнопка этажа ниже текущего:
        - начать движение вниз
3. Ждать, пока этаж не будет соответствовать нажатой кнопке
4. Открыть двери

◁ **Программа для лифта**

Чтобы лифт работал правильно и безопасно, каждый шаг программы должен быть точным, понятным и учитывать все возможные случаи. Программисты должны убедиться, что они составили подходящий алгоритм.

Как стать программистом

Программисты (или кодеры) — это люди, которые создают программы, управляющие всем, что мы видим и делаем на компьютере. Ты сможешь писать собственные программы, если освоишь какой-нибудь язык программирования.

СМОТРИ ТАКЖЕ

Что такое Scratch? 22–23 ❯

Что такое Python? 86–87 ❯

Языки программирования

Языков программирования очень много. Каждый из этих языков можно использовать для выполнения разных задач. Вот некоторые из самых популярных языков и цели, для которых их чаще всего используют.

C Мощный язык для написания компьютерных операционных систем.

Ada Используется для управления космическими кораблями, спутниками и самолетами.

Java Работает на компьютерах, мобильных телефонах и планшетах.

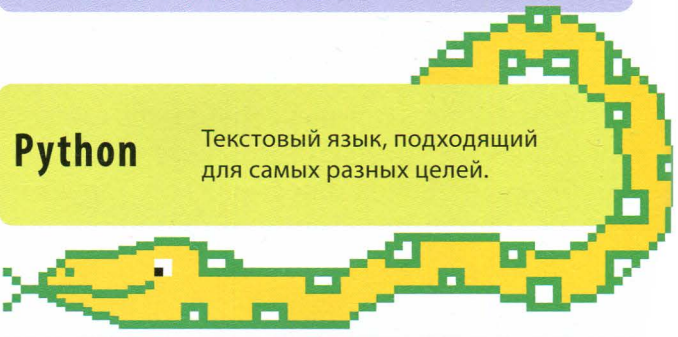
MATLAB Идеален, когда нужно выполнять много математических вычислений.

Ruby Для автоматического отображения информации на интернет-страницах.

Javascript Язык для написания интерактивных веб-сайтов.

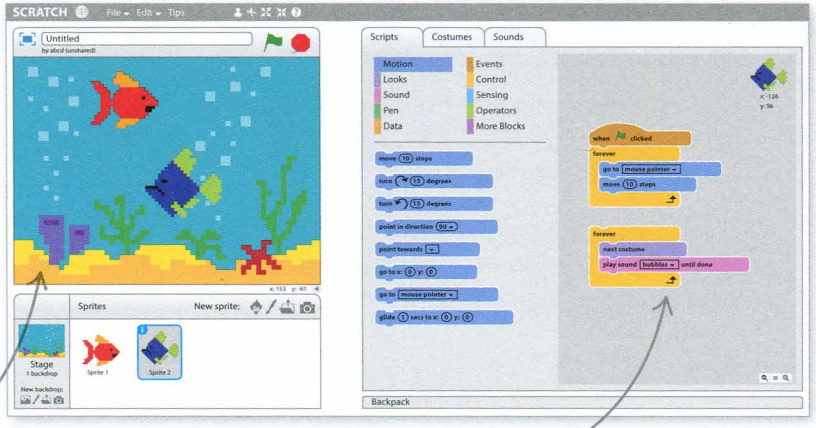
Scratch Визуальный язык, который идеально подходит для изучения программирования.

Python Текстовый язык, подходящий для самых разных целей.



Что такое Scratch?

Scratch (читается «скрэтч») отлично подходит, чтобы начать программировать. Программы на этом языке состоят не из текстовых инструкций, а из блоков кода, соединенных между собой. Scratch — простой и эффективный язык, знакомящий с основными понятиями, необходимыми, чтобы программировать на других языках.



Программа запускается на этой части экрана

Код создают, соединяя цветные блоки между собой

Что такое Python?

Люди по всему миру используют Python (читается «пайтон») для создания игр, приложений и веб-сайтов. Освоив этот замечательный язык, можно писать самые разные программы. Код на Python состоит из слов английского языка и символов.

IDLE File Edit Shell Debug Window Help

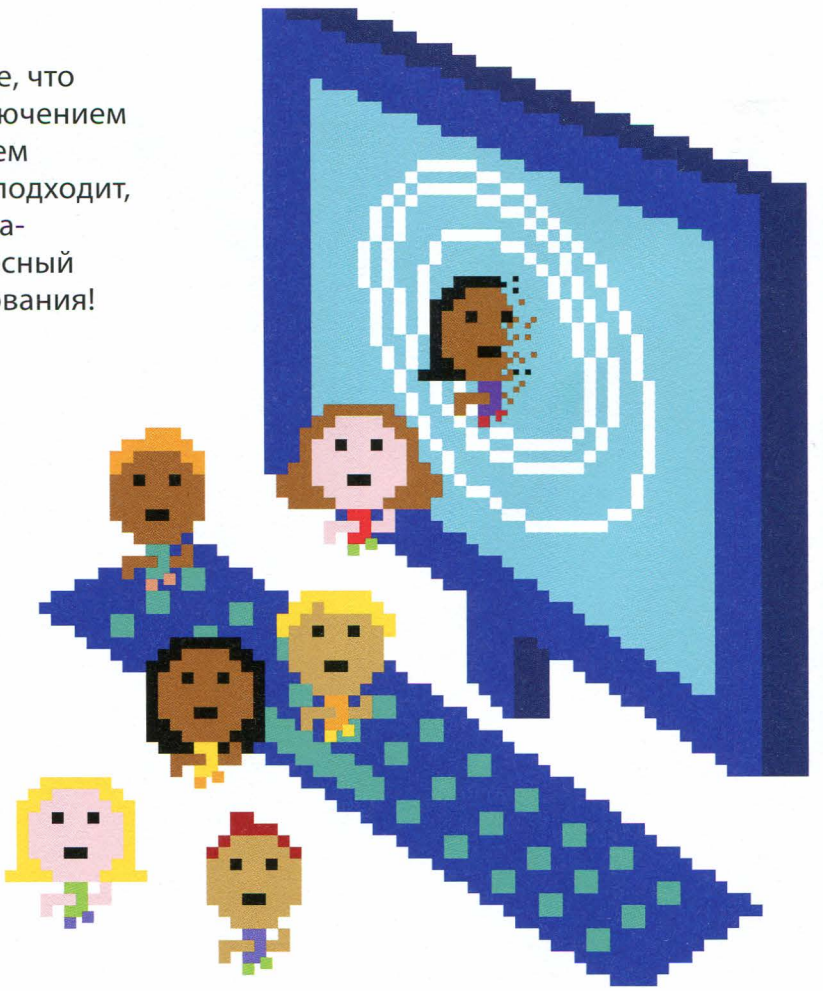
ghostgame

```
# Ghost Game
from random import randint
print('Ghost Game')
feeling_brave = True
score = 0
while feeling_brave:
    ghost_door = randint(1, 3)
    print('Three doors ahead...')
```

Программа, написанная на Python

Приступим

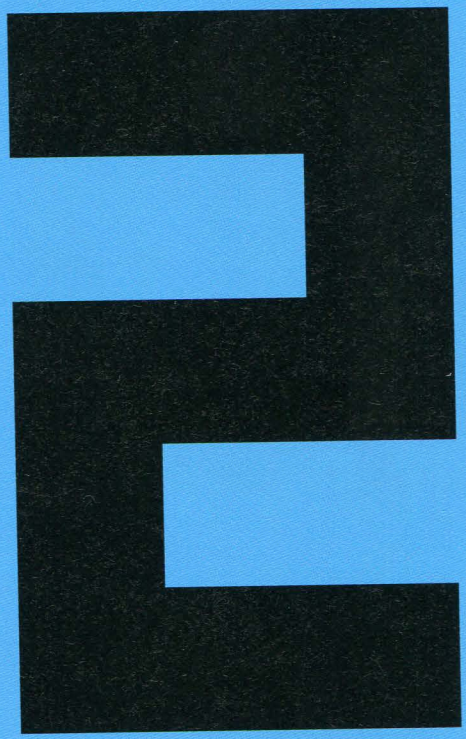
Пора начать программировать! Все, что тебе нужно, — компьютер с подключением к интернету. В этой книге мы начнем с языка Scratch, который отлично подходит, чтобы встать на путь программиста-знатока. Приготовься войти в чудесный мир компьютерного программирования!



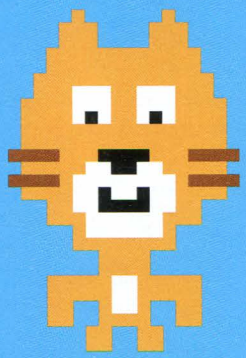
СОВЕТЫ ЭКСПЕРТА

Экспериментируй!

Тебе как программисту-новичку полезно экспериментировать с кодом своих программ. Один из лучших способов чему-то научиться — это менять разные части кода и смотреть, что произойдет. Играя с программой, ты найдешь новые способы решения задач, узнаешь больше о программировании, и тебе станет еще интереснее!



Начнем со Scratch



Что такое Scratch?

Scratch — это визуальный язык, программировать на котором очень просто. С его помощью можно создавать самые разные веселые и интересные программы.

СМОТРИ ТАКЖЕ

Установка и запуск Scratch **24–25 ›**

Интерфейс Scratch **26–27 ›**

Цветные блоки и скрипты **30–31 ›**

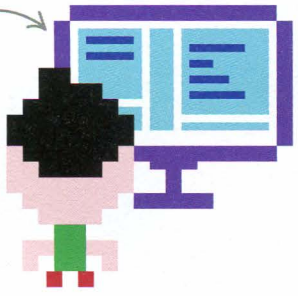
Основы Scratch

Scratch отлично подходит для создания игр и анимаций. Для него есть обширные наборы (или библиотеки) картинок и звуков, с которыми можно позабавиться.

1 Начнем программировать

Scratch — язык программирования. Он почти не требует ввода текста с клавиатуры и легок в освоении.

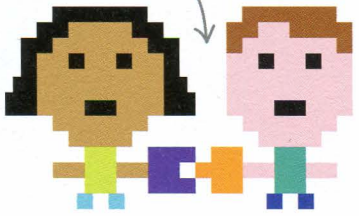
Напиши свою первую программу на Scratch!



2 Соединим программные блоки

Программируют в Scratch с помощью цветных блоков кода. Выбирая блоки и соединяя их вместе, можно составить набор инструкций — скрипт.

Блоки соединяются, как элементы головоломки

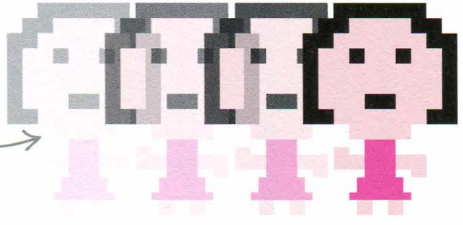


3 Заставим спрайты двигаться и говорить

В программу можно добавлять объекты: людей, автомобили, животных. Такие объекты называются спрайтами. С помощью скриптов их можно научить двигаться и разговаривать.

Спрайты можно запрограммировать так, чтобы они говорили.

Спрайты можно «научить» двигаться, бегать и танцевать



СЛЕНГ

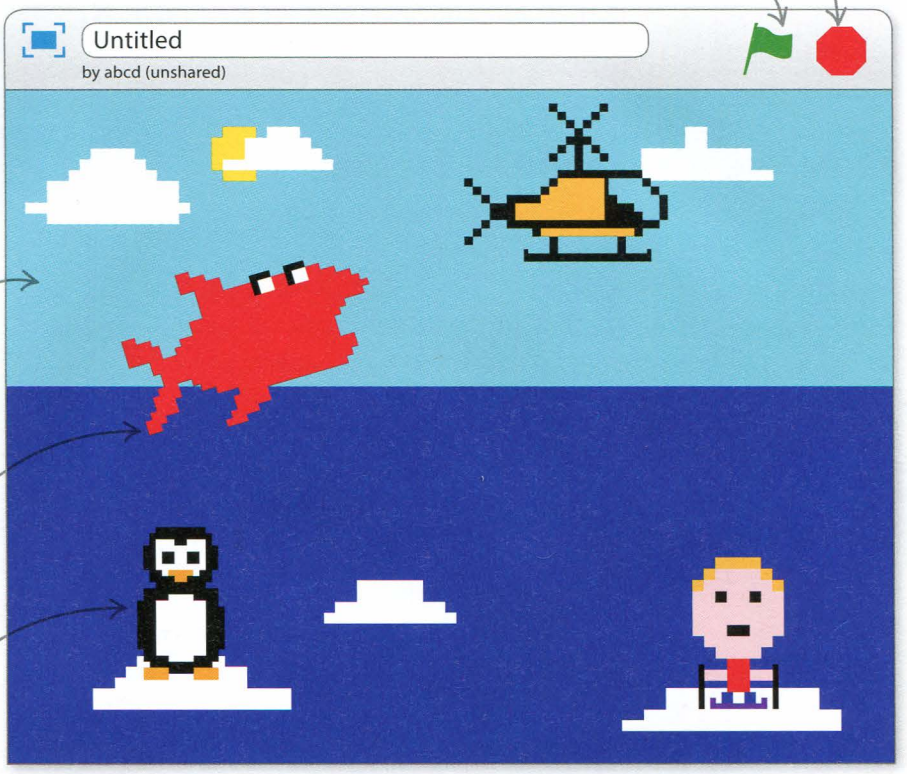
Что значит название Scratch?

Scratching — это способ смешения звуков для создания музыки. Язык Scratch позволяет объединять картинки, звуки и скрипты для создания программ.

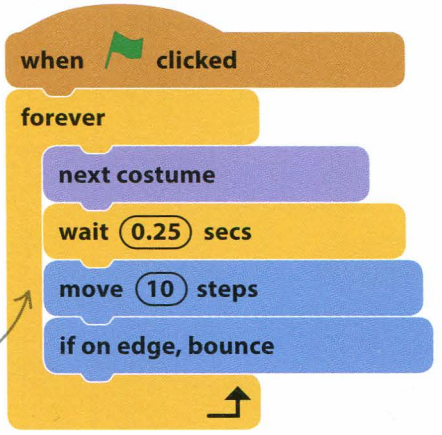
Типичная программа на Scratch

Это пример Scratch-программы. Действие происходит в области экрана, которая называется сценой. На сцену можно добавлять фоновые картинки и спрайты, «оживляя» их с помощью скриптов.

▷ Выполнение программы
Запуск программы называют также ее выполнением. Чтобы выполнить программу на Scratch, нажми на зеленый флажок над областью сцены.



▷ Скрипты для «оживления» спрайтов
Скрипты в Scratch состоят из блоков. Этот скрипт заставляет акулу плавать туда-сюда по экрану. Благодаря блоку next costume («следующий костюм») акула при каждом движении открывает и закрывает пасть.



Блок forever («всегда») делает перемещение спрайта бесконечным

ЗАПОМНИ

Программы на Scratch

Если сохранить свою работу в Scratch, получится проект. Проект содержит все использованные спрайты, фоны, звуки и скрипты. Если загрузить проект заново, все его элементы окажутся в прежнем состоянии — в каком были на момент сохранения. Проект Scratch — это компьютерная программа.

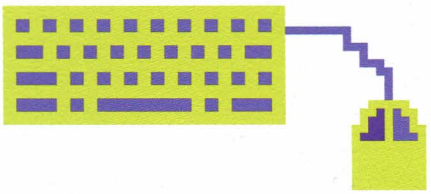
Установка и запуск Scratch

Чтобы начать программировать, понадобится программное обеспечение Scratch. Его можно установить на компьютер или работать с ним по сети — онлайн.

ЗАПОМНИ

Интернет-сайт Scratch

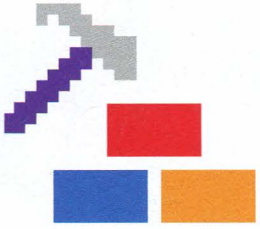
Сайт Scratch находится по адресу: **http://scratch.mit.edu/**



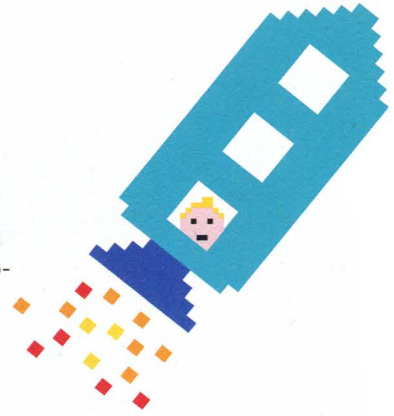
Создаем учетную запись Scratch

Учетная запись позволяет выкладывать готовые программы на сайт Scratch. Она также нужна, чтобы сохранять проекты при работе по сети. Зайди на сайт Scratch по адресу http://scratch.mit.edu и нажми «Вступайте в Скретч», чтобы создать учетную запись.

Примечание. Вы можете пользоваться русифицированной версией Scratch. Однако в большинстве языков программирования используют команды на английском, например в Python (см. сравнение команд на с. 102–105). В этой книге все примеры приведены на английском языке.



◁ **Начинаем работу**
Способ установки Scratch зависит от того, использовать ли его через интернет (по сети) или загрузить на компьютер (с диска).

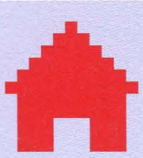


1 Установка

По сети

Зайди на сайт **http://scratch.mit.edu** и выбери «Регистрация». Заполни форму, создав имя пользователя и пароль. Убедись, что родители не возражают против твоей регистрации на сайте.

После регистрации на сайте Scratch выбери «Вход» и введи свое имя пользователя и пароль. Кликни «Создавай» вверху экрана, чтобы создать новую программу.



С диска

Скачай Scratch с сайта **http://scratch.mit.edu.** Запусти программу установки — на рабочем столе появится иконка Scratch.

Сделай двойной клик по иконке на рабочем столе — и Scratch запустится, готовый к вводу программ.

2 Запуск Scratch

СОВЕТЫ ЭКСПЕРТА

Управление мышкой

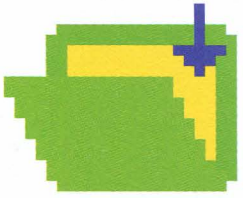
«Клик» («щелчок») означает нажатие левой кнопки мышки. «Правый клик» означает нажатие правой кнопки. Если у тебя мышка с единственной кнопкой, для правого клика следует нажать эту кнопку, удерживая клавишу Ctrl на клавиатуре.

Разные версии Scratch

В этой книге мы рассмотрим версию 2.0 языка Scratch. Постарайся использовать именно ее. Более старые версии будут работать немного иначе.

△ **Scratch 1.4**
В старой версии Scratch сцена находится с правой стороны экрана.

△ **Scratch 2.0**
В последней версии Scratch добавлены новые команды, а сцена находится слева.



3 Сохранение программы

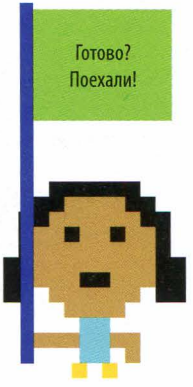
После входа на сайт Scratch автоматически сохраняет твою работу. Чтобы увидеть сохраненные программы, кликни по своему имени в правом верхнем углу экрана и выбери My Stuff («Мои работы»).

Открой меню File («Файл») вверху экрана и выбери Save as («Сохранить как»). Спроси у родителей, куда лучше всего сохранять свои работы.

4 Операционные системы

Сетевая версия Scratch работает на компьютерах с Windows, Ubuntu и Mac OS. Однако для ее работы нужен Adobe Flash, поэтому на некоторых планшетах она не запустится.

Скачанная версия Scratch работает на компьютерах с Windows или Mac OS. На компьютерах с Ubuntu она не запустится — если у тебя Ubuntu, попробуй сетевую версию.

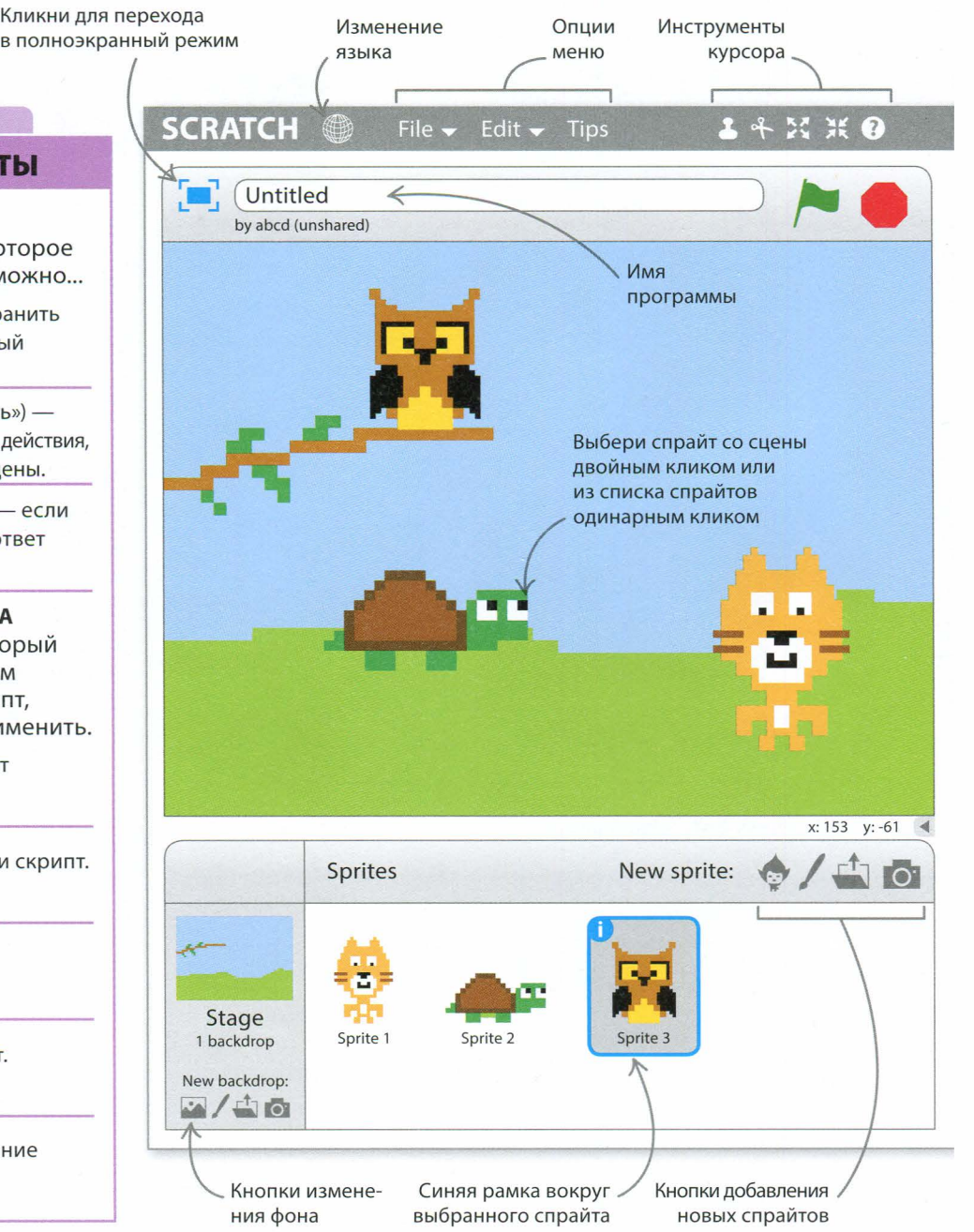


Интерфейс Scratch

Так выглядит экран, или интерфейс Scratch. Сцена находится слева, а справа можно создавать программы.

▽ Экспериментируй

Экспериментируй с интерфейсом Scratch, кликая по кнопкам и закладкам. Как их использовать, мы разберемся, выполняя проекты.



СОВЕТЫ ЭКСПЕРТА

Меню и инструменты

ОПЦИИ МЕНЮ

С помощью опций меню, которое находится вверху экрана, можно...

File — **File** («Файл») — сохранить работу, создать новый проект.

Edit — **Edit** («Редактировать») — отменить ошибочные действия, изменить размер сцены.

Tips — **Tips** («Подсказки») — если запутаешься, ищи ответ здесь.

ИНСТРУМЕНТЫ КУРСОРА

Кликни на инструмент, который хочешь использовать, затем кликни на спрайт или скрипт, к которому хочешь его применить.

Копировать спрайт или скрипт.

Удалить спрайт или скрипт.

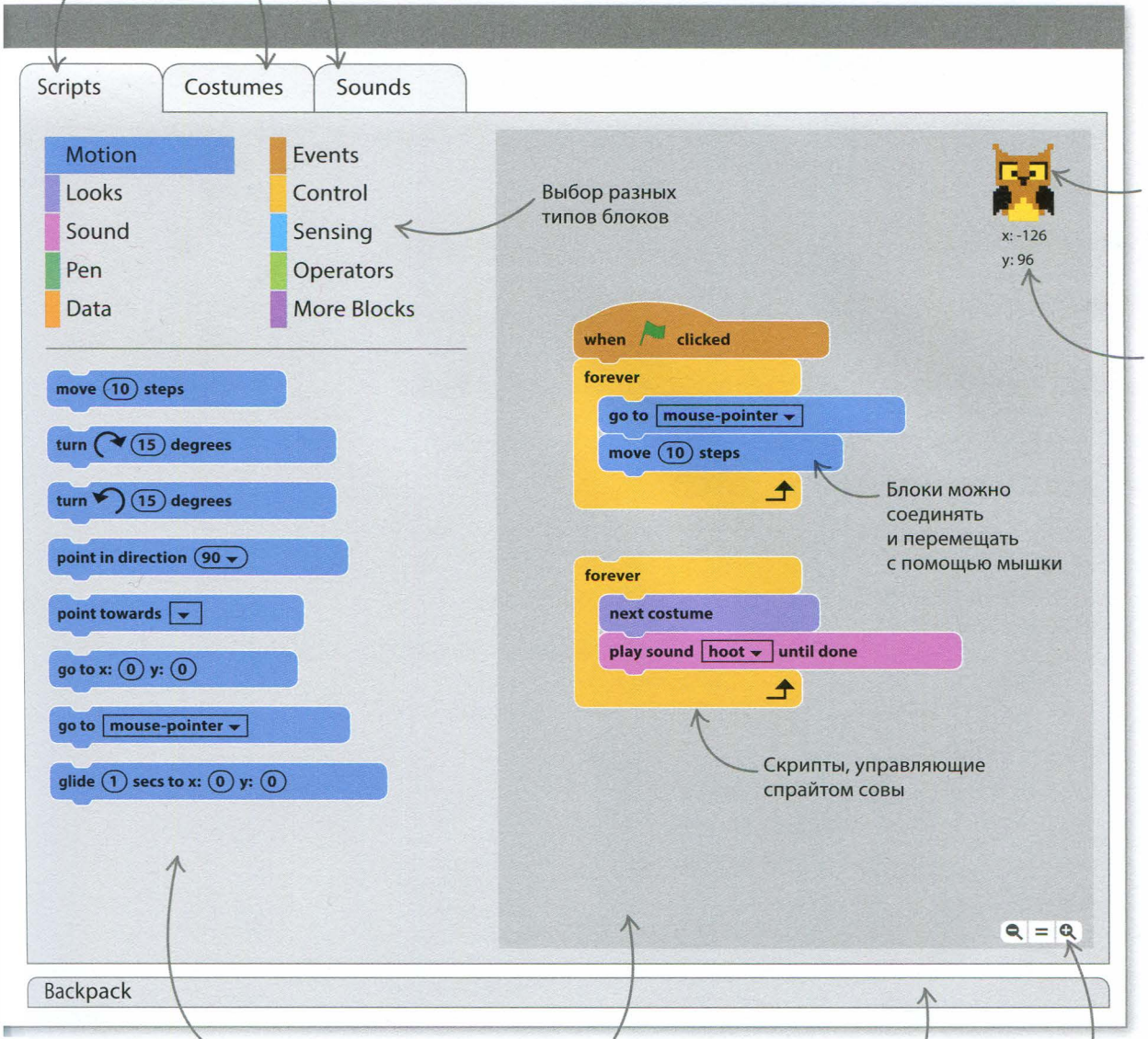
Увеличить спрайт.

Уменьшить спрайт.

Посмотреть описание блока.

▷ Карта Scratch

Сцена — область, где выполняются программы. Со спрайтами можно работать в списке спрайтов, а скриптовые блоки выбирать из палитры блоков. Область скриптов нужна, чтобы составлять скрипты.



Спрайты

Спрайты — основные элементы в Scratch. Каждая Scratch-программа состоит из спрайтов и скриптов для управления ими. В программе «Убеги от дракона!» со страниц 32–37 используются спрайты кота, дракона и пончика.

СМОТРИ ТАКЖЕ

❮ 26–27 Интерфейс Scratch

Костюмы 40–41 ❯

Прятки 42–43 ❯

Что могут спрайты?

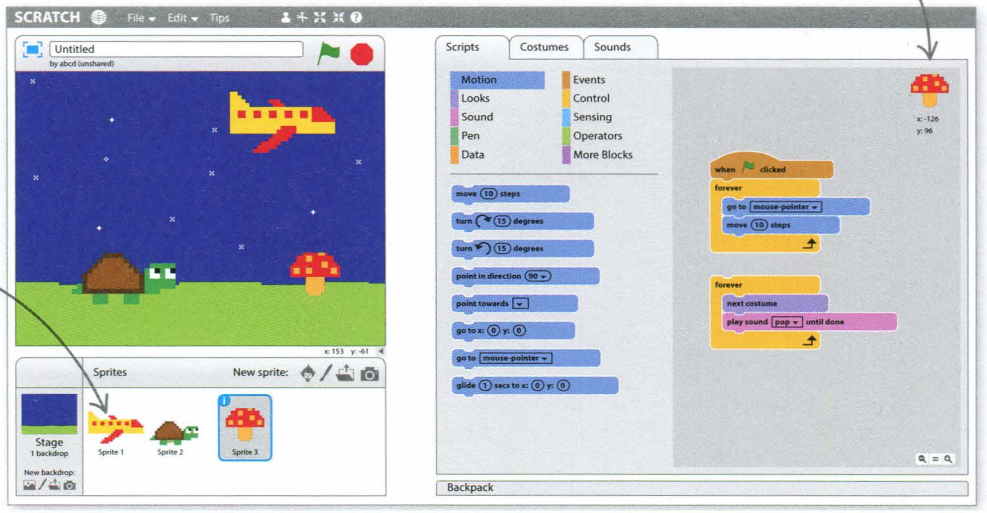
Спрайты — это изображения на сцене, и для управления ими служат скрипты. Спрайты можно научить реагировать на другие спрайты и на действия пользователя программы. Например, вот что могут спрайты.

- **Перемещаться по сцене**
- **Говорить через баллоны, вот так**
- **Изменять свой облик**
- **Выполнять команды пользователя**
- **Издавать звуки и играть музыку**
- **Реагировать на столкновения с объектами**

Спрайты в интерфейсе Scratch

В проекте может быть много спрайтов, и у каждого спрайта могут быть свои скрипты. Важно не перепутать, к какому спрайту какие скрипты добавлять, и помнить, как переключаться между спрайтами.

Скрипты на картинке принадлежат этому спрайту



Переключайся между спрайтами, кликая по ним

▷ **Спрайты и скрипты**

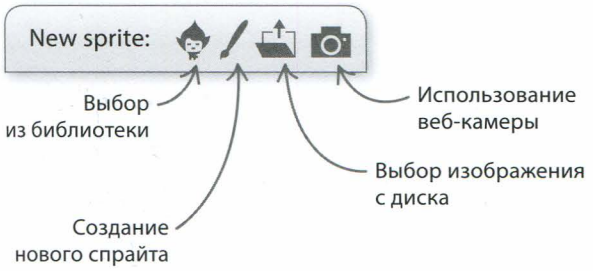
В проекте может быть много спрайтов, и каждый спрайт может иметь много скриптов.

Создание и редактирование спрайтов

Чем больше в игре спрайтов, которые могут сталкиваться, маневрировать и преследовать друг друга, тем она увлекательнее. Создавать, копировать и удалять спрайты очень просто.

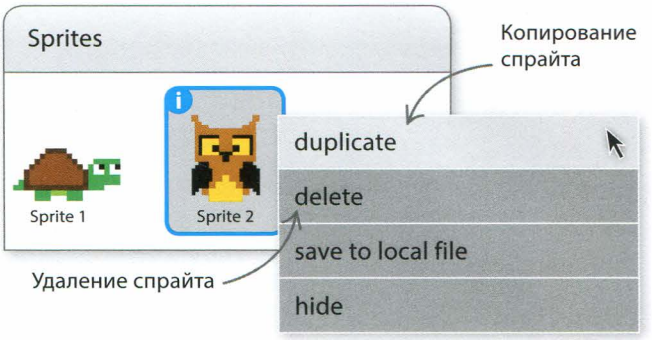
▽ **Создаем спрайт**

Используй кнопки над списком спрайтов, чтобы добавить спрайт в программу или создать новый.



▽ **Копирование или удаление спрайта**

Чтобы скопировать спрайт и его скрипты, сделай правый клик по спрайту и выбери duplicate («дублировать»).

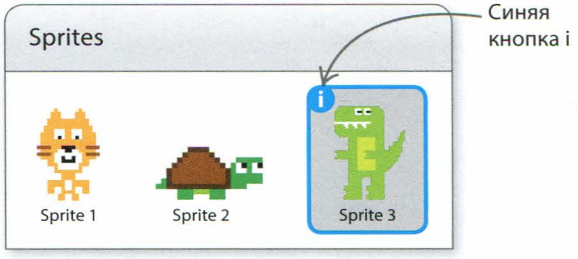


Название спрайта

Если создать новую программу в Scratch, спрайт кота будет называться Sprite1. Программировать будет легче, если ты дашь своим спрайтам более понятные имена. Кроме того, это упростит работу со скриптами.

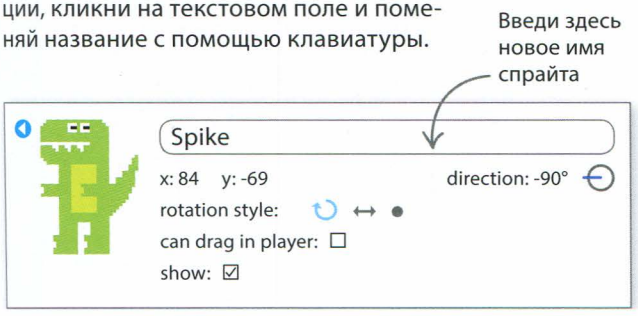
1 Выбери спрайт

Выбери спрайт в списке спрайтов и кликни по кнопке i в углу рамки.



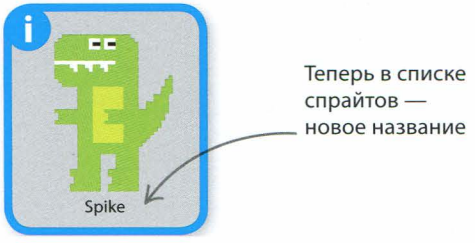
2 Измени название

Когда откроется панель информации, кликни на текстовом поле и поменяй название с помощью клавиатуры.



3 Спрайт переименован

Кликни по синей стрелочке слева от спрайта, чтобы закрыть панель.



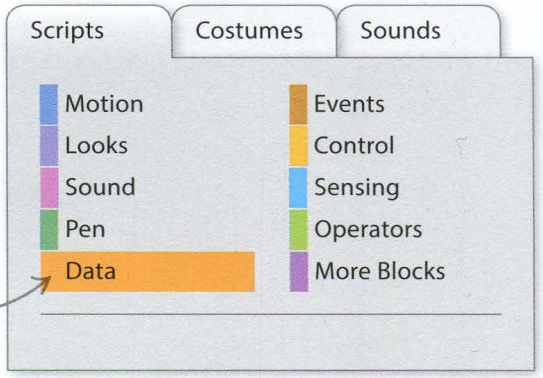
Цветные блоки и скрипты

СМОТРИ ТАКЖЕ
❮ 26–27 Интерфейс Scratch
Убеги от дракона! 32-37 ❯

Блоки раскрашены в разные цвета в зависимости от их назначения. При их соединении получаются скрипты, в которых блоки выполняются по очереди.

Цветные блоки

В Scratch есть блоки десяти разных типов. Переключайся между ними с помощью кнопок палитры блоков. Кликни по цветной кнопке, чтобы увидеть все блоки в секции.



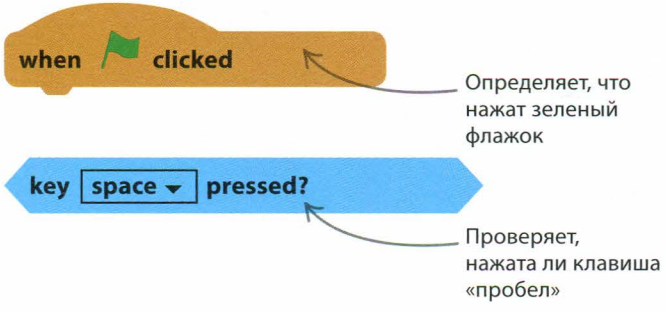
Кнопка, показывающая оранжевые блоки Data («Данные»)

Назначение блоков

Блоки разных типов имеют различные функции в программе. Некоторые из них двигают спрайты, некоторые управляют звуками, некоторые определяют, что должно происходить.

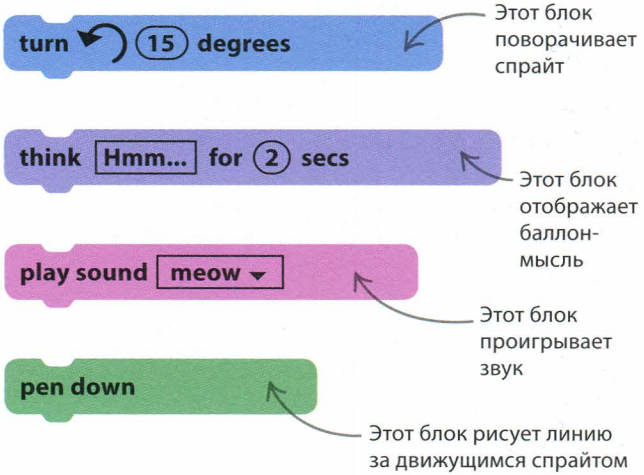
▽ **«События» и «Сенсоры»**

Коричневые блоки Events («События») управляют событиями. Голубые блоки Sensing («Сенсоры») дают информацию о клавиатуре, о мышке и о столкновениях спрайтов.

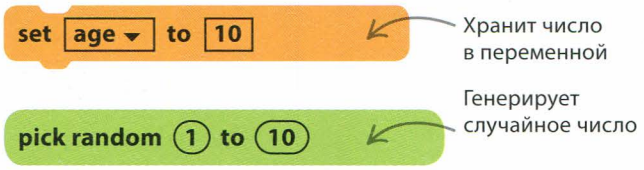


▽ **«Движение», «Внешность», «Звук» и «Перо»**

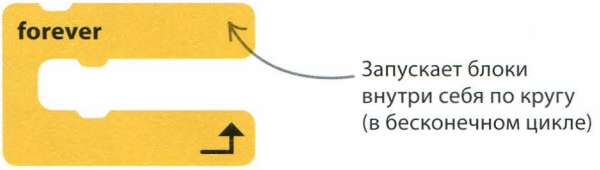
Motion, looks, sound, pen — эти блоки управляют поведением спрайта на экране, то есть выводом программы. Выбери спрайт и попробуй каждый из блоков, чтобы узнать, что он делает.



▽ **«Данные» и «Операторы»**
Оранжевые блоки Data («Данные») и зеленые блоки Operators («Операторы») хранят числа и слова и совершают с ними разные действия.

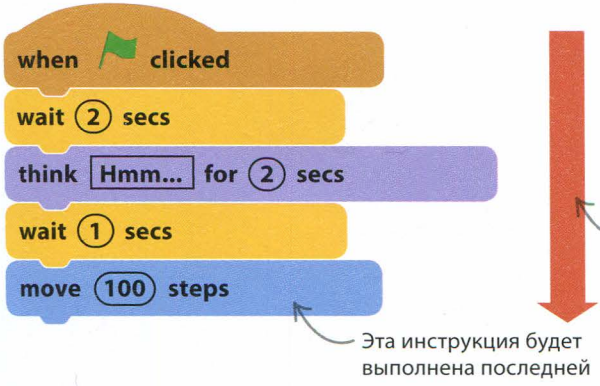


▽ **«Управление»**
Блоки Control («Управление») определяют, как запускать другие блоки. Их можно запрограммировать на повторение инструкций.



Порядок выполнения скриптов

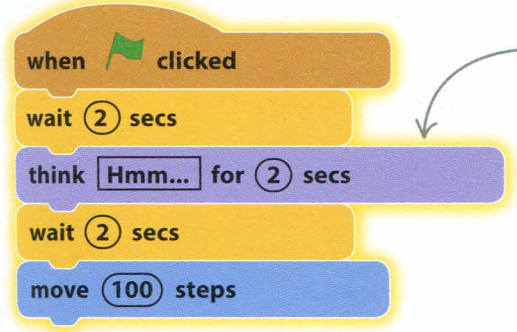
Когда программа запущена, Scratch выполняет инструкции блоков. Он делает это, начиная с верхнего блока в скрипте и далее книзу.



◁ **Задумчивый кот**
Если добавить этот скрипт к спрайту кота, кот подождет две секунды, задумается и затем двинется с места.

Выполнение скриптов

Когда скрипт работает, он подсвечивается. Используй зеленый флажок для запуска скрипта или кликни по скрипту или блоку, чтобы запустить его.

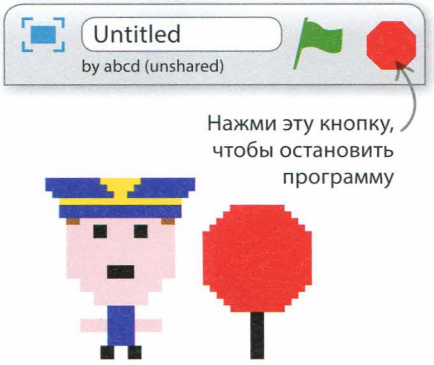


◁ **Проверка скриптов**
Чтобы убедиться, что скрипт работает правильно, кликни по нему.

ЗАПОМНИ

Остановка скриптов

Чтобы остановить все скрипты, которые выполняются в программе, кликни по шестиугольной красной кнопке над сценой — она находится рядом с зеленым флажком запуска программы.



ПРОЕКТ 1

Убеги от дракона!

В этом проекте показаны основы Scratch-программирования. Ты узнаешь, как создать игру, цель которой — помочь спрайту кота увернуться от огнедышащего дракона.

СМОТРИ ТАКЖЕ

❮ 24-25 Установка и запуск Scratch

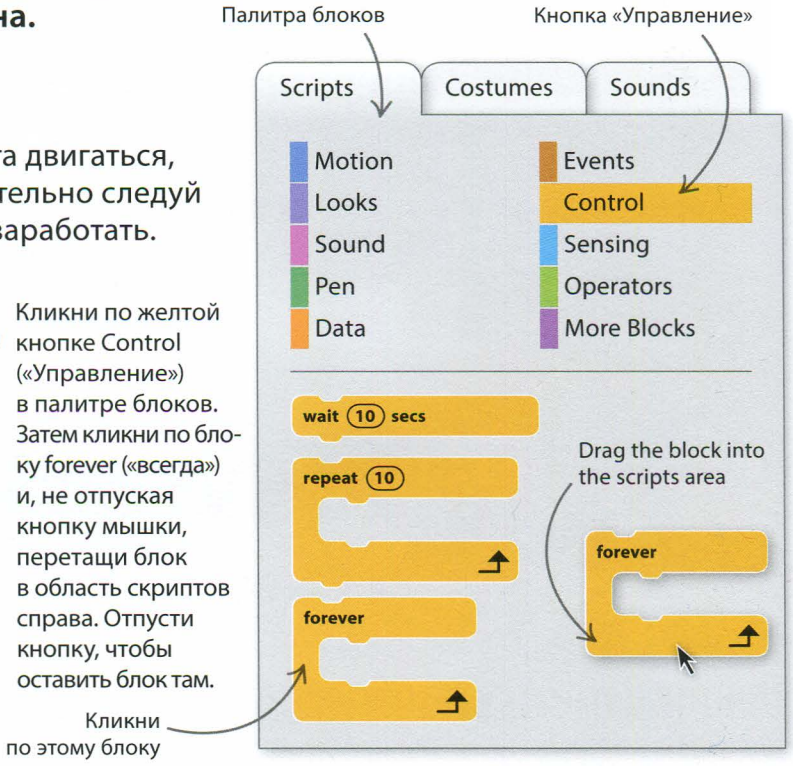
❮ 26–27 Интерфейс Scratch

Научим кота двигаться

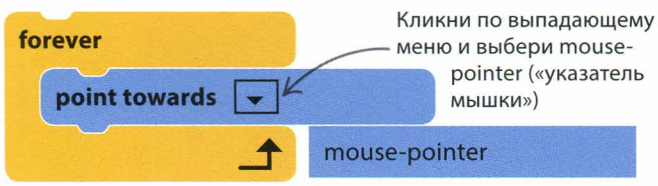
На этом этапе мы научим спрайт кота двигаться, преследуя указатель мышки. Внимательно следуй инструкциям, иначе игра может не заработать.

1 Открой Scratch. Кликни по опции File («Файл») в меню и выбери New («Новый»), чтобы создать новый проект. Появится спрайт кота.

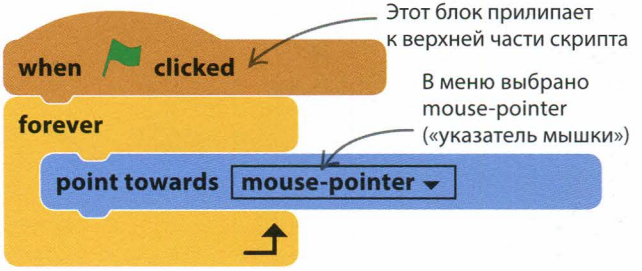
2 Кликни по желтой кнопке Control («Управление») в палитре блоков. Затем кликни по блоку forever («всегда») и, не отпуская кнопку мышки, перетащи блок в область скриптов справа. Отпусти кнопку, чтобы оставить блок там.



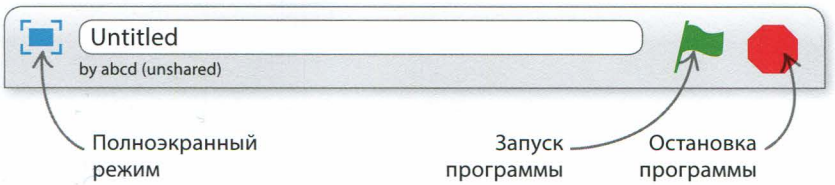
3 Кликни по синей кнопке Motion («Движение») в палитре блоков. Появятся синие блоки движений. Перетащи блок point towards («повернуться к») в область скриптов и помести его внутрь блока forever («всегда»). Кликни по черной стрелочке в блоке и выбери из выпадающего меню mouse-pointer («указатель мышки»).



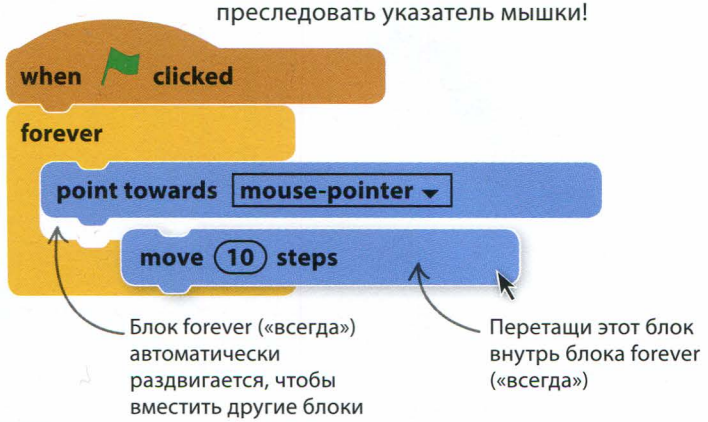
4 Кликни по кнопке Events («События») в палитре блоков. Перетащи блок when green flag clicked («когда щелкнут по зеленому флажку») в область скриптов. Присоедини его к скрипту сверху.



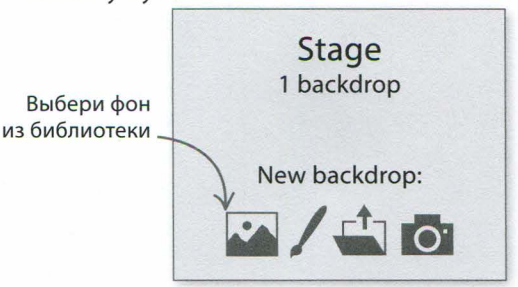
5 Попробуй запустить программу, кликнув по зеленому флажку сверху сцены. Если поводить над сценой указателем мышки, кот будет поворачиваться в его сторону.

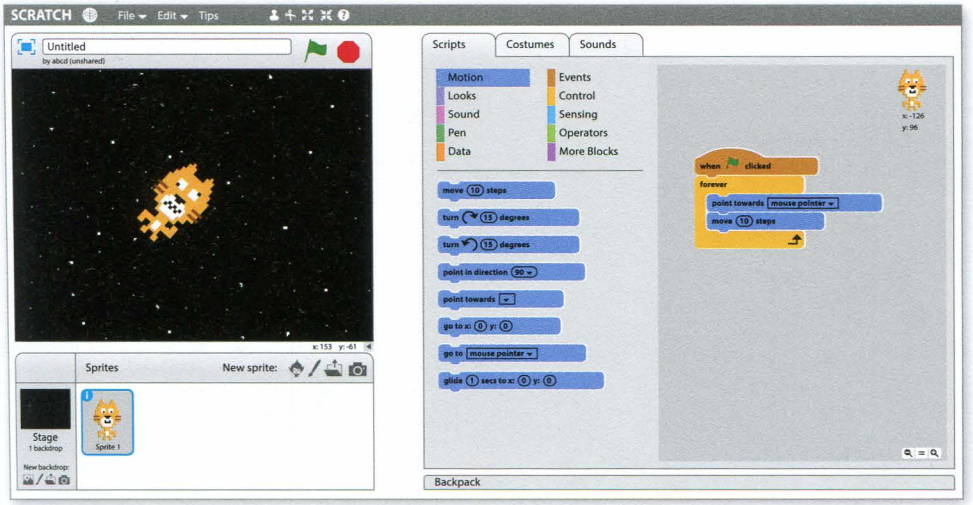


6 Снова кликни по кнопке Motion («Движение») и перетащи блок move 10 steps («идти 10 шагов») в область скриптов, внутрь блока forever («всегда»). Нажми на зеленый флажок, и кот начнет преследовать указатель мышки!



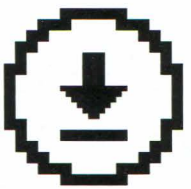
7 Изображение позади спрайтов называется фоном. Слева от списка спрайтов есть кнопка добавления фона из библиотеки. Кликни по ней и выбери из списка слева тему Cosmos («Космос»). Кликни по картинке stars, а затем по кнопке OK в правом нижнем углу.





◁ Кот в космосе

Теперь интерфейс Scratch выглядит так. Запусти программу, и кот будет гоняться за указателем мышки в космосе.



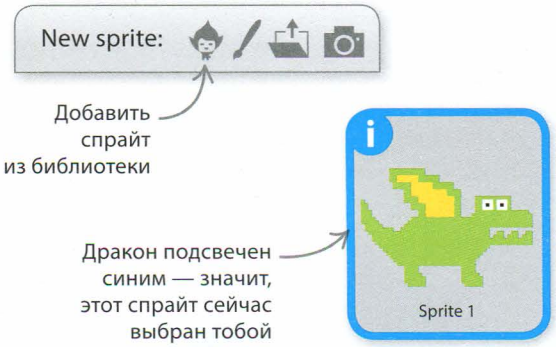
Сетевая версия Scratch автоматически сохраняет проект. Чтобы сохранить работу в дисковой версии, кликни File и выбери Save As.

УБЕГИ ОТ ДРАКОНА!

Добавим огнедышащего дракона

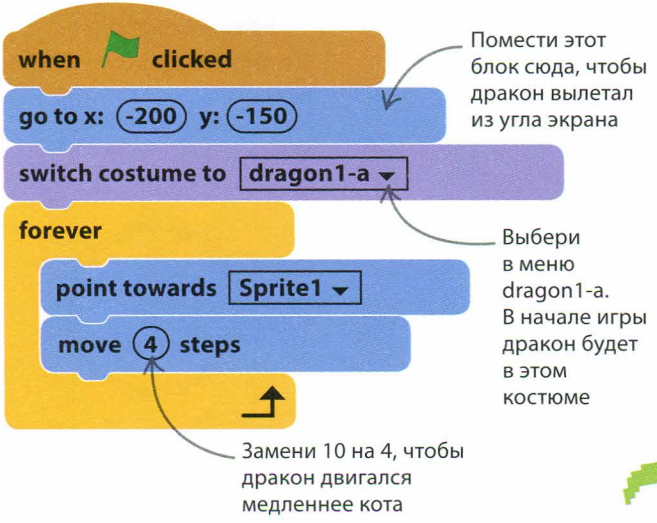
Теперь, когда кот преследует мышку, сделаем так, чтобы за котом гонялся дракон. Нельзя допустить, чтобы дракон поймал кота, а то он его поджарит.

8 Над списком спрайтов есть кнопка добавления спрайта из библиотеки. Кликни по ней, выбери категорию Fantasy («Фантастика») в меню слева, затем выбери спрайт Dragon. Кликни по кнопке OK в правом нижнем углу.



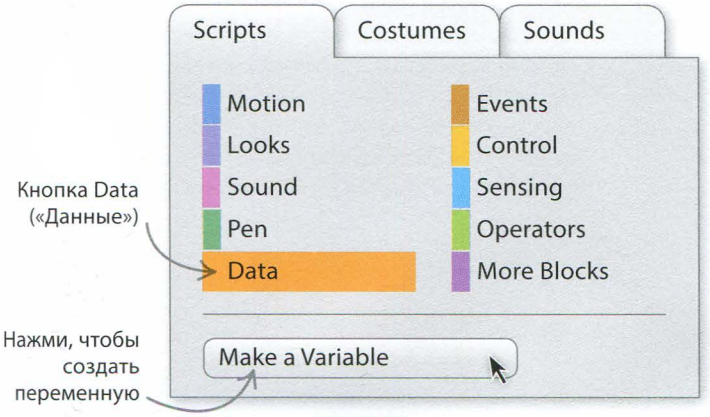
9 Добавь к спрайту дракона этот скрипт. С помощью цветных кнопок в палитре спрайтов найди и перетащи показанные ниже блоки в область скриптов. Теперь дракон будет гоняться за котом.

10 Кликни по синей кнопке Motion («Движение») и перетащи блок go to x:0 y:0 («перейти в x:0 y:0») в скрипт. Кликая по числам в блоке, замени их на –200 и –150. Кликни по фиолетовой кнопке Looks («Внешность») и добавь в свой скрипт блок switch costume to («сменить костюм на»).

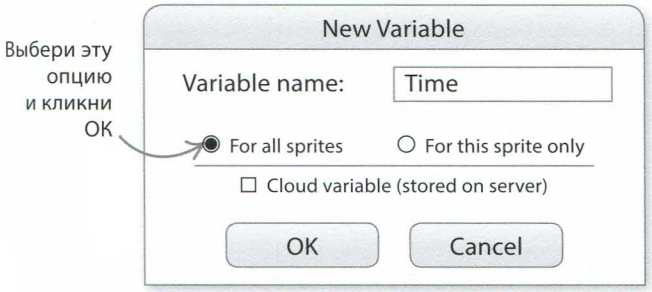


11 Когда спрайт дракона подсвечен, добавь еще один скрипт в область скриптов. Блок wait until («ждать до») ищи в секции Control («Управление»), а блок touching («касается») — в секции Sensing («Сенсоры»). Теперь дракон, столкнувшись с котом, выдыхает огонь.

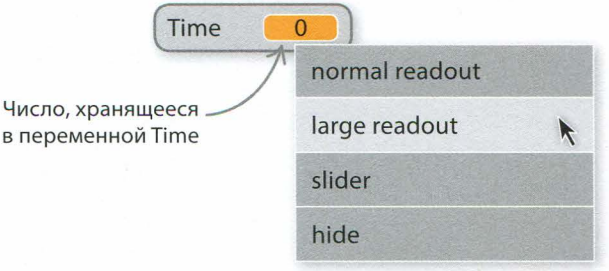
12 Переменные нужны для хранения информации. На этом шаге мы используем переменную, чтобы создать таймер, показывающий, сколько игрок смог продержаться. Кликни по кнопке Data («Данные»), а затем по кнопке Make a Variable («Создать переменную»).



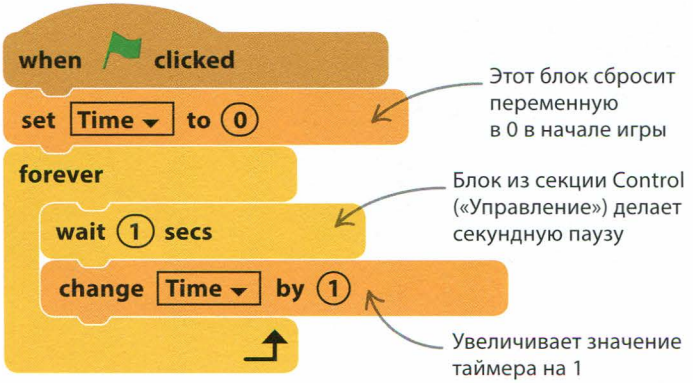
13 Введи имя переменной Time («Время»), убедись, что внизу выбрана опция For all sprites («Для всех спрайтов»), и кликни OK. Эта опция означает, что переменную смогут использовать и кот, и дракон, и любые другие спрайты.



14 Имя переменной и хранящееся в ней число появятся на сцене в рамочке. Сделай на ней правый клик и выбери large readout («крупный вид»). Теперь в рамке останется только число.



15 Создание переменной добавляет новые блоки в секцию Data («Данные») палитры блоков. Перетащи блоки set Time 0 («задать Time значение 0») и change Time 1 («изменить Time на 1») из секции Data («Данные») в область скриптов, чтобы создать этот новый скрипт. Добавить его можно к любому спрайту.



СОВЕТ ЭКСПЕРТА

Сделай игру сложнее

Попробуй изменить скорости или размеры спрайтов.

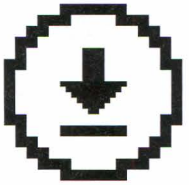
Сделать дракона быстрее:

move (5) steps

Сделать дракона больше или меньше:

кликни по этой иконке, а затем по спрайту, чтобы его увеличить;

кликни по этой иконке, а затем по спрайту, чтобы его уменьшить.



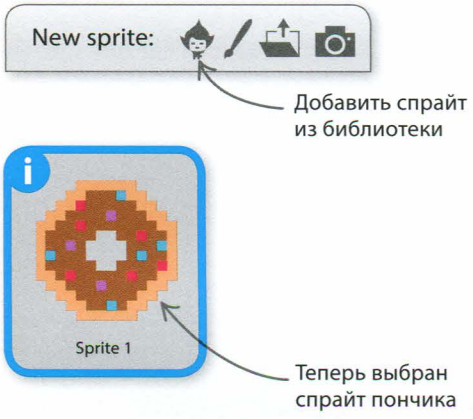
Не забудь сохранить свою работу.

УБЕГИ ОТ ДРАКОНА!

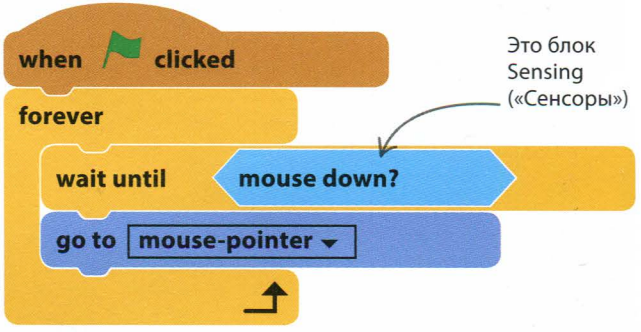
Добавим вкусный пончик

В библиотеке Scratch много спрайтов. Сделаем игру сложнее, добавив спрайт пончика, за которым будет гоняться кот.

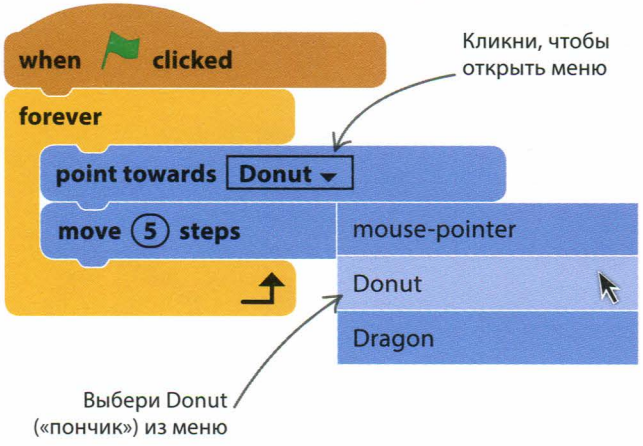
16 Кликни по кнопке над списком спрайтов, чтобы добавить новый спрайт из библиотеки. Выбери Donut из категории Things («Предметы») слева и кликни OK.



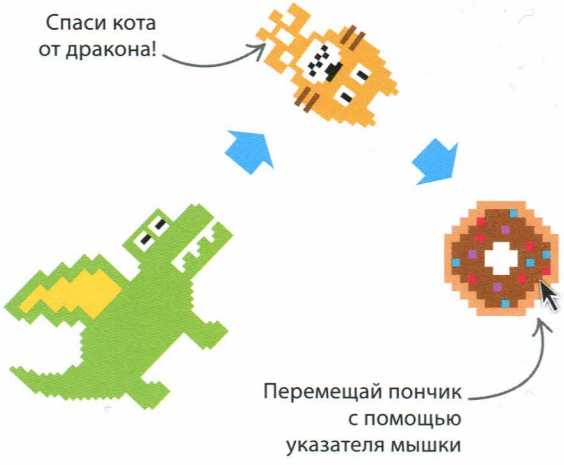
17 Добавь к пончику этот скрипт. Блок mouse down («мышка нажата?») находится в секции Sensing («Сенсоры»), а блок go to mouse-pointer («перейти в указатель мышки») — в секции Motion («Движение»). При нажатии кнопки мышки этот скрипт перемещает пончик к ее указателю.



18 Выбери кота в списке спрайтов, чтобы появились его скрипты. Кликни по меню в блоке point towards mouse-pointer («повернуться к указателю мышки»). Поменяй значение, чтобы кот гонялся за пончиком, а не за указателем мышки.

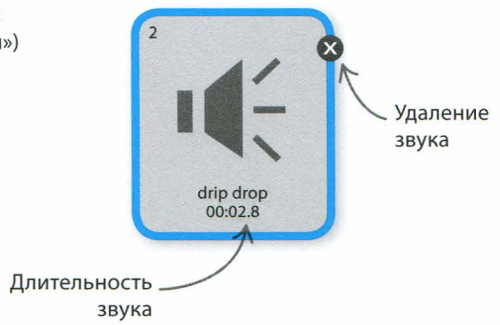


19 Кликни по зеленому флажку, чтобы запустить программу. Нажми кнопку мышки, и пончик переместится к указателю. Кот гоняется за пончиком, а дракон — за котом.

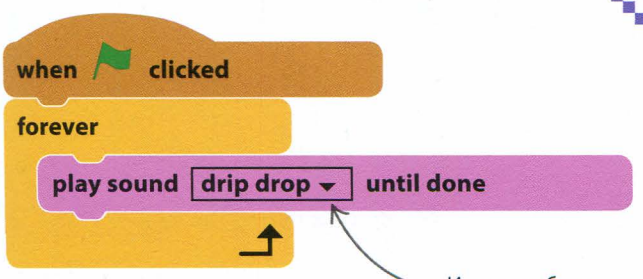


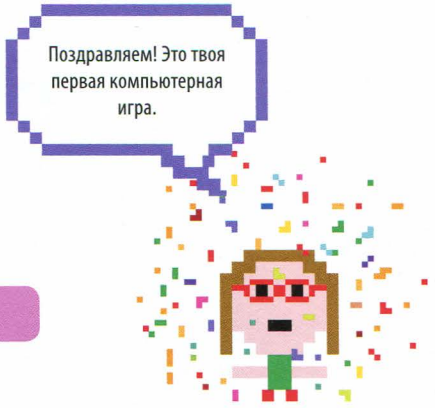
20 Теперь добавим музыку. Кликни по вкладке Sounds («Звуки») над палитрой блоков. У каждого спрайта есть свои звуки, и здесь ими можно управлять. Кликни по кнопке слева, чтобы добавить звук из библиотеки.

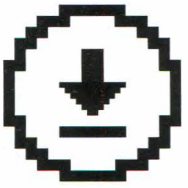
21 Выбери звук drip drop и кликни по кнопке OK в нижнем правом углу. Теперь звук добавлен к спрайту кота и появился под вкладкой Sounds («Звуки»).



22 Кликни по вкладке Scripts («Скрипты»), чтобы вернуться к области скриптов. Добавь этот скрипт к спрайту кота, чтобы в игре звучала музыка. Запусти программу и наслаждайся!







Не забудь сохранить свою работу.

ЗАПОМНИ

Твои успехи

Этот проект продемонстрировал некоторые возможности Scratch. Вот список твоих достижений.

Создание программы: создание игры с помощью скриптов, состоящих из блоков.

Добавление картинок: использование фонов и спрайтов.

Перемещение спрайтов: преследующие друг друга игровые спрайты.

Использование переменной: создание игрового таймера.

Использование костюмов: изменение внешности дракона с помощью разных костюмов.

Добавление музыки: звук, играющий непрерывно во время игры.

Перемещение объектов

СМОТРИ ТАКЖЕ
‹ 28–29 Спрайты
Координаты 56–57 ›

Компьютерные игры — это выстрелы, маневры, засады и атаки. Персонажи могут бегать, управлять звездолетами или водить гоночные автомобили. Чтобы создавать захватывающие игры в Scratch, нужно сперва научиться перемещать спрайты.

Блоки Motion

Синие блоки Motion («Движение») управляют движением спрайтов. Создай новый проект, открыв меню File («Файл») и выбрав New («Новый»). Проект начинается с кота, стоящего посреди сцены, и этот кот уже рвется в бой.

Scratch не позволяет спрайтам уйти со сцены, так что ты нас не потеряешь.

1 Первые шаги

Перетащи в область скриптов блок move 10 steps («идти 10 шагов») из секции Motion («Движение») палитры блоков. При каждом клике по блоку кот двигается.

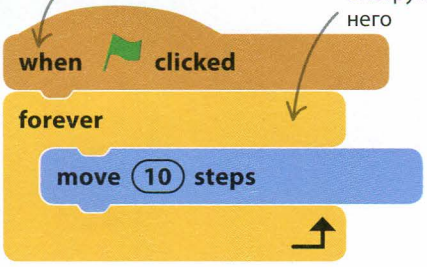
Кликни по числу в блоке и впиши другое значение, чтобы изменить скорость перемещения кота

2 Продолжим движение

Перетащи желтый блок forever («всегда») из палитры блоков так, чтобы блок move 10 steps («идти 10 шагов») оказался у него внутри. Запусти программу, кликнув по зеленому флажку над сценой. Кот будет двигаться, пока не упрется в край сцены.

Добавь этот блок, чтобы Scratch знал, когда запустить скрипт

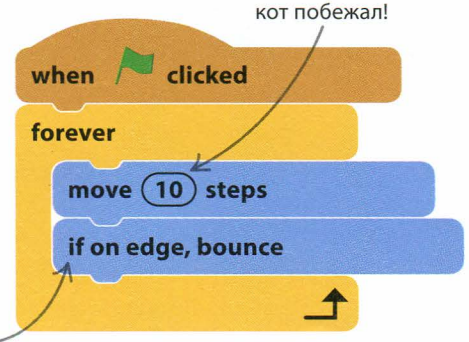
Блок forever («всегда») бесконечно повторяет все инструкции внутри него



3 Отскоки

Перетащи блок if on edge, bounce («если на краю, оттолкнуться») внутрь блока forever («всегда»). Теперь кот, упершись в край сцены, отскочит от него. При движении налево кот будет перевернут вверх тормашками.

Попробуй заменить число 10 на 30, чтобы кот побежал!



Этот блок развернет кота, когда тот упрется в край сцены

СОВЕТЫ ЭКСПЕРТА

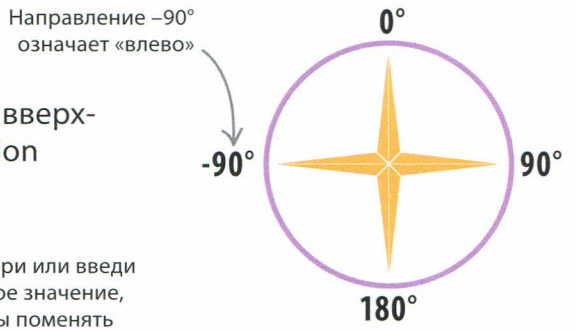
Типы разворота

Найди кота в списке спрайтов в левой нижней части экрана и кликни по кнопке i в левом верхнем углу рамки. Теперь можно изменить тип разворота спрайта — коту необязательно ходить на голове!

- Кот смотрит в направлении движения, порой оказываясь вверх ногами.
- Кот смотрит влево или вправо и всегда стоит на ногах.
- Спрайт кота остается неизменным.

В каком направлении?

Теперь кот ходит по экрану вправо-влево. Можно изменить направление движения, чтобы кот ходил вверх-вниз или даже по диагонали. Используя блоки Motion («Движение»), можно создать игру в кошки-мышки.



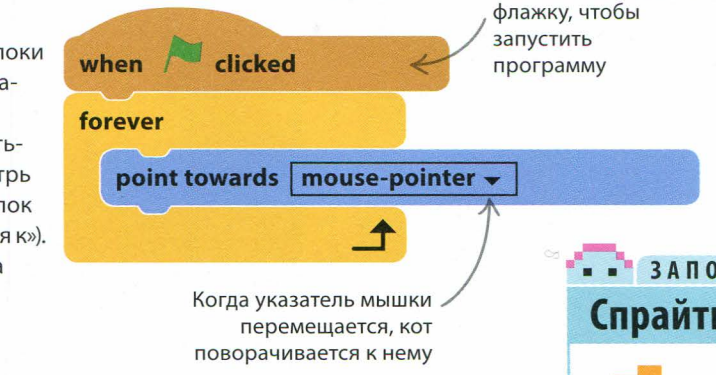
△ **Компас**
Направления измеряются в градусах, где 0° означает верх. Можно использовать любые числа в диапазоне от –179° до +180°.

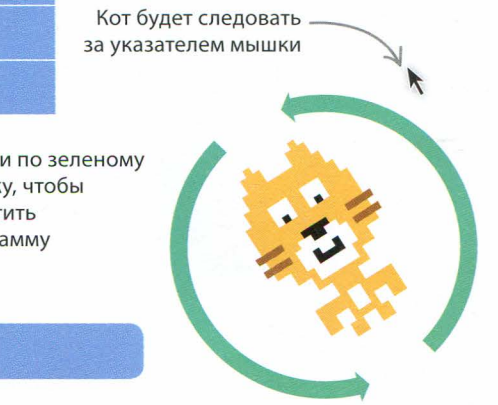
4 Двигаемся в правильном направлении

Перетащи блок point in direction («повернуть в направлении») в область скриптов и открой его выпадающее меню, где можно выбрать одно из четырех направлений. Также можно ввести свое направление, кликнув по числу в блоке.

5 Кот и мышка

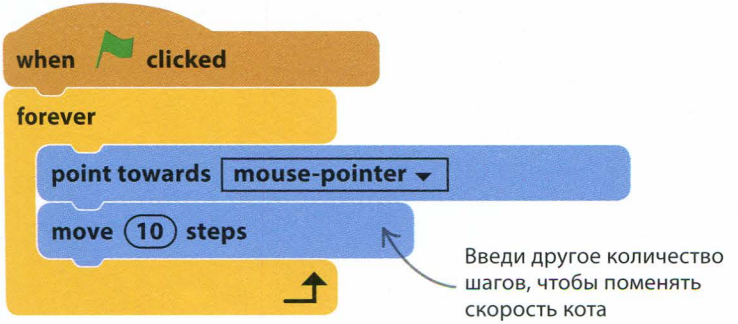
Удали из скрипта блоки move 10 steps («идти 10 шагов») и if on edge, bounce («если на краю, оттолкнуться»). Затем перетащи внутрь блока forever («всегда») блок point towards («повернуться к»). Открой меню этого блока и выбери mouse-pointer («указатель мышки»).





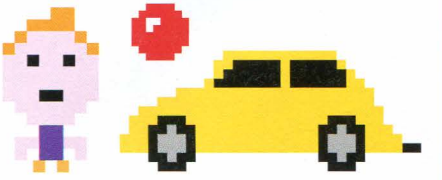
6 Погоня за мышкой

Как сделать, чтобы кот гонялся за мышкой? Перетащи блок move 10 steps («идти 10 шагов») внутрь блока forever («всегда»). Теперь кот пойдет к указателю мышки.



ЗАПОМНИ

Спрайты



Спрайты — это объекты Scratch-программы, которые можно перемещать по экрану (см. с. 28–29). В каждом новом проекте будет спрайт кота, но ты можешь добавить из библиотеки спрайты автомобилей, динозавров, танцоров и многие другие. Можно даже создать свой спрайт.

Костюмы

Чтобы изменить внешний вид спрайта, например выражение лица или позу персонажа, нужно сменить его костюм. Костюмы — это изображения спрайта в разных позах.

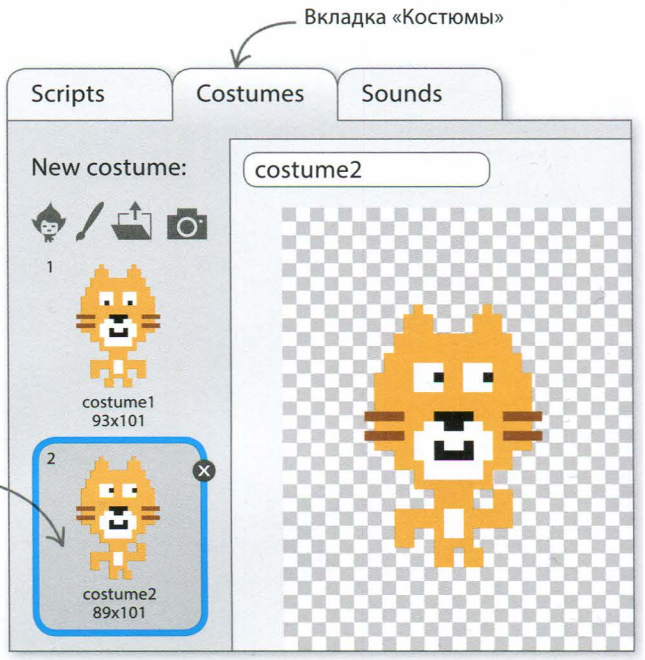
СМОТРИ ТАКЖЕ

❮ 38–39 Перемещение объектов

Обмен сообщениями 70–71 ❯

Вкладка костюмов

Смена костюмов может заставить спрайт двигать руками и ногами. Если переключаться между двумя костюмами, которые есть у кота, возникнет впечатление, что кот идет. Начни новый проект и поэкспериментируй с костюмами.

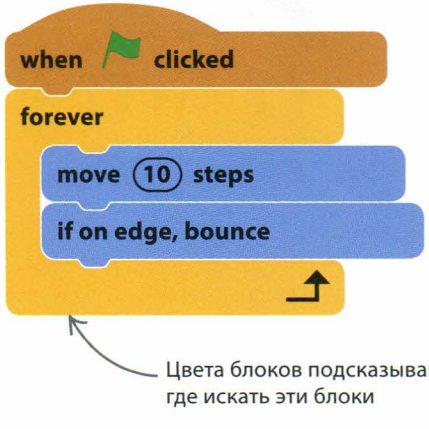


1 Разные костюмы

Кликни по вкладке Costumes («Костюмы») и посмотри на костюмы кота. Это изображения кота с разным положением лап.

2 Научим кота двигаться

Добавь этот скрипт, чтобы кот начал двигаться. Кот будет скользить по экрану, не меняя положения лап, потому что не меняется его изображение.



3 Меняем коту костюм

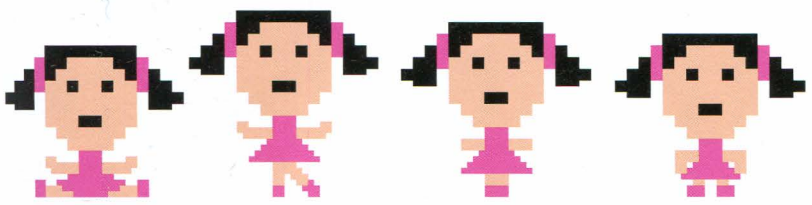
Добавь блок next costume («следующий костюм») из секции Looks («Внешность») палитры блоков, чтобы костюм кота менялся при каждом шаге. Теперь его лапы будут двигаться.

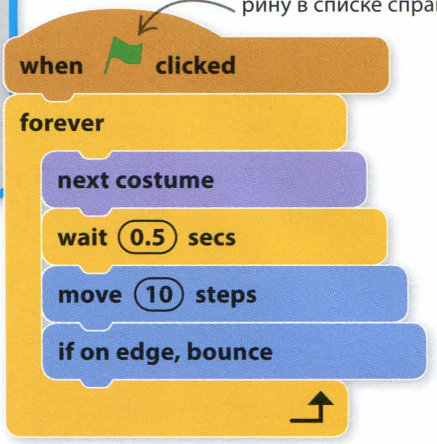
Танцующая балерина

Теперь попробуем научить балерину танцевать. Добавь спрайт балерины из библиотеки. Выбери спрайт кота и перетащи его скрипт на изображение балерины в списке спрайтов. У балерины появится копия этого скрипта.

Перетащи скрипт на балерину в списке спрайтов

Балерина начнет танцевать, если кликнуть на зеленый флажок





△ **Скрипт балерины**

Один и тот же скрипт годится и для балерины, и для кота. У балерины четыре костюма, которые сменяют друг друга, когда она танцует.

СОВЕТЫ ЭКСПЕРТА

Переключение

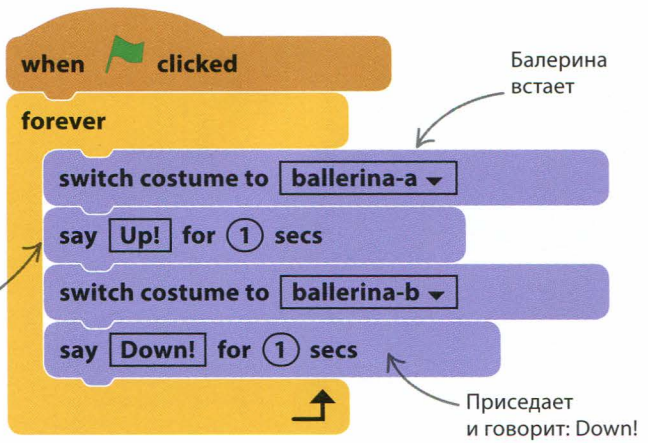
С помощью блока switch costume to («сменить костюм на») можно переключить костюм спрайта на заданный. Используй этот блок, чтобы показать определенное изображение спрайта.

Смена костюмов: Выбери нужный костюм из меню блока.

Смена фона: Этот блок задает фоновую картинку сцены.

Добавим баллоны прямой речи

Можно использовать баллоны, чтобы спрайт «разговаривал» во время смены костюма. Добавь блок say Hello! for 2 secs («говорить Hello! в течение 2 секунд») и измени текст в нем, чтобы спрайт говорил что-нибудь другое.



Балерина встает

Балерина говорит: Up!

Приседает и говорит: Down!

Прятки

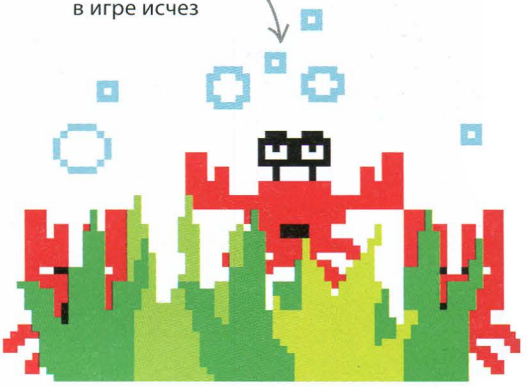
СМОТРИ ТАКЖЕ
‹ 38–39 Перемещение объектов
Обмен сообщениями 70–71 ›

Добро пожаловать в студию спецэффектов! Фиолетовые блоки Looks («Внешность») могут научить спрайты исчезать и появляться, увеличиваться и уменьшаться, таять и проявляться.

Прячем спрайты

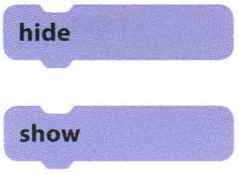
Если нужно, чтобы спрайт исчез, используй блок hide («скрыть»). Спрайт останется на сцене и сможет по ней двигаться, но будет незаметен до тех пор, пока блок show («показать») не вернет ему видимость.

Используй блок hide («скрыть»), если нужно, чтобы спрайт в игре исчез



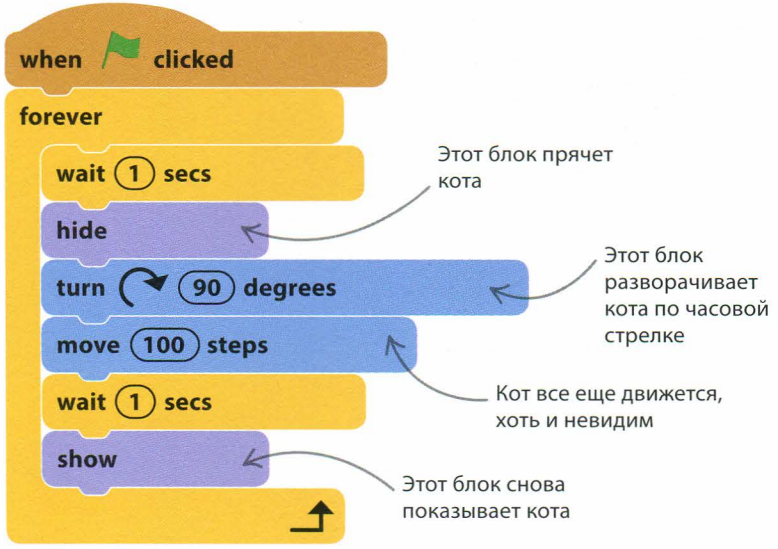
▷ Скрыть и показать

Чтобы спрайт исчез, используй блок hide («скрыть»). Когда настанет время показать его снова, используй блок show («показать»). Эти блоки находятся в секции Looks («Внешность») палитры блоков.



▽ Исчезающий кот

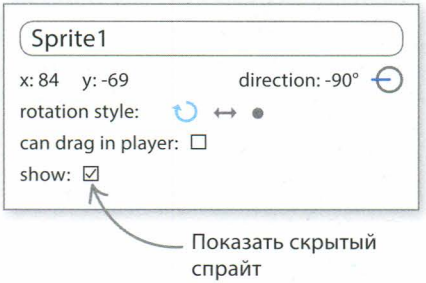
Попробуй этот скрипт со спрайтом кота. Кот будет исчезать и появляться, продолжая движение, даже когда он невидим.



СОВЕТЫ ЭКСПЕРТА

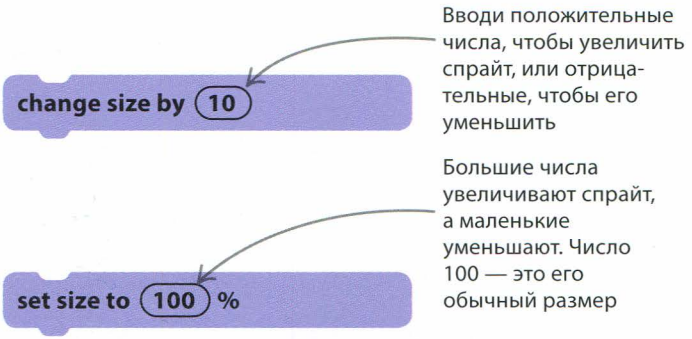
Показать спрайт

Выбери спрайт в списке спрайтов. Кликни по кнопке i, чтобы открыть панель свойств. С помощью отметки show («показать») можно менять видимость спрайта.

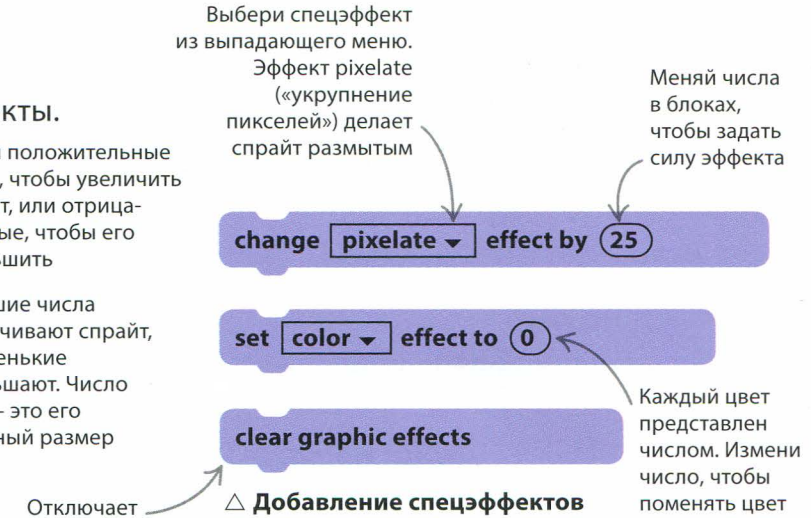


Размеры и эффекты

В скриптах можно менять размер спрайта и добавлять ему спецэффекты.



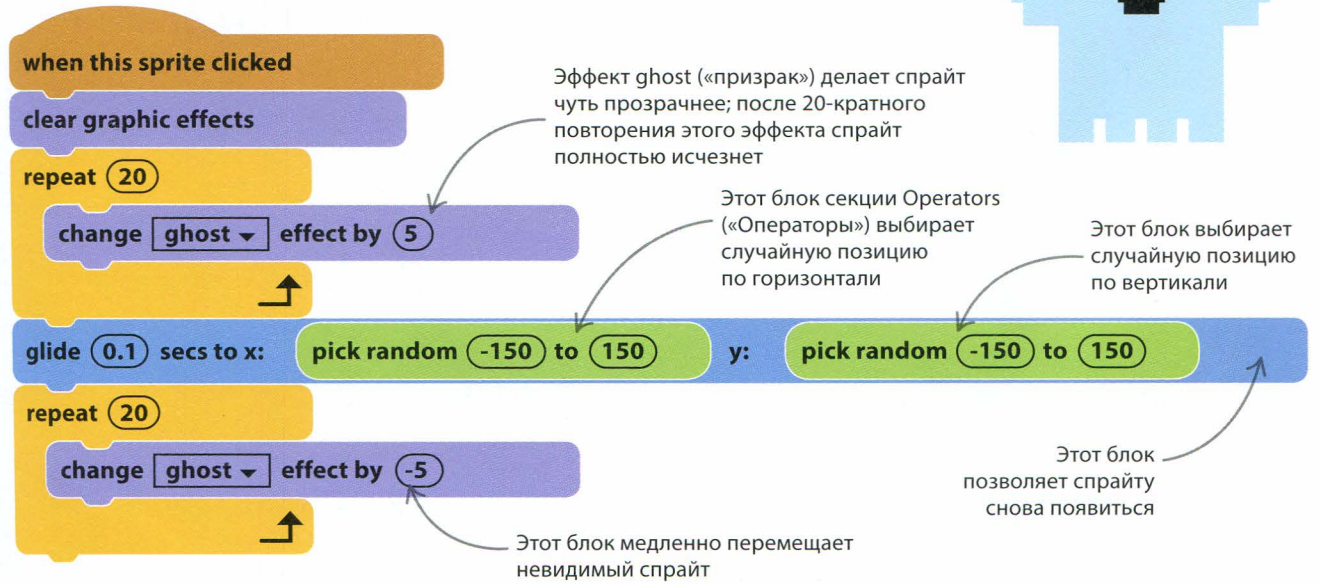
△ **Изменение размера спрайта**
Эти два блока могут увеличивать или уменьшать спрайт — на определенную величину либо в процентах от его размера.



△ **Добавление спецэффектов**
Графические эффекты служат для изменения вида спрайта или искажения его формы. С ними интересно экспериментировать.

Телепортация с помощью эффектов

Загрузи из библиотечной категории Fantasy («Фантастика») спрайт привидения и добавь ему этот скрипт. Привидение телепортируется, если кликнуть по нему мышкой.



События

СМОТРИ ТАКЖЕ

Считывание и распознавание **66–67 ❯**

Обмен сообщениями **70–71 ❯**

Коричневые блоки Events («События») запускают скрипты при определенных событиях — например, если пользователь нажмет клавишу, кликнет по спрайту, использует веб-камеру или микрофон.

Клик мышкой

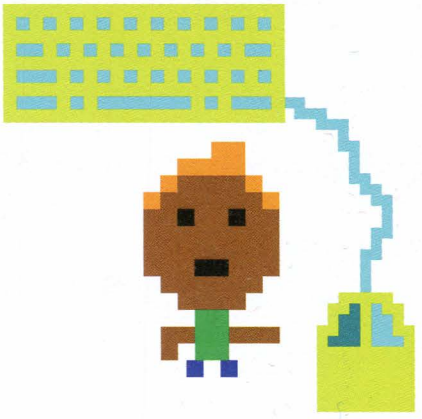
К спрайту можно добавить скрипт, делающий что-то, если пользователь кликнет по этому спрайту мышкой. Экспериментируя с разными блоками, узнай, как спрайт может реагировать на клик.

△ **Клик по спрайту**
Этот скрипт научит спрайт кота мяукать при клике мышкой.

СЛЕНГ

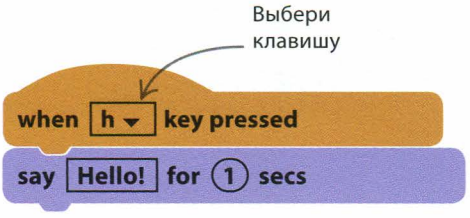
Что такое событие?

Событие — это некое действие, например нажатие на клавишу или клик по зеленому флажку. Блоки, реагирующие на события, должны быть вверху скрипта. Скрипт будет ожидать, пока событие не произойдет, и лишь тогда заработает.

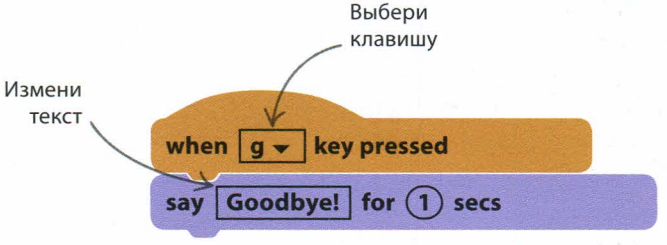


Нажатия клавиш

Программы могут реагировать на нажатия клавиш на клавиатуре. Есть и другой способ работы с клавиатурой, более подходящий для игр (см. с. 66–67).



△ **Скажи «Привет»!**
Добавь этот скрипт, и спрайт будет говорить Hello! при нажатии клавиши H.



△ **Скажи «До свидания»!**
Спрайт с этим скриптом говорит Goodbye! при нажатии G.

Звуковые события

Если на твоем компьютере есть микрофон, спрайты могут определять громкость звуков в комнате в диапазоне от 0 (очень тихо) до 100 (очень громко). Используй блок when loudness > 10 («когда громкость > 10»), чтобы скрипт запускался от звука.

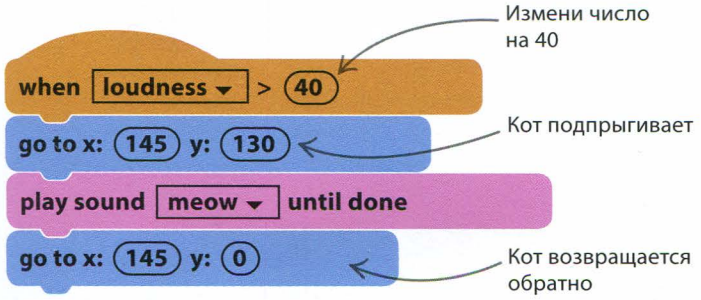
СОВЕТЫ ЭКСПЕРТА

Разрешение

Scratch спрашивает разрешения на использование веб-камеры и микрофона. В окне с вопросом выбери Allow («Разрешить»).

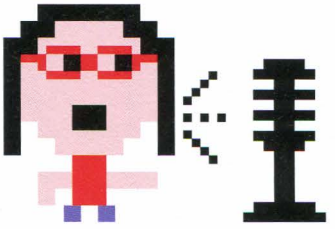
1 **Учим кота реагировать на звук**

Начни новый проект и добавь в него фон room3 из категории Indoors («В помещении») библиотеки фонов. Перетащи кота на стул и добавь ему показанный ниже скрипт.



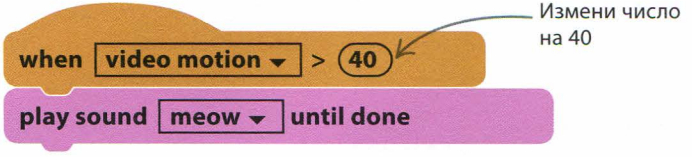
2 **Кричим на кота**

Крикни в микрофон — кот в ужасе подпрыгнет и замяукает. Так же он будет реагировать на музыку и другие громкие звуки.



Распознавание движения

Если у тебя есть веб-камера, ее тоже можно использовать в Scratch. Добавь коту этот скрипт, и он замяукает, если помахать ему рукой через веб-камеру.



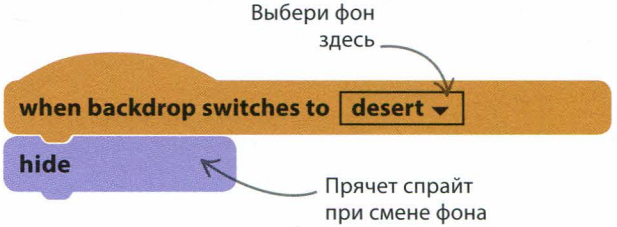
△ **Распознавание движения**

Используй блок when loudness > 10 («когда громкость > 10»). В выпадающем меню выбери video motion («движение видео») вместо loudness («громкость»). Если подвигаться перед камерой, скрипт запустится.

СОВЕТЫ ЭКСПЕРТА

Изменение фона

Спрайт может реагировать на изменение фона. Например, если установлен определенный фон, спрайт может исчезать. В списке сцен в левой нижней части экрана добавь новый фон, затем добавь блок when backdrop switches to backdrop1 («когда фон меняется на фон1»).



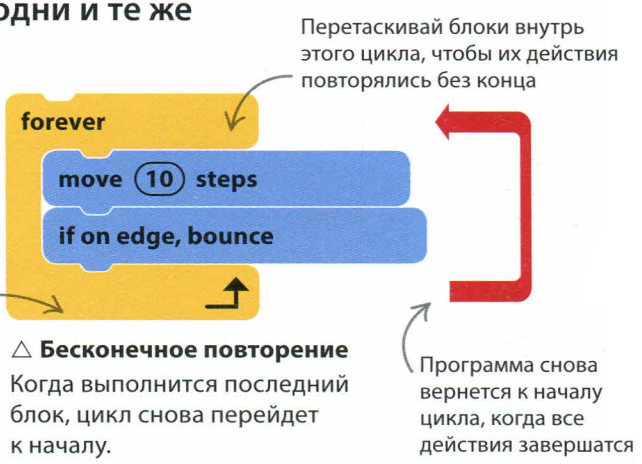
Простые циклы

СМОТРИ ТАКЖЕ
Сложные циклы 68–69 ›
Циклы в Python 122–123 ›

Цикл — часть программы, повторяющаяся снова и снова. Блоки циклов из секции Control («Управление») обозначают, какие блоки нужно повторять и сколько раз. Это избавляет от необходимости добавлять одни и те же блоки многократно.

Бесконечный цикл

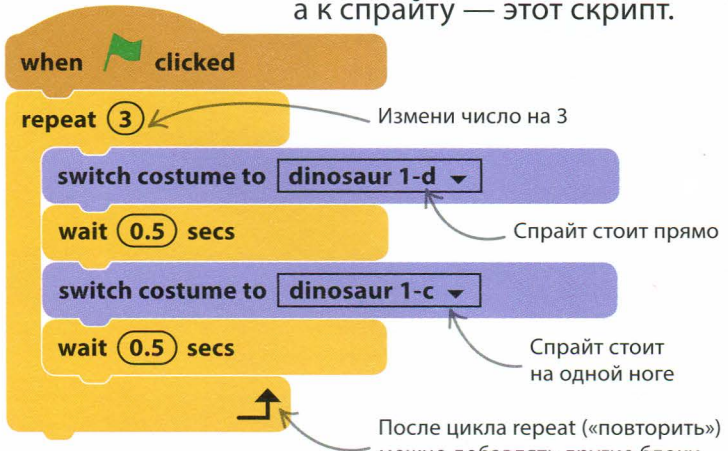
Все, что ты поместишь внутрь блока forever («всегда»), будет повторяться бесконечно. После него нельзя добавлять другие блоки, потому что цикл forever («всегда») никогда не закончится.



△ **Бесконечное повторение**
Когда выполнится последний блок, цикл снова перейдет к началу.

Цикл «Повторить»

Чтобы повторить действие заданное количество раз, используй блок repeat 10 («повторить 10»). Измени число в блоке на соответствующее нужному количеству повторов. Добавь в новый проект спрайт «Dinosaur1», а к спрайту — этот скрипт.

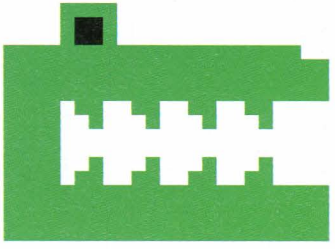


△ **Танцующий динозавр**
Если кликнуть по зеленому флажку, динозавр начнет танцевать. Он повторит движения танца три раза.

ЗАПОМНИ

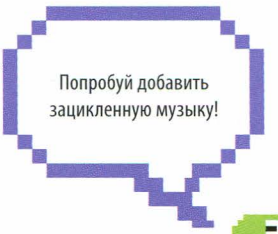
Форма блоков циклов

Блоки циклов имеют форму, напоминающую пасть животного. Перетащи блоки, которые нужно повторять, в пасть, и цикл сомкнется вокруг них. Если добавить еще блоки, пасть растянется, чтобы их вместить.



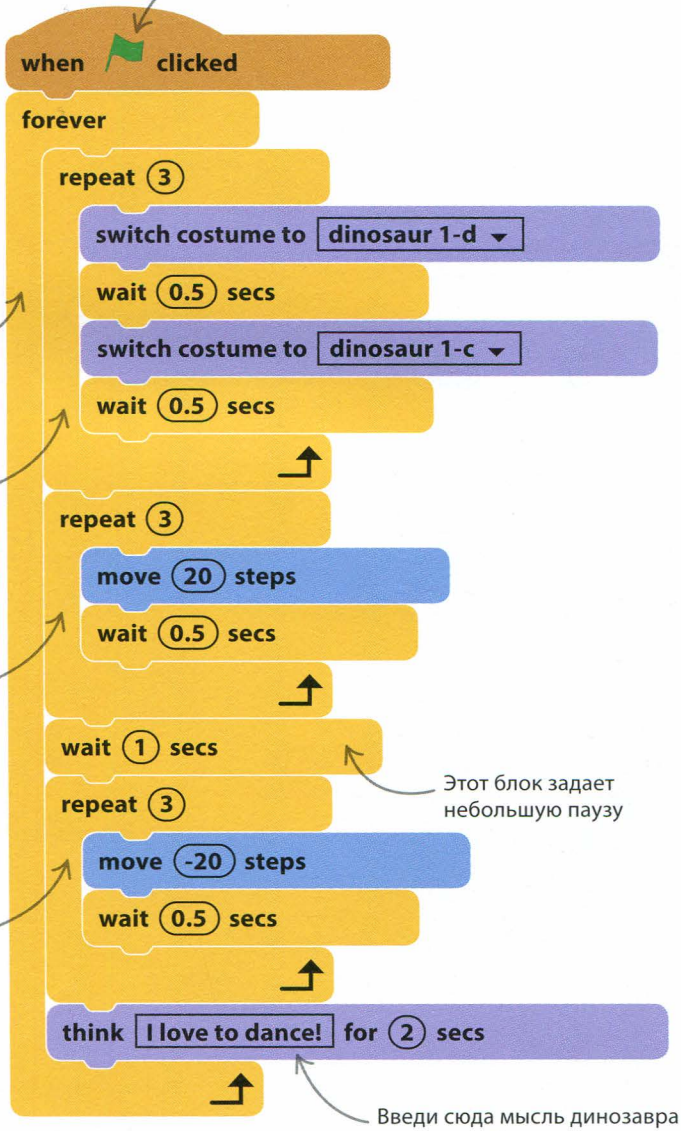
Вложенные циклы

Циклы могут быть вложенными — это значит, что циклы можно вкладывать в другие циклы. В этом скрипте динозавр, закончив танцевать, пойдет направо, потом налево и задумается. Переведя дыхание, он снова начнет танцевать и успокоится, только если нажать кнопку остановки программы.



▷ **Циклы в циклах**
Внутри этого цикла forever («всегда») — несколько других циклов. Проверь, в нужных ли циклах находятся блоки, иначе программа не будет работать правильно.

Кликни по зеленому флажку, чтобы запустить скрипт



Блок forever («всегда») охватывает остальные блоки

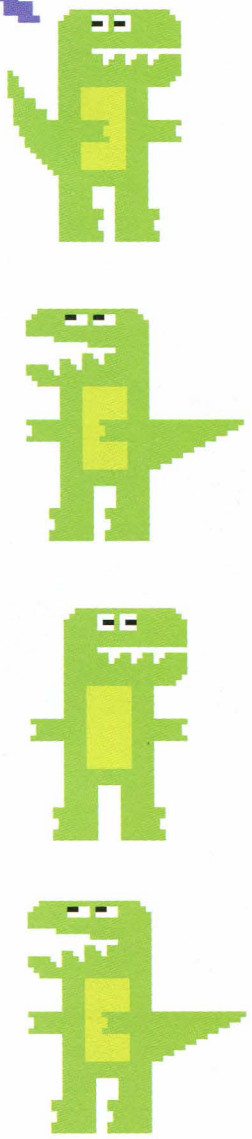
Движения танца

Динозавр делает три шага вправо

Этот блок задает небольшую паузу

Динозавр делает три шага влево

Введи сюда мысль динозавра — она появится в баллоне у него над головой



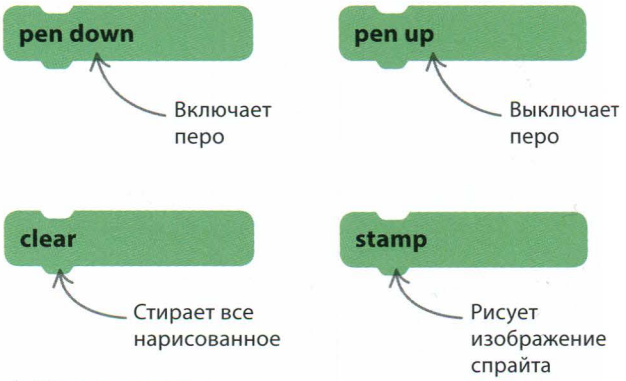
Перья и черепашки

СМОТРИ ТАКЖЕ
‹ 44–45 События
‹ 46–47 Простые циклы

У каждого спрайта есть инструмент «перо», который может рисовать линию вслед за движущимся спрайтом. Чтобы что-нибудь нарисовать, включи перо и двигай спрайт по сцене, словно карандаш по бумаге.

Блоки «Перо»

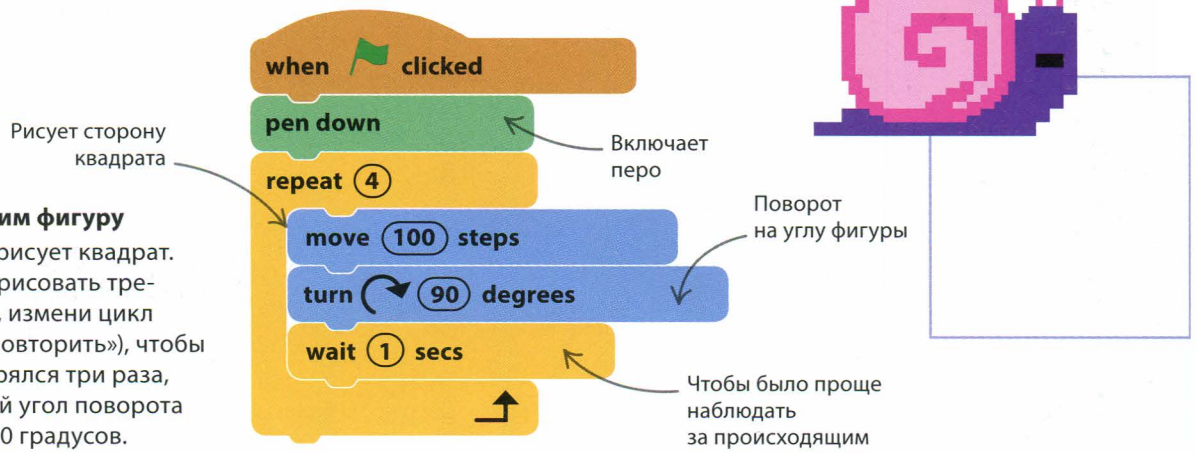
Темно-зеленые блоки служат для управления пером. У каждого спрайта есть свое перо, которое включается блоком pen down («опустить перо») и выключается блоком pen up («поднять перо»). Также можно менять размер пера и цвет линии.



△ **Игры с пером**
Экспериментируй, рисуя картинки с помощью блоков управления пером.

Рисуем квадрат

Чтобы нарисовать квадрат, включи у спрайта перо и двигай спрайт по квадрату. Используй цикл для рисования четырех сторон и поворотов на углах.



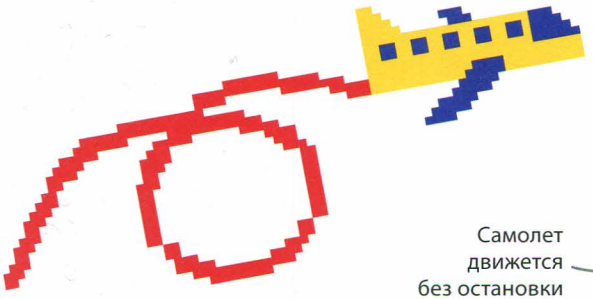
▷ **Изменим фигуру**
Этот код рисует квадрат. Чтобы нарисовать треугольник, измени цикл repeat («повторить»), чтобы он повторялся три раза, и поменяй угол поворота с 90 на 120 градусов.

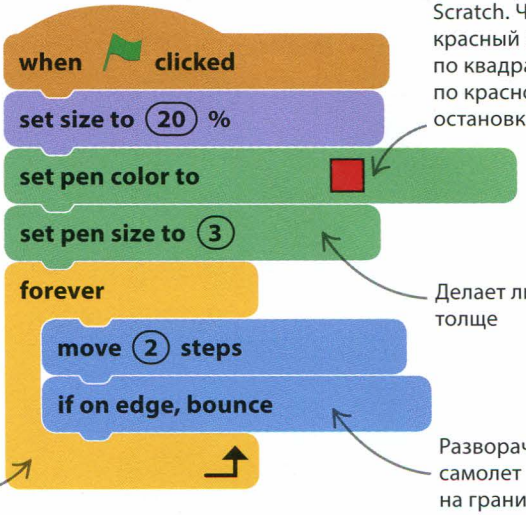
След в небе

Управляй самолетом, который оставляет за собой дымный след, так, чтобы рисовать им в воздухе. Создав новый проект, добавь спрайт самолета и этот скрипт.

▷ В полет

Используй клавиши «Стрелка влево» и «Стрелка вправо» для управления самолетом. Включай след клавишей A и выключай его клавишей Z. Чтобы стереть рисунок, нажми пробел.





Выбирать можно только цвета, которые есть в интерфейсе Scratch. Чтобы выбрать красный цвет, кликни по квадратику, а затем по красной кнопке остановки программы

Делает линию толще

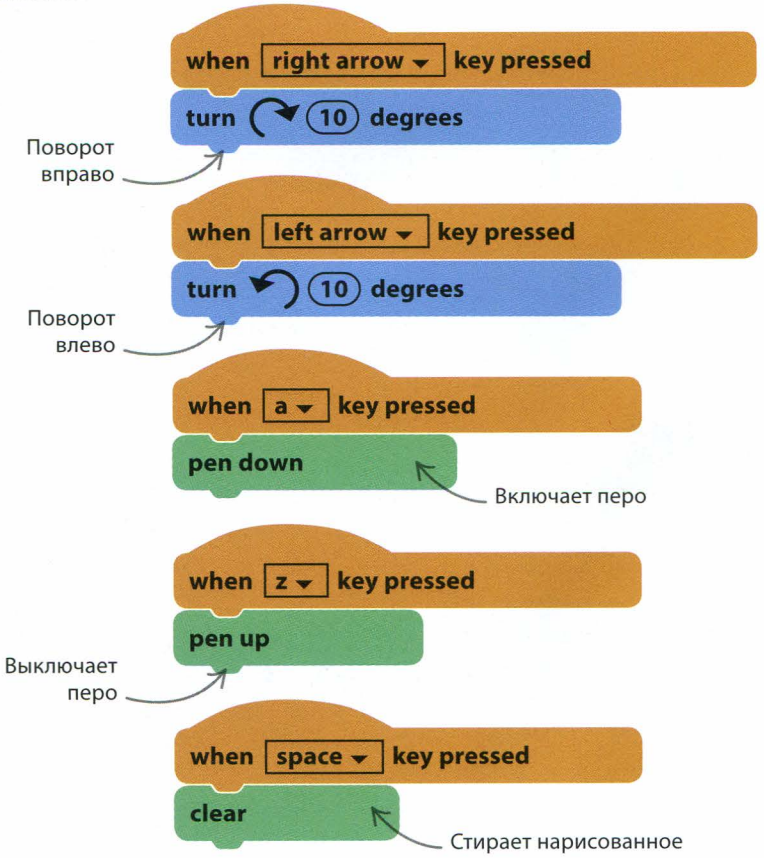
Самолет движется без остановки

Разворачивает самолет на границах сцены

СЛЕНГ

Черепашья графика

Рисование с помощью спрайтов называют черепашьей графикой. Черепашкой звали робота, который, двигаясь по полу, оставлял за собой след. Первый язык программирования, использовавший черепашью графику, назывался LOGO.



Поворот вправо

Поворот влево

Включает перо

Выключает перо

Стирает нарисованное

Переменные

Переменной называют именованное место для хранения информации. Например, в переменной можно запомнить счет игры, имя игрока или скорость персонажа.

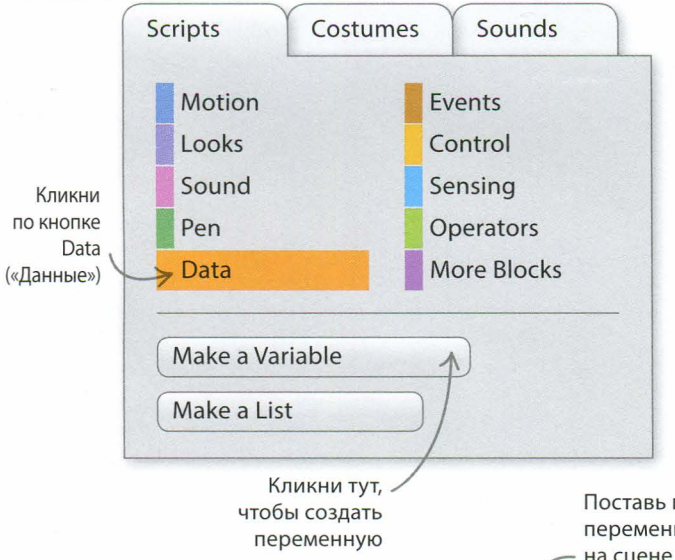
СМОТРИ ТАКЖЕ
Вычисления **52–53** ❯
Переменные в Python **108–109** ❯

Создание переменной

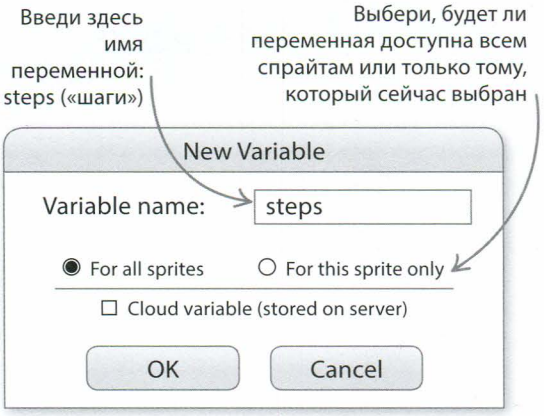
Переменную можно создать в секции Data («Данные») палитры блоков. Как только переменная создана, в палитре появляются новые, готовые к использованию блоки.

◁ **Хранение данных**
Переменные похожи на ящички для хранения частей информации, которую можно использовать в программе.

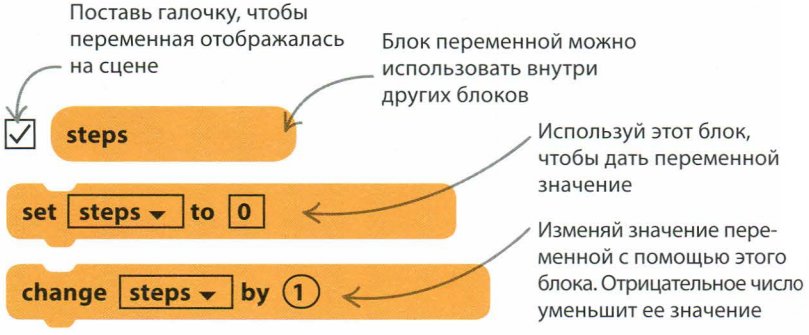
1 Создаем переменную
Сперва кликни по кнопке Data («Данные») в палитре блоков, затем по кнопке Make a Variable («Создать переменную»).



2 Дай новой переменной имя
Дай переменной имя, которое будет напоминать о ее назначении. Выбери, каким спрайтам нужно использовать переменную, и кликни OK.



3 Новая переменная создана
Как только переменная создана, в палитре появятся новые блоки. Меню этих блоков позволяет выбрать, к какой переменной они относятся (если переменных больше, чем одна).



Использование переменной

Переменные можно использовать для изменения скорости спрайтов. Этот простой скрипт покажет, как это сделать.

1 Дадим переменной значение

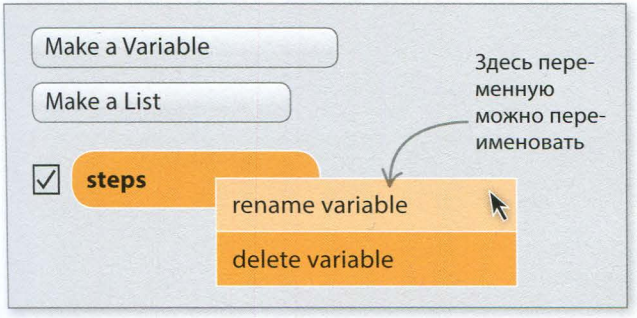
Создай этот скрипт. Используй блок set steps to 0 («задать шаги значение 0»), заменив число на 5. Добавь в скрипт блок move 10 steps («идти 10 шагов»), затем перетащи блок переменной steps («шаги») поверх числа 10.

2 Изменим значение переменной

Чтобы увеличить значение переменной steps («шаги») на 1, используй блок change steps by 1 («изменить шаги на 1»). Перетащи его внутрь блока forever («всегда»), чтобы кот двигался все быстрее и быстрее.

Удаление переменных

Когда переменная больше не нужна, сделай по ней правый клик в палитре блоков и выбери delete variable («удалить переменную»). Хранившаяся в переменной информация будет потеряна.



СОВЕТЫ ЭКСПЕРТА

Защищенные переменные

Значения некоторых переменных задает сам Scratch, и изменять их нельзя. Однако это все-таки переменные, так как их значения меняются. Такие блоки называют считывающими блоками.

distance to — Отслеживает дистанцию до чего-либо, например до указателя мышки.

costume # — Сообщает, сколько у спрайта костюмов.

direction — Показывает направление, в котором движется спрайт.

Вычисления

СМОТРИ ТАКЖЕ
‹ 50–51 Переменные
Вычисления в Python 112–113 ›

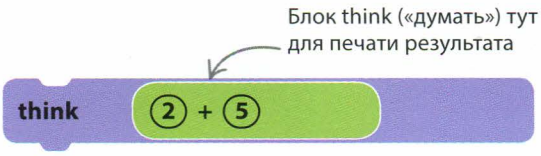
Scratch может не только хранить числа в переменных (см. с. 50–51), но и выполнять расчеты с помощью блоков секции Operators («Операторы»).

Арифметика

Для простейших вычислений есть четыре блока секции Operators («Операторы»). Это сложение, вычитание, умножение и деление.

△ **Сложение**
Блок «+» складывает вместе два числа.

△ **Вычитание**
Блок «–» вычитает второе число из первого.



△ **Печать результата**
Перетащи блок think («думать») в область скриптов, затем перетащи внутрь него блок «+». Теперь введи оба числа и кликни по блоку, чтобы увидеть ответ.

△ **Умножение**
В программировании умножение обозначается знаком «*».

△ **Деление**
Для обозначения деления в Scratch используется символ «/».

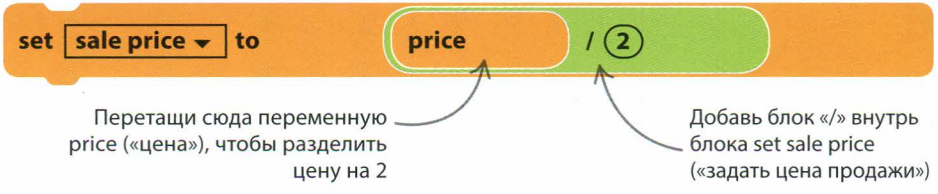
Результат в переменной

Для более сложных вычислений, например определения цены продажи продукта, в арифметических блоках вместо чисел можно использовать переменные. Результат также можно сохранить в переменной.

1 Создаем переменные
Создай в секции Data («Данные») две переменные — price («цена») и sale price («цена продажи»).

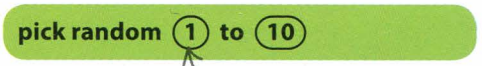
2 Зададим цену
Выбери блок set price («задать цена») и выставь значение цены в 50.

3 Вычисляем цену продажи
Этот скрипт вычисляет половину цены продукта и выставляет ее как цену продажи.



Случайные числа

Блок pick random («выдать случайное») служит для получения случайного числа в диапазоне между двумя значениями. Он пригодится, когда нужно бросить в игре жребий или перемешать костюмы спрайта.



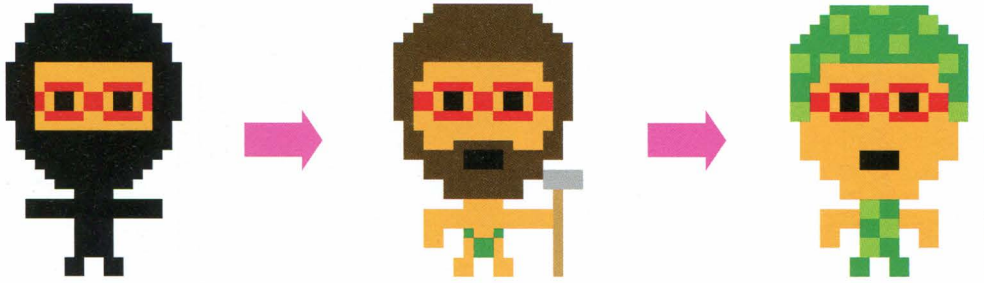
◁ **Получим случайное значение**
Чтобы получить случайный номер месяца, измени числа: выбор должен быть от 1 до 12.

СОВЕТЫ ЭКСПЕРТА

Игры

Случайные числа часто используют, чтобы добавить в игру элемент неожиданности. Например, инопланетянин может возникать в случайном месте или в случайное время. Можно взять случайное число для жеребьевки или выбрать спрайту случайный костюм.

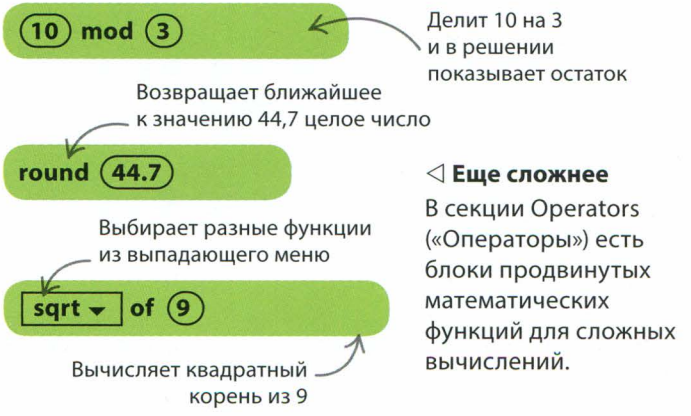
◁ **Смена костюмов**
Этот скрипт меняет костюм спрайта на случайный каждые две секунды.



◁ **Случайный костюм**
Благодаря костюмам спрайт может совершать движения или появляться в разной одежде, как показано здесь.

Сложные вычисления

Простых блоков из секции Operators («Операторы») достаточно для большинства вычислений, но Scratch способен и на большее. Блок Mod возвращает остаток от деления двух чисел. Блок round («округлить») округляет число до ближайшего целого значения, а блок sqrt («квадратный корень») вычисляет квадратный корень.



◁ **Еще сложнее**
В секции Operators («Операторы») есть блоки продвинутых математических функций для сложных вычислений.

Строки и списки

СМОТРИ ТАКЖЕ
❮ 50–51 Переменные
Строки в Python 114–115 ❯

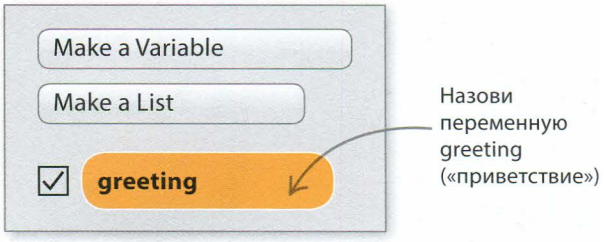
Строкой в программировании называют последовательность символов. Строки могут быть любой длины и содержать любые клавиатурные символы (включая пробелы). Строки также можно объединять в списки.

Работа со словами

Нередко в программе нужно запоминать слова, например имя игрока. Для этого пригодятся переменные. Еще такая программа может задать пользователю вопрос, ответ на который нужно вводить в текстовом окне. Этот скрипт спрашивает имя пользователя, чтобы спрайт с ним поздоровался.

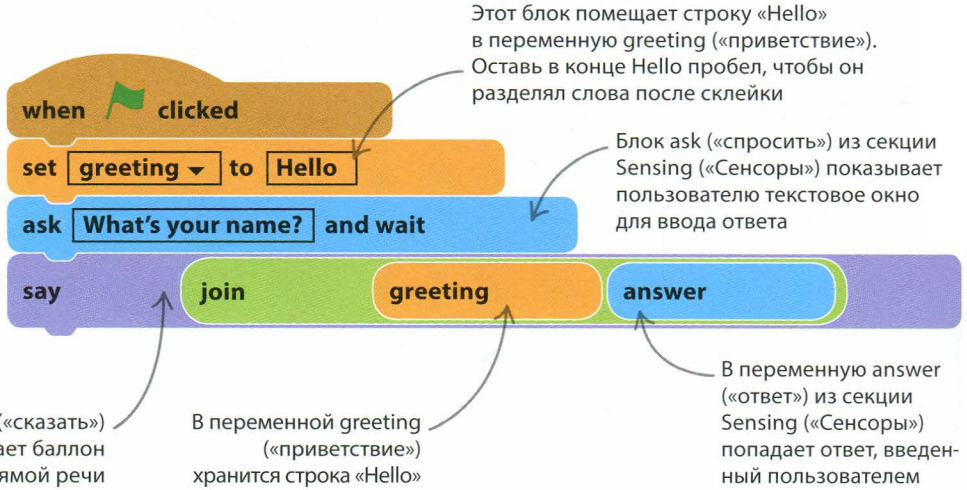
1 Создаем новую переменную

Кликни по кнопке Data («Данные») в палитре блоков, а затем по кнопке Make a Variable («Создать переменную»). Создай переменную с именем greeting («приветствие»).



2 Задаем вопрос

Этот скрипт задает пользователю вопрос. То, что пользователь введет в текстовом окне, попадает в переменную answer («ответ»). Затем скрипт склеивает строки, хранящиеся в переменных greeting («приветствие») и answer («ответ»), чтобы поприветствовать пользователя.

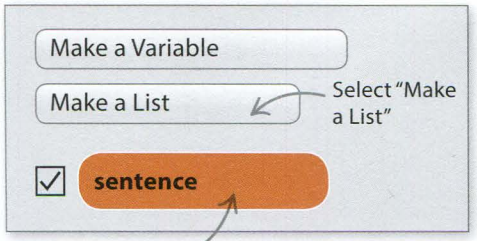


Создание списков

Переменные хороши, если нужно запомнить лишь одно значение. Для запоминания множества похожих значений можно использовать списки. Список способен хранить много элементов (чисел и строк) одновременно, например набор лучших результатов в игре. Эта программа демонстрирует один из способов использования списка.

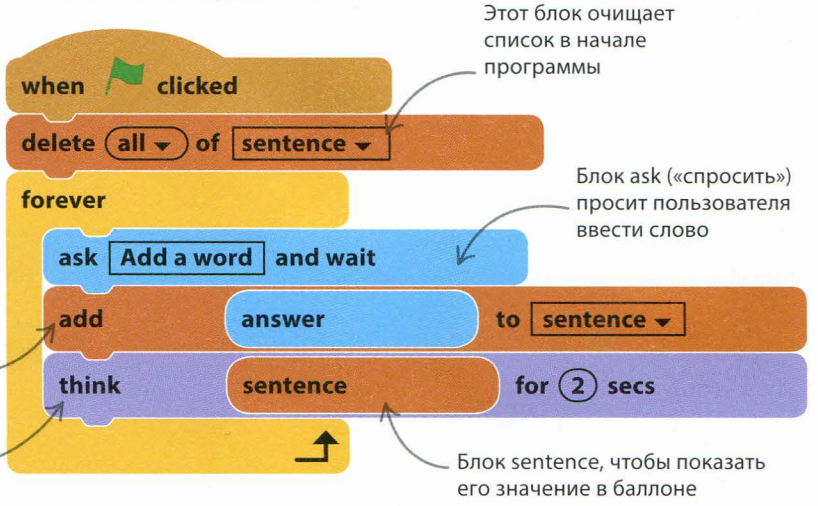
1 Создадим список

Начни новый проект. Зайди в секцию Data («Данные») палитры блоков и кликни по кнопке Make a List («Создать список»). Дай списку имя sentence («предложение»).



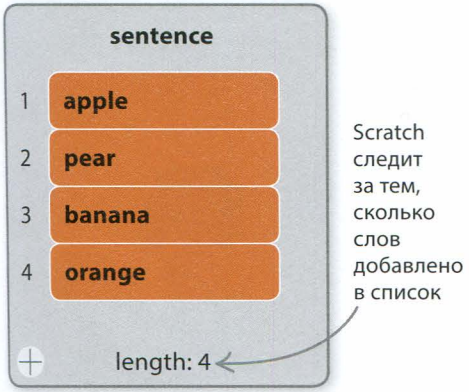
2 Используем список

Этот скрипт запрашивает у пользователя слова. Каждое слово попадает в список, который показывается в баллоне над спрайтом.



3 Содержимое списка

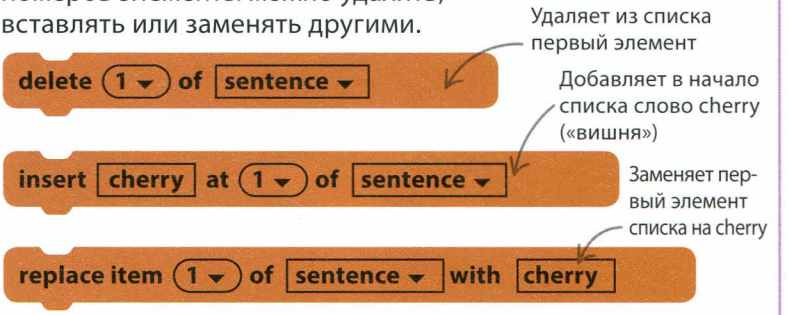
Если поставить галочку рядом со списком в палитре блоков, содержимое списка будет отображаться на сцене.



СОВЕТЫ ЭКСПЕРТА

Игры со списками

С помощью этих блоков можно менять содержимое списков. У каждого элемента в списке есть номер — для первого элемента это 1, и так далее. С помощью номеров элементы можно удалять, вставлять или заменять другими.



Координаты

Чтобы поместить спрайт в определенное место или узнать его положение, нужны координаты — пара чисел, указывающая положение спрайта на сетке X–Y.

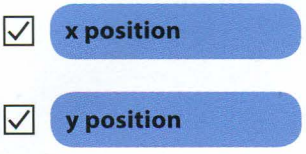
СМОТРИ ТАКЖЕ
‹ 38–39 Перемещение объектов
‹ 52–53 Вычисления

Позиции X и Y

X- и Y-позиции спрайта и указателя мышки отображаются в интерфейсе Scratch. Знание координат спрайта может пригодиться при создании скриптов.

◁ **Положение спрайта**
Текущие координаты спрайта можно увидеть в правом верхнем углу области скриптов.

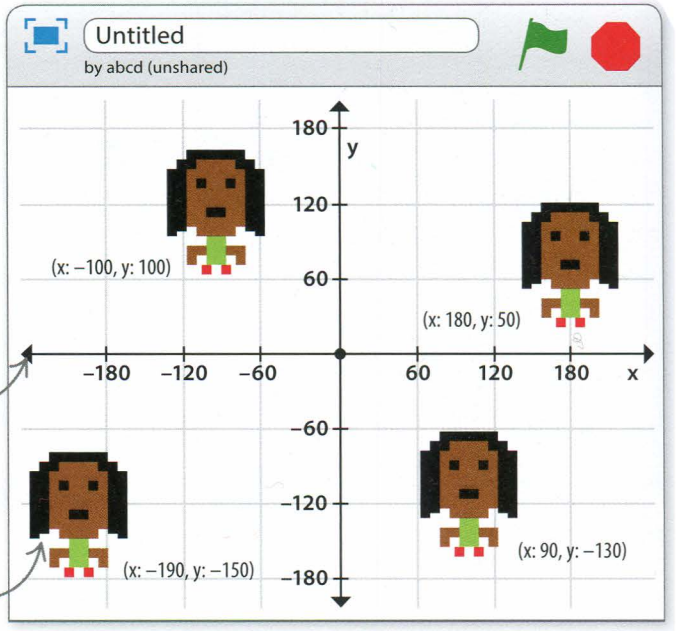
△ **Положение указателя мышки**
Координаты указателя мышки отображаются справа под сценой. Подвигай мышкой над сценой и посмотри, как меняются координаты.



◁ **Отображение координат на сцене**
Поставь отметки рядом с блоками X position («положение по оси *x*») и Y position («положение по оси *y*»), чтобы координаты показывались на сцене.

Сетка X–Y

Чтобы узнать положение точки, посчитай шаги влево или вправо, а также вверх или вниз от середины сцены. Количество шагов влево или вправо называется X, а вверх или вниз — Y. Для отсчета влево и вниз используются отрицательные числа.

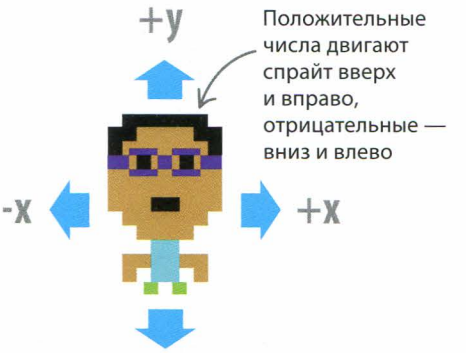


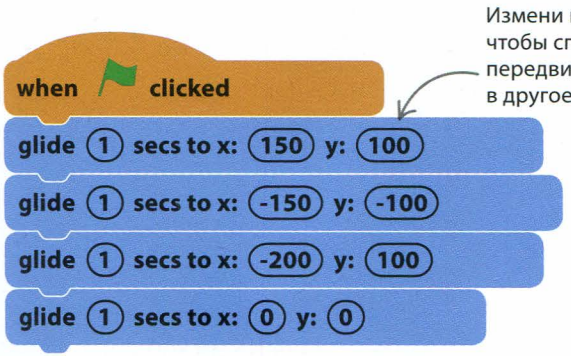
На сцену наложена сетка X–Y

Этот спрайт в 190 шагах слева (–190) и 150 шагах вниз (–150) от середины сцены

Перемещение спрайта

Координаты используются для перемещения спрайта в нужную точку сцены, и не важно, близко или далеко эта точка. Блок glide 1 secs to x: 0 y: 0 («плыть 1 секунд в точку x: 0 y: 0») из секции Motion («Движение») плавно передвигает спрайт в середину сцены.



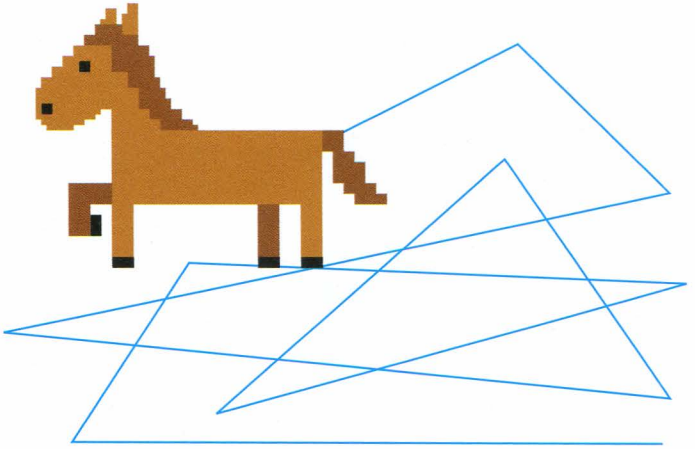


△ **Управляем спрайтом из скрипта**
Догадаешься ли ты, как будет двигаться спрайт, если запустить этот скрипт? Попробуй — и узнаешь!

△ **Меняем X и Y по отдельности**
Эти блоки могут изменять X, не меняя Y, и наоборот.

Путь безумной лошади

Поупражняйся с координатами на примере этого забавного скрипта. Выбери спрайт Horse1 и добавь ему скрипт, показанный ниже. Программа использует блок go to x: 0 y: 0 («перейти в x: 0 y: 0») для перемещения лошади в случайные позиции и рисует следом за ней линии.



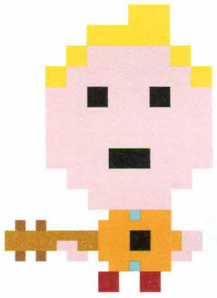
Пошумим!

СМОТРИ ТАКЖЕ
Считывание и распознавание 66–67 ❯
Бешеные обезьяны 74–81 ❯

Scratch-программам не обязательно быть беззвучными. Используй розовые блоки Sound («Звук») для звуковых эффектов и создания музыки. Можно добавлять звуки, которые уже есть на компьютере, или записывать новые.

Добавление звуков к спрайтам

Чтобы проиграть звук, его надо добавить к спрайту. У каждого спрайта есть свой набор звуков. Чтобы редактировать его, кликни по вкладке Sounds («Звуки») над палитрой блоков.

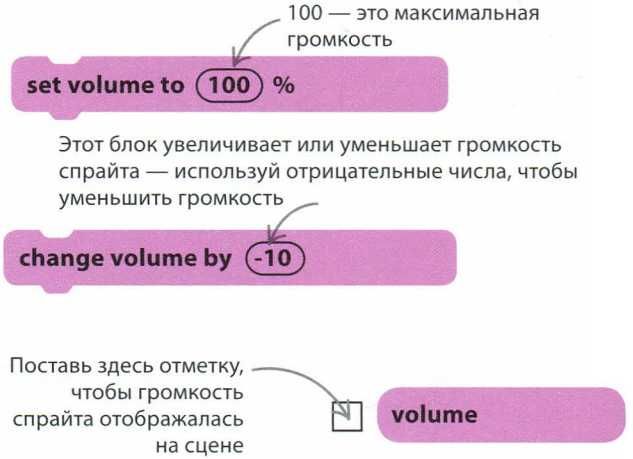


Проигрывание звука

Есть два блока, проигрывающих звук: play sound («играть звук ») и play sound until done («играть звук до завершения»). Блок play sound until done («играть звук до завершения») приостанавливает скрипт до тех пор, пока звук не затихнет.

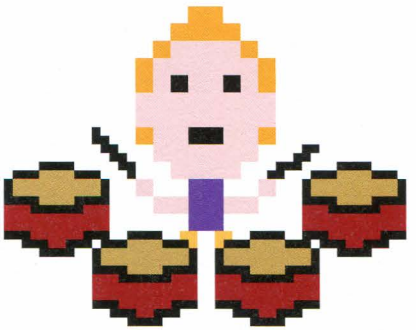
Прибавим громкости

У каждого спрайта есть свой уровень громкости, который задается числом от 0 (тишина) до 100 (предельная громкость).



Создание собственной музыки

В Scratch есть блоки, подходящие для сочинения музыки. В твоем распоряжении целый оркестр инструментов и ударная установка. Длительность каждой ноты измеряется в долях такта, или «ударах».



Определяет, насколько высоко или низко звучит нота

play note (60) for (0.5) beats

Большие числа делают ноту протяжнее. Длительность может быть меньше одной доли такта, как здесь

set instrument to (1)

Кликни для выбора инструмента из меню

play drum (1) for (0.25) beats

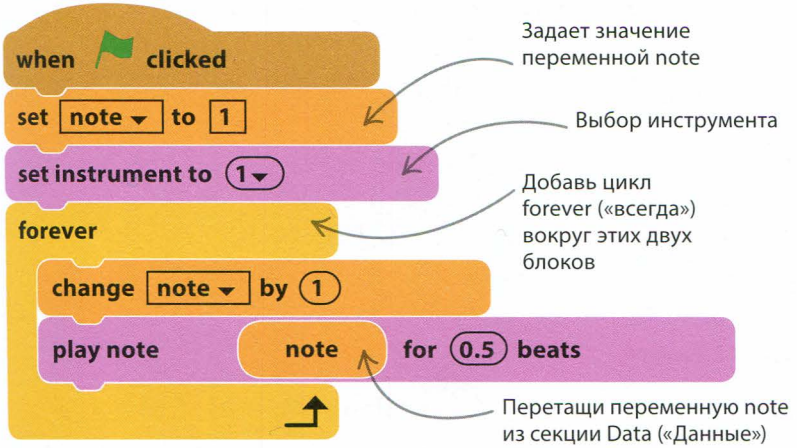
Используй меню для выбора звука барабана

rest for (0.25) beats

Этот блок выдерживает в музыке паузу. Чем больше число, тем пауза дольше

Проигрывание музыки

Если соединить ноты вместе, получится мелодия. Создай переменную с именем note (см. с. 50–51) и добавь этот скрипт к любому спрайту, чтобы заиграла музыка.



△ **Восходящая гамма**

Этот скрипт играет последовательность нот, каждая из которых звучит в течение половины доли такта. Высота ноты с каждым шагом увеличивается на 1.

СОВЕТЫ ЭКСПЕРТА

Темп

Скорость проигрывания музыки называется темпом. Темп определяет длительность доли такта. Для работы с темпом есть три блока.

set tempo to (60) bpm

Темп измеряется в «ударах» в минуту, обозначается bpm.

change tempo by (60)

Увеличь темп, чтобы ускорить музыку, или используй отрицательное число, чтобы ее замедлить.

☐ tempo

Поставь галочку в квадрате, чтобы темп спрайта отображался на сцене.

ПРОЕКТ 2

Катись, кубик

Даже простая программа может быть интересной и полезной. В этом проекте мы создадим игральный кубик. Его можно бросать, соревнуясь, у кого выпадет большее число, или использовать вместо настоящего кубика в настольной игре.

СМОТРИ ТАКЖЕ

- ❮ 40–41 Костюмы
- ❮ 46–47 Простые циклы
- ❮ 50–51 Переменные
- ❮ 52–53 Вычисления

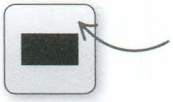
Как сделать игральный кубик

Для кубика понадобится шесть костюмов. Каждый костюм изображает одну из граней кубика с соответствующим количеством точек — от одной до шести.

1 Кликни по кнопке с кисточкой внизу сцены, чтобы нарисовать новый спрайт.

Нарисовать новый спрайт

2 Кликни по кнопке с прямоугольником слева от области рисования. Чтобы кубик был красочным, выбери цвет из палитры (см. ниже). Затем в области рисования нажми левую кнопку мыши, удерживая при этом клавишу Shift, и двигай указатель так, чтобы нарисовать в центре квадрат.

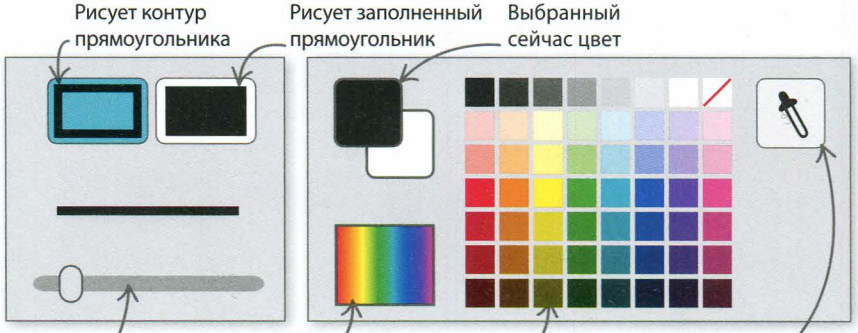


Если зажата клавиша Shift, будет нарисован прямоугольник или квадрат

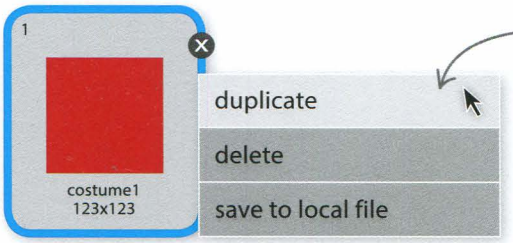
СОВЕТЫ ЭКСПЕРТА

Инструменты цвета

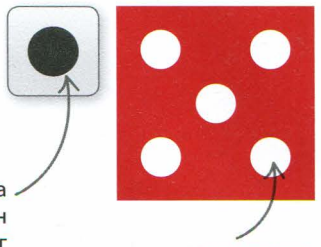
Под областью рисования есть панель инструментов. Кликни по сплошному прямоугольнику, чтобы нарисовать прямоугольник, заполненный цветом. Кликни по пустому прямоугольнику, чтобы нарисовать контур прямоугольника или квадрата. Меняй толщину линий ползунком. Цвет можно выбрать из палитры, кликнув по нему.



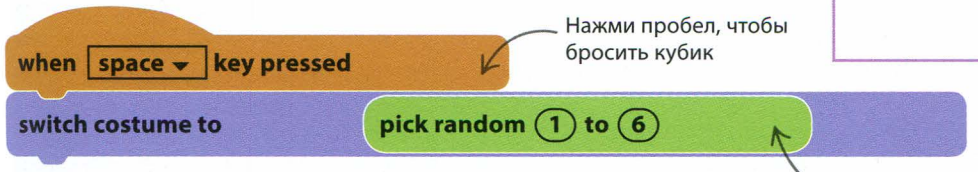
3 Сделай правый клик по костюму слева от области рисования и выбери duplicate («дублировать»). Повторяй этот шаг, пока костюмов не станет шесть.



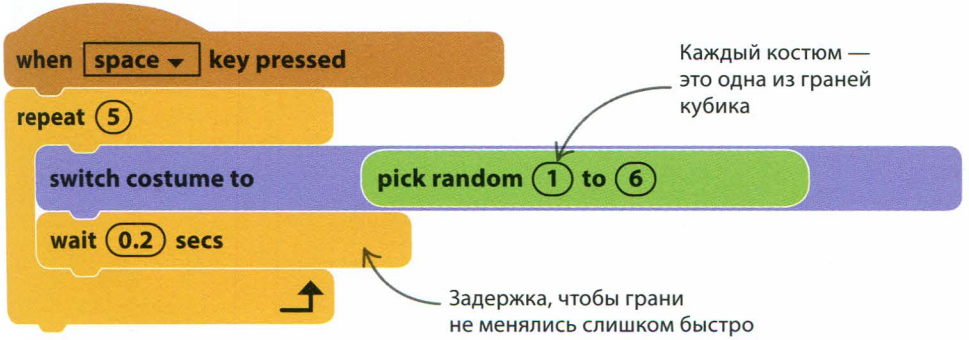
4 Выбери костюм. Кликни по кнопке с кругом в области рисования и выбери белый цвет из палитры. Нарисуй кружочки на каждом из костюмов, чтобы получилось шесть граней для кубика.



5 Добавь спрайту кубика этот скрипт. Нажми пробел, чтобы бросить кубик. Сделай это несколько раз, чтобы увидеть все костюмы.



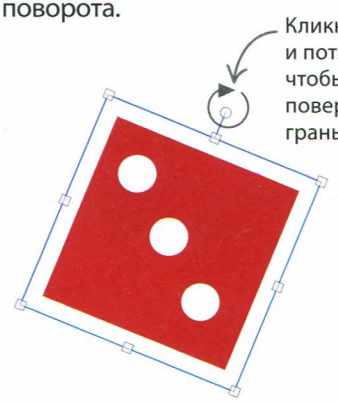
6 Если одна и та же грань выпадет два раза подряд, картинка не изменится, и это будет выглядеть так, будто ничего не произошло. Этот скрипт поменяет костюмы пять раз, прежде чем кубик остановится, — чтобы создалось впечатление, будто он катится.

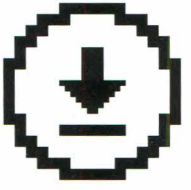


СОВЕТЫ ЭКСПЕРТА

Инструмент поворота

Чтобы было похоже, будто кубик катится, можно повернуть костюмы граней на различные углы. Кликни по кнопке Convert to vector («Конвертировать в векторную графику») в нижнем правом углу. Если теперь кликнуть по области рисования, появится инструмент поворота.





Не забудь сохранить свою работу.

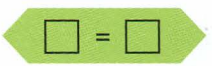
Истина или ложь?

СМОТРИ ТАКЖЕ
Решения и ветвление 64–65 ❯
Принятие решений 118–119 ❯

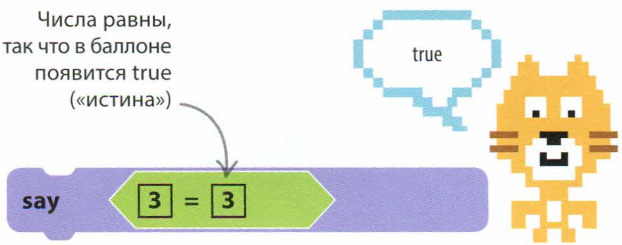
Компьютеры принимают решения о том, что делать дальше, задавая вопросы и анализируя ответы «да» или «нет». Истина или ложь? Вопросы, на которые есть лишь два варианта ответа, называют бу́левыми (логическими) выражениями.

Сравнение чисел

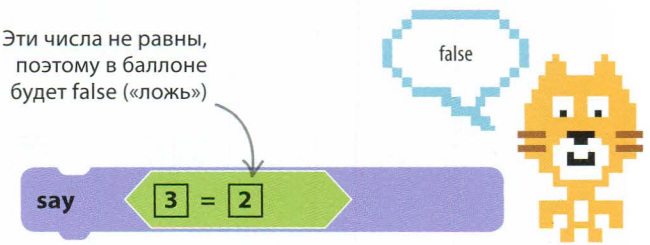
Числа можно сравнивать с помощью блока «=» из секции Operators («Операторы») палитры блоков.



◁ **Блок «=»**
Этот блок дает один из двух вариантов ответа: «истина», если числа равны, и «ложь», если нет.



△ **Ответ «истина»**
Используй блок «=» внутри блока say («сказать»), чтобы увидеть ответ true («истина») или false («ложь») в баллоне над спрайтом.

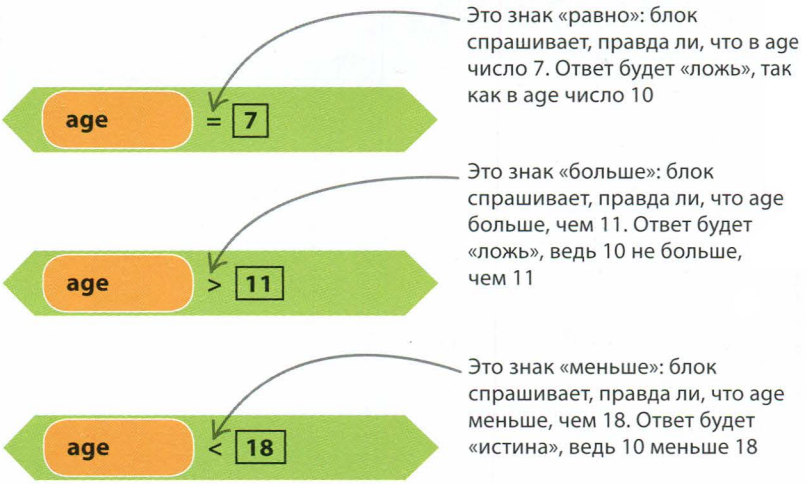


△ **Ответ «ложь»**
Если числа отличаются, в баллоне над спрайтом появится слово false («ложь»).

Сравнение переменных

Внутри блоков сравнения можно использовать переменные: зачем сравнивать фиксированные числа, если результат всегда один и тот же?

△ **Создаем переменную**
Кликни по кнопке Data («Данные») в палитре блоков и создай переменную с именем age («возраст»). Задай ей значение 10 (кликни по блоку, чтобы убедиться, что значение поменялось). Перетаскивай эту переменную в блоки сравнения.



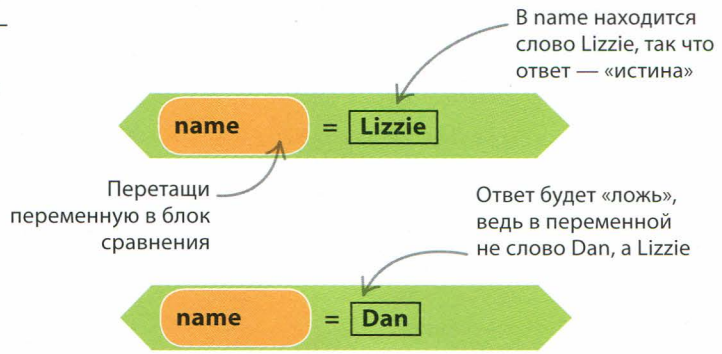
△ **Сравниваем числа**
Зеленые блоки сравнения находятся в секции Operators («Операторы»). Можно выяснить, являются ли числа равными или одно больше другого.

СОВЕТЫ ЭКСПЕРТА

Сравнение слов

Блок «=» подходит не только для чисел — можно узнать, совпадают ли две строки. Разницу между заглавными и строчными буквами этот блок не учитывает.

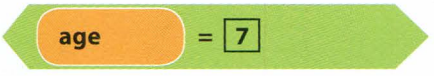
△ **Создаем переменную**
Чтобы поупражняться в сравнении строк, создай новую переменную с именем name («имя») и задай ей значение Lizzie.



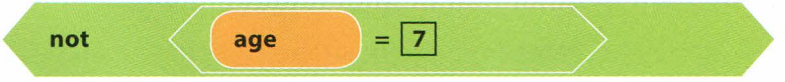
«Не»

Блок not («не») пригодится, когда надо поменять значение булева выражения на противоположное. Например, проще проверить age на не равенство 10, чем на равенство каждому из остальных чисел.

◁ **Блок not**
Блок not («не») меняет ответ на противоположный: истина становится ложью, а ложь — истиной.



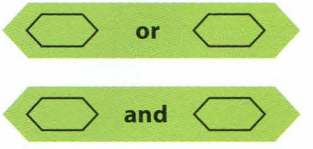
△ **Без блока not**
Числа 10 и 7 различны, так что ответ будет «ложь».



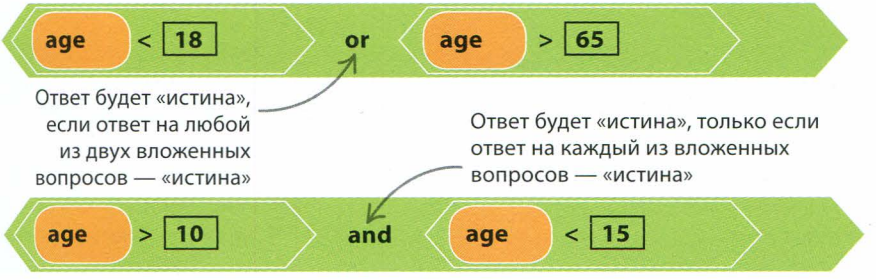
△ **С блоком not**
Если добавить блок not, результат изменится. Поскольку 10 не равно 7, теперь ответ будет «истина».

Объединение вопросов

Чтобы задавать более сложные вопросы, состоящие из нескольких простых, блоки сравнения можно объединять.



△ **Блоки сравнения**
Блоки or («или») и and («и») нужны, чтобы по-разному объединять булевы выражения.



◁ **Пример**
Верхний блок проверяет, что age меньше 18 или больше 65. Нижний блок проверяет, что age равна 11, 12, 13 или 14.

Решения и ветвление

Сведения об истинности или ложности можно использовать, чтобы объяснить компьютеру, что делать дальше. Он выполнит разные действия в зависимости от того, был ли ответ «истина» или «ложь».

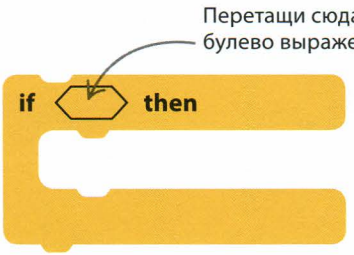
СМОТРИ ТАКЖЕ

❮ 62–63 Истина или ложь?

Считывание и распознавание 66–67 ❯

Принятие решений

Блоки if («если») используют булевы выражения, чтобы решить, что делать дальше. Если поместить внутрь блока if другие блоки, они будут выполняться, лишь когда булево выражение в заголовке if дает истину.

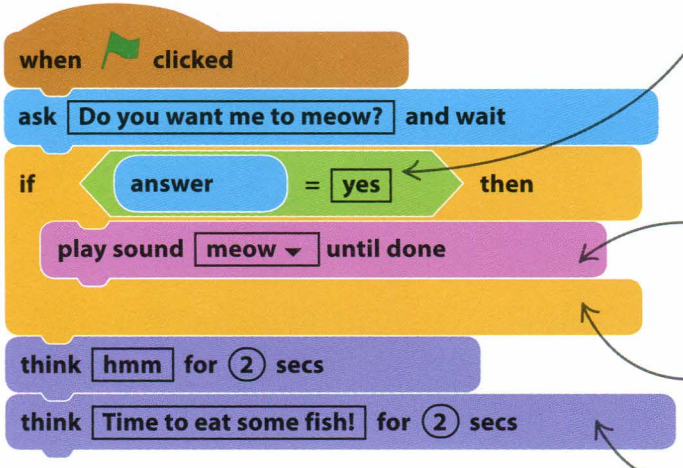


△ **Блок if-then**
Если булево выражение истинно, блоки внутри if-then («если, то») выполняются.

△ **Блок if-then-else**
«Если, то, иначе»: если булево выражение истинно, выполнится первый набор блоков, иначе — второй набор.

Использование блока if-then

Блок if-then позволяет на основе значения булева выражения решить, нужно ли выполнять часть скрипта. Чтобы попробовать этот скрипт, добавь его к спрайту кота.



△ **Мяукающий кот**
Эта программа проверяет значение булева выражения и запускает блоки внутри if-then, если это значение истинно. Кот замяукает, только если его об этом попросить.

Перетащи блок «=» из секции Operators («Операторы») в заголовок блока if-then («если, то»). Затем помести в «=» переменную answer («ответ») из секции Sensing («Сенсоры»)

Эти блоки находятся внутри if-then («если, то» — кот мяукает, только если на вопрос был дан ответ yes («да»))

Конец блока if-then («если, то»)

Эти блоки think («думать») — за пределами блока if-then («если, то»), а значит, они будут выполнены в любом случае

△ **Как это работает**
Программа проверяет, истинно ли булево выражение. Если это так, она выполняет блоки внутри if-then («если, то»).

Инструкции ветвления

Часто нужно, чтобы программа сделала что-то одно, если условие выполняется, и что-то другое, если нет. Блок if-then-else («если, то, иначе») задает два варианта исполнения программы, которые называют ветвями. В зависимости от ответа на булево выражение запустится лишь одна из ветвей.

▽ **Ветвление программы**
У этой программы две ветви: если ответ был yes («да»), запустится одна ветвь, иначе — другая.

Если ответить yes («да»), запустится эта ветвь

Если ответить что угодно, кроме yes («да»), запустится эта ветвь

△ **Как это работает**
Программа смотрит, было ли введено слово yes («да»), и, если было, показывает первое сообщение, иначе — второе.

СОВЕТЫ ЭКСПЕРТА

Форма блоков

В Scratch у блоков булевых выражений заостренные края, однако в лунки другой формы их тоже можно перетаскивать.

△ **Блоки Sensing**
Эти блоки проверяют, не нажата ли кнопка мышки.

△ **Блоки Control**
В некоторых блоках Control («Управление») есть лунки заостренной формы для булевых выражений.

▷ **Ветви**
Ветви программы словно ветви дерева, которые расходятся и растут в разных направлениях.

Считывание и распознавание

Блоки из секции Sensing («Сенсоры») рассказывают скрипту, что происходит с компьютером. Они могут считывать нажатия клавиш и позволяют спрайтам реагировать на столкновения.

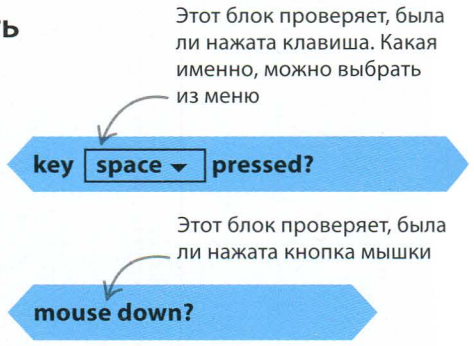
СМОТРИ ТАКЖЕ

❮ 40–41 Костюмы

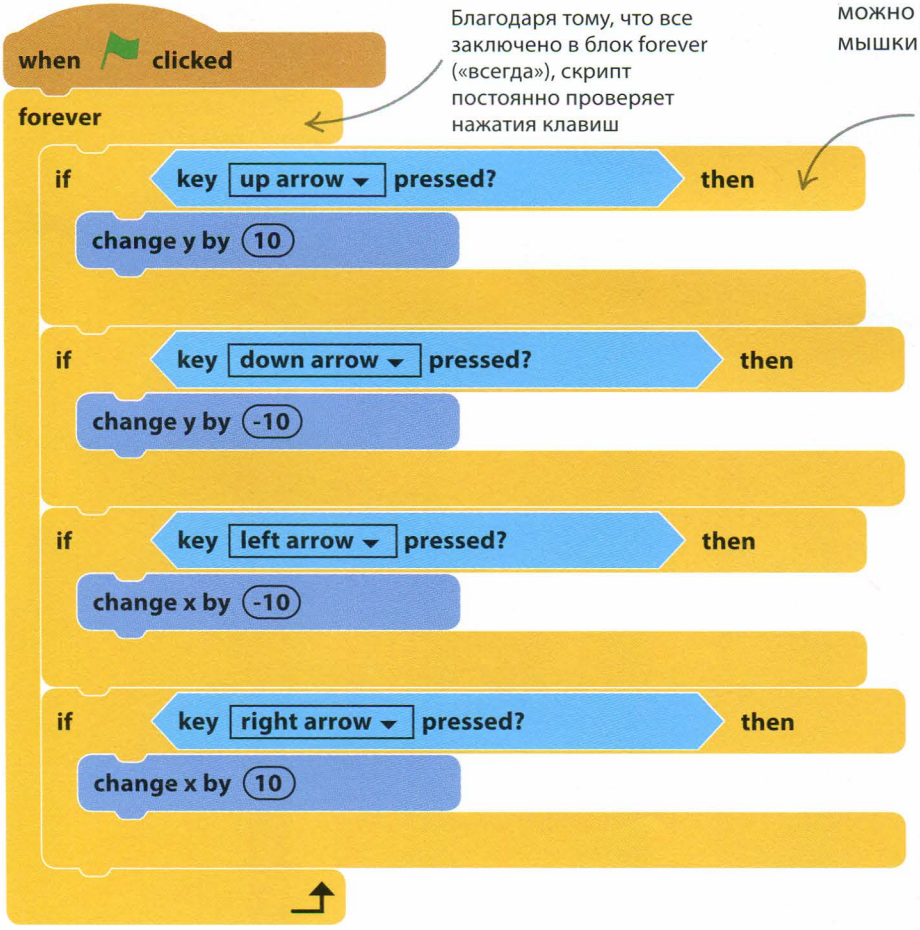
❮ 56–57 Координаты

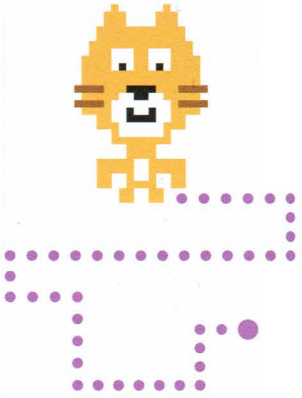
Управление с клавиатуры

С помощью блоков Sensing («Сенсоры») и блока If-then можно двигать спрайт по экрану, управляя им с клавиатуры. В меню блока key pressed? («мышка нажата?») есть большинство клавиш клавиатуры, и спрайт сможет на них реагировать. Также можно выполнять действия при нажатии кнопок мышки.



△ **Блоки Sensing**
Используя эти блоки с блоком if-then, можно узнать, была ли нажата кнопка мышки или клавиша на клавиатуре.



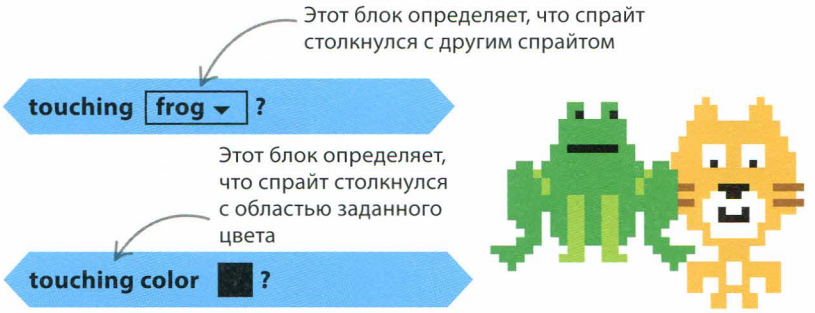


△ **Управляем спрайтами**
Клавиатура подходит для точного управления спрайтами — как раз то, что нужно для игр.

◁ **Скрипт управления**
Этот скрипт позволяет двигать спрайты вверх, вниз, влево и вправо с помощью стрелок на клавиатуре.

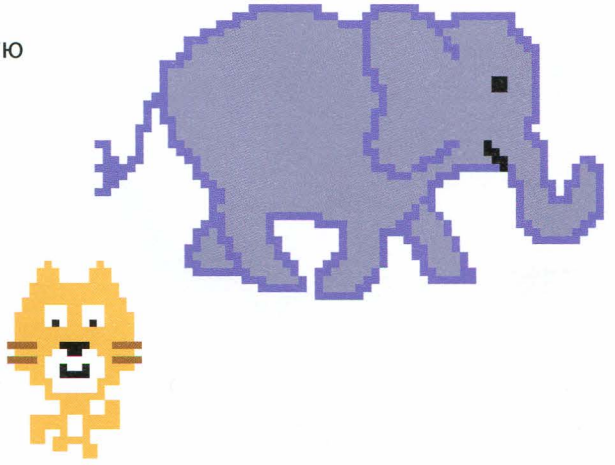
Столкновения спрайтов

Бывает важно знать, когда спрайты соприкасаются один с другим — например, в играх. Используй блоки Sensing («Сенсоры»), чтобы реагировать на столкновения спрайтов или на касание спрайтом области заданного цвета.



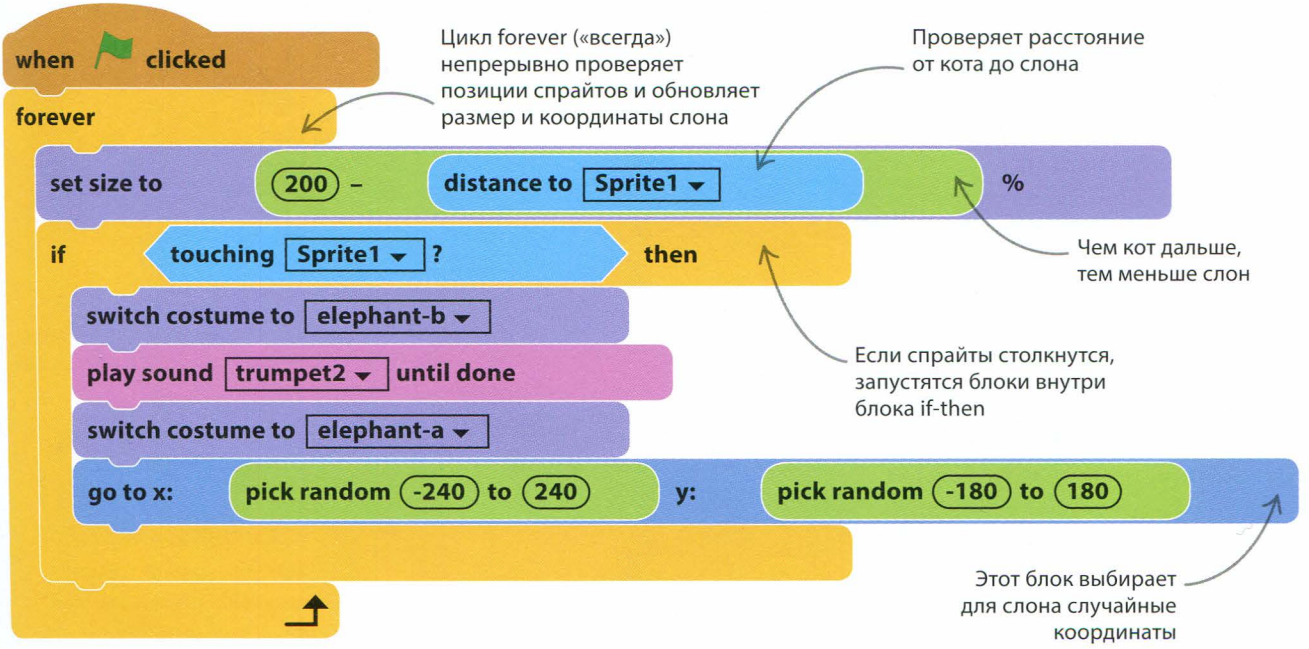
Использование блоков Sensing

Сделаем из нашего управляемого кота настоящую игру. Сперва добавь к спрайту кота скрипт передвижения со страницы 66, затем добавь на сцену фон room1 и спрайт слона. Используя вкладку Sounds («Звуки»), добавь слону звук trumpet2. Также добавь ему следующий скрипт.



▽ Найди слона

В этом скрипте блоки Sensing («Сенсоры») используются, чтобы управлять взаимоотношениями кота и слона. Чем ближе кот подходит к слону, тем больше тот становится. Если кот прикоснется к слону, слон сменит костюм, затрубит в хобот и убежит в другое место.



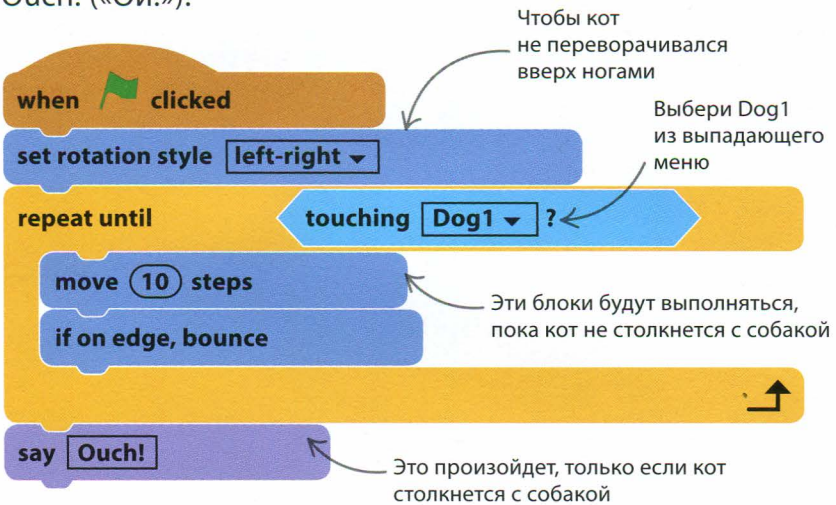
Сложные циклы

СМОТРИ ТАКЖЕ
‹ 46–47 Простые циклы
‹ 62–63 Истина или ложь?

Простые циклы нужны, чтобы повторять части программы или бесконечно, или заданное количество раз. Чтобы повторять инструкции лишь в определенных случаях, есть другие, более хитрые циклы.

Повтор, пока что-то не произойдет

Загрузи в проект спрайт Dog1 и добавь к спрайту кота следующий скрипт. Если его запустить, блок repeat until («повторять, пока не») будет двигать кота, пока тот не наткнется на собаку. Тогда кот остановится и скажет: Ouch! («Ой!»).

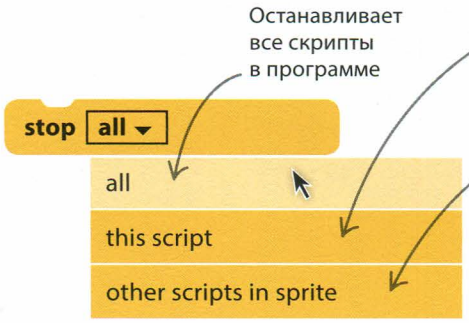


△ **Проверка программы**
Убери собаку подальше от кота и запусти программу. Затем перетащи собаку, оставив ее на пути у кота, и смотри, что произойдет.

△ **Блок Repeat until**
Блоки, находящиеся внутри блока repeat until («повторять, пока не»), будут повторяться, пока условие не станет истинно (пока кот и собака не столкнутся).

Стоп!

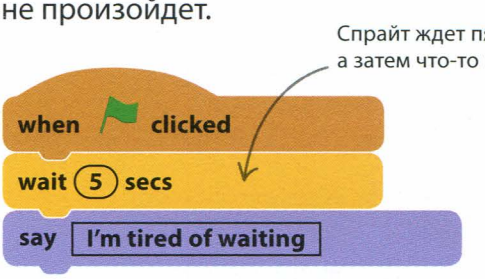
Еще один полезный блок из Control («Управление») — это stop all («стоп все»), который останавливает выполнение скрипта. Например, он пригодится, если надо остановить спрайты в конце игры.



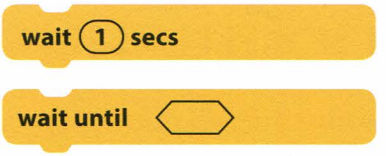
◁ **Остановка скриптов**
Из выпадающего меню можно выбрать, какие скрипты останавливать.

Ожидание

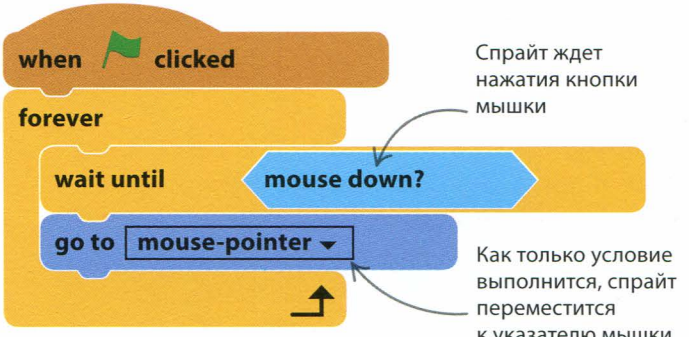
Играть в игру или следить за выполнением программы легче, если скрипт делает небольшие паузы. Есть блоки, которые могут приостановить скрипт на несколько секунд или до тех пор, пока что-то не произойдет.



△ **Блок Wait secs**
Используя wait secs («ждать секунд»), можно приостанавливать скрипт на заданное количество секунд.



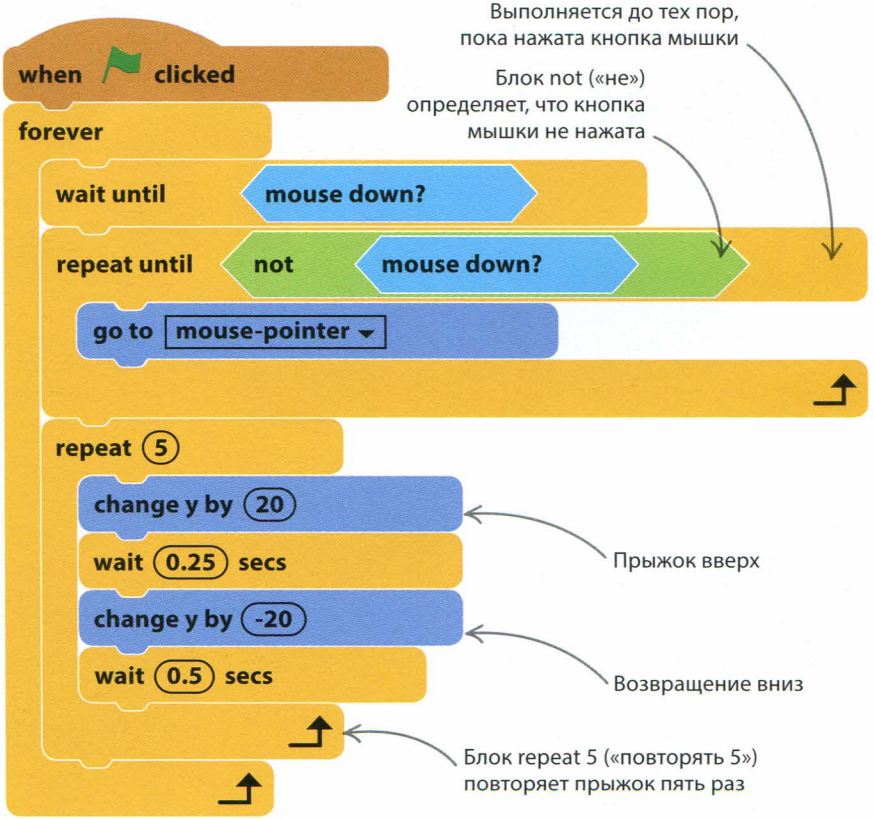
◁ **Блоки ожидания**
Блок wait secs («ждать секунд») делает паузу на заданное время. Блок wait until («ждать до») ожидает соблюдения условия.



△ **Блок Wait until**
«Ждать до»: этот блок держит паузу, пока булево выражение не станет истинно.

Притягательная мышка

Различные циклы можно использовать вместе. Эта программа начинает работать, если нажать кнопку мышки. Спрайт движется за указателем мышки, пока не отпустишь кнопку. Потом он подскакивает вверх-вниз пять раз. Затем, благодаря циклу forever («всегда»), все повторяется.



▷ **Вложенные циклы**
Обрати внимание, что циклы repeat («повторять») вложены в цикл forever («всегда»).

Обмен сообщениями

Порой спрайтам нужно «договариваться» между собой. Для этого они могут посылать друг другу сообщения. Также можно сделать так, чтобы спрайты беседовали друг с другом.

СМОТРИ ТАКЖЕ

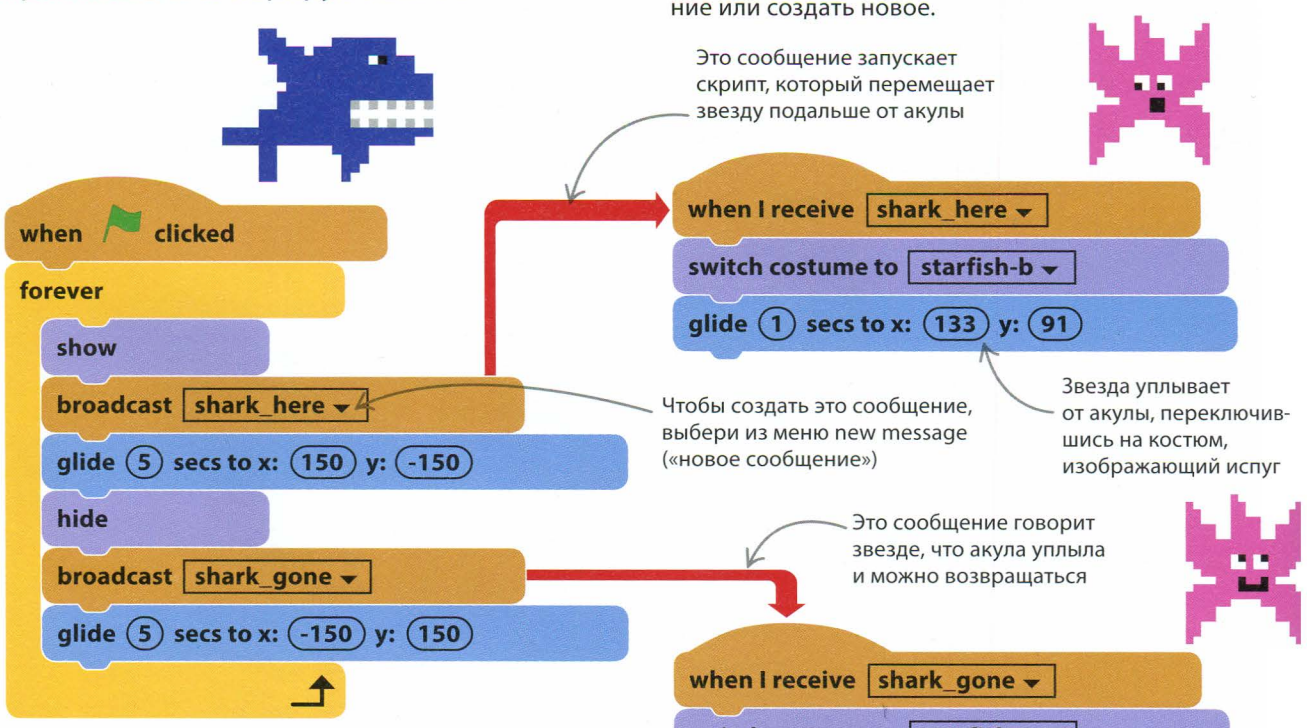
- ❮ 38–39 Перемещение объектов
- ❮ 40–41 Костюмы
- ❮ 44–45 События

Связь

Блоки связи из секции Events («События») позволяют спрайтам посылать и принимать сообщения. Сообщения состоят из одного лишь имени, но и этого достаточно для взаимодействия спрайтов. Спрайты реагируют только на те сообщения, которые они запрограммированы принимать, и игнорируют все остальные.

△ **Блоки связи**

Один тип блоков связи позволяет спрайту отправлять сообщения, другой говорит спрайту, что сообщение нужно принять. Можно использовать готовое сообщение или создать новое.



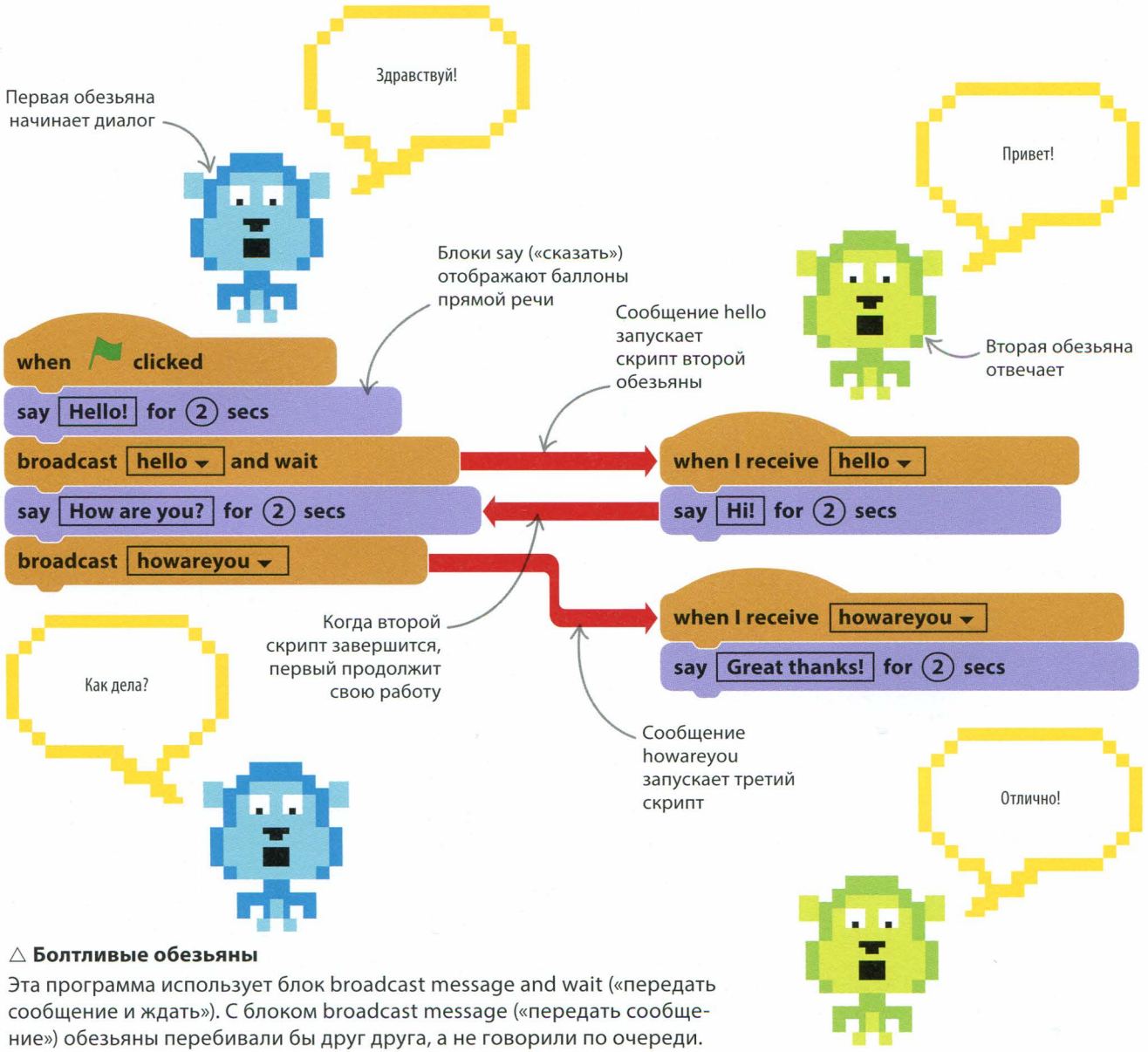
△ **Берегись акулы**

Выбери два спрайта — акулу и морскую звезду. Добавь акуле верхний скрипт, а звезде — два скрипта справа. Когда акула появляется, она шлет сообщение, получив которое звезда отплывает в сторонку.

Диалоги

Если нужно, чтобы спрайты вели диалог, можно использовать блоки broadcast message and wait («передать сообщение и ждать») с блоками say («сказать»), отображающими баллоны прямой речи. Начни новый проект и добавь в него два спрайта обезьяны. Добавь скрипт снизу к одной обезьяне, а два скрипта справа — к другой.

△ **Отправка и ожидание**
Этот блок отправляет сообщение и ждет, пока все скрипты, принимающие это сообщение, завершат свою работу.



△ **Болтливые обезьяны**
Эта программа использует блок broadcast message and wait («передать сообщение и ждать»). С блоком broadcast message («передать сообщение») обезьяны перебивали бы друг друга, а не говорили по очереди.

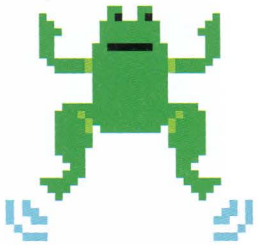
Создание блоков

СМОТРИ ТАКЖЕ
〈 50–51 Переменные
Приступим к экспериментам! 82–83 〉

Чтобы не повторять одну и ту же последовательность блоков несколько раз, можно создать новый блок. Каждый новый блок может содержать разные инструкции.

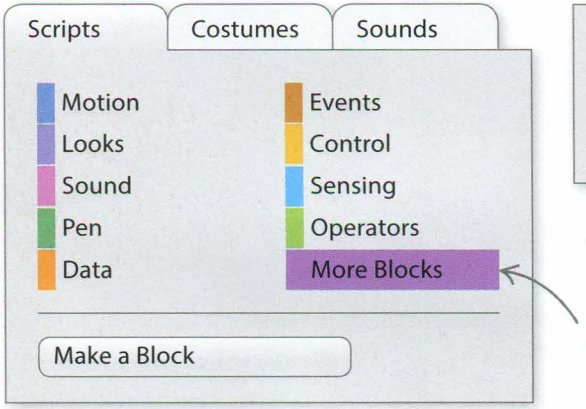
Создание своего блока

Можно создать свой блок, который, если его запустить, выполнит некий скрипт. Как это делается, мы выясним сейчас на примере. Программисты называют такие выделенные части кода подпрограммами или функциями.



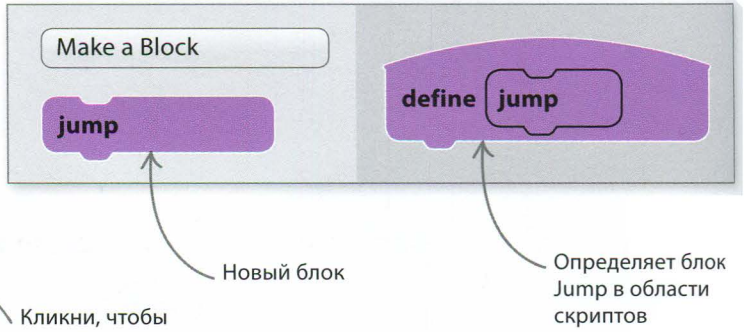
1 Создаем новый блок

Кликни по кнопке More Blocks («Новые блоки») и выбери Make a Block («Создать блок»). Введи слово jump и кликни OK.



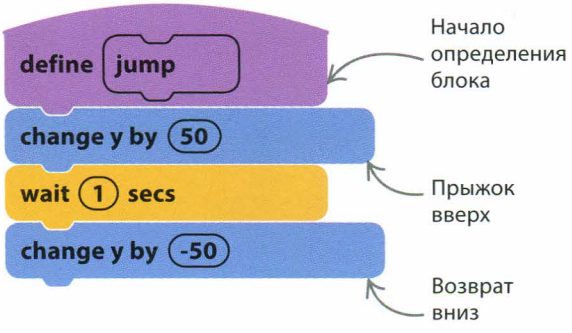
2 Новый блок появился

Новый блок Jump появился в палитре блоков, а в области скриптов возник блок define («определить»).



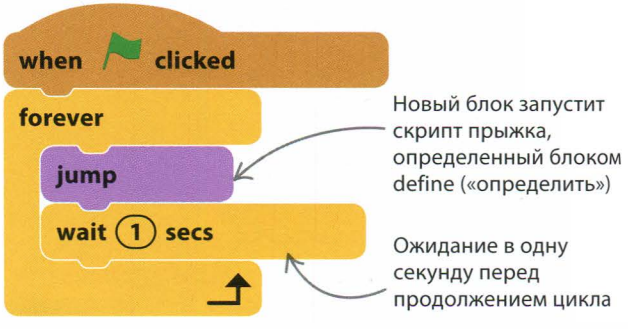
3 Блок define («определить»)

Блок define («определить») определяет, что нужно делать при запуске нового блока. Задай определение блока, добавив этот скрипт.



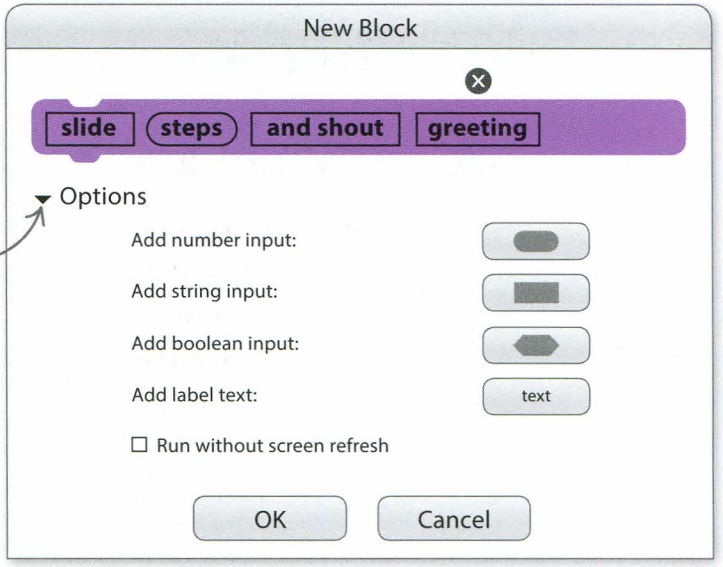
4 Используем блок в скрипте

Теперь новый блок можно использовать в любом скрипте — так, будто блоки для прыжка стоят на его месте.



Блоки с параметрами

Через лунки параметров в новый блок можно передавать числа или строки. Например, можно указать, на какое расстояние передвинуть спрайт.



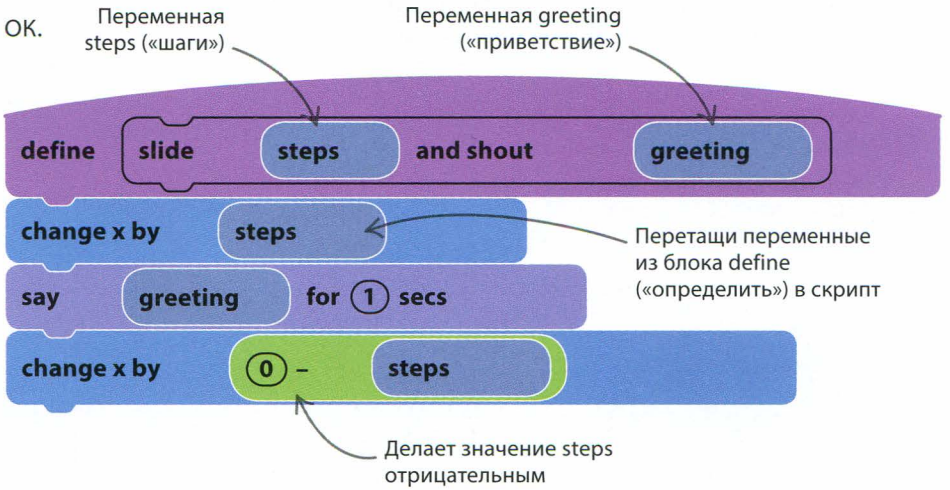
Кликни тут, чтобы увидеть все опции

1 Создаем новый блок

Создай новый блок по имени slide и кликни Options («Параметры»). Теперь выбери input («Добавить поле ввода чисел») и введи steps. Выбери Add label text («Добавить текст подписи») и введи and shout. Кликни Add string input («Добавить поле ввода строк») и введи greeting. Теперь нажми OK.

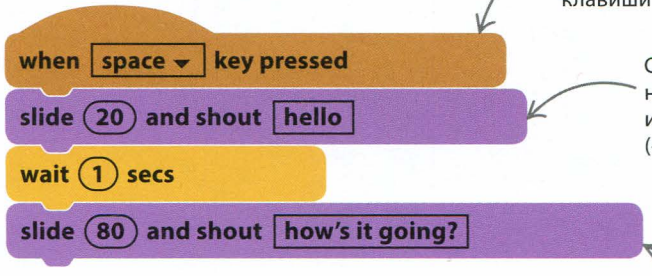
2 Определение блока

В блоке define («определить») вместо лунок параметров появились переменные steps и greeting. Добавь к спрайту этот скрипт, перетаскивая переменные из блока define («определить») туда, где они нужны.



3 Используем новый блок в скрипте

Теперь добавь к спрайту следующий скрипт. Если передавать в блок различные значения для steps и greeting, спрайт будет вести себя по-разному.



Выбирай для своих блоков понятные имена, чтобы программу было легче читать и изменять.

ПРОЕКТ 3

Бешеные обезьяны

В этой захватывающей, динамичной игре мы используем все изученные возможности Scratch. Следуй инструкциям, чтобы создать собственную игру «Бешеные обезьяны», и поглядим, сумеешь ли ты сбить летучую мышь бананом!

СМОТРИ ТАКЖЕ

- 40–41 Костюмы
- 38–39 Перемещение объектов
- 66–67 Считывание и распознавание

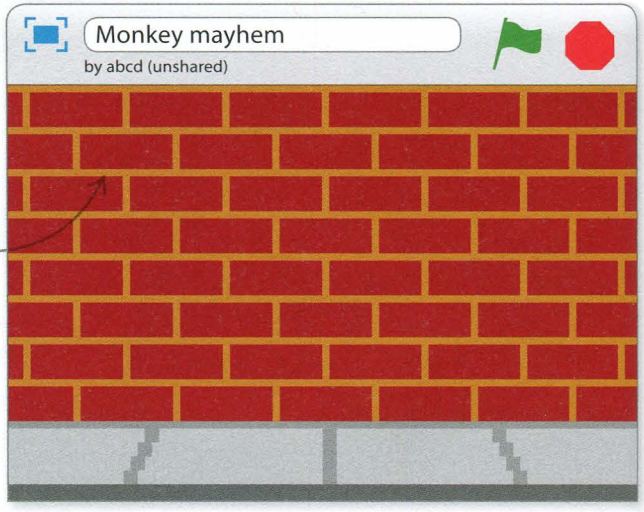
Приступим

Создай новый Scratch-проект. Спрайт кота в этой игре не понадобится — чтобы удалить его, сделай правый клик по коту в списке спрайтов и выбери из меню delete («удалить»). Теперь перед тобой чистый проект.

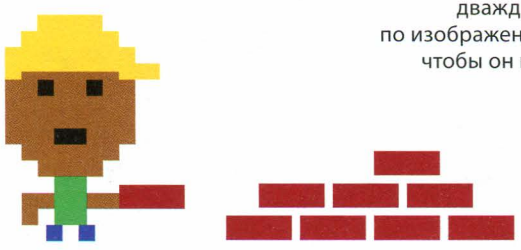
1. Добавь новый фон из библиотеки с помощью кнопки, которая находится слева от списка спрайтов.

Кликни, чтобы добавить фон из библиотеки фонов

2. Двойным кликом выбери фон brick wall1. Кирпичная стена хорошо подходит для этой игры, но при желании ты можешь выбрать и другой фон.



В библиотеке фонов дважды кликни по изображению фона, чтобы он появился на сцене



СОВЕТЫ ЭКСПЕРТА

Предотвращаем ошибки

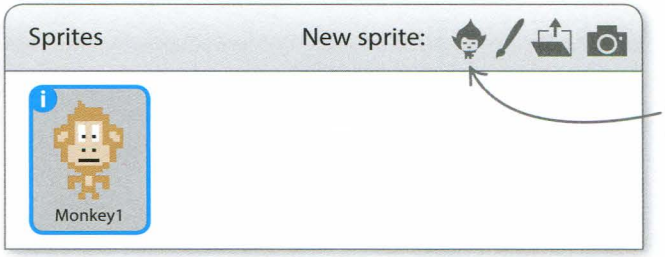
Эта программа — самая большая из всех, что встречались тебе до сих пор, и сразу она может не заработать. Вот несколько советов, как избежать ошибок при работе над проектом.

Убедись, что ты добавляешь скрипты к нужным спрайтам.

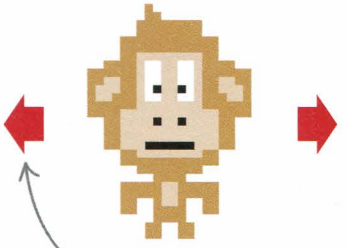
Внимательно следуй инструкциям. Не забывай создавать переменную перед ее использованием.

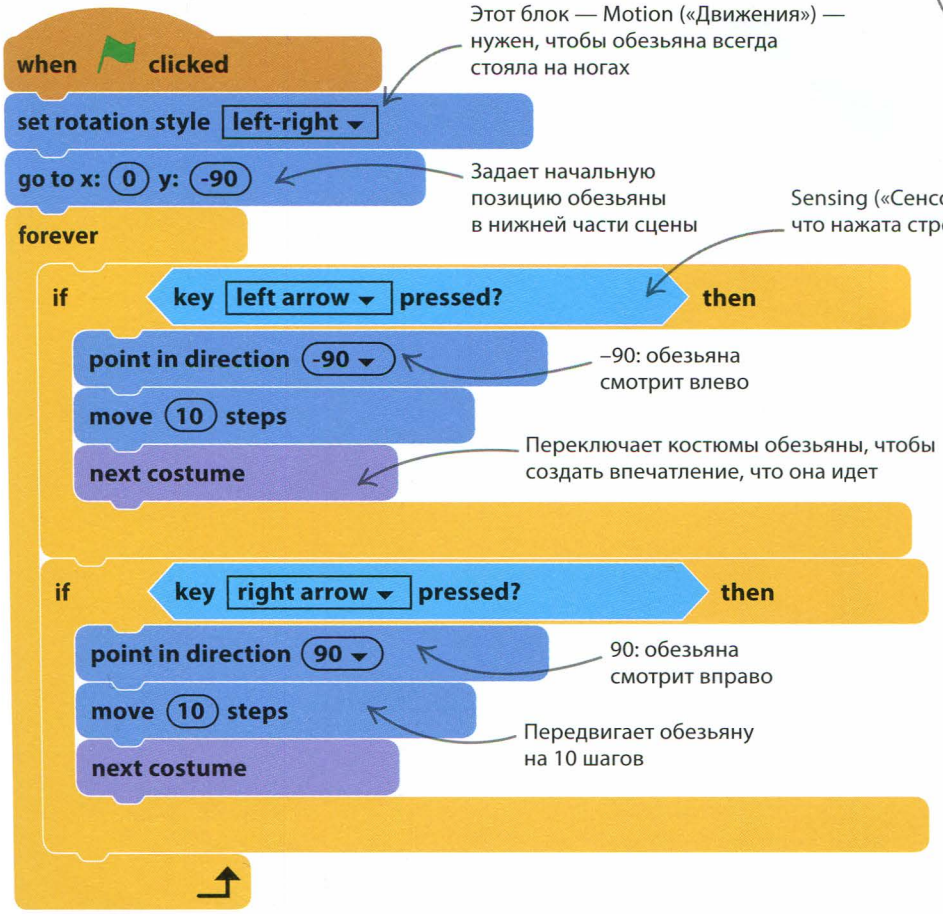
Проверь, что все числа в блоках записаны без ошибок.

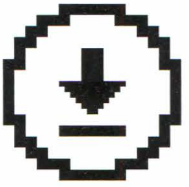
3 Добавь новый спрайт из библиотеки — выбери Monkey1 из раздела Animals («Животные»). Этим спрайтом тебе предстоит управлять в игре.



4 Добавь к спрайту обезьяны следующий скрипт. Помни: все блоки можно найти в палитре блоков, где они отсортированы по цвету. Блоки Sensing («Сенсоры») используются в этом скрипте, чтобы нажатиями клавиш-стрелок двигать обезьяну по сцене. Запусти скрипт и убедись, что он работает.





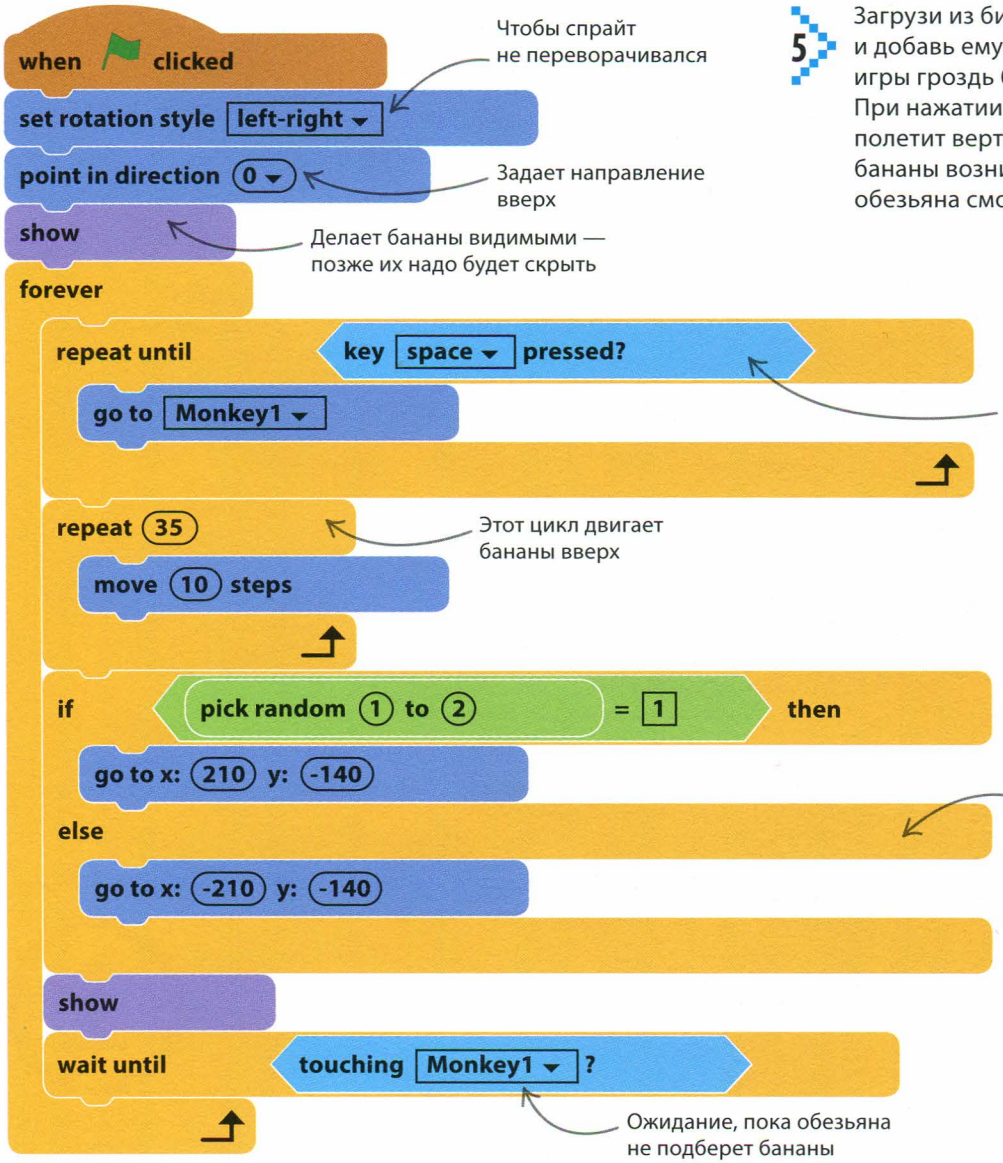


Не забудь сохранить свою работу.

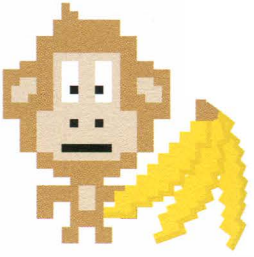
БЕШЕНЫЕ ОБЕЗЬЯНЫ

Добавим еще спрайтов

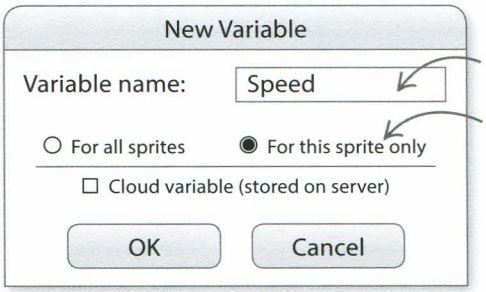
Теперь обезьяну можно двигать по сцене вправо и влево клавишами-стрелками. Чтобы игра стала интереснее, добавим еще спрайтов и снабдим обезьяну бананами, чтобы кидаться ими в летучую мышь!



5 Загрузи из библиотеки спрайт Bananas и добавь ему этот скрипт. В начале игры гроздь бананов будет у обезьяны. При нажатии клавиши «пробел» гроздь полетит вертикально вверх. Затем бананы возникнут на краю сцены, где обезьяна сможет их подобрать.



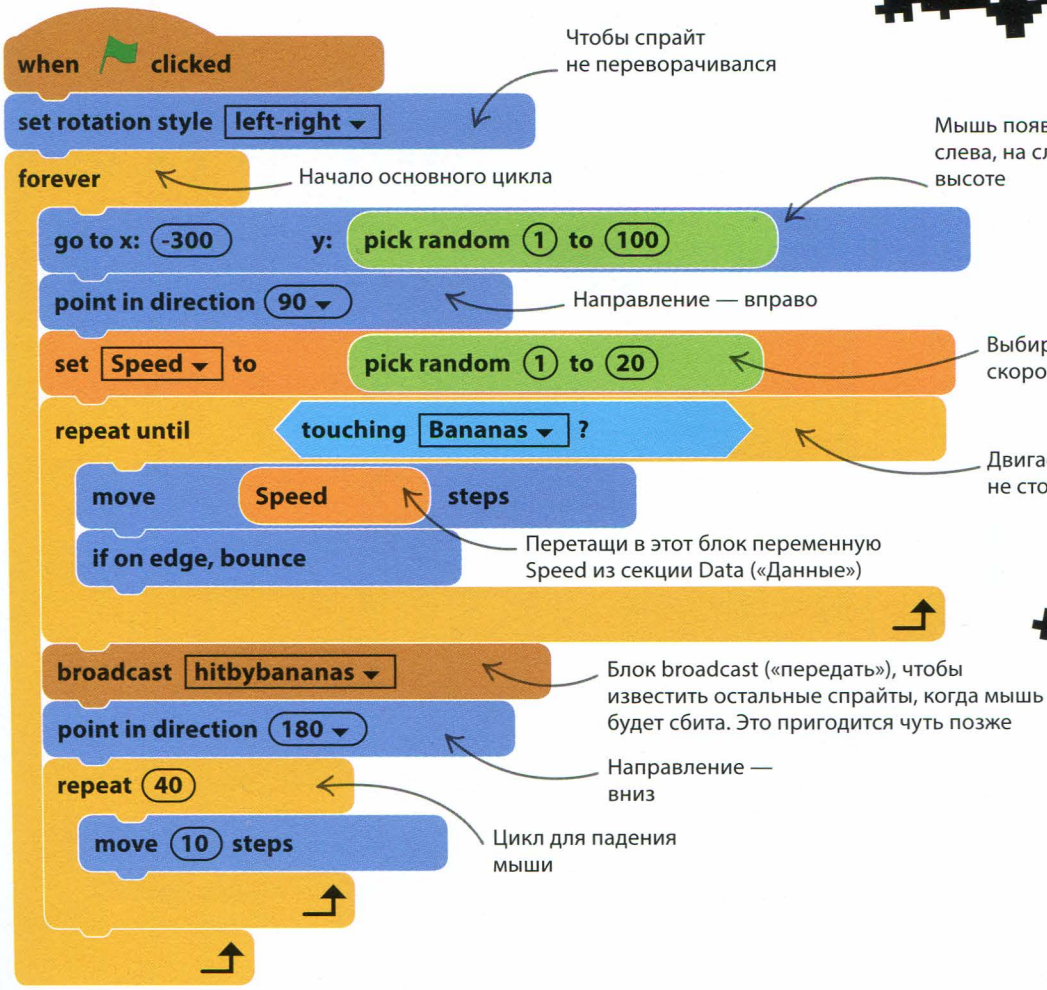
6 Теперь самое время заняться летучей мышью, которая должна падать, если попасть в нее бананами. Добавь из библиотеки спрайт Bat2, затем создай переменную с именем Speed (только для спрайта мыши). Для этого кликни по кнопке Data («Данные») в палитре блоков и выбери Make a Variable («Создать переменную»). Сними отметку рядом с переменной Speed, чтобы убрать ее со сцены.



Назови переменную Speed

Эта переменная нужна только для спрайта мыши

7 Добавь к спрайту летучей мыши следующий скрипт. В основном цикле forever («всегда») мышь перемещается в случайную позицию в левой части сцены и со случайно выбранной скоростью летает туда-обратно, пока не встретится с бананами. Тогда сбитая мышь падает.



Чтобы спрайт не переворачивался

Начало основного цикла

Мышь появляется слева, на случайной высоте

Направление — вправо

Выбирает случайную скорость

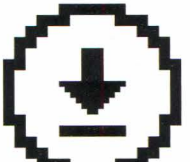
Двигает мышь, пока она не столкнется с бананами

Перетащи в этот блок переменную Speed из секции Data («Данные»)

Блок broadcast («передать»), чтобы известить остальные спрайты, когда мышь будет сбита. Это пригодится чуть позже

Направление — вниз

Цикл для падения мыши



Не забудь сохранить свою работу.

БЕШЕНЫЕ ОБЕЗЬЯНЫ

Последние штрихи

Чтобы игра стала еще интереснее, можно добавить ограничение по времени, запоминать количество заработанных очков в переменной и сделать экран конца игры, чтобы показывать его, когда время истечет.

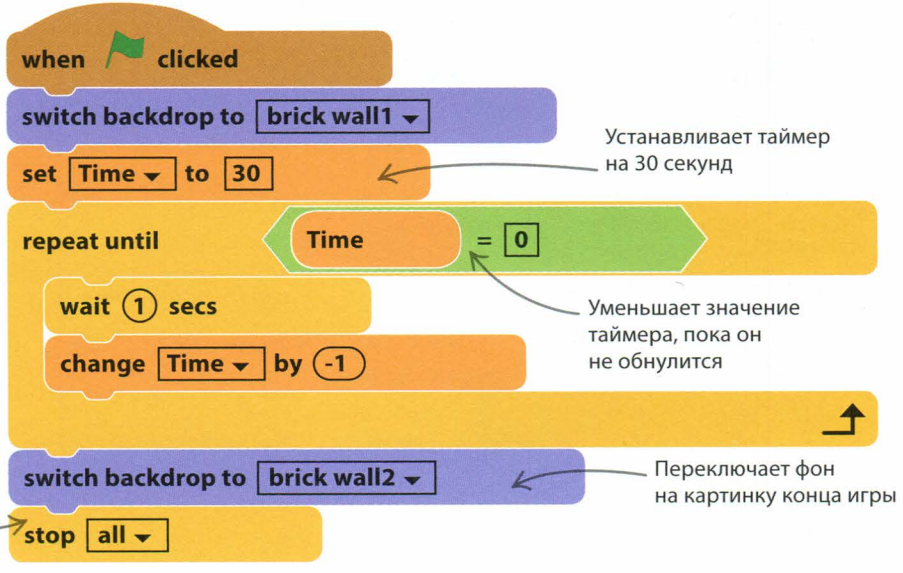
8 Создай переменную с именем Time («Время»). Выбери опцию For all sprites («Для всех спрайтов»), чтобы переменная была доступна для всех спрайтов. Поставь отметку рядом с переменной в секции Data («Данные»), чтобы таймер отображался на сцене.

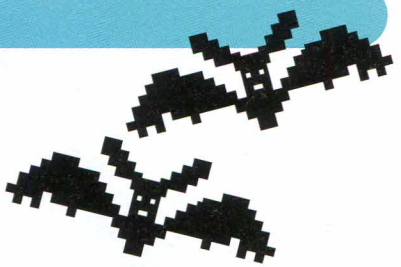
9 Кликни по изображению сцены слева от списка спрайтов, затем открой вкладку Backdrops («Фоны») над палитрой блоков. Сделай правый клик по изображению фона слева от области рисования и выбери duplicate («дублировать»). Добавь поверх скопированного фона надпись GAME OVER.

Используй инструмент «текст», чтобы сделать надпись

Экран конца игры будет выглядеть примерно так

10 Кликни по вкладке Scripts («Скрипты») и добавь к сцене этот скрипт для работы с таймером. Скрипт задает значение таймера и начинает цикл обратного отсчета. Когда таймер обнулится, выставляется фон конца игры и программа останавливается.

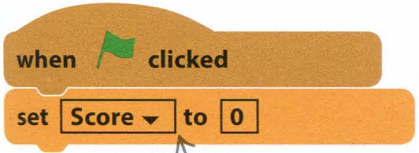




11 Кликни по спрайту бананов в списке спрайтов. Создай новую переменную с именем Score («Счет») и сделай ее доступной для всех спрайтов. Перетащи ее изображение в верхний правый угол сцены.

Поставь галочку, чтобы счет игры отображался на сцене

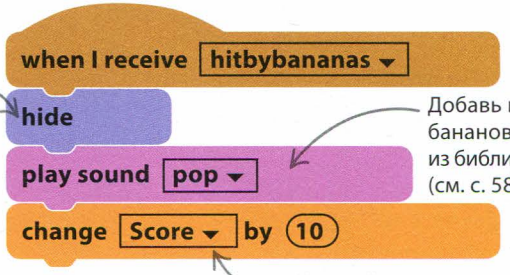
12 Добавь этот скрипт к спрайту бананов. Он нужен, чтобы обнулить счет в начале игры.



Сбрасывает счет игры

13 Добавь к спрайту бананов еще и этот скрипт. Если летучая мышь подбита бананами, он проиграет звук, увеличит счет на 10 и сделает бананы невидимыми.

Прячет бананы



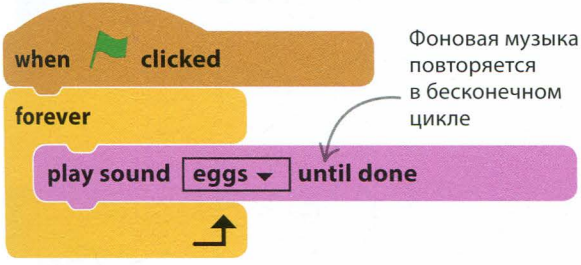
Добавь к спрайту бананов звук pop из библиотеки звуков (см. с. 58–59)

Каждый раз при попадании в мышь счет увеличивается на 10 очков

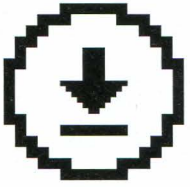
14 Пусть в игре звучит музыка. Кликни по изображению сцены слева от списка спрайтов и открой вкладку Sounds («Звуки») над палитрой блоков. Добавь звук eggs из библиотеки звуков.

Добавь звук eggs во вкладке Sounds («Звуки»)

15 Добавь к сцене следующий скрипт — он будет в цикле «играть музыку из файла eggs» и перестанет звучать, лишь когда блок stop all («стоп все») остановит игру.



Фоновая музыка повторяется в бесконечном цикле



Не забудь сохранить свою работу.

ЗАПОМНИ

Твои успехи

Поздравляем с созданием полноценной Scratch-игры! Вот некоторые твои достижения.

Спрайт, кидающий предметы в другой спрайт.

Спрайт, падающий со сцены при попадании в него.

Ограничение времени игры.

Фоновая музыка, которая звучит, пока продолжается игра.

Экран конца игры, появляющийся, когда истекло время.

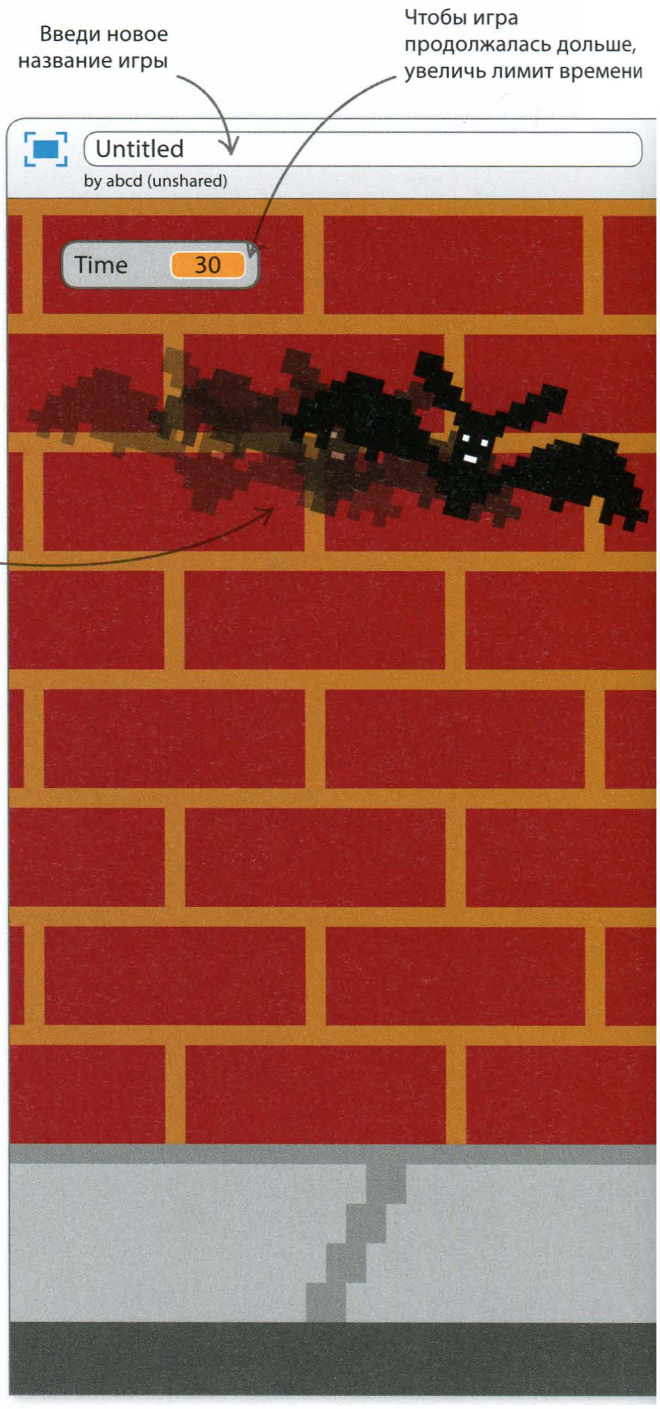
БЕШЕНЫЕ ОБЕЗЬЯНЫ

Пора играть

Теперь можно играть. Кликни по зеленому флажку, и посмотрим, сколько раз ты собьешь летучую мышь, прежде чем закончится время.

△ **Управление**

Двигай обезьяну вправо и влево стрелками клавиатуры. Нажимай пробел, чтобы швырнуть бананы в летучую мышь.

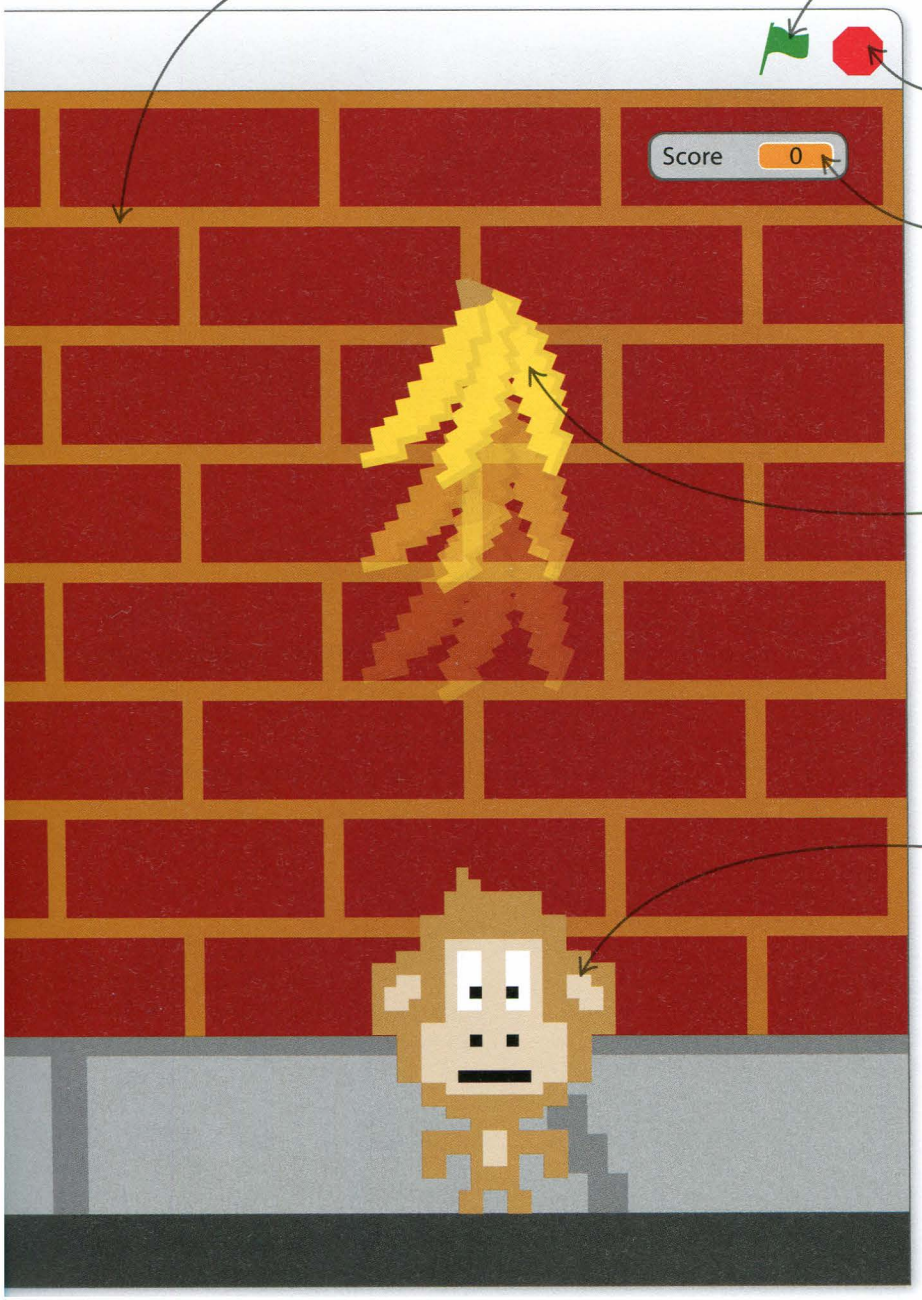


СОВЕТЫ ЭКСПЕРТА

Еще больше спрайтов

Чтобы добавить еще летучих мышей-целей, сделай правый клик по спрайту мыши и выбери duplicate («дублировать»). Появится новая мышь, с теми же скриптами, что и первая. Попробуй добавить другие летающие объекты.

1. Добавь спрайт из библиотеки. Летающий бегемот (Hippo1) отлично подойдет.
2. Кликни по летающей мыши в списке спрайтов.
3. Кликни по скрипту мыши, но не отпускай кнопку.
4. Перетащи скрипт мыши на изображение нового спрайта в списке спрайтов.
5. У нового спрайта появится копия скрипта.



Пробуй разные фоны и смотри на результат

Сыграй три раза и узнай, сколько очков ты сможешь набрать

Кликни по кнопке остановки программы, чтобы прервать игру

Измени программу, чтобы за попадание начислялось больше очков

Чтобы играть было сложнее, измени программу, уменьшив скорость бананов

Попробуй заменить обезьяну другим спрайтом

◁ Развлекаемся

Эту игру можно доработать множеством способов. Меняя скорости, подсчет очков, звуки и спрайты, ты можешь создать свою, уникальную версию игры.

Приступим к экспериментам!

Теперь, зная основы Scratch, ты можешь попробовать более продвинутые его функции. Чем больше практики, тем лучше ты будешь программировать.

СМОТРИ ТАКЖЕ

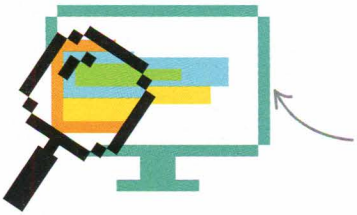
Что такое Python? 86–87 ❯

Простые команды 102–103 ❯

Чем заняться

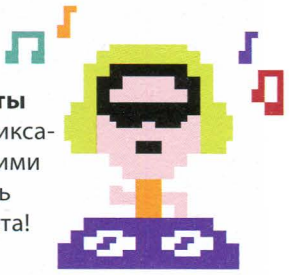
Не знаешь, что еще сделать со Scratch? Вот несколько идей. Если ты думаешь, что пока не готов написать программу самостоятельно, можешь взять уже написанную и что-нибудь в ней поменять.

◁ **Вступи в кружок программирования**
Есть ли в твоей школе кружок программирования? Там ты можешь повстречать других пользователей Scratch и обменяться идеями.



Сайт Scratch позволяет изучать все выложенные на нем проекты

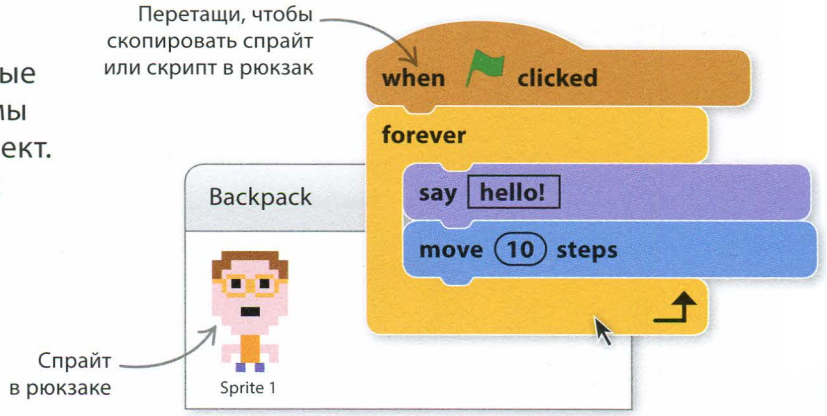
△ **Изучай программы**
Исследование чужих программ — прекрасный способ обучения. Посмотри на проекты, выложенные на сайте Scratch. Что ты сможешь почерпнуть из них?



▷ **Изменяй готовые проекты**
Делай ремиксы! Своими ремиксами можно поделиться с другими пользователями, но не забудь поблагодарить автора проекта!

Рюкзак

В рюкзаке можно хранить полезные скрипты, спрайты, звуки и костюмы и переносить их из проекта в проект. Рюкзак находится в нижней части экрана Scratch.



▷ **Перетаскивай**
Спрайты и скрипты можно перетаскивать в рюкзак, а потом добавлять в другие проекты.

Помогите!

Бывает сложно писать программу, не зная о некоторых нужных для нее блоках. В Scratch есть меню помощи, где описан каждый блок.

1 Помощь по блоку

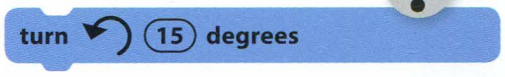
Чтобы узнать больше о конкретном блоке, кликни по кнопке «Помощь по блоку» среди инструментов курсора вверху экрана.

Кнопка «Помощь по блоку»

2 Спрашивай

Курсор примет вид вопросительного знака. Кликни по блоку, о котором ты хочешь узнать.

Курсор превращается в знак вопроса



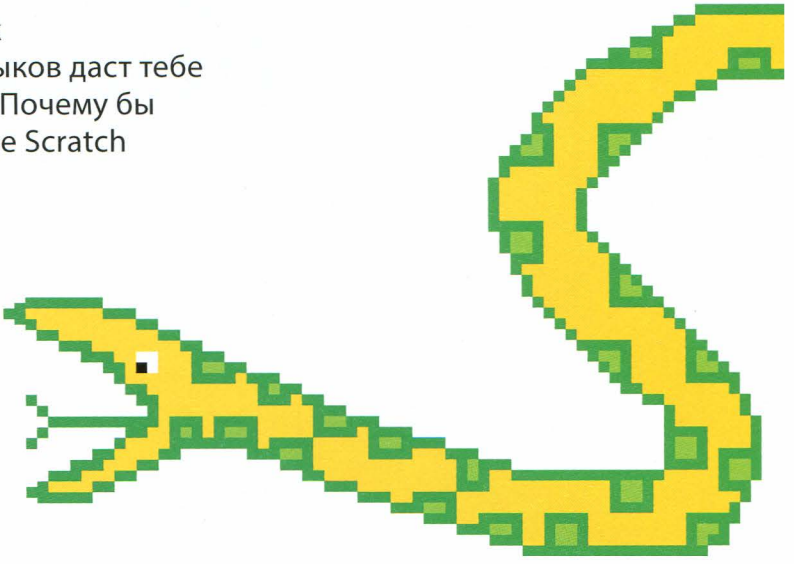
3 Окно помощи

Откроется окно с описанием работы блока и примерами его использования.

Окно помощи расскажет про каждый блок

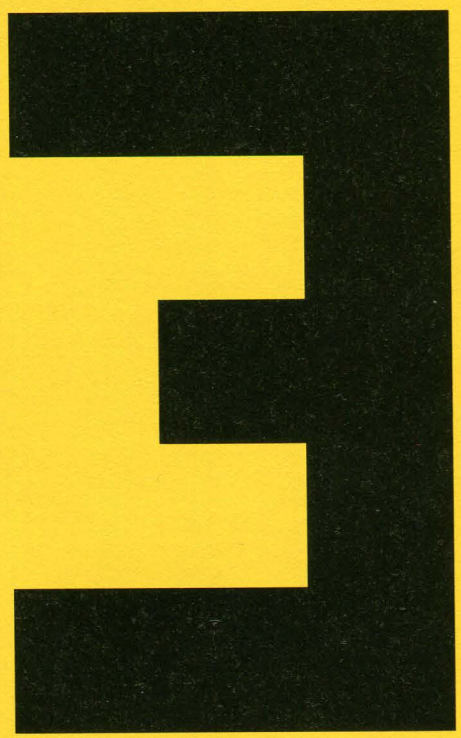
Изучи другой язык

Сейчас ты осваиваешь свой первый язык программирования. Изучение других языков даст тебе возможность писать другие программы. Почему бы не попробовать язык Python? Твое знание Scratch поможет в нем разобраться.

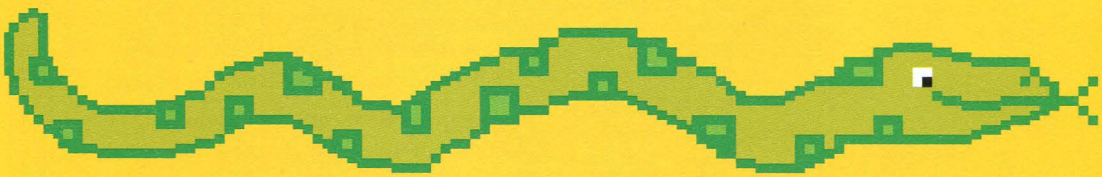


▷ **Похоже на Scratch**

В Python тоже есть циклы, переменные и ветвление. Используй знание Scratch, чтобы изучить Python!



Игры с Python



Что такое Python?

Python — это текстовый язык программирования. Он немного сложнее в изучении, чем Scratch, зато и возможностей дает гораздо больше.

СМОТРИ ТАКЖЕ

Установка Python **88–91 ❯**

Простые команды **102–103 ❯**

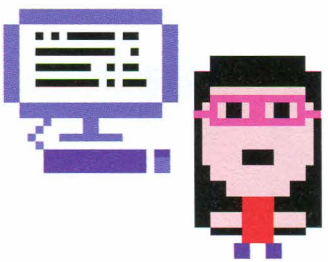
Команды посложнее **104–105 ❯**

Полезный язык

Python — это универсальный язык, пригодный для создания самых разных программ, от текстовых процессоров до веб-браузеров. Вот несколько веских причин изучить Python.

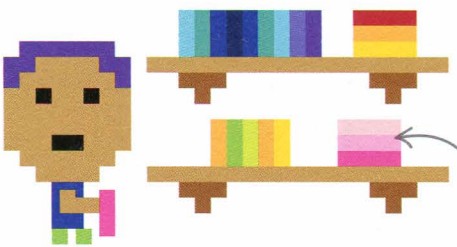
1 Легок в изучении и применении

Python — простой и удобный язык. По сравнению со многими другими языками читать и составлять программы на Python совсем не сложно.



2 Содержит мощные библиотеки

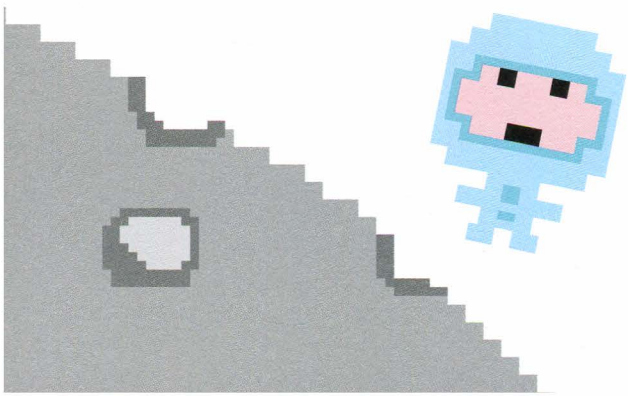
В Python есть библиотеки готовых процедур для использования в своих программах. Это позволяет создавать сложные программы быстро.



В Python есть много готовых процедур — бери и пользуйся

3 Используется серьезными фирмами

Python подходит для создания серьезных программ. Например, его используют в Google, NASA и на студии Pixar.



СОВЕТЫ ЭКСПЕРТА

С чего начать

Прежде чем научиться программировать на Python, стоит разобраться, как он устроен. На следующих страницах рассказывается о том, как:

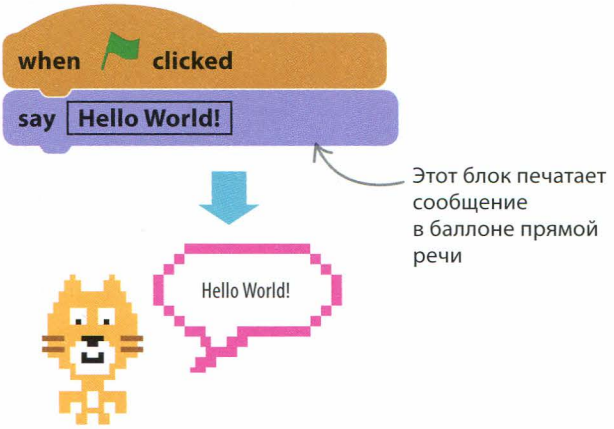
установить Python: Python бесплатен, но его нужно устанавливать на компьютер (с. 88–91);

использовать его интерфейс: создай простую программу и сохрани ее на диске;

экспериментировать: введи несколько простых программ и узнай, как они работают.

Scratch и Python

Для многих элементов языка Scratch есть аналоги в Python, правда, выглядят они иначе. Вот некоторые функции, общие для этих двух языков.



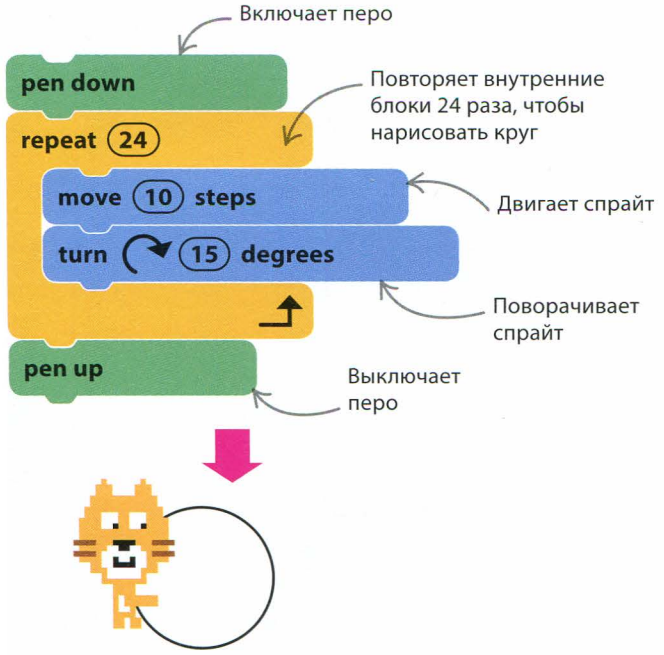
△ **Сообщения в Scratch**
Чтобы показать сообщение на экране, в Scratch используется блок say («сказать»).

```
print('Hello World!')
```
Введи сообщение здесь

Сообщение возникнет на экране

```
Hello World!
```

△ **Сообщения в Python**
В Python для отображения текста на экране служит команда print («печать»).



△ **Черепашья графика в Scratch**
В этом скрипте блок Pen down («Опустить перо») используется, чтобы, перемещая спрайт кота, нарисовать круг.

```
from turtle import *
pendown()
for n in range(24):
    forward(10)
    right(15)
penup()
```
Начало цикла

Поворот на 15 градусов по часовой стрелке

△ **Черепашья графика в Python**
В Python тоже есть черепашка. Это код для рисования круга.

Установка Python

Перед тем как начать программировать на Python, его нужно скачать и установить на компьютер. Python 3 бесплатен, прост в установке и работает на компьютерах с Windows, Mac OS и Linux (например, Ubuntu).

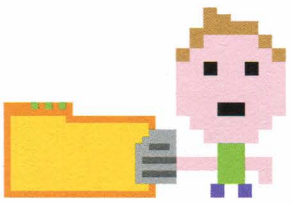
Что такое IDLE?

Вместе с Python 3 на компьютер будет установлена программа IDLE. Это ориентированная на начинающих среда разработки, в которой есть несложный текстовый редактор для написания и отладки Python-программ.

СОВЕТЫ ЭКСПЕРТА

Сохранение программы

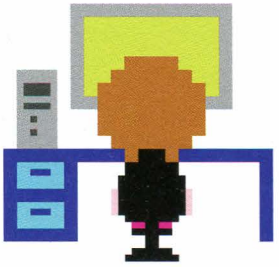
Чтобы сохранить программу на Python, нужно зайти в меню File и, выбрав Save As, ввести имя файла. Сперва создай для своих файлов папку, назвав ее, к примеру, «PythonПрограммы». Посоветуйся с родителями, как это лучше сделать.



WINDOWS

△ **Windows**

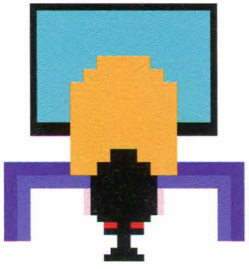
Перед тем как скачать Python, выясни, какая на компьютере операционная система. Если это Windows, узнай, какая версия — 32- или 64-битная. Для этого кликни по кнопке «Пуск», сделай правый клик по опции «Мой компьютер» и кликни «Свойства». Если появится опция «Система», выбери ее.



Mac OS

△ **Mac OS**

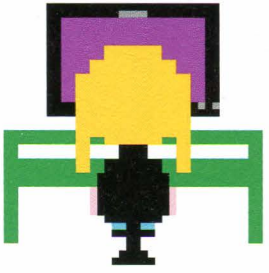
Если у тебя Apple Mac, выясни, какая на нем операционная система. Кликни по яблоку в левом верхнем углу и выбери About This Mac.



UBUNTU

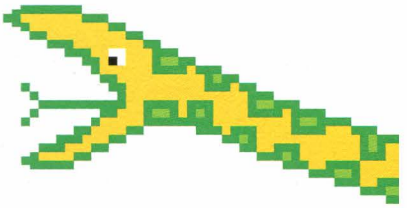
△ **Ubuntu**

Ubuntu Linux — бесплатная операционная система, похожая на Windows и Mac OS. О том, как установить Python на Ubuntu, читай на странице 91.



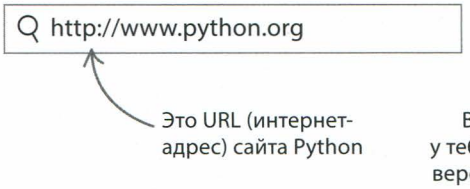
Python 3 и Windows

Прежде чем устанавливать Python 3 на Windows-компьютер, убедись, что родители не возражают против этого. Возможно, во время установки тебе также понадобится пароль администратора.



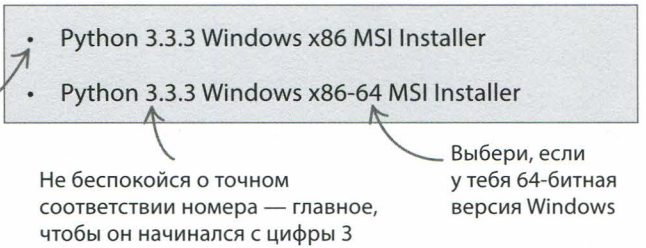
1 Зайди на сайт Python

Введи адрес сайта Python в строку браузера. Кликни Downloads, чтобы открыть страницу скачивания.



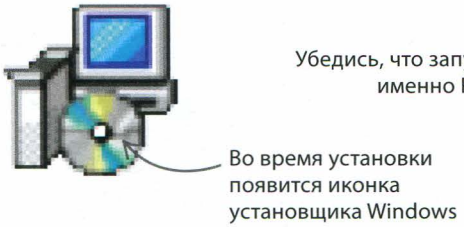
2 Скачай Python

Кликни по самой новой Windows-версии Python, номер которой начинается с цифры 3, — она будет где-то в начале списка.



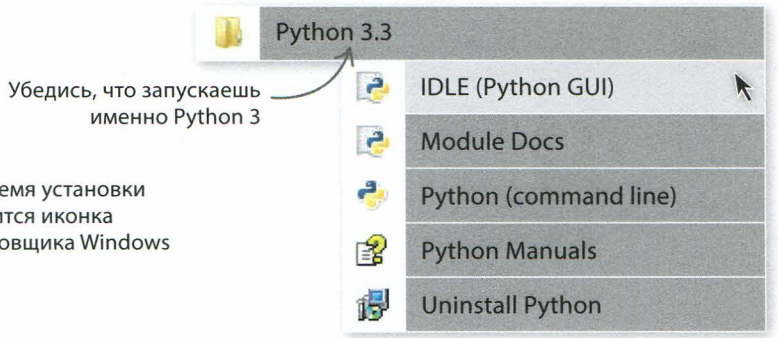
3 Установка

После того как скачается установочный файл, сделай по нему двойной клик, чтобы установить Python. Выбери «Установить для всех пользователей» и на каждый запрос кликай по кнопке «Далее».



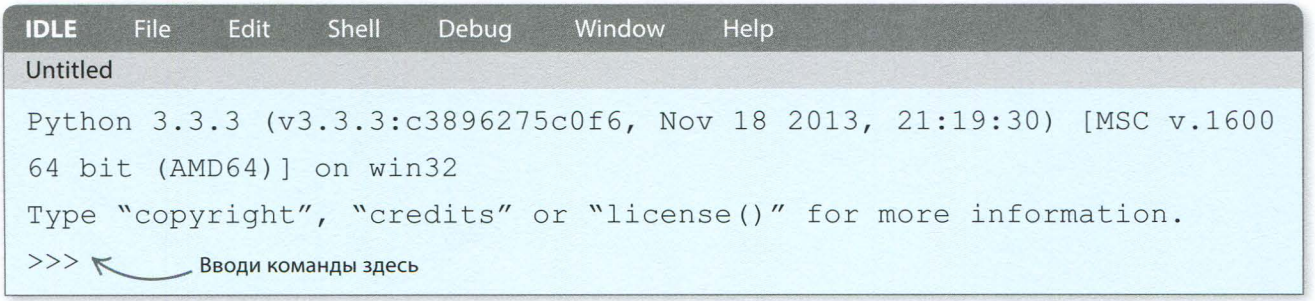
4 Запусти IDLE

Убедись, что программа установилась без ошибок. Для этого кликни по кнопке «Пуск» и выбери «Все программы», Python, IDLE.



5 Откроется окно Python

Появится окошко, примерно как на картинке ниже. В нем уже можно программировать, вводя команды после значка «>>>».



Python 3 и Mac OS

Прежде чем устанавливать Python 3 на компьютер с операционной системой Mac, убедись, что родители не возражают против этого. Возможно, во время установки тебе также понадобится пароль администратора.

1 Зайди на сайт Python

Введи адрес сайта Python в строку браузера. Кликни Download, чтобы открыть страницу скачивания.

http://www.python.org

2 Скачай Python

Узнай, какая версия Mac OS установлена на компьютере (см. с. 88), и выбери подходящую версию Python 3. Сохрани .dmg-файл на рабочий стол Mac OS.

Версия для новой Mac OS

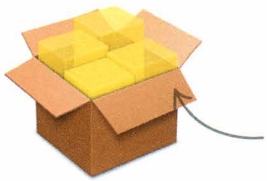
- Python 3.3.3 Mac OS X 64-bit... (for Mac OS X 10.6 and later)
- Python 3.3.3 Mac OS X 32-bit... (for Mac OS X 10.5 and later)

Не беспокойся о точном соответствии номера, главное, чтобы он начинался с цифры 3

Версия для большинства компьютеров с этой операционной системой

3 Установка

Сделай двойной клик по скачанному .dmg — откроется окно с несколькими файлами. Чтобы начать установку, сделай двойной клик по файлу Python.mpkg.

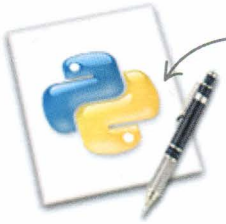


Установщик Python

Python.mpkg

4 Запусти IDLE

Во время установки кликай Next на каждый вопрос. После завершения установки открой на компьютере папку Applications, а в ней папку Python (убедись, что это Python 3, а не Python 2). Сделав двойной клик по иконке IDLE, проверь, что IDLE запускается.



Иконка IDLE

5 Откроется окно Python

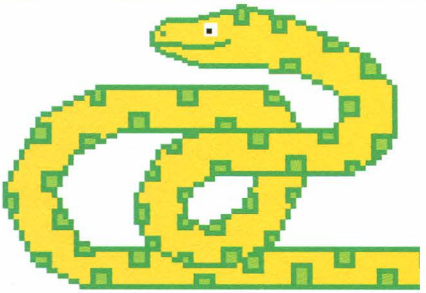
Появится окошко, примерно как на картинке ниже. Теперь уже можно программировать, вводя команды после значка «>>>».

IDLE File Edit Shell Debug Window Help

Untitled

```
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 16 2013, 23:39:35)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Python 3 и Ubuntu



Чтобы установить Python 3 на Linux-систему Ubuntu, необязательно пользоваться браузером — просто следуй дальнейшим инструкциям. Если используется другая версия Linux, попроси владельца компьютера установить для тебя Python 3.

1 Открой Ubuntu Software Center

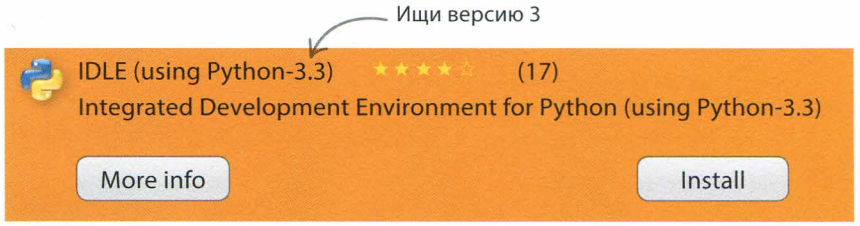
Найди иконку Ubuntu Software Center и сделай по ней двойной клик.

2 Введи в строке поиска Python

В правом верхнем углу ты увидишь строку поиска, введи там Python и нажми Enter.

3 Выбери IDLE и кликни Install

Ищи в списке строку IDLE (using Python). Выбери версию, номер которой начинается с цифры 3, и кликни Install.



4 Кликни по иконке Dash

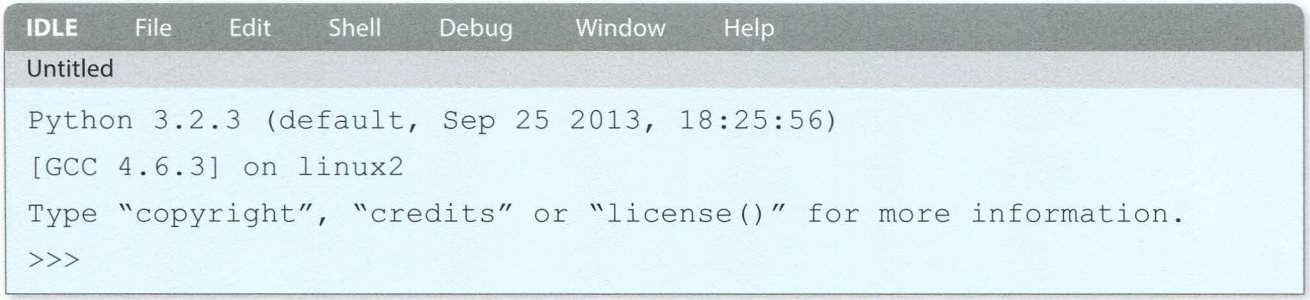
После окончания установки убедись, что программа работает. Кликни по иконке Dash в верхнем правом углу.

5 Запусти IDLE

Введи в строке поиска IDLE и сделай двойной клик по сине-желтой иконке IDLE (using Python 3)

6 Откроется окно Python

Появится окошко, примерно как на картинке ниже. Теперь уже можно программировать, вводя команды после значка «>>>».



Знакомство с IDLE

IDLE поможет тебе писать и выполнять программы на Python. Чтобы разобраться, как работает IDLE, введи эту простую программу, печатающую на экране сообщение.

СМОТРИ ТАКЖЕ
‹ 88–91 Установка Python
Разные окна 106–107 ›

Работа в IDLE

Чтобы научиться вводить, сохранять и выполнять Python-программы с помощью IDLE, выполни эти шаги.

1 Запусти IDLE

Запусти IDLE, выполнив инструкции для твоей операционной системы (см. с. 88–91). Откроется окно консоли. В этом окне отображается вывод программы (то, что она печатает) и ошибки.

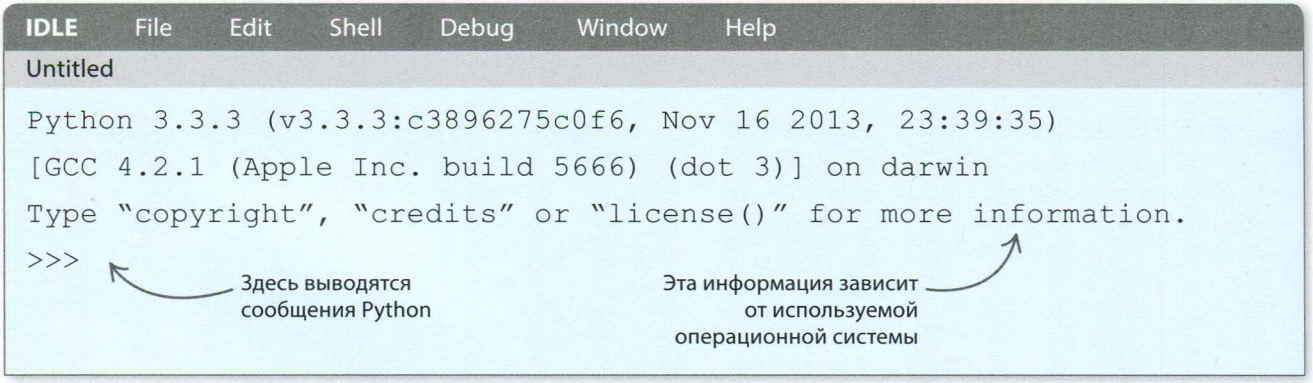
СОВЕТЫ ЭКСПЕРТА

Разные окна

В Python используются два разных окна — окно консоли и окно программы (см. с. 106–107). Чтобы отличать их, мы будем использовать разные цвета.

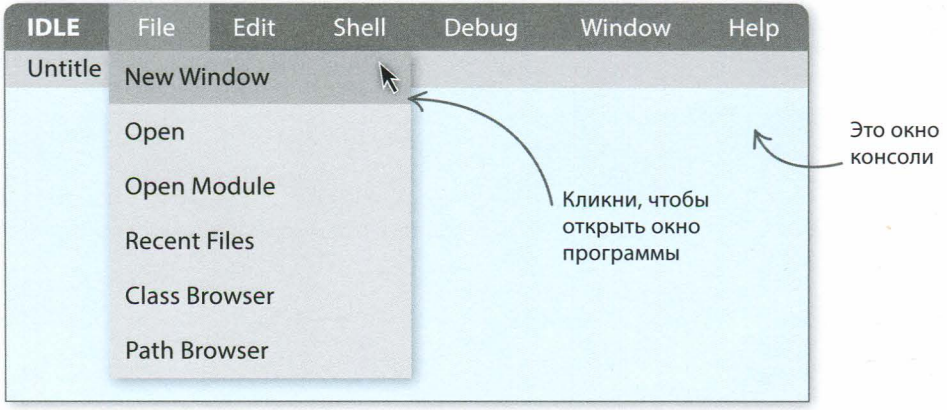
Окно консоли

Окно программы



2 Открой новое окно

Кликни по меню File вверху окна консоли и выбери New Window. Откроется окно программы.



3 Введи программу

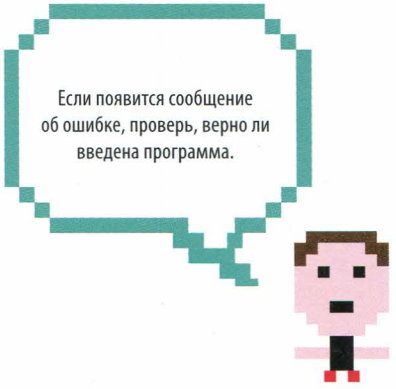
Введи в окне программы этот текст — команду для печати слов Hello World!

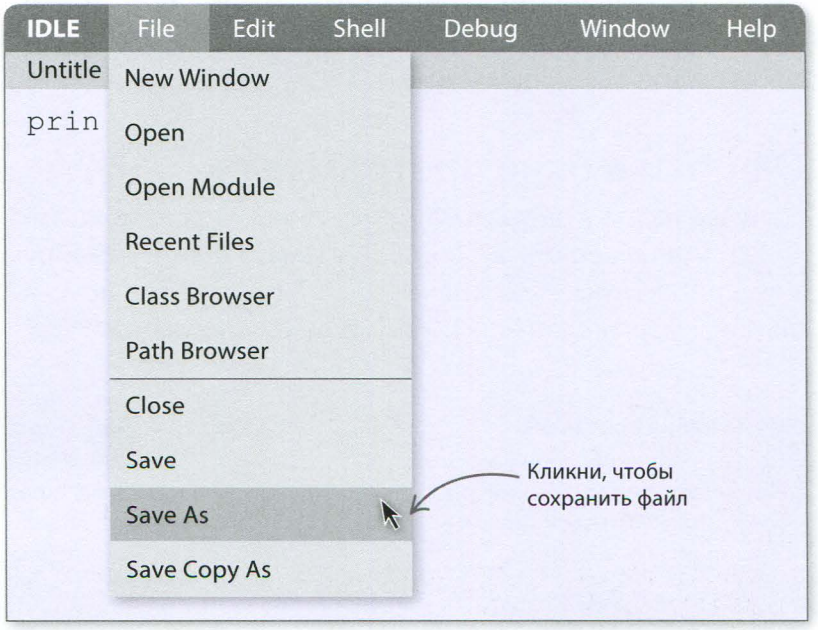
```
print('Hello World!')
```

Используй одинарные кавычки

4 Сохрани программу

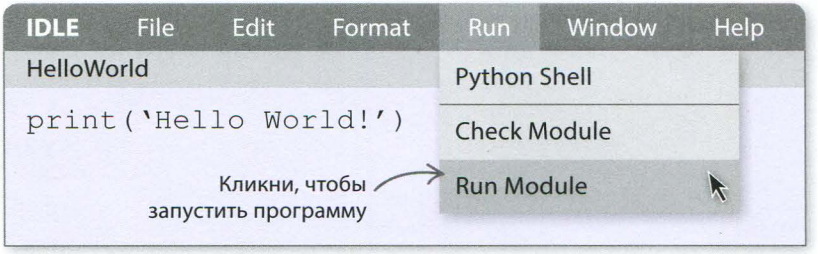
Кликни по меню File и выбери Save As. Введи имя файла HelloWorld и кликни Save.





5 Запусти программу

В окне программы кликни по меню Run и выбери Run Module. После этого программа запустится в окне консоли.



6 Результат в окне консоли

Проверь окно консоли — там должно появиться сообщение Hello World! Это твоя первая программа на Python!

```
>>>
Hello World!
>>>
```

Сообщение печатается без кавычек

ЗАПОМНИ

Как работает IDLE

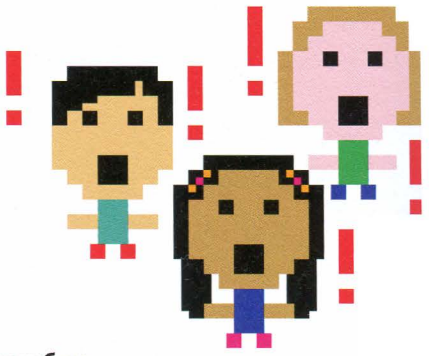
Всегда работай с IDLE так: напиши программу, сохрани ее и лишь потом запускай. Помни: программу нельзя запустить, пока она не сохранена, — при попытке это сделать появится предупреждение.

Enter code → Save → Run

Ошибки

СМОТРИ ТАКЖЕ
Ошибки и отладка 148–149 ❯
Что дальше? 176–177 ❯

Бывает, что программа сначала не запускается, но это всегда можно исправить. Когда что-то в тексте программы не так, Python отображает сообщение об ошибке с информацией о том, в чем проблема.

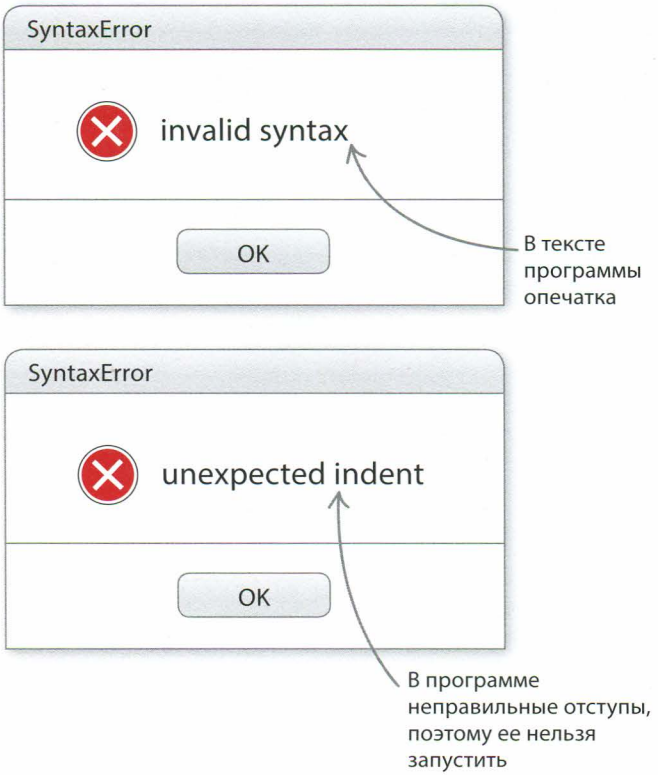


Ошибки в окне программы

Если запускать код из окна программы, может появиться всплывающее окно с сообщением об ошибке (например, SyntaxError). Это означает, что в программе есть ошибка, мешающая ее запуску, и ее нужно исправить.

1 Ошибка синтаксиса

Если появится окошко с сообщением SyntaxError, обычно это значит, что в тексте программы что-то записано неверно.



2 Подсветка ошибки

Кликни OK во всплывшем окне, чтобы вернуться к программе. Поверх строки с ошибкой или рядом с ней появится красная подсветка. Внимательно проверь эту строку.

```
print('Hello World!)
```

Здесь не хватает кавычки

Ошибка подсвечивается

СОВЕТЫ ЭКСПЕРТА

Классические ошибки

Некоторые ошибки встречаются в программах особенно часто. Обрати внимание на типичные проблемы.

Заглавные или строчные: регистр букв важен. Если ты напишешь Print вместо print, Python не поймет, что это за команда.

Одинарные и двойные кавычки: не путай разные виды кавычек. Открывающей кавычке должна соответствовать такая же закрывающая.

Минус и подчеркивание: не перепутай минус (–) и знак подчеркивания (_).

Разные скобки: у скобок различной формы, таких как (), {} и [], разное назначение. Не путай их и убедись, что открывающей скобке соответствует закрывающая того же типа.

Ошибки в окне консоли

Иногда ошибки печатаются красным цветом в окне консоли. Они тоже мешают запуску программы.

1 Ошибка названия

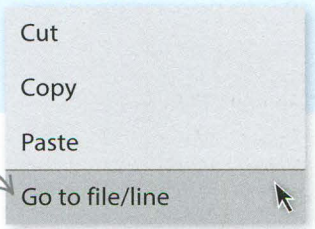
Ошибка NameError означает, что Python не понимает какое-то слово в программе. Если в окне консоли появилась эта ошибка, сделай правый клик по строке, которая начинается с File, и выбери Go to file/line.

Строка в коде (в окне программы), в которой найдена ошибка

```
>>>
Traceback (most recent call last):
    File "C:\PythonCode\errors.py", line 1, in <module>
        pront('Hello World!')
NameError: name 'pront' is not defined
```

Слово, которое Python не понимает

Кликни, чтобы подсветить строку с ошибкой в окне программы



2 Исправь ошибку

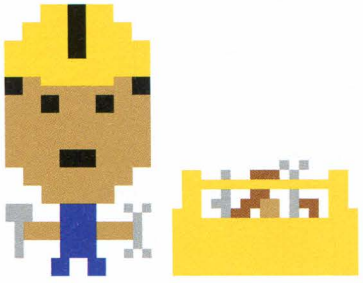
В окне программы подсвечена строка с ошибкой — вместо print введено слово pront. Отредактируй это слово в тексте программы, чтобы исправить ошибку.

```
pront('Hello World!')
```

Исправь на print

Поиск ошибок

Пользуйся подсказками с этой и предыдущей страниц, чтобы найти в тексте программы строку с ошибкой, и внимательно проверь эту строку. Ответь на вопросы из списка справа, чтобы выяснить, в чем проблема.



▷ **Если что-то не так**

Вот список вопросов, которые помогут найти ошибку в программе.

ОХОТА ЗА ОШИБКАМИ	
Проверь свой код по следующим пунктам	✓
Нужный ли текст программы скопирован из книги?	✓
Все ли слова в тексте написаны правильно?	✓
Если программа что-то печатает, заключены ли сообщения в одинарные кавычки?	✓
Нет ли в начале строки лишних пробелов? Отступы очень важны в Python.	✓
Все ли в порядке в строках сразу перед и после выделенной строки с ошибкой? Бывает, что проблема там.	✓
Проверял ли твой код кто-то еще? Порой другой человек может заметить свежим взглядом упущенную ошибку.	✓
Используешь ли ты Python 3, а не Python 2? Программы, составленные для Python 3, могут не работать на Python 2.	✓

ПРОЕКТ 4

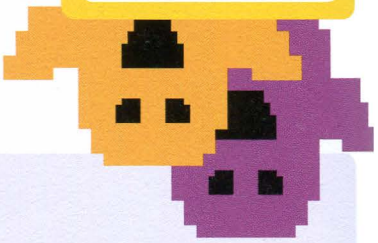
Дом с привидениями

Разбирая эту несложную игру, мы подчеркнем моменты, на которые стоит обращать внимание при программировании на Python. Введи код программы и запусти ее — сумеешь ли ты спастись из дома с привидениями?

СМОТРИ ТАКЖЕ

Разбор «Дома с привидениями» **98–99 ❯**

Выполнение программ **100–101 ❯**



1 Запустив IDLE, открой новое окно с помощью меню File. Расположи оба окна так, чтобы видеть их одновременно. Введи этот текст в окно программы и сохрани ее под именем ghostgame.

```
# Ghost Game
from random import randint
print('Ghost Game')
feeling_brave = True
score = 0
while feeling_brave:
    ghost_door = randint(1, 3)
    print('Three doors ahead...')
    print('A ghost behind one.')
    print('Which door do you open?')
    door = input('1, 2, or 3?')
    door_num = int(door)
    if door_num == ghost_door:
        print('GHOST!')
        feeling_brave = False
    else:
        print('No ghost!')
        print('You enter the next room.')
        score = score + 1
print('Run away!')
print('Game over! You scored', score)
```

Используй одинарные кавычки

Используй заглавные буквы только там, где они показаны

Это знак подчеркивания, а не минус

Убедись, что здесь стоит двоеточие

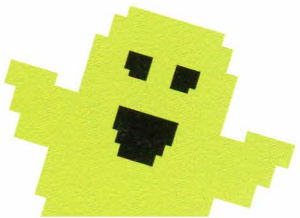
Перед этими строками должен быть отступ в 4 пробела. Если отступ не появился автоматически, проверь, стоит ли после feeling_brave двоеточие

Здесь должно быть 2 знака равенства

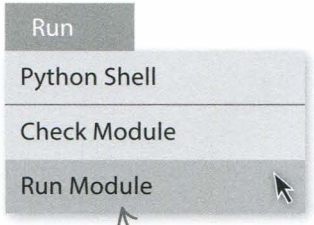
При вводе текста тут будет отступ в 8 пробелов, и его надо будет уменьшить до 4 пробелов

Здесь убери все пробелы

score нужно вводить без кавычек



2 Внимательно набрав текст программы, открой меню Run и выбери Run Module. Перед этим программу нужно сохранить.



Выбери Run Module из меню Run окна программы

3 Игра начнется в окне консоли. За одной из трех дверей скрывается привидение. Какую дверь ты выберешь? Напечатай 1, 2 или 3 и нажми Enter.

```
Ghost Game
Three doors ahead...
A ghost behind one.
Which door do you open?
1, 2, or 3?
```

Введи свой вариант ответа

4 Твоя задача в игре — выбрать дверь, за которой нет привидения. Угадав, ты перейдешь в следующую комнату, где игра продолжится.

```
Ghost Game
Three doors ahead...
A ghost behind one.
Which door do you open?
1, 2, or 3?3
No ghost!
```

Число, которое ты введешь, появится здесь

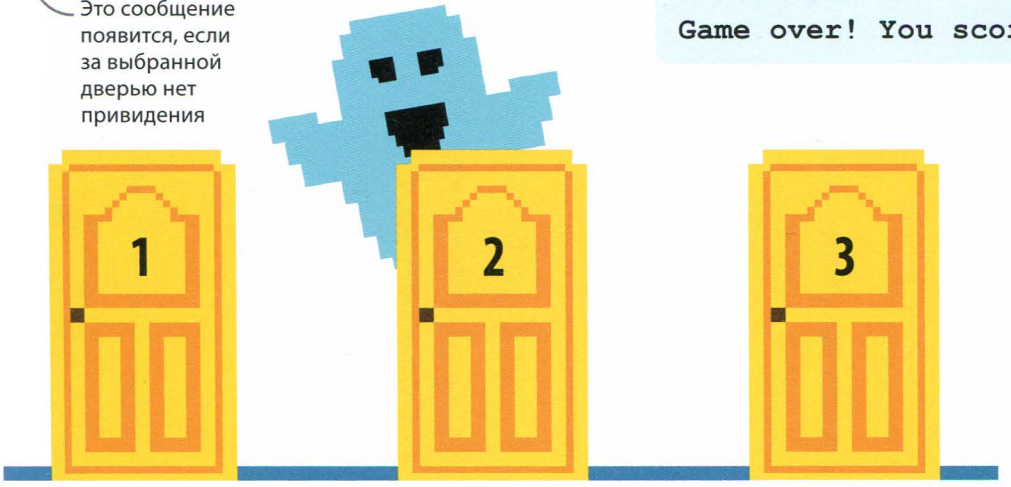
Это сообщение появится, если за выбранной дверью нет привидения

5 Если тебе не повезло и за дверью прячется привидение, игра закончится. Запусти программу еще раз — сумеешь ли ты превзойти свой прошлый результат?

```
Ghost Game
Three doors ahead...
A ghost behind one.
Which door do you open?
1, 2, or 3?2
GHOST!
Run away!
Game over! You scored 0
```

Это сообщение появится, если за дверью есть привидение

Это счет игры — количество пройденных комнат



Разбор «Дома с привидениями»

СМОТРИ ТАКЖЕ

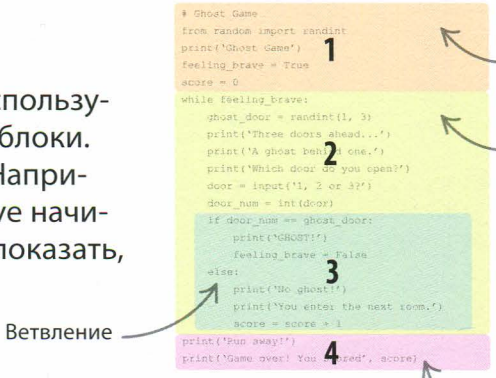
❮ **96–97** Дом с привидениями

Выполнение программ **100–101** ❯

В этой игре показаны некоторые основные возможности Python. Стоит разобрать программу по частям, чтобы понять, как она устроена и что делает каждая из частей.

Структура программы

В Python пробелы в начале строк используются, чтобы объединять команды в блоки. Эти пробелы называют отступами. Например, строки после while feeling_brave начинаются с четырех пробелов, чтобы показать, что они внутри основного цикла.



◁ Схема программы
На этой диаграмме показана структура игры. Далее мы подробно разберем пронумерованные части.

Это комментарий. Во время игры он нигде не отображается

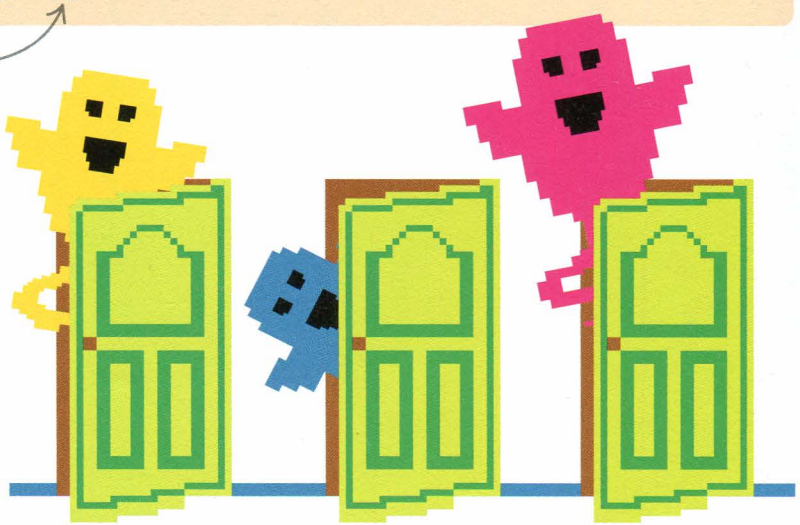
1 Подготовка к игре
Эти команды выполняются только один раз — в начале игры. Они печатают название игры, настраивают переменные и команду randint.

```
# Ghost Game
from random import randint
print('Ghost Game')
feeling_brave = True
score = 0
```

Настраивает команду randint, которая генерирует случайные числа

Команда print печатает текст в игре

Сбрасывает счет игры в 0



СОВЕТЫ ЭКСПЕРТА

Вводи текст внимательно

Вводи текст Python-программ очень внимательно. Если ты пропустишь запятую, кавычку или скобку, программа не будет нормально работать. Также важно не путать заглавные и строчные буквы и следить за величиной отступов.

2 Основной цикл

В этом цикле печатаются правила игры и запрашивается ответ игрока. Пока игрок не столкнулся с привидением, цикл продолжается. Если же за дверью привидение, переменная feeling_brave принимает значение False (Ложь) и цикл завершается.

3 Ветвление

Программа выполняет разные действия в зависимости от того, есть ли за выбранной дверью привидение. Если игрок встретил привидение, переменная feeling_brave принимает значение False, а если нет, счет игры увеличивается на единицу.

```
while feeling_brave:
    ghost_door = randint(1, 3)
    print('Three doors ahead...')
    print('A ghost behind one.')
    print('Which door do you open?')
    door = input('1, 2 or 3?')
    door_num = int(door)
    if door_num == ghost_door:
        print('GHOST!')
        feeling_brave = False
    else:
        print('No ghost!')
        print('You enter the next room.')
        score = score + 1
```

Выбирает случайное число от 1 до 3

Команда print печатает текст на экране

Запрашивает выбор игрока

Эта ветвь выполняется, если за выбранной дверью есть привидение

Если привидения нет, игрок увидит это сообщение

Счет увеличивается на 1 каждый раз, когда игрок входит в новую комнату, не встретив привидения

4 Конец игры

Эта часть выполняется только один раз, когда игрок встретит привидение. Благодаря отсутствию отступов Python знает, что эти команды вне цикла.

```
print('Run away!')
print('Game over! You scored', score)
```

Дает игроку совет бежать прочь от привидения

Счет игры — это переменная, значение которой меняется в зависимости от числа пройденных комнат

ЗАПОМНИ

Твои успехи

Поздравляем с твоей первой игрой на Python! Скоро ты узнаешь обо всех этих командах еще больше, но и сейчас сделано немало.

Ввод программы: тобой введена и сохранена первая Python-программа.

Запуск программы: теперь ты знаешь, как запустить Python-программу.

Структурирование программы: программы структурируются с помощью отступов.

Переменные: использование переменной для хранения счета игры.

Отображение текста: печать сообщений на экране.

Выполнение программ

Прежде чем продолжить изучение Python, важно разобраться, как работают программы. Основы программирования, изученные для Scratch, годятся и для Python.

СМОТРИ ТАКЖЕ

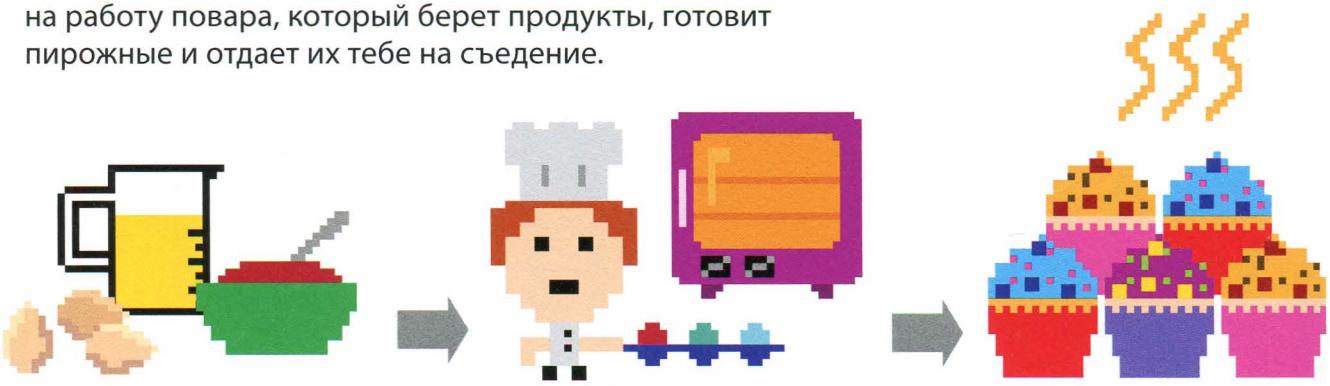
❮ 30–31 Цветные блоки и скрипты

Простые команды 102–103 ❯

Команды посложнее 104–105 ❯

От ввода до вывода

Программа принимает входные данные (ввод), обрабатывает их и затем возвращает результат (вывод). Это похоже на работу повара, который берет продукты, готовит пирожные и отдает их тебе на съедение.



Ввод

- Команда input
- Клавиатура
- Мышка

Обработка

- Переменные
- Вычисления
- Циклы
- Ветвление
- Функции

Вывод

- Команда print
- Экран
- Графика

△ **Выполнение программ на Python**

В Python клавиатура и мышка используются для ввода информации, которая затем обрабатывается с помощью циклов, ветвления и переменных. После этого результат отображается на экране.

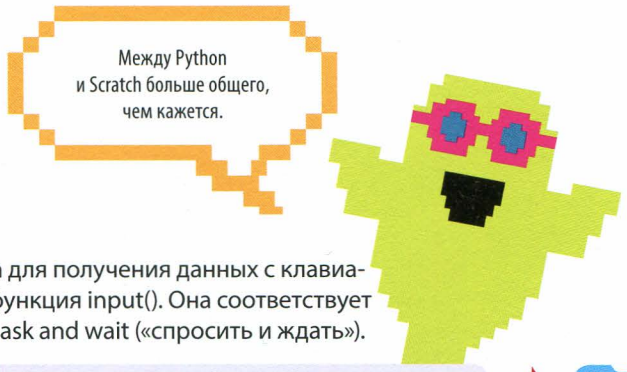
Взгляд на «Дом с привидениями» глазами Scratch

Программы выполняются одинаковым образом в большинстве языков программирования. Вот примеры ввода, обработки и вывода данных в игре «Дом с привидениями» и то, как это могло бы выглядеть в Scratch.

СОВЕТЫ ЭКСПЕРТА

Один скрипт за раз

Между Scratch и Python есть важное различие. В Scratch множество скриптов могут работать одновременно, тогда как в Python программа состоит лишь из одного скрипта.



1 Ввод

В Python для получения данных с клавиатуры служит функция input(). Она соответствует Scratch-блоку ask and wait («спросить и ждать»).

```
door = input('1, 2 or 3?')
```

Вопрос отображается на экране

Вопрос в блоке Scratch

Scratch-блок ask and wait («спросить и ждать»)

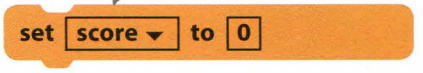
2 Обработка

Чтобы следить за счетом игры, используется переменная, а функция randint выбирает случайную дверь. В Scratch для этого служат разные блоки.

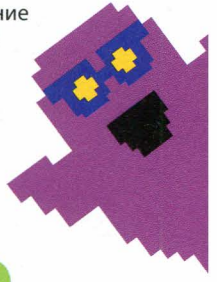
```
score = 0
```

Устанавливает переменную score в 0

Этот Scratch-блок устанавливает значение переменной score в 0

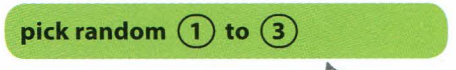


Scratch-блок set score 0 («задать score значение 0»)



```
ghost_door = randint(1, 3)
```

Выбирает случайное число от 1 до 3



Scratch-блок pick random («выдать случайное»)

Этот Scratch-блок выбирает случайное число

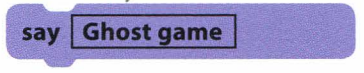
3 Вывод

В Python для печати на экране служит функция print(), а в Scratch — блок say («сказать»).

```
print('Ghost game')
```

Печатает на экране Ghost game

Отображает баллон со словами Ghost game



Scratch-блок say («сказать»)

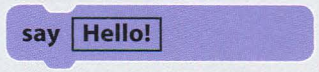
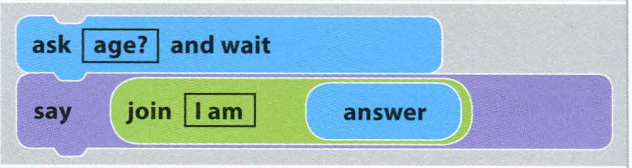
Простые команды

На первый взгляд Python может показаться сложным, особенно по сравнению со Scratch. Однако эти два языка не такие уж и разные. Вот список соответствий между основными командами в Python и Scratch.

СМОТРИ ТАКЖЕ

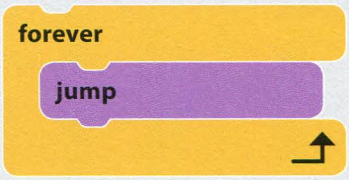
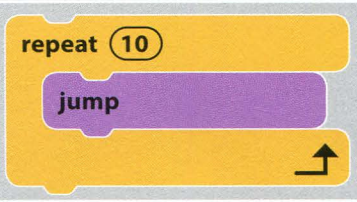
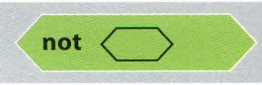
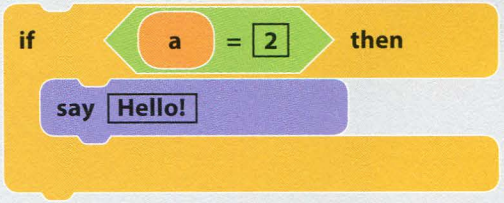
‹ 86–87 Что такое Python?

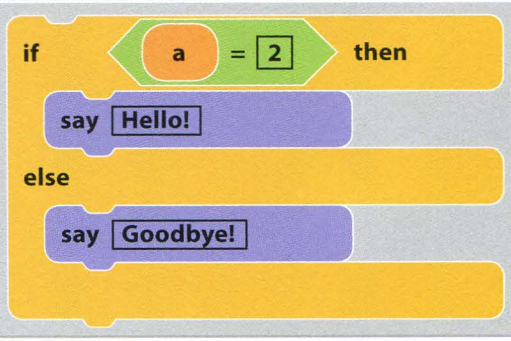
Команды посложнее **104–105 ›**

Команда	Python 3	Scratch 2.0
Запустить программу	Меню Run или клавиша F5 в окне программы	
Остановить программу	CTRL-C на клавиатуре (в окне консоли)	
Вывод текста	`print('Hello!')`	
Задать переменной числовое значение	`magic_number = 42`	
Задать переменной строковое значение	`word = 'dragon'`	
Считать текст с клавиатуры в переменную	`age = input('age?')` `print('I am ' + age)`	

Прибавить число к переменной	`cats = cats + 1`
	or
	`cats += 1`
		

Сложение	`a + 2`	
Вычитание	`a - 2`	
Умножение	`a * 2`	
Деление	`a / 2`	

Команда	Python 3	Scratch 2.0
Бесконечный цикл	`while True:` `    jump()`	
Повторить 10 раз	`for i in range (10):` `    jump()`	
Равно?	`a == 2`	
Меньше?	`a < 2`	
Больше?	`a > 2`	
Булево «не»	`not`	
Булево «или»	`or`	
Булево «и»	`and`	
Ветвление «если, то»	`if a == 2:` `    print('Hello!')`	

Ветвление «если, то, иначе»	`if a == 2:`
	`    print('Hello!')`
	`else:`
	`    print('Goodbye!')`
		

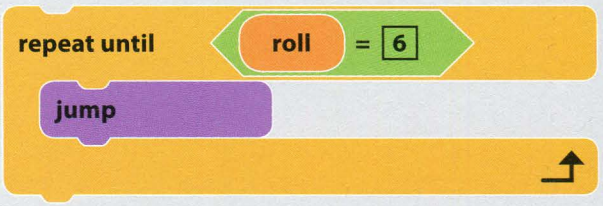
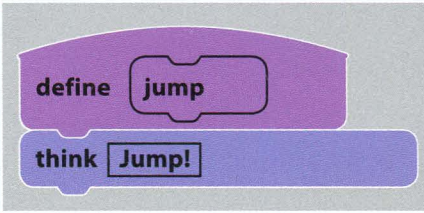
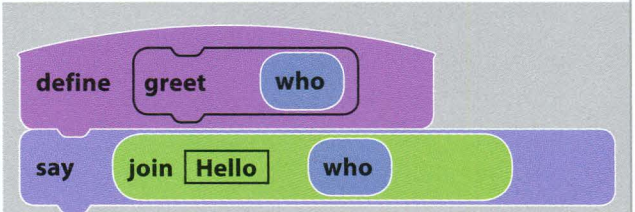
Команды посложнее

В Python можно делать и более сложные вещи из тех, что возможны в Scratch, например сложные циклы, действия со строками и списками, рисование с помощью черепашки.

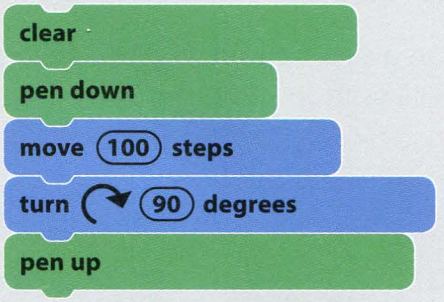
СМОТРИ ТАКЖЕ

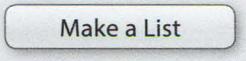
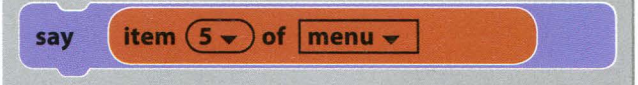
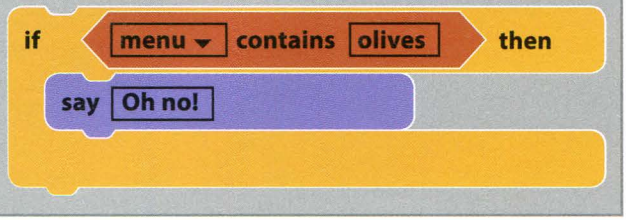
❮ 86–87 Что такое Python?

❮ 102–103 Простые команды

Команда	Python 3	Scratch 2.0
Цикл с условием	`while roll != 6:` `    jump()`	
Ожидание	`from time import sleep` `sleep(2)`	
Случайные числа	`from random import randint` `roll = randint(1, 6)`	
Определить функцию или подпрограмму	`def jump():` `    print('Jump!')`	
Вызвать функцию или подпрограмму	`jump()`	
Определить функцию или подпрограмму с параметрами	`def greet(who):` `    print('Hello ' + who)`	
Вызвать функцию или подпрограмму	`greet('chicken')`	

Команда	Python 3	Scratch 2.0

Черепашья графика	`from turtle import *`
	`clear()`
	`pendown()`
	`forward(100)`
	`right(90)`
	`penup()`
		

Склеить строки	`print(greeting + name)`	
Получить один символ из строки	`name[0]`	
Длина строки	`len(name)`	
Создать пустой список	`menu = list()`	
Добавить элемент к списку	`menu.append(thing)`	
Число элементов в списке	`len(menu)`	
Значение 5-го элемента списка	`menu[4]`	
Удалить 2-й элемент списка	`del menu[1]`	
Есть ли элемент в списке?	`if 'olives' in menu:` `  print('Oh no!')`	

Разные окна

В IDLE есть два вида окон. В окне программы можно писать и сохранять программный код, а в окне консоли — сразу выполнять команды Python.

СМОТРИ ТАКЖЕ

❮ **92–93** Знакомство с IDLE

❮ **96–97** Проект «Дом с привидениями»

Окно программы

Окно программы служит для того, чтобы писать и редактировать текст программы. Программу можно ввести, сохранить, запустить и посмотреть на результат в окне консоли.

▽ **Запуск программ**
Вот порядок запуска Python-программ. Прежде чем запускать программу, ее обязательно нужно сохранить.

Ввод программы → Сохранение → Запуск → Результат

1 Введи код в окне программы
Введи этот код в окне программы, сохрани его и выбери Run module из меню Run, чтобы запустить программу.

```
a = 10
b = 4
print(a + b)
print(a - b)
```

Задать *a* значение 10

Задать *b* значение 4

Команда print печатает результаты расчетов

2 Результат в окне консоли
Когда программа запущена, ее вывод (результат) отображается в окне консоли.

```
>>>
14
6
```

Результаты расчетов в окне консоли

Окно консоли

Python также может выполнять команды, введенные в окне консоли. Эти команды выполняются, как только их напечатаешь, и результат виден сразу.

```
>>> a = 10
>>> b = 4
>>> a + b
14
>>> a - b
6
```

Первые две команды не отображают результата, они лишь присваивают значения переменным

Результат отображается сразу

◁ **Команды и результат**
В окне консоли отображаются и команды, и результат их выполнения. Если вводить команды в окне консоли, проще понять, к какой команде какой результат относится.

△ **Исследуй разные команды**
В окне консоли результат появляется сразу — это очень удобно, чтобы пробовать разные команды, выясняя, что они делают.

Лаборатория Python

В окне консоли сразу видно, что делают разные команды Python, в том числе команды рисования. Для рисования на экране в Python, как и в Scratch, используется черепашка.

Загружает команды для работы с черепашкой

```
>>> from turtle import *
>>> forward(100)
>>> right(120)
>>> forward(100)
```

Двигает черепашку вперед

◁ **Вводим команды**
Введи эти команды в окне консоли. Они выполнятся одна за другой, по очереди. Черепашка будет двигаться, оставляя за собой линию.

◁ **Черепашья графика**
Сможешь нарисовать другие фигуры — например квадрат или пятиугольник? Чтобы очистить область рисования, введи в окне консоли clear().

Какое окно лучше?

Какое из окон IDLE удобнее: окно программы или окно консоли? Это зависит от того, какую программу ты пишешь и нужно ли запускать ее несколько раз.

СОВЕТЫ ЭКСПЕРТА

Цвета в тексте программы

IDLE раскрашивает текст программы в различные цвета. Эти цвета — подсказки, как Python воспринимает разные части текста.

◁ **Встроенные функции**
Команды Python, например print, — фиолетового цвета.

◁ **Строки в кавычках**
Зеленым цветом помечены строки. Если скобки вокруг текста тоже зеленые, где-то не хватает кавычки.

◁ **Большинство символов**
Бóльшая часть кода — черного цвета.

◁ **Вывод**
Результаты выполнения команд отображаются синим цветом.

◁ **Ключевые слова**
Ключевые слова языка, такие как if и else, оранжевого цвета. Python не позволяет использовать эти слова как имена переменных.

◁ **Ошибки**
Красным цветом помечаются ошибки в окне консоли.

▷ **Окно программы**
Окно программы хорошо подходит для больших программ, которые нужно сохранять и редактировать. Это проще, чем печатать команды каждый раз, когда они понадобятся. Однако программу придется сохранять и запускать с помощью меню Run.

◁ **Окно консоли**
Окно консоли идеально для экспериментов — например, если нужно разобраться, что делает команда. Кроме того, консоль можно использовать как калькулятор. Однако, если какие-то команды нужно повторять, лучше использовать окно программы.

Переменные в Python

Переменные нужны, чтобы запоминать фрагменты данных в программе. Они похожи на подписанные ящички для хранения данных.

СМОТРИ ТАКЖЕ

- Типы данных **110–111** ❯
- Вычисления в Python **112–113** ❯
- Строки в Python **114–115** ❯
- Ввод и вывод **116–117** ❯
- Функции **130–131** ❯

Создание переменной

С помощью знака «=» число или строку можно поместить в переменную — это называется заданием значения переменной. Введи эти команды в окне консоли.

Имя переменной — Значение переменной

```
>>> bones = 3
```

△ **Числовое значение**
Чтобы задать переменной числовое значение, введи имя переменной, знак «=» и затем число.

Имя переменной — Значение переменной

```
>>> dogs_name = 'Bruno'
```

△ **Строковое значение**
Чтобы задать строковое значение переменной, введи имя переменной, знак «=» и затем строку в кавычках.

ЗАПОМНИ

Переменные в Scratch

Команда задания значения переменной в Python соответствует этому блоку Scratch. Однако в Python для создания переменной не нужно нажимать кнопки: она будет создана автоматически при задании ее значения.

Scratch-блок задания значения переменной

Печать переменной

Команда print нужна, чтобы печатать текст на экране (и принтер тут ни при чем). С помощью этой команды можно узнать значение переменной.

```
>>> print(bones)
3
```

Имя переменной

△ **Печать числа**
В переменной bones («кости») — число 3, и это значение появится в окне консоли.

```
>>> print(dogs_name)
Bruno
```

Кавычки здесь не нужны

△ **Печать строки**
В переменной dogs_name («имя собаки») — строка, так что в окне консоли тоже появится строка, причем без кавычек.

Изменение значения переменной

Чтобы изменить значение переменной, просто задай ей новое значение. В переменной gifts («подарки») — число 2. После задания нового значения оно поменяется на 3.

```
>>> gifts = 2
>>> print(gifts)
2
>>> gifts = 3
>>> print(gifts)
3
```

Изменяет значение переменной

Использование переменных

С помощью знака «=» можно присвоить одной переменной значение другой. Так, если в переменной rabbits («кролики») — количество кроликов, можно присвоить ее значение переменной hats («шляпы»), чтобы у каждого кролика было по шляпе.

1 Задаем значения

Эти команды дают переменной rabbits значение 5, затем дают это же значение переменной hats.

Имя переменной

Значение переменной

```
>>> rabbits = 5
>>> hats = rabbits
```

Теперь hats имеет то же значение, что и rabbits

2 Печатаем значения

Чтобы вывести на экран значения двух переменных, укажи их в скобках после команды print, поставив между ними запятую. Значение hats и rabbits — число 5.

```
>>> print(rabbits, hats)
5 5
```

Поставь после запятой пробел

3 Изменяем значение rabbits

Если изменить значение rabbits, значение hats останется прежним — оно изменится, только если задать для hats новое значение.

```
>>> rabbits = 10
>>> print(rabbits, hats)
10 5
```

Дает rabbits новое значение

Значение hats не меняется

СОВЕТЫ ЭКСПЕРТА

Имена переменных

Существуют правила, как можно и как нельзя называть переменные.

Любые буквы и цифры можно использовать.

Имя не может начинаться с цифры.

Знаки, такие как -, /, # или @, использовать нельзя.

Пробелы использовать нельзя.

Знак подчеркивания (_) можно использовать вместо пробела.

Заглавные и строчные буквы различаются: Dogs и dogs в Python — две разные переменные.

Нельзя называть переменные именами команд, например print.

Типы данных

В Python используются данные разных типов. Обычно Python сам понимает, какой тип использовать, но иногда нужно вручную менять один тип данных на другой.

СМОТРИ ТАКЖЕ

Вычисления в Python **112–113 ❯**

Строки в Python **114–115 ❯**

Принятие решений **118–119 ❯**

Списки **128–129 ❯**

Числа

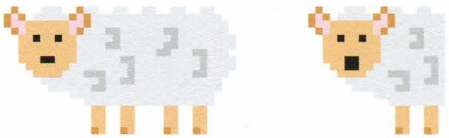
В Python есть два типа числовых данных: целые (int), то есть числа без дробной части, и вещественные (float) — дробные числа с десятичной точкой. Целые числа хороши для подсчета чего-либо, а вещественные — для измерения таких свойств, как вес.

```
>>> sheep = 1
>>> print(sheep)
1
```

Целое число

△ **Целые**

Целое — это число без десятичной точки, например 1 в переменной sheep («овца»).



```
>>> sheep = 1.5
>>> print(sheep)
1.5
```

1.5 — это вещественное число

△ **Вещественные**

Вещественные числа записывают с десятичной точкой, например 1.5 (= 1,5). Для подсчета количества их обычно не используют.

Строки

В Python, так же как и в Scratch, строкой называют фрагмент текста. Строки могут содержать буквы, числа и символы (например, точки и запятые). Обычно их записывают в одинарных кавычках.

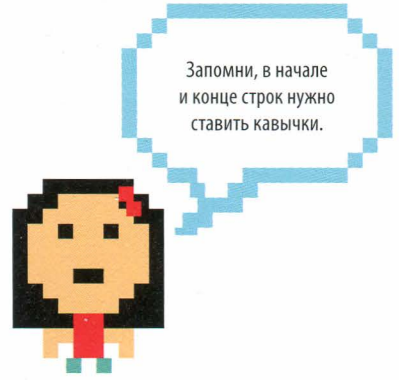
▷ **Использование строк**

Чтобы дать переменной строковое значение, укажи текст в одинарных кавычках.

Строка в кавычках

```
>>> a = 'Coding is fun!'
>>> print(a)
Coding is fun!
```

Печать значения переменной *a*



Булевы значения

Булев тип в Python может принимать одно из двух значений — True («Истина») или False («Ложь»). Эти слова надо писать с заглавной буквы.

Кавычки не нужны

▷ True

Если переменной дать значение True, это будет переменная булева типа.

```
>>> a = True
>>> print(a)
True
```

Печать булевой переменной

▷ False

Если переменной дать значение False, это тоже будет переменная булева типа.

```
>>> a = False
>>> print(a)
False
```

Печать булева значения

СОВЕТЫ ЭКСПЕРТА

Как узнать тип

В Python много типов данных. Чтобы узнать тип чего-либо, можно воспользоваться командой type.

Команда type

```
>>> type(24)
<class 'int'>
>>> type(24.3)
<class 'float'>
>>> type('24')
<class 'str'>
```

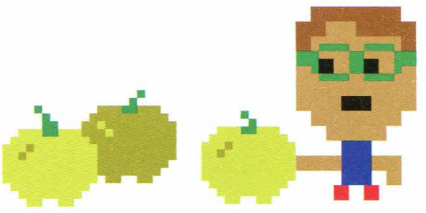
24 — целое (int)

24.3 — вещественное (float)

'24' — строка («str»), потому что в кавычках

Преобразование типов данных

В переменных могут храниться значения любого типа. Однако не все типы данных в программе совместимы. Порой один тип нужно преобразовать в другой, иначе возникнет ошибка.



Имя переменной

Строка с текстом вопроса

▷ Смешение типов

Команда input всегда возвращает строку, даже если было введено число. В этом примере при попытке прибавить к строке apple число возникнет ошибка.

```
>>> apple = input('Enter number of apples ')
Enter number of apples 2
>>> print(apple + 1)
TypeError
```

Попытка прибавить к apple («яблоко») единицу

Программа выдаст ошибку, потому что Python не знает, как сложить число и строку

▷ Преобразование типа

Команда int() преобразует строку в целое число.

```
>>> print(int(apple) + 1)
3
```

Теперь программа работает и выводит результат

Переменная преобразуется к типу «целое», а к нему можно прибавлять числа

Вычисления в Python

В Python можно выполнять разные математические операции, например складывать, вычитать, умножать и делить. В вычислениях часто используют переменные.

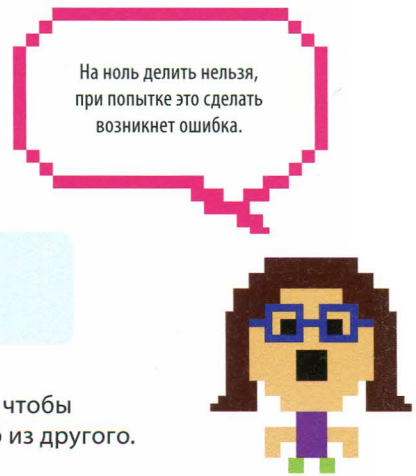
СМОТРИ ТАКЖЕ

❮ 52–53 Вычисления

❮ 108–109 Переменные в Python

Простые вычисления

Простые выражения можно вычислять, вводя их в окне консоли. Для этого не нужна команда print() — Python напечатает результат и так. Попробуй ввести в окне консоли эти примеры.




```
>>> 12 + 4
16
```

В окне консоли тут же появится результат

△ **Сложение**
Складывай числа с помощью знака «+».

Результат появится после нажатия Enter

```
>>> 12 - 4
8
```

△ **Вычитание**
Используй знак «–», чтобы вычесть одно число из другого.

В программировании умножение обозначается знаком «*»

```
>>> 12 * 4
48
```

△ **Умножение**
Перемножай числа с помощью знака «*».

```
>>> 12 / 4
3.0
```

Деление в Python дает вещественный результат (число с десятичной точкой)

△ **Деление**
Используй знак «/», чтобы разделить одно число на другое.

Использование скобок

С помощью скобок можно указать, какую часть выражения вычислять первой. Python вычислит то, что в скобках, и лишь потом займется оставшейся частью выражения.

Сначала вычислит, что 6 + 5 = 11, затем умножит 11 на 3

```
>>> (6 + 5) * 3
33
```

△ **Сначала сложение**
Скобки в этом выражении указывают Python, что сперва надо выполнить сложение.

Сперва вычисляет, что 5 * 3 = 15, затем складывает 6 и 15

```
>>> 6 + (5 * 3)
21
```

Другой результат

△ **Сначала умножение**
Здесь скобки используются, чтобы сперва выполнить умножение.

Результат в переменной

В выражениях можно использовать переменные с числовыми значениями. Если же присвоить выражение переменной, там окажется не само выражение, а его результат.

1 Простое сложение

Эта программа складывает значения переменных ants и spiders («муравьи» и «пауки») и помещает результат в переменную bugs («насекомые»).

```
>>> ants = 22
>>> spiders = 35
>>> bugs = ants + spiders
>>> print(bugs)
57
```

Складывает значения двух переменных

Печатает значение bugs («насекомые»)

2 Другое значение переменной

Измени значение переменной spiders. Снова сложи переменные и помести результат в переменную bugs.

```
>>> ants = 22
>>> spiders = 18
>>> bugs = ants + spiders
>>> print(bugs)
40
```

Новое значение в spiders

Сложение переменных

Результат изменился

3 Без присваивания

Если не присваивать результат переменной bugs, ее значение не изменится, что бы ни происходило с переменными ants и spiders.

```
>>> ants = 11
>>> spiders = 17
>>> print(bugs)
40
```

Печать значения bugs

Результат не изменился (по-прежнему 18 + 22)

Случайные числа

Чтобы получить случайное число, сперва загрузи в Python функцию randint. Для этого используй команду import. Функция randint() уже запрограммирована на выбор случайного целого числа.

```
>>> from random import randint
>>> randint(1, 6)
3
```

Загружает функцию randint()

Выбирает случайное число от 1 до 6

Выпало число 3

△ Случайное число

Функция randint() выбирает случайное число в диапазоне от первого до второго числа в скобках. В этом примере randint(1, 6) выберет значение от 1 до 6.

ЗАПОМНИ

Блок «выдать случайное»

Функция randint работает так же, как Scratch-блок pick random («выдать случайное»). В Scratch наименьшее и наибольшее значения вводят в лунках параметров блока. В Python эти числа записывают в скобках, разделяя запятой.

△ Целые числа

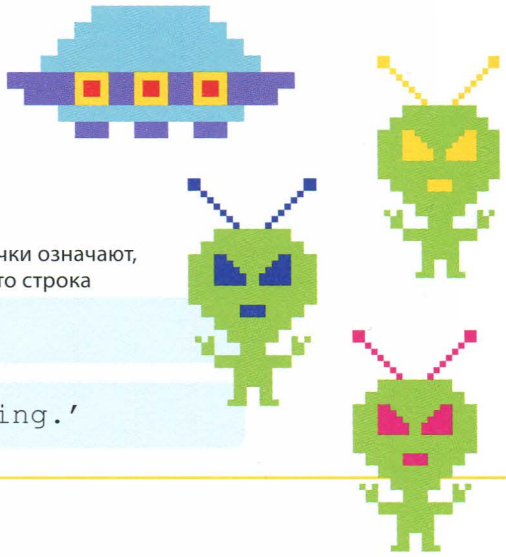
И Python-функция randint(), и Scratch-блок pick random возвращают целое число — результат никогда не бывает вещественным.

Строки в Python

Python отлично подходит для работы с фрагментами текста. Строки (последовательности символов) можно склеивать вместе или вырезать из них выбранные части.

СМОТРИ ТАКЖЕ

- ❮ 54–55 Строки и списки
- ❮ 110–111 Типы данных



Создание строки

Строка может состоять из букв, чисел, знаков или пробелов — то есть из символов. Строки можно хранить в переменных.

Кавычки означают, что это строка

```
>>> a = 'Run! '
>>> b = 'Aliens are coming.'
```

▷ **Строки в переменных**
В переменных можно хранить строки. Помести эти две строки в переменные *a* и *b*.

Сложение строк

Подобно тому как сложение двух чисел дает новое число, можно сложить две строки. Получится строка, состоящая из исходных строк, склеенных вместе.

Переменные *a* и *b* вместе дают переменную *c*

```
>>> c = a + b
>>> print(c)
Run! Aliens are coming.
```

△ **Сложение строк**
Символ «+» склеивает две строки. Результат попадает в переменную *c*.

В переменную *c* добавляется еще одна строка

```
>>> c = b + ' Watch out! ' + a
>>> print(c)
Aliens are coming. Watch out! Run!
```

△ **Добавление третьей строки в середине**
Можно добавить между двух строк еще одну, как в этом примере.

Новая строка появится в середине

СОВЕТЫ ЭКСПЕРТА

Длина строки

Длину строки можно узнать с помощью функции len(). Python посчитает все символы, включая пробелы, и выдаст их количество.

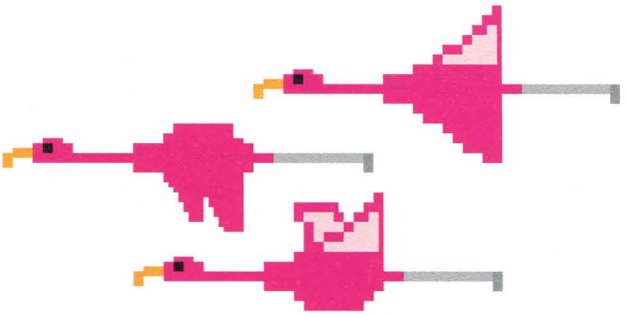
Вычисляет длину строковой переменной *a* (Run!)

```
>>> len(a)
4
>>> len(b)
18
```

Длина строковой переменной *b* (Aliens are coming.) — 18 символов

Нумерация символов

У каждого символа в строке есть номер, который соответствует его позиции. Этот номер можно использовать, чтобы узнать отдельный символ или чтобы вырезать его из строки.



1 Отсчет ведется с нуля

Считать позиции символов Python начинает с 0. У второго символа в строке — позиция 1, у третьего — 2 и так далее.

```
>>> a = 'FLAMINGO'
```

У шестого символа, N, позиция 5

У последнего символа, O, позиция 7

У первого символа, F, позиция 0

2 Отсчет символов

Номер позиции называется индексом. По индексу можно узнать конкретный символ в строке.

```
>>> a[3]
'M'
```

Индекс указывают в квадратных скобках

Символ в позиции 3 переменной *a*

3 Срез

С помощью двух индексов можно получить часть строки — сделать ее срез. Символ в позиции второго индекса в срез не войдет.

```
>>> a[1:7]
'LAMING'
```

Срез указывается с помощью двоеточия

Срез переменной *a* с индекса 1 по индекс 6

4 С начала или до конца

Если не указывать первый или последний индекс, Python автоматически сделает срез с первого или по последний символ.

```
>>> a[:3]
'FLA'
>>> a[3:]
'MINGO'
```

С индекса 0...

...по индекс 7

Апострофы

Строки можно задавать и в одинарных, и в двойных кавычках, но начинаться и заканчиваться строка должна кавычками одного вида. В этой книге мы используем одинарные кавычки (знак апострофа). Но что если апостроф нужен внутри строки?

```
>>> print('It\'s a cloudy day.')
It's a cloudy day.
```

Теперь апостроф — часть строки

△ Экранирование апострофа

Чтобы Python не принял апостроф за конец строки, впиши перед ним символ «\». Это называется экранированием.

Ввод и вывод

Программы взаимодействуют с пользователем с помощью ввода и вывода. Вводить данные можно с клавиатуры, а вывод — это все, что программа печатает на экране.

СМОТРИ ТАКЖЕ

- ‹ 100–101 Выполнение программ
- ‹ 110–111 Типы данных
- Циклы в Python 122–123 ›

Ввод

Команда input() служит для ввода данных с клавиатуры. Она ждет, пока пользователь напечатает что-нибудь и нажмет Enter.

1 Использование input()

Программа может подсказать пользователю, что́ нужно ввести. Сообщение с подсказкой указывается в скобках после input.

С пробелом после двоеточия вывод программы выглядит аккуратнее

```
name = input ('Enter your name: ')
print('Hello', name)
```

От того, какая строка введена, зависит, что напечатает программа

2 Вывод в окне консоли

Если запустить программу, сообщение Enter your name: и введенный ответ появятся в окне консоли.

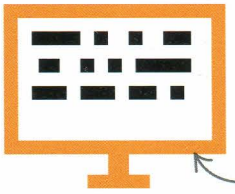
```
Enter your name: Jina
Hello Jina
```

Программа выводит сообщение

Пользователь ввел свое имя

Вывод

Функция print() служит для печати данных в окне консоли. Она может одновременно выводить и текст, и значения переменных.



1 Создаем переменные

Создай для этого примера три переменные — две строковые и одну числовую (целое число).

```
>>> a = 'Dave'
>>> b = 'is'
>>> c = 12
```

Кавычки означают, что это строки

Кавычек нет, это число

2 Использование print()

В скобках функции print() можно указать несколько значений. Они могут быть разных типов, можно даже печатать строки вместе с переменными.

```
>>> print(a, b, c)
Dave is 12
>>> print('Goodbye', a)
Goodbye Dave
```

Между значениями ставятся запятые

Два способа разделять вывод

До сих пор мы выводили все одной строкой, с пробелом между значениями. Однако есть два других способа разделять значения.

```
>>> print(a, b, c, sep='-')
Dave-is-12
```

Символ, разделяющий значения

△ Через дефис

Между выводимыми значениями можно ставить дефис или другие символы, например «+» или «*».

Разделитель

```
>>> print(a, b, c, sep='\n')
Dave
is
12
```

Каждое значение — с новой строки

△ С новой строки

В параметре sep можно указать, чем разделять выводимые значения. Если sep равно \n, каждое значение будет печататься с новой строки.

Три способа завершить вывод

Есть несколько способов завершить вывод функции print.

```
>>> print(a, '.')
Dave .

>>> print(a, end='.')
Dave.
```

Точка как строковое значение

Точка как параметр end

△ Точка в конце

Чтобы завершить вывод print точкой, можно напечатать ее как строку, но тогда перед точкой появится пробел. Чтобы этого избежать, используй «end='.'».

Цикл, чтобы напечатать строку три раза

Пробел как end-параметр

```
>>> for n in range(3):
    print('Hurray!' end=' ')
Hurray! Hurray! Hurray!
```

Все выводится в одну строку

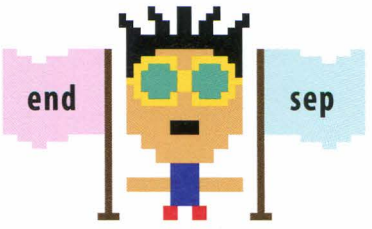
△ Вывод без перевода строки

Обычно каждая команда print выводит данные с новой строки. Чтобы печатать все в одну строку, укажи пробел в параметре end.

СОВЕТЫ ЭКСПЕРТА

Параметры print()

Метки end и sep указывают, что аргумент является параметром настройки print(), а не очередным значением для печати.




```
>>> print(a, end='\n\n\n\n')
Dave


>>>
```

Каждое \n начинает новую строку

Пустые строки

△ Пустые строки в конце

Если end равен \n, каждый вывод print()начнется с новой строки. Указав несколько \n подряд, можно завершить вывод пустыми строками.

Принятие решений

Программы принимают решения о том, что делать дальше, сравнивая переменные, числа и строки с помощью булевых выражений — тех, что возвращают либо True, либо False.

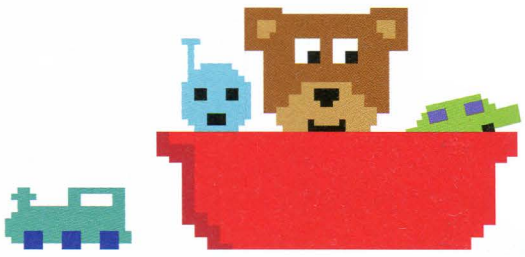
СМОТРИ ТАКЖЕ

❮ 62–63 Истина или ложь?

❮ 108–109 Переменные в Python

Логические операции

Для сравнения переменных с числами, строками или другими переменными служат логические операции. Их результат — True либо False.



==	Операция «равно»
!=	Операция «не равно»
<	Операция «меньше, чем»
>	Операция «больше, чем»
<=	Операция «меньше либо равно»
>=	Операция «больше либо равно»

△ **Виды операций сравнения**

В Python есть шесть операций сравнения. Чтобы сравнить два значения на равенство, используют два знака «=» (одинарный знак «=» служит для задания значения переменной).

▷ **Используй консоль**

Логические операции работают и в окне консоли. Выполни этот пример, чтобы познакомиться с логическими операциями, в числе которых — not, or и and.

```
>>> toys = 10
>>> toys == 1
False
>>> toys > 1
True
>>> toys < 1
False
>>> toys != 1
True
>>> toys <= 10
True
>>> not toys == 1
True
>>> toys == 9 or toys == 10
True
>>> toys == 9 and toys == 10
False
```

Проверяет, равна ли toys («игрушки») единице

Проверяет, больше ли toys, чем 1

Проверяет, меньше ли toys, чем 1

Проверяет toys на неравенство 1

Проверяет, меньше или равна toys 10 либо нет

Операция not меняет результат на противоположный (здесь — с False на True)

Операция or проверяет, равна ли toys 9 или 10

Операция and проверяет, равна ли toys одновременно 9 и 10. Поскольку это невозможно, ответ будет False

У Эллы день рождения?

Элла родилась 28 июля. Эта программа запрашивает номер дня и месяца, а затем с помощью логических операций выясняет, не пора ли праздновать.

1 Проверка на день рождения

Создай переменные для дня и месяца. Используй операцию and, чтобы проверить, соответствуют ли они 28 июля.

```
>>> day = 28
>>> month = 7
>>> day == 28 and month == 7
True
```

Операция and проверяет, истинны ли оба условия

Не забудь поставить двойной знак равенства

У Эллы день рождения!

2 Проверка, что это не день рождения

Операция not меняет результат на противоположный — ответ будет True для любого дня, кроме дня рождения Эллы.

```
>>> day = 28
>>> month = 7
>>> not (day == 28 and \
    month == 7)
False
```

Этот символ нужен, чтобы записать выражение в 2 строки

У Эллы день рождения — ответ False

3 Это день рождения или Новый год?

С помощью операции or можно проверить, день ли это рождения Эллы или Новый год. Используй скобки, чтобы совместить проверки обеих дат.

```
>>> day = 28
>>> month = 7
>>> (day == 28 and month == 7) \
    or (day == 1 and month == 1)
True
```

Проверка на 28 июля

Если это день рождения Эллы или Новый год, результат будет True

Строки

Две строки можно сравнивать операциями «==» и «!=». Равными считаются только абсолютно одинаковые строки.

```
>>> dog = 'Woof woof'
>>> dog == 'Woof woof'
True
>>> dog == 'woof woof'
False
>>> dog == 'Woof woof '
False
```

Строки полностью совпадают, ответ True

Строки не совпадают из-за разницы между заглавной и строчной W

Строки не совпадают из-за лишнего пробела в конце

△ **Полное совпадение**

Строки равны, только если они полностью совпадают. Заглавные буквы, пробелы и специальные знаки — все это должно быть одинаковым.

СОВЕТЫ ЭКСПЕРТА

Строковые операции

Для того чтобы узнать, содержится ли внутри одной строки другая, служит операция in. Используй ее, чтобы определить, содержит ли строка определенный символ или последовательность символов.

Проверяет, есть ли *a* в строке *abc*

```
>>> 'a' in 'abc'
True
>>> 'd' in 'abc'
False
```

В *abc* нет буквы *d*, ответ False

Ветвление

На основе результата булева выражения (True или False, Истина или Ложь) можно принимать решение, какую часть программы выполнять дальше. Это называется ветвлением.

СМОТРИ ТАКЖЕ

‹ 64–65 Решения и ветвление

‹ 118–119 Принятие решений

Выполнять или нет

Команда if выполняет блок команд, если условие дает True. Если же условие дает False, этот блок будет пропущен. Блок нужно вводить после if, всегда помечая его отступом в четыре пробела.

1 Условие if

Эта программа спрашивает, не день ли рождения у пользователя. Если тот ответит *y*, программа напечатает поздравление.

```
ans = input('Is it your birthday? (y/n)')
if ans == 'y':
    print('Happy Birthday!')
```

Подсказывает, что нужно ввести

Этот блок выполняется, только если пользователь ответил *y*

Отступ в 4 пробела

2 Если условие дает True

Запусти программу и введи *y* — ты увидишь сообщение. Оно не появится, если ввести не *y*, а что-нибудь другое.

```
Is it your birthday? (y/n) y
Happy Birthday!
```

Введи *y*

Появилось поздравление

Выполнить одно или другое

Вместе с if можно использовать команду else. Это работает так: если условие дает True, выполняется один блок команд, если же нет, то другой.

1 Условие if-else

Если ввести *y*, программа выведет новогоднее поздравление. Если ввести что-либо кроме *y*, появится другое сообщение.

```
ans = input('Is it New Year? (y/n)')
if ans == 'y':
    print('Happy New Year!')
    print('Time for Fireworks.')
else:
    print('Not yet!')
```

Не забудь про двоеточие

Будет напечатано, только если пользователь ввел *y*

Здесь тоже нужно двоеточие

Выполняется, только если пользователь ввел не *y*

2 **Результат, если условие дает True**
Запусти программу и введи *y*. Программа напечатает поздравление. Другое сообщение показано не будет.

```
Is it New Year? (y/n)y
Happy New Year!
Time for Fireworks.
```

Введи *y*

3 **Блок «else»**
Введи *n*, и поздравление напечатано не будет. Вместо него появится сообщение «Not yet!».

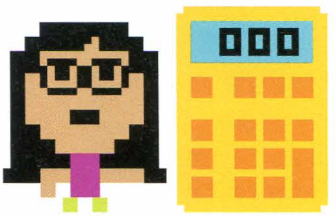
```
Is it New Year? (y/n)n
Not yet!
```

Введи *n*

Появилось другое сообщение

Выполнить одно из...

Команда elif означает «иначе-если»: если условие дает True — выполнить один блок, иначе — проверить другое условие и, если оно дает True, — выполнить другой блок. В этой программе-калькуляторе используется команда elif.



1 **Условие if-elif-else**
Эта программа проверяет, что ввел пользователь. Если это одна из строк add, sub, mul или div, будет показан результат соответствующей операции.

Запрос на ввод числа

Не забудь про кавычки и скобки

```
a = int(input('a = '))
b = int(input('b = '))
op = input('add/sub/mul/div:')
if op == 'add':
    c = a + b
elif op == 'sub':
    c = a - b
elif op == 'mul':
    c = a * b
elif op == 'div':
    c = a / b
else:
    c = 'Error'
print('Answer = ',c)
```

Введи add, чтобы сложить переменные

Введи div, чтобы выполнить деление

Если было введено что-то другое, кладет в «c» сообщение об ошибке

Показывает результат или сообщение об ошибке

2 **Результат для условия, дающего True**
Запусти программу. Введи два числа и затем sub. Будет показан результат вычитания второго числа из первого.

```
a = 7
b = 5
add/sub/mul/div:sub
Answer = 2
```

Введи два числа

Введи sub, чтобы вычесть 5 из 7

Результат получается вычитанием переменной *a* из переменной *b*

3 **Результат для блока else**
Блок else выполнится, если было введено что-то кроме add, sub mul или div, — появится сообщение об ошибке.

```
a = 7
b = 5
add/sub/mul/div:try
Answer = Error
```

Введи что-нибудь другое

Сообщение об ошибке

Циклы в Python

Программу, в которой встречаются блоки повторяющихся команд, неудобно писать и сложно читать. Гораздо удобнее использовать циклы. Простейшие циклы for повторяют блок команд заданное количество раз.

СМОТРИ ТАКЖЕ

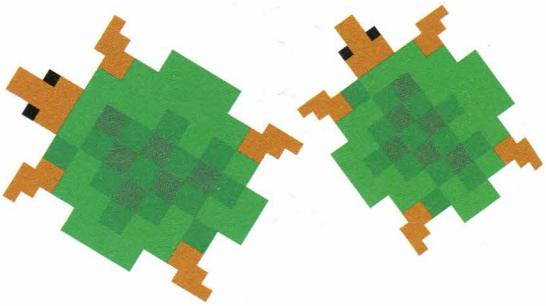
❮ 48–49 Перья и черепашки

Цикл while 124–125 ❯

Выход из цикла 126–127 ❯

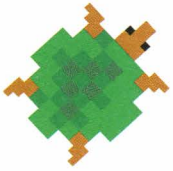
Повтор команд

Цикл for повторяет команды, избавляя от необходимости писать одно и то же несколько раз. Например, может понадобиться напечатать имена всех учеников в классе из 30 человек.



1 Управление черепашкой

С помощью цикла for программу можно также сделать компактнее. В этом примере мы нарисуем на экране треугольник, управляя черепашкой, которая при движении оставляет за собой линию.



```
from turtle import *
forward(100)
right(120)
forward(100)
right(120)
forward(100)
right(120)
```

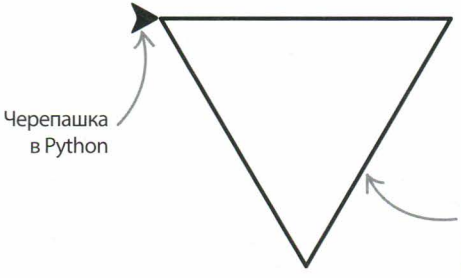
Загружает команды для работы с черепашкой

Двигает черепашку вперед

Поворот на 120 градусов вправо

2 Черепашка рисует треугольник

Программа рисует треугольник, задавая длину каждой из его сторон и углы между сторонами. Когда ты запустишь программу, черепашка появится в отдельном окне.



3 Используем цикл for

Вместо того чтобы три раза, по разу для каждой стороны треугольника, давать черепашке одни и те же команды forward(100) и right(120), можно давать ей эти же команды внутри цикла for. Запусти этот новый вариант программы.

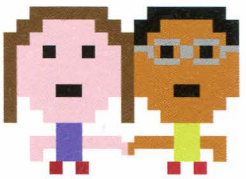
```
for i in range(3):
    forward(100)
    right(120)
```

Цикл for повторяет команды 3 раза

Блок команд внутри цикла выделен отступом в 4 пробела

Переменные цикла

Переменная цикла отсчитывает, сколько раз цикл повторился. Отсчет начинается с первого числа в диапазоне «range (от, до, шаг)» и заканчивается за 1 до последнего.



Переменная цикла

Цикл повторяется трижды

```
for i in range(10):
    print(i, end=' ')
```

Отсчет заканчивается на 1 раньше последнего числа

```
>>> 0 1 2 3 4 5 6 7 8 9
```

△ **Простая переменная цикла**

В этом примере не говорится, с какого числа начинать отсчет, поэтому Python считает с 0, подобно тому как отсчитываются позиции символов в строке.

Указывает, что считать нужно с шагом 2

```
for i in range(2, 11, 2):
    print(i, end=' ')
```

```
>>> 2 4 6 8 10
```

Отсчет по двойкам

△ **Отсчет по двойкам**

В этом цикле указано, откуда начинать отсчет (2), и шаг отсчета (2). Отсчет заканчивается на числе 10 — за 1 до полного количества повторений (11).

Указывает, что отсчитывать нужно назад

```
for i in range(10, 0, -1):
    print(i, end=' ')
```

```
>>> 10 9 8 7 6 5 4 3 2 1
```

△ **Обратный отсчет**

На этот раз программа считает от 10 назад — подобно отсчету перед стартом ракеты. Переменная цикла пробегает значения от 10 до 1 с шагом –1.

Вложенные циклы

Цикл внутри другого цикла называется вложенным. В этом случае внешний цикл повторяется лишь после того, как внутренний цикл выполнит все свои повторы.

Чтобы цикл повторился *n* раз, последним числом диапазона должно быть *n* + 1

Внешний цикл

```
n = 3
for a in range(1, n + 1):
    for b in range(1, n + 1):
        print(b, 'x', a, '=', b * a)
```

Внутренний цикл

Это выражение будет вычислено и напечатано девять раз

△ **Цикл внутри цикла**

В этом примере на каждый повтор внешнего цикла приходится три повтора цикла внутреннего. Выходит, что всего внешний цикл повторится три раза, а внутренний — девять раз.

Значение *a*

Значение *b*

```
>>>
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
```

Первый повтор внешнего цикла (внутренний цикл повторился трижды)

Второй повтор внешнего цикла

Третий повтор внешнего цикла

△ **Что делает программа**

Вложенные циклы выводят первые три строки таблицы умножения для чисел 1, 2 и 3. Значение *a* меняется при каждом повторе внешнего цикла. Значение *b* пробегает от 1 до 3 для каждого *a*.

Цикл while

Циклы for хороши, когда количество повторений известно заранее. Но иногда цикл нужно повторять, пока что-то в программе не изменится. Цикл while будет повторяться до тех пор, пока это нужно.

СМОТРИ ТАКЖЕ

❮ 118–119 Принятие решений

❮ 122–123 Циклы в Python

Выход из цикла 126–127 ❯

Цикл while

Цикл while будет повторяться, пока выполняется заданное условие. Это условие называется условием цикла и возвращает True или False.

1 Используем цикл while

Дадим значение переменной answer, которая проверяется в условии цикла. Сперва это условие должно давать True, иначе цикл не повторится ни разу.

▷ Как это работает

Цикл while проверяет условие. Если оно истинно (дает True), цикл повторяется, если же нет, он заканчивается.

Блок команд внутри цикла нужно пометить отступом в 4 пробела

```
answer = 'y'
while answer == 'y':
    print('Stay very still')
    answer = input('Is the monster friendly? (y/n)')
print('Run away!')
```

Переменной answer дается значение *y*

Цикл выполняется, только если условие дает True

Если условие дает False, команда без отступа после цикла выдаст другое сообщение

2 Что происходит в программе

Введенное значение сохраняется в переменной answer. Условие цикла «answer == *'y'*», поэтому, если ввести *y* — цикл продолжится, а если *n* — завершится.

```
>>>
Stay very still
Is the monster friendly? (y/n)y
Stay very still
Is the monster friendly? (y/n)y
Stay very still
Is the monster friendly? (y/n)n
Run away!
```

Введено *y*, цикл продолжается

Введено *n*, цикл прекращается, выводится другое сообщение

ЗАПОМНИ

Блок repeat until

Цикл while в Python похож на блок repeat until («повторить, пока не») в Scratch. И тот и другой циклы выполняются, пока что-то в программе не изменится.

Повторяет внутренние блоки, пока условие не станет истинно

Бесконечный цикл

Если в качестве условия while задать True, оно никогда не станет ложным — цикл никогда не завершится. Это может быть как полезно, так и очень неприятно.

△ **Зациклились**
Цикл с условием True называют бесконечным, потому что он никогда не заканчивается.

1 Создадим бесконечный цикл

Здесь условием цикла задано True. Что бы ни случилось в цикле, это условие никогда не даст False и цикл никогда не завершится.

Всегда True значит без конца

Введенное значение сохраняется в переменной answer

```
while True:
    answer = input('Type a word and press enter: ')
    print('Please do not type \'' + answer + '\' again.')
```

2 Что происходит в программе

В предыдущем примере в условии цикла была проверка ответа пользователя: цикл прекращался, если введено y. Но в этом примере такой проверки нет — пользователь никак не сможет остановить цикл.

Что бы ни ввел пользователь, цикл продолжает повторяться

```
>>>
Type a word and press enter: tree
Please do not type 'tree' again
Type a word and press enter: hippo
Please do not type 'hippo' again
Type a word and press enter: water
Please do not type 'water': again
Type a word and press enter
```

СОВЕТЫ ЭКСПЕРТА

Остановка цикла

Ты можешь остановить бесконечный цикл из IDLE. Для этого в окне консоли, удерживая клавишу Ctrl, нажми клавишу C — IDLE будет отправлен запрос на остановку программы. Возможно, придется нажать Ctrl-C несколько раз. Это похоже на нажатие красной кнопки в Scratch.

ЗАПОМНИ

Блок forever

Помнишь блок forever («всегда») в Scratch? Он выполнял блоки внутри себя, пока не нажата красная кнопка. Цикл while True делает в точности то же самое. Используй его, когда нужно, чтобы программа всегда делала одно и то же, по кругу.

Блок forever перемещает спрайт без остановки

Выход из цикла

Программа может зациклиться, но есть и способы это прекратить. Команда break завершает цикл (даже бесконечный), а команда continue сразу переходит к следующему повтору цикла.

СМОТРИ ТАКЖЕ

❮ 122–123 Циклы в Python

❮ 124–125 Цикл while

Выход

Вызов break внутри цикла приводит к его мгновенному завершению — даже если условие дает True. Любые команды внутри цикла после вызова break игнорируются.

1 Пишем простую программу

Эта программа проверяет, умеет ли пользователь умножать на 7. Она выполняется, пока на каждый из 12 вопросов не будет дан ответ. Введи этот код в окне программы.

```
table = 7
for i in range(1, 13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

Переменная *i* считает от 1 до 12

i — это переменная цикла

Обратный слеш («\») говорит, что следующий символ — это апостроф, а не конец строки

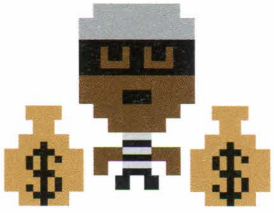
2 Добавим break

Чтобы пользователь мог выйти из цикла, можно использовать break — программа завершится, если пользователь напечатает stop.

```
table = 7
for i in range(1,13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    if guess == 'stop':
        break
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

Если guess равняется stop, программа выйдет из цикла и напечатает Finished («Завершено»)

В переменной ans — правильный ответ на вопрос



```
>>>
What's 1 x 7 ?
1
No, it's 7
What's 2 x 7 ?
14
Correct!
What's 3 x 7 ?
stop
Finished
```

При первом повторе цикла *i* равно 1

При втором повторе *i* примет значение 2

Приведет к вызову break и выходу из цикла

3 Как это работает

Если пользователю надоест отвечать и он введет stop, будет выполнена команда break и программа выйдет из цикла.

Пропуск вопроса

С помощью команды continue («продолжить») можно пропустить вопрос, не выходя из цикла. Эта команда говорит, что нужно завершить текущий повтор цикла и сразу перейти к следующему.

```
table = 7
for i in range(1,13):
    print('What\'s', i, 'x', table, '?')
    guess = input()
    if guess == 'stop':
        break
    if guess == 'skip':
        print('Skipping')
        continue
    ans = i * table
    if int(guess) == ans:
        print('Correct!')
    else:
        print('No, it\'s', ans)
print('Finished')
```

При первом повторе задаст вопрос: What's 1 x 7? («Сколько будет 1 x 7?»)

Переход к следующему повтору

4 Добавим continue

Добавь внутри цикла еще один if, проверяющий, не введено ли skip. Если это так, программа напечатает Skipping («Переход») и выполнит continue, чтобы перейти к следующему повтору.

5 Что получилось

Если пользователь не хочет отвечать на вопрос, он может ввести skip, чтобы перейти к следующему.

```
>>>
What's 1 x 7 ?
skip
Skipping
What's 2 x 7 ?
14
Correct!
What's 3 x 7 ?
```

Введи skip, чтобы пропустить вопрос

Был введен правильный ответ, цикл продолжается

Списки

Если нужно держать много данных в одном месте, их можно поместить в список. В списке можно хранить числа, строки, другие списки или все это одновременно.

СМОТРИ ТАКЖЕ

❮ 54–55 Строки и списки

Забавные фразы 132–133 ❯

Что такое список?

Список — это структура, элементы которой хранятся по порядку. Каждому элементу соответствует номер, по которому к нему можно обратиться. Можно менять, удалять или добавлять элементы в любой момент.

▽ Пример списка

Каждый элемент записан в одинарных кавычках и отделен от следующего запятой. Весь список заключен в квадратные скобки.

```
>>> mylist = ['apple', 'milk', 'cheese', 'icecream', \
'lemonade', 'tea']
```

Список хранится в переменной mylist

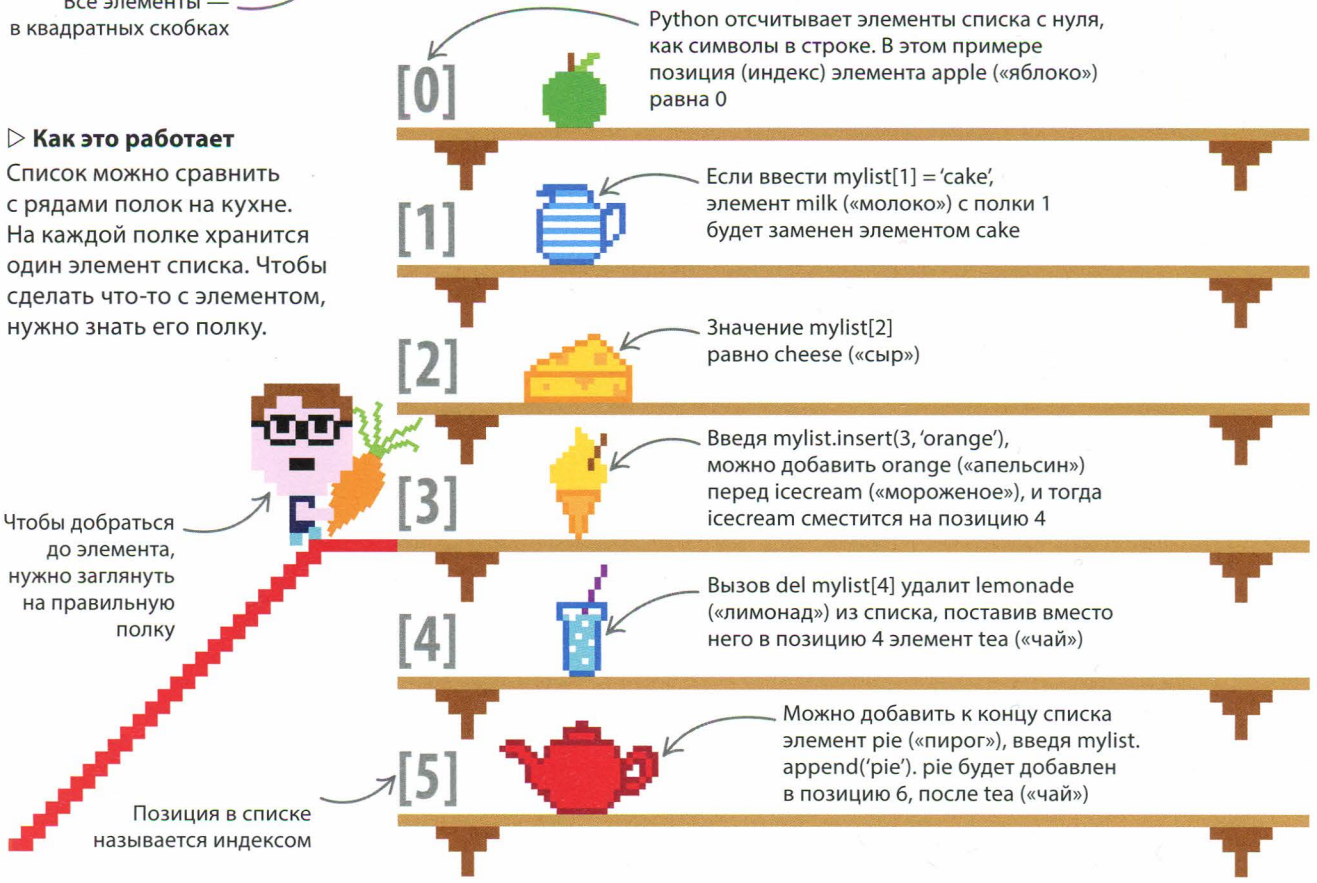
Слеш нужен, чтобы записать код в две строки

Элементы отделены запятыми

Все элементы — в квадратных скобках

▷ Как это работает

Список можно сравнить с рядами полок на кухне. На каждой полке хранится один элемент списка. Чтобы сделать что-то с элементом, нужно знать его полку.



Использование списков

Когда список создан, можно написать программу для работы с его содержимым, например в цикле. Также можно составлять из списков другие списки.

```
>>> names = ['Simon', 'Kate', 'Vanya']
>>> for item in names:
        print('Hello', item)

Hello Simon
Hello Kate
Hello Vanya
```

Список хранится в переменной names

Тело цикла нужно помечать отступом в 4 пробела

Эта программа напечатает Hello для каждого имени из списка

СЛЕНГ

Изменяемые объекты

Списки в Python изменяемые — то есть можно добавлять и удалять элементы, а также менять их порядок. В Python есть и неизменяемые объекты, которые нельзя менять после их создания, например кортежи (см. с. 134–135).

◁ **Списки и циклы**

Чтобы перебрать все элементы списка, можно использовать цикл. Эта программа выводит приветствие для каждого имени из списка.

```
x = [1, 2, 3, 4]
y = [5, 6, 7, 8]
z = x + y
print(z)
z = [1, 2, 3, 4, 5, 6, 7, 8]
```

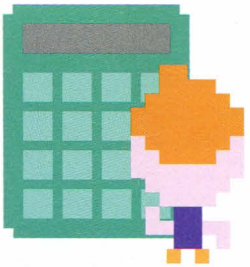
Запомни, списки указываются в квадратных скобках

Складывает списки

В новом списке — все элементы из *x*, за которыми идут все элементы из *y*

◁ **Сложение списков**

Два списка можно сложить. Новый список будет содержать элементы из обоих списков.



▽ **Списки списков**

Элементы списков тоже могут быть списками. Список suitcase («чемодан») из этого примера содержит два других списка (предметы одежды).

```
>>> suitcase=[['hat', 'tie', 'sock'],['bag', 'shoe', 'shirt']]
>>> print(suitcase)
[['hat', 'tie', 'sock'],['bag', 'shoe', 'shirt']]
>>> print(suitcase[1])
['bag', 'shoe', 'shirt']
>>> print(suitcase[1][2])
shirt
```

Список в квадратных скобках становится элементом списка suitcase — suitcase[0]

suitcase[1]

Напечатает весь список

Напечатает содержимое второго списка, suitcase[1]

Напечатает элемент с индексом 2 из suitcase[1] — помни, в Python индексы считаются с нуля

Функции

СМОТРИ ТАКЖЕ

Забавные фразы **132–133 ›**

Переменные и функции **138–139 ›**

Функция — это именованная часть программы, выполняющая определенную задачу. По ее имени функцию в любой момент можно вызвать. Функции служат для того, чтобы не вводить одни и те же команды несколько раз.

Полезные функции

В Python уже есть множество полезных функций на разные случаи жизни. При вызове функции Python находит содержащиеся в ней команды и выполняет их. После этого программа возвращается к строке, откуда функция была вызвана, и переходит к следующей команде.

print()

△ **Функция print()**
Выводит информацию для пользователя — например, инструкции или результаты вычислений.

input()

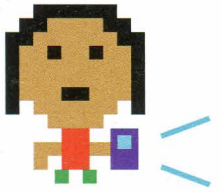
△ **Функция input()**
Противоположность функции print() — позволяет пользователю ввести данные и передать их в программу.

randint()

△ **Функция randint()**
Возвращает случайное число. Может пригодиться, если в программе нужен элемент случайности.

Создание и вызов функций

Вызывать можно не только функции, идущие в комплекте с Python. Чтобы создать новую функцию, запиши нужные команды в специальной «обертке» и дай ей имя. По имени функцию можно будет вызвать, как только она понадобится.



1 Определение функции
Определение функции всегда начинается с ключевого слова def и имени функции.

```
def greeting():
    print('Hello!')
```

Двоеточие отмечает конец имени функции и начало содержащихся в ней команд

Команда внутри функции

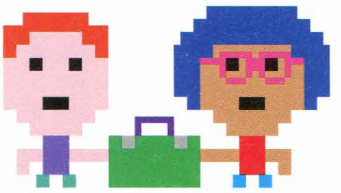
2 Вызов функции
Если в окне консоли ввести имя функции, а затем скобки, она будет вызвана, и на экране появится результат.

```
>>> greeting()
Hello!
```

Вызвана функция greeting («приветствие»), результат на экране

Скобки показывают, что это функция, а не переменная

Передача данных в функцию



В функцию можно передавать значения, с которыми она будет работать. Например, print(a, *b*, *c*) означает, что в функцию print передаются значения переменных *a*, *b* и *c*. Вызов height(1, 45) означает, что в height передаются значения 1 и 45.

1 Параметры функции

Значения, которые передаются в функцию, называют параметрами. При определении функции параметры указывают в скобках следом за ее именем.

m и *cm* — параметры

```
def height(m, cm):
    total = (100 * m) + cm
    print(total, 'cm tall')
```

Выводит значение total и строку cm tall

Переводит метры в сантиметры, умножая их на 100

2 Передача значений

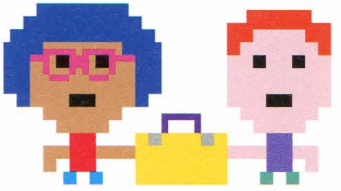
Команды внутри функции используют переданные значения.

Вызывает функцию с параметрами *m* = 1 и *cm* = 45

```
>>> height(1, 45)
145 cm tall
```

Показывает, что 1 метр 45 сантиметров — это 145 сантиметров

Получение данных из функции



Удобно, когда функции передают данные обратно в программу — возвращают значение. Чтобы функция возвращала значение, добавь в нее команду return, а следом значение, которое нужно вернуть.

1 Определим функцию, которая возвращает число

Функция input() всегда возвращает строку, даже если было введено число. Эта новая функция возвращает числовое значение.

```
def num_input(prompt):
    typed = input(prompt)
    num = int(typed)
    return num

a = num_input('Enter a')
b = num_input('Enter b')
print('a + b =', a + b)
```

Строка, полученная от input, хранится в переменной typed

Преобразует строку в число и сохраняет его в переменной num

Возвращает значение переменной

2 Число на выходе

Если бы в программе использовалась функция input, *a* + *b* означало бы соединение строк 10 и 7 — получилось бы 107.

```
Enter a 10
Enter b 7
a + b = 17
```

Сложение *a* + *b* дает 17, потому что функция num_input возвращает число, а не строку

ПРОЕКТ 5

Забавные фразы

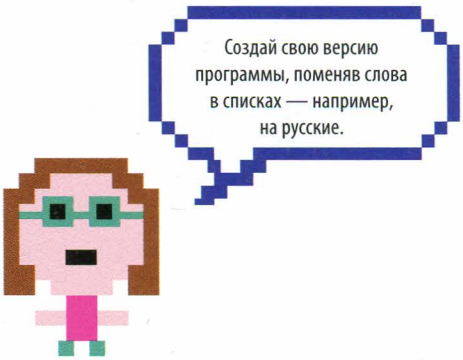
Циклы, функции и списки очень полезны сами по себе. Но если использовать их вместе, можно писать программы для решения еще более интересных и сложных задач.

СМОТРИ ТАКЖЕ

- **124–125** Цикл while
- **128–129** Списки
- **130–131** Функции

Генератор забавных фраз

Эта программа составляет фразы на английском языке на основе трех списков со словами. Из каждого списка она берет по случайному слову и соединяет их в смешную фразу.



1 Открой новое окно программы и введи эти строки — определения списков, из которых мы будем выбирать слова.

Одинарные кавычки означают, что каждый элемент списка — строка

```
name = ['Neha', 'Lee', 'Sam']
verb = ['buys', 'rides', 'kicks']
noun = ['lion', 'bicycle', 'plane']
```

Квадратные скобки означают, что это список

2 Чтобы составить фразу, нужно выбрать из списков случайные слова. Определим для этого функцию — ведь выбор придется делать трижды, по разу для каждого из списков.

Загружает функцию получения случайного числа (randint)

```
from random import randint
def pick(words):
    num_words = len(words)
    num_picked = randint(0, num_words - 1)
    word_picked = words[num_picked]
    return word_picked
```

Вычисляет, сколько слов в списке (работает для списков любой длины)

Выбирает случайное число, соответствующее одному из элементов списка

Сохраняет случайное слово в переменной word_picked

3 Составь случайную забавную фразу, вызвав функцию pick для каждого из списков со словами. Для вывода на экран используй команду print.

Добавь *a*, чтобы получилась правильная фраза на английском

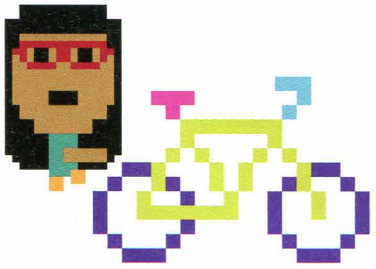
```
print(pick(name), pick(verb), 'a', pick(noun), end='.\n')
```

Добавляет в конец фразы точку, а после нее начинает новую строку

4 Сохрани программу, запусти ее — и увидишь фразу, составленную на основе списков имен, глаголов и существительных.

```
Neha kicks a bicycle.
```

При каждом запуске программа составит новую случайную фразу



Забавные фразы навсегда!

В программу можно добавить бесконечный цикл, чтобы она работала без остановки, пока пользователь не нажмет Ctrl-C.

1 Если выполнять команду print в цикле while True, программа будет печатать смешные фразы снова и снова.

```
while True:
    print(pick(name), pick(verb), 'a', pick(noun), end='.')
    input()
```

Заключает команду печати в цикл

Выдает новую фразу после каждого нажатия Enter

2 Функция input() ждет, пока пользователь не нажмет Enter перед выводом очередной фразы. Без этого программа печатала бы слишком быстро.

Программа будет составлять всё новые и новые случайные фразы

```
Sam rides a lion.
Neha kicks a plane.
Lee buys a bicycle.
```

СОВЕТЫ ЭКСПЕРТА

Читаемый код

Важно писать программы так, чтобы потом в них было легко разобраться. Это упростит их доработку в будущем, ведь в таком случае не придется сначала выяснять, как программа работает.

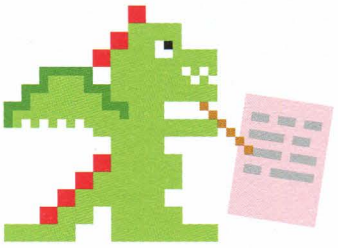
Кортежи и словари

СМОТРИ ТАКЖЕ
❮ 110–111 Типы данных
❮ 128–129 Списки

Списки нужны для упорядоченного хранения информации. Для этого служат и два других типа данных — кортежи и словари. Типы данных, предназначенные для хранения множества элементов, называют контейнерными.

Кортежи

Кортежи напоминают списки, однако хранящиеся в них элементы нельзя изменять. Кортеж всегда остается таким, каким был создан.

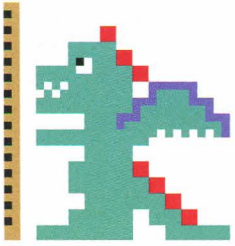


Кортежи записывают в круглых скобках

```
>>> dragonA = ('Sam', 15, 1.70)
>>> dragonB = ('Fiona', 16, 1.68)
```

Элементы кортежа разделяют запятыми

◁ Что такое кортеж?
Кортеж содержит элементы, которые отделены запятыми и заключены в круглые скобки. В кортежах удобно хранить свойства объектов — к примеру, имя, возраст и рост дракона.



▷ Получение элемента кортежа
Чтобы получить элемент кортежа, используй его позицию (индекс). Индексы отсчитываются с нуля, так же как для списков и строк.

```
>>> dragonB[2]
1.68
```

Выбирает элемент в позиции 2

◁ Получение переменных из кортежа
Приравняй кортеж dragonA трем переменным: name, age и height («имя», «возраст» и «рост»). Python выделит из кортежа три элемента, поместив каждый из них в соответствующую переменную.

```
>>> name, age, height = dragonA
>>> print(name, age, height)
Sam 15 1.7
```

Элементы кортежа dragonA, напечатанные по отдельности

▷ Создание списка кортежей
Контейнерные типы могут быть вложены один в другой, а значит, кортежи можно поместить в список.

Создадим список кортежей под именем dragons

Списки задают в квадратных скобках

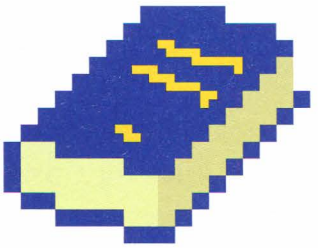
```
>>> dragons = [dragonA, dragonB]
>>> print(dragons)
[('Sam', 15, 1.7), ('Fiona', 16, 1.68)]
```

Кортежи в круглых скобках, снаружи — квадратные скобки списка

Python напечатает все содержимое списка, а не просто имена кортежей

Словари

Словари похожи на списки, но вместо индексов их элементы помечают специальными ярлыками — ключами. У каждого элемента словаря есть ключ и значение, причем очередность элементов роли не играет. Содержимое словаря можно изменять.



▷ Создадим словарь

В этой программе создается словарь под именем age. Ключ каждого элемента — это имя человека, а значение — его возраст.

```
>>> age = {'Mary': 10, 'Sanjay': 8}
```

Словари записывают в фигурных скобках

Элементы словаря разделяют запятыми

Ключ и значение разделяют двоеточием

Ключ используют вместо индекса элемента

Значение элемента словаря (всегда указывается после двоеточия)

```
>>> print(age)
{'Sanjay': 8, 'Mary': 10}
```

Имя словаря

Sanjay — ключ этого элемента

10 — значение элемента Mary

◁ Распечатаем словарь

Очередность элементов может меняться, поскольку позиции элементов в словаре не фиксированы.

▷ Добавим новый элемент

В словарь можно добавить значение, пометив его новым ключом.

```
>>> age['Owen'] = 11
>>> print(age)
{'Owen': 11, 'Sanjay': 8, 'Mary': 10}
```

Имя словаря

Новый ключ

Добавляет элемент в словарь

В словаре появилось новое значение

Прежние значения остаются в словаре

◁ Изменение значения

Чтобы изменить значение элемента, приравняй его ключу новое значение.

```
>>> age['Owen'] = 12
>>> print(age)
{'Owen': 12, 'Sanjay': 8, 'Mary': 10}
```

Задает элементу с ключом Owen новое значение

Значение элемента Owen изменилось

▷ Удаление элемента

Удаление элемента из словаря никак не влияет на остальные элементы, поскольку они помечены ключами, а не индексами.

```
>>> del age['Owen']
>>> print(age)
{'Sanjay': 8, 'Mary': 10}
```

Удаляет элемент с ключом Owen

Элемента с ключом Owen больше нет в словаре

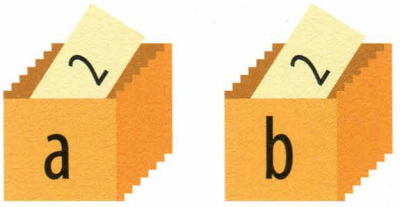
Списки в переменных

СМОТРИ ТАКЖЕ
❮ 108–109 Переменные в Python
❮ 128–129 Списки

Может показаться, что при хранении списков в переменных Python ведет себя довольно странно. Разберемся, что при этом происходит, и все станет на свои места.

Переменные просто хранят значения

Переменные похожи на ящички для хранения значений. Значение можно скопировать из одной переменной в другую — то есть создать копию значения из ящичка *a* и поместить ее в ящичек *b*.



△ **Как устроены переменные**
Каждая переменная — как ящичек, в котором лежит листок бумаги с записанным значением.

1 Зададим значение переменной
Задай переменной *a* значение 2, затем присвой переменной *b* значение из переменной *a*. При этом возникнет копия числа 2, которая и попадет в *b*.

```
>>> a = 2
>>> b = a
>>> print('a =', a, 'b =', b)
a = 2 b = 2
```

Копирует содержимое *a* в *b*

Теперь и в *a*, и в *b* — число 2

Печатает имена и значения переменных

2 Изменим значение
Если изменить значение, хранящееся в одной переменной, на значение другой переменной это не повлияет, так же как если изменить надпись на листке в ящичке *a*, — надпись на листке в *b* останется прежней.

```
>>> a = 100
>>> print('a =', a, 'b =', b)
a = 100 b = 2
```

Изменим значение в *a*, теперь это 100

Теперь в *a* число 100, а в *b* по-прежнему 2

3 Изменим другое значение
Поменяем значение в *b* на 22. В переменной *a* по-прежнему 100. Хоть первоначальное значение *b* и было скопировано из *a*, теперь эти переменные независимы — изменение *b* никак не влияет на *a*.

```
>>> b = 22
>>> print('a =', a, 'b =', b)
a = 100 b = 22
```

b содержит 22, но в *a* по-прежнему 100

А если поместить в переменную список?

При копировании значения переменной создается независимая копия. Это так, если значение — число, но как насчет других типов данных? Если в переменной список, все происходит немного иначе.

1 Копируем список
Сохрани список [1, 2, 3] в переменной listA. Теперь присвой значение listA другой переменной — listB. В итоге обе переменные содержат список [1, 2, 3].

При создании списка используй квадратные скобки

```
>>> listA = [1, 2, 3]
>>> listB = listA
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 2, 3] listB = [1, 2, 3]
```

Печатает имена переменных и их значения

У listA и listB одно и то же значение

2 Изменим listA
Поменяй значение в listA[1] на число 1000. Теперь listB[1] тоже содержит 1000 — изменение списка привело к изменению его копии!

Меняет второй элемент списка (отсчет ведется с нуля)

```
>>> listA[1] = 1000
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 1000, 3] listB = [1, 1000, 3]
```

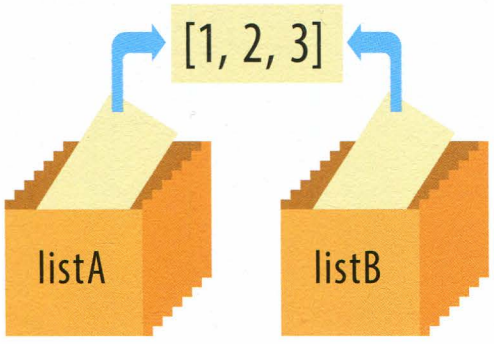
Второй элемент обоих списков поменялся

3 Изменим listB
Поменяй значение в listB[2] на число 75. В listA[2] теперь тоже 75. Изменение копии привело к изменению первоначального списка.

Это третий элемент списка

```
>>> listB[2] = 75
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 1000, 75] listB = [1, 1000, 75]
```

Третий элемент обоих списков поменялся



△ **Что же произошло?**
В переменных хранятся не сами списки, а лишь ссылки на них. Копирование значения listA привело к копированию ссылки, и теперь listA и listB ссылаются на один и тот же список.

СОВЕТЫ ЭКСПЕРТА

Копирование списков

Чтобы сделать независимую копию списка, используй функцию copy. В listC будет ссылка на новый список, элементы которого скопированы из listA. Изменение listC не приведет к изменению listA, а изменение listA — к изменению listC.

```
>>> listC = listA.copy()
```

Переменные и функции

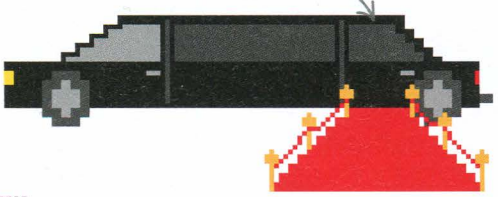
СМОТРИ ТАКЖЕ
‹ 130–131 Функции
Рисование фигур 158–159 ›

Переменные, созданные внутри функции (локальные переменные), и переменные, созданные в основной программе (глобальные переменные), ведут себя по-разному.

Локальные переменные

Локальные переменные существуют лишь в пределах одной функции, поэтому основной программе и другим функциям они недоступны. Попытка обратиться к локальной переменной извне приведет к ошибке.

Локальные переменные похожи на кинозвезд, сидящих в машине с зеркальными стеклами: они внутри (в функции), но никто их не видит



1 Переменная в функции
Создай в функции func1 локальную переменную *a*. Напечатай значение *a*, вызвав func1 из основной программы.

```
>>> def func1():
        a = 10
        print(a)
>>> func1()
10
```

Вызов func1 приводит к печати значения *a*

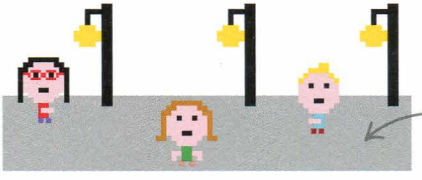
2 Переменная вне функции
Если ты попробуешь напечатать *a* из основной программы, это закончится ошибкой: *a* существует только для func1.

```
>>> print(a)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
```

Сообщение об ошибке: основная программа не знает, что такое *a*

Глобальные переменные

Переменная, созданная в основной программе, называется глобальной. Функции могут узнать ее значение, но не изменить его.



Глобальные переменные похожи на людей, идущих по улице, — они у всех на виду

1 Переменная вне функции
Создай в основной программе глобальную переменную *b*. Новая функция func2 может прочесть значение *b* и напечатать его.

```
>>> b = 1000
>>> def func2():
        print(b)
>>> func2()
1000
```

func2 видит значение *b*, поскольку *b* — глобальная переменная

При вызове func2 печатается значение *b*

2 Общая глобальная переменная
Кроме того, мы можем напечатать *b* прямо из основной программы. *b* видна отовсюду, потому что создана не в функции.

```
>>> print(b)
1000
```

К глобальной переменной *b* можно обратиться из любого места основной программы

Переменные как параметры функций

Если переменная используется как входной параметр функции, ее значение копируется в новую локальную переменную. Изменение значения этой новой переменной внутри функции не влияет на значение первоначальной переменной.

1 Изменение значения переменной

func3 использует входной параметр *y* как локальную переменную. Она печатает значение *y*, затем меняет его на bread и печатает новое значение.

```
>>> def func3(y):
        print(y)
        y = 'bread'
        print(y)
>>> z = 'butter'
>>> func3(z)
butter
bread
```

В *y* — значение, переданное при вызове func3

Теперь в *y* — строка bread

Создание глобальной переменной *z*

Теперь входной параметр *y* содержит значение *z*, переданное в момент вызова func3

2 Печать переменной

Если распечатать значение *z* после вызова func3, видно, что оно не изменилось. Вызов func3 копирует значение *z* (butter) в локальную переменную *y*, но на саму *z* это не влияет.

```
>>> print(z)
butter
```

Печать значения глобальной переменной *z* после вызова func3

Локальная для func3 переменная *y* хранит копию значения *z*. Хоть значение *y* и изменилось на bread («хлеб»), это не затрагивает глобальную переменную *z*, значение которой по-прежнему butter («масло»)

Маскирование глобальной переменной

Глобальную переменную нельзя изменить из функции. Если функция попытается это сделать, будет создана локальная переменная с таким же именем, которая скрывает (маскирует) глобальную переменную.

1 Изменение глобальной переменной

Глобальной переменной *c* присвоено значение 12345. func4 дает *c* значение 555 и печатает его. Похоже, глобальную переменную *c* удалось изменить.

```
>>> c = 12345
>>> def func4():
        c = 555
        print(c)
>>> func4()
555
```

Первоначальное значение глобальной переменной *c*

Печатает значение *c* из функции func4

2 Печать переменной

Если напечатать *c* из основной программы, станет ясно, что ее значение не поменялось. func4 печатает значение новой локальной переменной, которая тоже называется *c*.

```
>>> print(c)
12345
```

Значение глобальной переменной *c* не поменялось

СОВЕТЫ ЭКСПЕРТА

Вызов функций

Есть два разных способа вызывать функции.

function(a)

Элементы данных в Python называют объектами. Некоторые функции вызывают, передавая им объект (a).

a.function()

Другие функции вызывают, добавляя их имя к имени объекта (a) через точку.

ПРОЕКТ 6

Чертежный автомат

СМОТРИ ТАКЖЕ
❮ 122–123 Циклы в Python
Библиотеки 152–153 ❯

Пора заняться более сложным проектом. Эта программа преобразует строку с простыми инструкциями в команды для черепашки, чтобы чертить фигуры на экране. Навыки, нужные для разработки этой программы, необходимы каждому программисту.

Выберем фигуру для проверки

При разработке программы для рисования любых фигур полезно выбрать какую-то одну фигуру для проверки. Используй этот контур домика, чтобы тестировать программу на разных этапах ее написания. В итоге домик можно будет нарисовать гораздо проще — с помощью строки с краткими командами (вроде F100).

▷ **Черепашка рисует домик**
Стрелка указывает на точку, где остановилась черепашка. Начав с левого нижнего угла, она двигалась по часовой стрелке, рисуя домик.

```
from turtle import *
reset()
left(90)
forward(100)
right(45)
forward(70)
right(90)
forward(70)
right(45)
forward(100)
right(90)
forward(100)
```

Загружает команды для работы с черепашкой

Обнуляет позицию черепашки и включает перо

Двигает черепашку вперед на 70 шагов

Поворачивает черепашку вправо на 90 градусов

△ **Программа для рисования домика**
Эти команды говорят черепашке, как рисовать домик. Выходит, что для такой простой программы нужно довольно много строк кода.

Три части программы

Автомат для рисования — довольно большая программа. Для удобства ее можно разбить на три части, каждая из которых выполняет свою задачу.

Function 1

△ **Контроллер черепашки**
Эта функция принимает от пользователя команду, состоящую из латинской буквы и числа, и преобразует ее в команду для черепашки.

Function 2

△ **Обработчик строки**
Программа принимает от пользователя строку с командами. Эта функция разбивает строку на отдельные команды, чтобы передать их контроллеру.

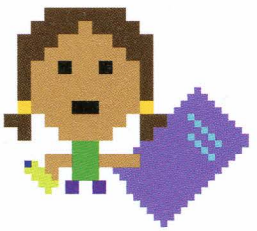
Основная программа

△ **Интерфейс пользователя**
Обработчику строки нужны входные данные. Интерфейс пользователя позволяет ввести строку с клавиатуры и передать ее обработчику.

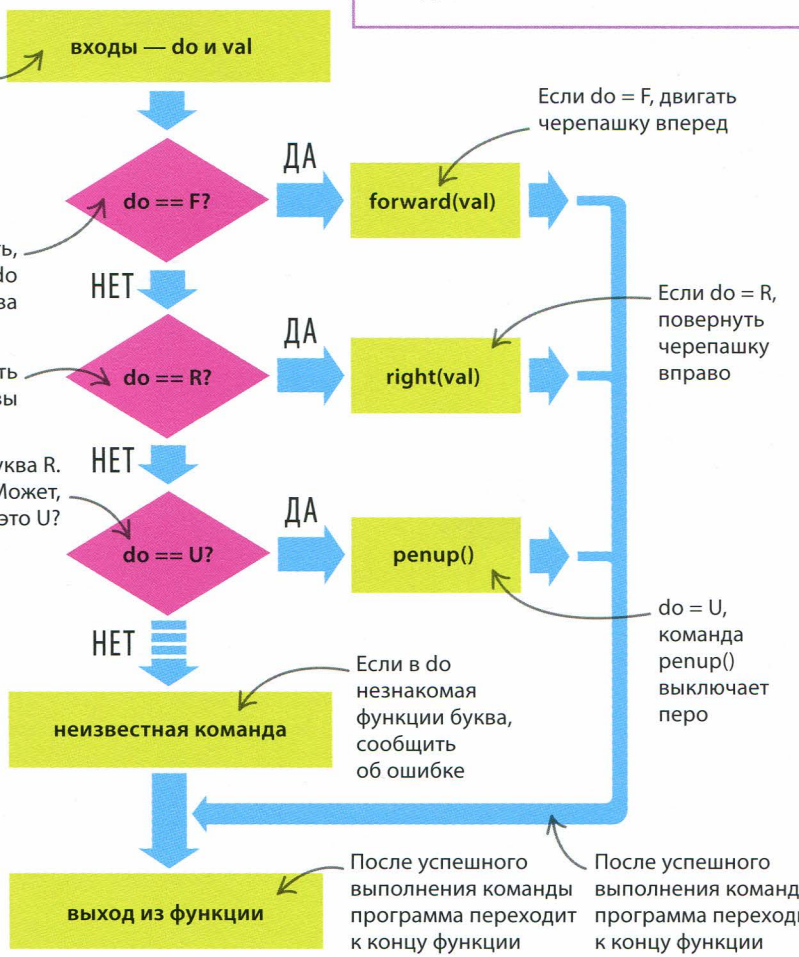
Рисуем блок-схему

Чтобы писать хороший код и делать меньше ошибок, программисты часто проектируют код на бумаге. Один из способов это делать — рисование блок-схемы, диаграммы шагов и решений, которым должна следовать программа.

1 На этой блок-схеме показано, как должна работать функция контроллера черепашки. Функция принимает букву (параметр do) и число (параметр val) и превращает их в команду для черепашки. Например, F и 100 нужно превратить в команду forward(100). Если встречена незнакомая буква, функция выдает сообщение об ошибке.



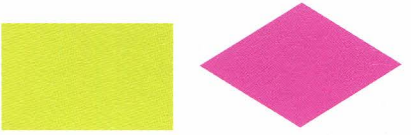
Каждой команде соответствуют две переменные: do (строка) говорит черепашке, что нужно сделать, а val (целое число) говорит, как это сделать — например, на сколько шагов двигаться



СОВЕТЫ ЭКСПЕРТА

Прямоугольники и ромбы

Блок-схемы состоят из прямоугольников и ромбов. В прямоугольниках записывают действия, выполняемые программой, а ромбы — это точки ветвления программы (моменты принятия решений).



СОВЕТЫ ЭКСПЕРТА

Буквенные команды

Контроллер черепашки использует эти буквы для обозначения команд черепашки.

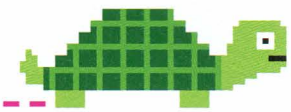
N = новый рисунок (очистка окна)
U/D = включить/выключить перо
F = вперед
B = назад
R = поворот направо
L = поворот налево

ЧЕРТЕЖНЫЙ АВТОМАТ

Контроллер черепашки

Первая часть программы — функция, управляющая черепашкой, которая выполняет по одной команде за раз, преобразуя параметры do и val в инструкции, понятные черепашке. Блок-схема этой функции изображена на предыдущей странице.

2 Это код контроллера черепашки. Он преобразует входные параметры do в направлении перемещения, а val — в углы и расстояния.



Начальная позиция черепашки

Указывает, чтобы черепашка начала рисовать

```
from turtle import *
def turtle_controller(do, val):
    do = do.upper()
    if do == 'F':
        forward(val)
    elif do == 'B':
        backward(val)
    elif do == 'R':
        right(val)
    elif do == 'L':
        left(val)
    elif do == 'U':
        penup()
    elif do == 'D':
        pendown()
    elif do == 'N':
        reset()
    else:
        print('Unrecognized command')
```

Загрузка команд управления черепашкой

Определяет do и val как входные параметры функции

Преобразует буквы в строке do в заглавные

Указывает, что значение F параметра do соответствует команде forward («вперед»)

Сверяет значение do с известными буквами — так же, как на блок-схеме

Указывает, чтобы черепашка перестала рисовать

Обнуляет позицию черепашки, перемещая ее в середину экрана

Это сообщение печатается, если значение do не соответствует известным командам

3 Вот несколько примеров использования контроллера черепашки. При каждом вызове он преобразует параметры do и val в понятные черепашке команды.

```
>>> turtle_controller('F', 100)
>>> turtle_controller('R', 90)
>>> turtle_controller('F', 50)
```

Вызов функции по имени

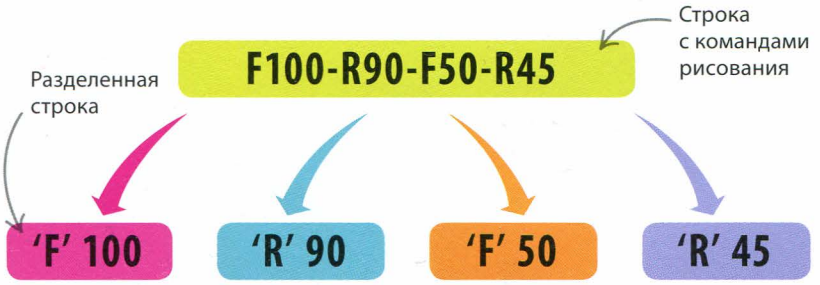
Эти значения do и val указывают черепашке сделать 100 шагов вперед

Поворачивает черепашку на 90 градусов вправо

Пишем псевдокод

Псевдокод — это еще один способ проектирования программ. «Псевдо» означает, что этот код нельзя запустить: это приблизительные инструкции, похожие на настоящий код, которые удобны для записи идей.

4 Пора спроектировать обработчик строки. Эта функция принимает строку с несколькими значениями do и val и разделяет ее на пары, состоящие из буквы и числа. Затем она по одной передает эти пары контроллеру черепашки.



5 Это обработчик строки, записанный в псевдокоде. Псевдокод помогает описать идею и структуру кода без необходимости думать о деталях.

```
функция string_artist(вход — строка команд):
    разделить строку на список с командами
    для каждой команды из списка:
        проверить, что это не пустая строка
            – если пустая, перейти к следующему элементу списка
        тип команды — это первая буква
        если есть еще символы
            – преобразовать их в число
        вызвать turtle_controller(тип команды, число)
```

Функция принимает от пользователя строку с командами (например, F100-R90)

Разделяет строку на список отдельных команд

Пропускает пустую команду

Берет первую букву как параметр do

Берет остальные знаки как параметр val

Передает команду контроллеру черепашки

СОВЕТЫ ЭКСПЕРТА

Понятный код

Программный код читают не только компьютеры, но и люди. Поэтому важно, чтобы твои программы были максимально легки для понимания.

Используй функции, чтобы разбить программу на части. Каждая функция должна выполнять только одну задачу.

Давай переменным говорящие имена: имя age_in_years («возраст_в_годах») более понятно, чем aiy.

Пиши комментарии (используя символ «#») с объяснениями, что происходит в программе. Откомментированный код читать легче.

Не используй символы, которые можно спутать с другими: заглавная O похожа на ноль, а строчная L напоминает заглавную i или единицу.

ЧЕРТЕЖНЫЙ АВТОМАТ

Пишем обработчик строки

Псевдокод с предыдущей страницы описывает функцию string_artist, которая разбивает строку на отдельные команды для контроллера черепашки. Теперь нужно сделать из псевдокода код на Python, и тут нам пригодится функция split().

6 Функция split() разбивает строку на список подстрок, используя один из символов как разделитель (в нашей программе это знак «-»).

Строка команд для рисования домика

```
>>> program = 'N-L90-F100-R45-F70-R90-F70-R45-F100-R90-F100'
>>> cmd_list = program.split('-')
>>> cmd_list
['N', 'L90', 'F100', 'R45', 'F70', 'R90', 'F70', 'R45', 'F100', 'R90', 'F100']
```

Функция split() разделяет строку на список с отдельными командами

7 Теперь перепиши псевдокод в командах Python. Используй функцию split() для разделения строки на отдельные команды.

```
def string_artist(program):
    cmd_list = program.split('-')
    for command in cmd_list:
        cmd_len = len(command)
        if cmd_len == 0:
            continue
        cmd_type = command[0]
        num = 0
        if cmd_len > 1:
            num_string = command[1:]
            num = int(num_string)
        print(command, ':', cmd_type, num)
        turtle_controller(cmd_type, num)
```

Указывает, что нужно разделить строку по каждому символу «-»

Цикл по списку строк: каждый элемент — это команда для черепашки

Узнает длину строки с командой

Если длина равна нулю (команда пустая), пропустить ее и перейти к следующей

Берет первый символ (отсчет символов ведется с 0) в качестве типа команды

Проверяет, есть ли в команде еще символы (число)

Преобразует строку в число

Берет оставшиеся символы команды, отрезая от нее первый символ

Печатает команду на экране, чтобы пользователь видел, что происходит

Передает команду черепашке

8 Если передать строку с командами обработчику строки, он напечатает полученные отдельные команды в окне консоли.

```
>>> string_artist('N-L90-F100-R45-F70-R90-F70-R45-F100-R90-F100')
N : N 0
L90 : L 90
F100 : F 100
R45 : R 45
F70 : F 70
R90 : R 90
F70 : F 70
R45 : R 45
F100 : F 100
R90 : R 90
F100 : F 100
```

Очищает окно и перемещает черепашку в его середину

Команды разделены символами «-»

В команде «F100» F — это тип команды, а 100 — это ее параметр

Поворачивает черепашку на 45 градусов перед рисованием крыши

Рисует правую сторону крыши

Поворачивает черепашку направо на 90 градусов перед рисованием нижней части домика

9 Когда каждая команда в строке, переданной обработчику строки, получена, распознана и выполнена, в графическом окне появится изображение домика.

ЗАПОМНИ

Команды

Вспомним команды, которые понимает эта программа. Некоторые из них состоят из одной буквы, а некоторые содержат число с количеством шагов или градусов поворота. При каждом вызове обработчик строки будет добавлять часть рисунка — до очистки окна командой N.

N = новый рисунок

D/U = включить/выключить перо

F = вперед

B = назад

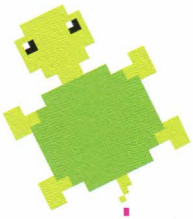
R = поворот направо

L = поворот налево

ЧЕРТЕЖНЫЙ АВТОМАТ

И наконец, добавим интерфейс пользователя

Чтобы программа чертежного автомата была удобнее, ей нужен интерфейс пользователя, позволяющий вводить строки команд с клавиатуры.



10 Этот код создает всплывающее окно, в котором пользователь может вводить команды. Цикл while True позволяет делать это многократно.

Тройные кавычки (''') указывают, что все последующие символы вплоть до следующих тройных кавычек (даже переносы строк) — это части одной строки

```
instructions = '''Enter a program for the turtle:
eg F100-R45-U-F100-L45-D-F100-R90-B50
N = New drawing
U/D = Pen Up/Down
F100 = Forward 100
B50 = Backwards 50
R90 = Right turn 90 deg
L45 = Left turn 45 deg'''
screen = getscreen()
while True:
    t_program = screen.textinput('Drawing Machine', instructions)
    print(t_program)
    if t_program == None or t_program.upper() == 'END':
        break
    string_artist(t_program)
```

Напоминает пользователю команды, которые можно ввести

Конец строки

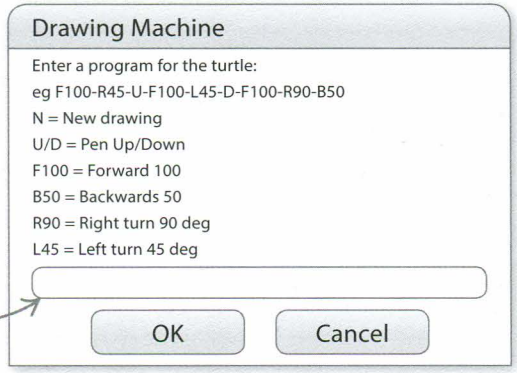
Получает данные, необходимые для создания всплывающего окна

Говорит, что нужно отобразить в окне

Останавливает программу, если введено слово END или нажата кнопка Cancel

Передает введенную строку обработчику строк

11 В этом окне, которое появится поверх графического окна с черепашкой, пользователь сможет ввести строку с командами.



Введи строку здесь и кликни OK, чтобы начать рисовать

△ Управление черепашкой

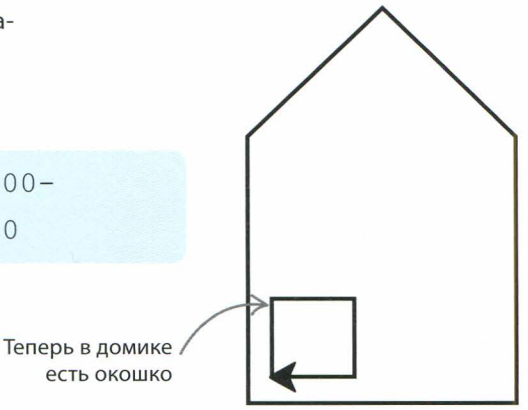
С помощью этой программы управлять черепашкой проще, к тому же программу не нужно перезапускать, чтобы нарисовать новую картинку.

12 Программа чертежного автомата годится не только для рисования контуров. Отключение пера перед перемещением черепашки дает возможность рисовать внутри контура. Попробуй запустить программу и ввести эту строку.

```
N-L90-F100-R45-F70-R90-F70-R45-F100-R90-F100-
B10-U-R90-F10-D-F30-R90-F30-R90-F30-R90-F30
```

Отключает перо черепашки, чтобы оно не оставляло за собой след

Включает перо, чтобы нарисовать окошко

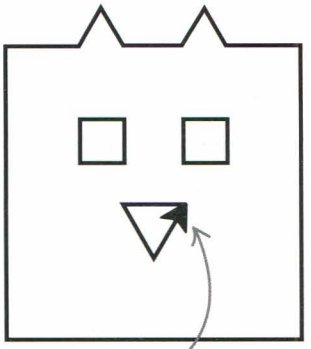


Попробуем кое-что другое

Теперь, зная, как добавлять детали внутри контура, можно развлечься как следует. Введи эту строку, чтобы нарисовать голову совы.

```
N-F100-L90-F200-L90-F50-R60-F30-L120-F30-R60-F40-
R60-F30-L120-F30-R60-F50-L90-F200-L90-F100-L90-U-
F150-L90-F20-D-F30-L90-F30-L90-F30-L90-F30-R90-U-
F40-D-F30-R90-F30-R90-F30-R90-F30-L180-U-F60-R90-
D-F40-L120-F40-L120-F40
```

Перо здесь выключается трижды — чтобы нарисовать нос и два глаза



ЗАПОМНИ

Твои успехи

Чтобы запрограммировать чертежный автомат, тебе пришлось выполнить несколько более простых задач:

использовать блок-схему для проектирования функции, обозначив действия и точки ветвления программы;

написать псевдокод для проектирования функции, перед тем как писать код на Python;

написать функцию turtle_controller, которая по заданным букве и числу определяет, какую команду рисования выполнить;

написать функцию string_artist, которая создает рисунок на основе строки с инструкциями;

создать интерфейс, позволяющий вводить команды рисования с клавиатуры.

Ошибки и отладка

СМОТРИ ТАКЖЕ
❮ 94–95 Ошибки
❮ 122–123 Циклы в Python
Что дальше? 176–177 ❯

Идеальных программистов не бывает, и в большинстве программ изначально имеются ошибки. Выявление этих ошибок называется отладкой.

Виды ошибок

В программах бывает три основных типа ошибок: ошибки синтаксиса, ошибки выполнения и логические ошибки. Некоторые из них легко обнаружить, с другими придется сложнее, но есть способы найти и исправить все.

Ключевое слово Python — for, а не fir

```
fir i in range(5):
print(i)
```

Вызовет ошибку, поскольку на ноль делить нельзя

```
a = 0
print(10 / a)
```

age не может быть одновременно меньше 5 и больше 8

```
if age < 5 and age > 8:
    print('Free ticket!')
```

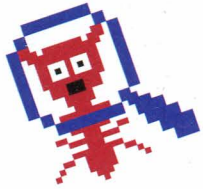
△ Легко найти
Синтаксические ошибки — это ошибки в написании текста программы, такие как опечатки в ключевых словах, отсутствующие скобки, неверные отступы.

△ Сложнее найти
Ошибки выполнения проявляются во время работы программы — например, при попытке сложить число и строку или делении на ноль.

△ Трудно найти
Логические ошибки — это ошибки в том, как продуман код. Например, это использование «<» вместо «>» или операция вычитания там, где должно быть сложение.

Находим и исправляем ошибку

Синтаксические ошибки найти легко, так как IDLE подсвечивает их красным при запуске программы. С ошибками выполнения и логическими ошибками все немного сложнее.



1 Подопытная программа

Эта программа должна печатать сумму всех чисел от 1 до числа, которое хранится в переменной top_num.

```
top_num = 5
total = 0
for n in range(top_num):
    total = total + n
print('Sum of numbers 1 to', top_num, 'is', total)
```

Наибольшее значение в последовательности складываемых чисел

Печатает на экране результат работы программы

2 Результат

Суммирование чисел должно дать (1 + 2 + 3 + 4 + 5) = 15, но программа печатает число 10. Разберемся почему.

```
Sum of numbers 1 to 5 is 10
```

Тут должно быть 15, а не 10

3 Добавим print и input()

Программа не сообщает, что она делает на каждом шаге вычислений. Если добавить команду print(), станет ясно, что происходит. Команда input() ожидает нажатия Enter перед следующим повтором цикла.

```
top_num = 5
total = 0
for n in range(top_num):
    total = total + n
    print('DEBUG: n=', n, 'total=', total)
    input()
print('Sum of numbers 1 to', top_num, 'is', total)
```

Печатает значение переменной цикла и суммы в текущий момент

4 Проблема

В цикле складываются числа от 0 до 4, а не от 1 до 5. Причина в том, что цикл for ведет отсчет с 0 (если не указано другое число) и останавливается за 1 до последнего числа в диапазоне.

```
DEBUG: n= 0 total= 0
DEBUG: n= 1 total= 1
DEBUG: n= 2 total= 3
DEBUG: n= 3 total= 6
DEBUG: n= 4 total= 10
Sum of numbers 1 to 5 is 10
```

В действительности это сумма чисел от 0 до 4, а не от 1 до 5

5 Исправляем строку с ошибкой

Диапазон чисел должен быть от 1 до top_num + 1, чтобы в цикле складывались числа от 1 до top_num (5).

```
top_num = 5
total = 0
for n in range(1, top_num + 1):
    total = total + n
    print('DEBUG: n=', n, 'total=', total)
    input()
print('Sum of numbers 1 to', top_num, 'is', total)
```

Новый диапазон для суммы — от 1 до top_num (на 1 меньше, чем top_num + 1)

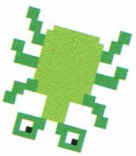
6 Правильный результат

Команда print подтверждает, что теперь программа складывает числа от 1 до 5 и вычисляет верный результат. Ошибка исправлена!

```
DEBUG: n= 1 total= 1
DEBUG: n= 2 total= 3
DEBUG: n= 3 total= 6
DEBUG: n= 4 total= 10
DEBUG: n= 5 total= 15
Sum of numbers 1 to 5 is 15
```

Когда $n = 3$, сумма равна (1 + 2 + 3)

Теперь появится верный результат



Алгоритмы

Алгоритм — это набор инструкций по выполнению задачи. Некоторые алгоритмы эффективней других и требуют меньше времени или ресурсов. Для простых задач вроде сортировки чисел тоже можно использовать разные алгоритмы.

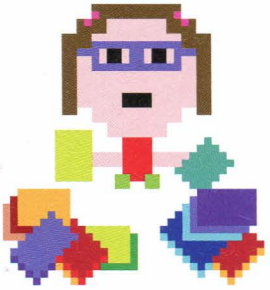
СМОТРИ ТАКЖЕ

❮ 16–17 Думай как компьютер

Библиотеки 152–153 ❯

Сортировка вставками

Представь, что в школе тебе надо отсортировать листки с выставленными оценками от меньших оценок к бóльшим. Сортировка вставками выделяет вверху стопки место под отсортированные данные и в нужном порядке вставляет туда каждый неотсортированный листок.

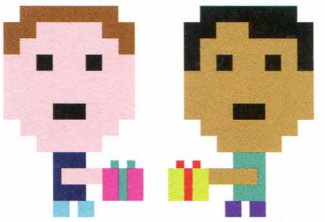


△ **Сортировка по порядку**
Сортировка вставками берет каждый листок и вставляет его в нужное место.

▽ **Как это работает**
Сортировка вставками проходит через каждый из этих шагов, сортируя числа быстрее, чем на это способен человек.

Сортировка выбором

Сортировка выбором работает иначе, чем сортировка вставками. Она не сдвигает элементы, а меняет пары элементов местами. Каждая смена ставит одно из чисел на его конечную (отсортированную) позицию.



△ **Обмен позиций**
Обмен позиций — быстрая операция, и она не влияет на остальные элементы списка.

СОВЕТЫ ЭКСПЕРТА

Сортировка в Python

Есть много алгоритмов сортировки, и у каждого свои преимущества и недостатки. Функция sort() в Python использует алгоритм Timsort, названный по имени его автора, Тима Петерса. Он основан на двух других алгоритмах: сортировке вставками и сортировке слиянием. Чтобы посмотреть, как он работает, введи эти команды.

a — список несортированных чисел

```
>>> a = [4, 9, 3, 8, 2, 6, 1, 5, 7]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Вызов функции sort()

Теперь числа в *a* отсортированы

Библиотеки

СМОТРИ ТАКЖЕ

Создание окон **154–155 ›**

Цвета и координаты **156–157 ›**

Написание нового программного кода требует времени, поэтому удобно повторно использовать части готовых программ. Эти фрагменты кода можно хранить в так называемых библиотеках.

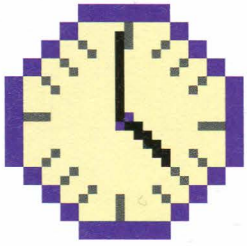
Модули из стандартной библиотеки

В составе Python есть стандартная библиотека, содержащая много фрагментов готового кода. Чтобы сделать Python еще мощнее, в него можно загружать модули — отдельные части библиотеки.

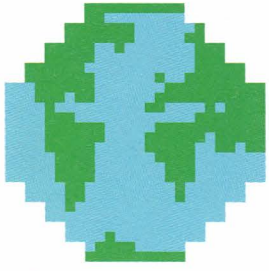
◁ **Batteries included**
Девиз Python — «батарейки в комплекте». Это значит, что в комплекте с Python идет много готового кода.

◁ **Random**
Этот модуль выбирает случайные числа или переставляет элементы списка в случайном порядке.

▽ **Turtle**
Этот модуль служит для рисования линий и фигур на экране.

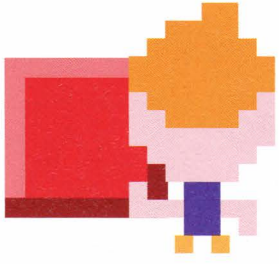


△ **Time**
Этот модуль подскажет текущие время и дату и может вычислять даты: например, какой день будет три дня спустя?

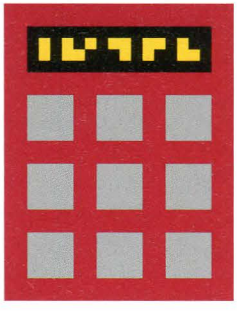


△ **Socket**
Этот модуль помогает компьютерам подключаться друг к другу по сети и через интернет.

▽ **Tkinter**
Нужен для создания кнопок, окон и других графических элементов для взаимодействия пользователя с программой.



▷ **Math**
Используй этот модуль для сложных математических вычислений.



Загрузка модулей

Перед тем как использовать модуль в своей программе, его надо загрузить: это позволит обращаться к хранящимся в нем фрагментам кода. Для загрузки модулей служит команда import («импортировать»). Python может загружать модули несколькими способами.

```
import random

random.randint(1, 6)
random.choice(my_list)
```

Перед каждой функцией указано имя модуля

◁ import random

Этот способ загрузки требует, чтобы имя модуля было указано перед вызываемым фрагментом кода. Такую программу легче читать, потому что сразу видно, какому модулю принадлежит код.

▷ from random import *

Такая загрузка модуля хорошо подходит для небольших программ. Однако чем больше программа, тем сложнее понять, из какого модуля та или иная функция.

Загружает все функции из модуля random

```
from random import *

randint(1, 6)
choice(my_list)
```

Тут не указано, какому модулю принадлежит функция

Загружает только функцию randint

```
from random import randint

randint(1, 6)
```

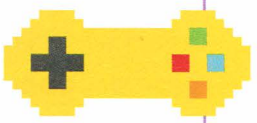
Доступна лишь функция randint

◁ from random import randint

Можно загрузить из модуля только одну функцию. Если это единственная функция, которая тебе нужна, такой способ эффективней загрузки модуля целиком.

СОВЕТЫ ЭКСПЕРТА

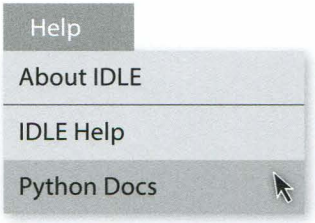
Pygame



Pygame — это Python-библиотека для создания видеоигр, позволяющая работать со звуком и специальной игровой графикой. Ты сможешь использовать Pygame, когда изучишь основы программирования на Python, изложенные в этой книге.

Справка

Не понимаешь, для чего нужен модуль или какие в нем есть функции? Это можно узнать из справки по библиотеке Python. Просто кликни по имени интересующего модуля. О доступных библиотеках, модулях и функциях полезно знать, чтобы не тратить время на разработку уже существующего кода.



◁ Справка

Кликни Help! («Помощь») вверху окна IDLE и выбери Python Docs. Откроется окно с разнообразной справочной информацией.

Создание окон

Многие программы используют такие элементы управления, как окна и кнопки. Это называется графическим интерфейсом пользователя и обозначается аббревиатурой GUI.

СМОТРИ ТАКЖЕ

Цвета и координаты **156–157 ›**

Рисование фигур **158–159 ›**

Изменение рисунков **160–161 ›**

Создадим простое окно

Основа GUI — это окно, в котором будут располагаться все остальные элементы. Для создания простого окна подойдет модуль Tkinter из стандартной библиотеки Python.

1 Введем код

Этот код загружает библиотечный модуль Tkinter и создает новое окно.

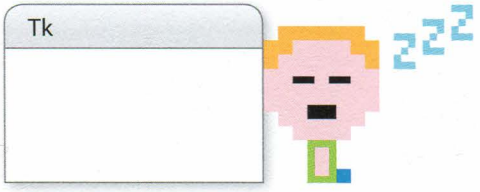
```
from tkinter import *
window = Tk()
```

Загружает Tkinter из библиотеки

Создает окно Tkinter

2 Появилось окно Tkinter

Запусти введенный код, и на экране появится окно. Выглядит оно скучновато, но это лишь первый этап создания GUI.



Добавим кнопки

Сделай GUI интерактивным, разместив в окне кнопки. При клике по каждой из кнопок будут отображаться разные сообщения.

1 Создай две кнопки

Введи этот код, чтобы создать простое окно с двумя кнопками.

```
from tkinter import *
def bAaction():
    print('Thank you!')
def bBaction():
    print('Ouch! That hurt!')
window = Tk()
buttonA = Button(window, text='Press me!', command=bAaction)
buttonB = Button(window, text='Don\'t press!', command=bBaction)
buttonA.pack()
buttonB.pack()
```

Это сообщение появится, если нажата кнопка A

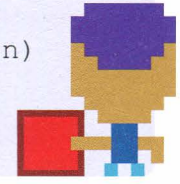
Это сообщение появится, если нажата кнопка B

Надпись на кнопке A

Указывает, какую функцию вызвать при нажатии кнопки

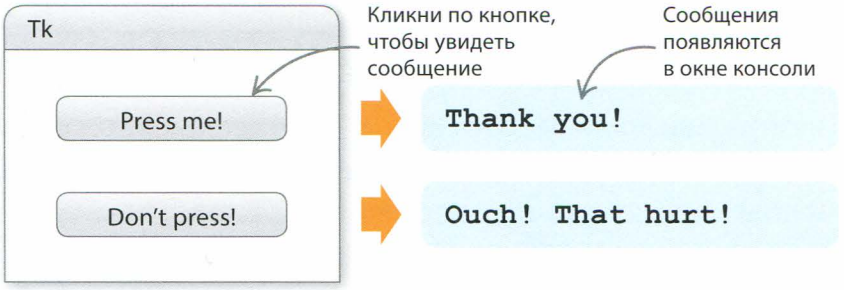
Надпись на кнопке B

Указание расположить кнопки внутри окна



2 Кликай по кнопкам, чтобы увидеть сообщения

После запуска программы появится окно с двумя кнопками, при нажатии на которые в окне консоли будут печататься два разных сообщения. Это GUI, реагирующий на команды пользователя.



Бросаем кубик

Создадим Tkinter-GUI для несложной программы, которая имитирует выбрасывание шестигранного кубика.

1 Пишем имитатор кубика

В этой программе создается кнопка, которая при нажатии вызывает функцию roll() для отображения случайного числа от 1 до 6.

```
from tkinter import *
from random import randint
def roll():
    text.delete(0.0, END)
    text.insert(END, str(randint(1,6)))
window = Tk()
text = Text(window, width=1, height=1)
buttonA = Button(window, text='Press to roll!', command=roll)
text.pack()
buttonA.pack()
```

Загружает функцию randint из модуля random

Очищает текстовое поле и печатает в нем случайное число от 1 до 6

Создает текстовое поле для отображения случайного числа

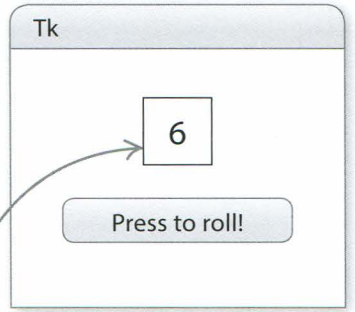
Указывает, какую функцию запустить при нажатии на кнопку

Надпись на кнопке

Размещает текстовое поле и кнопку в окне

2 Нажми кнопку, чтобы бросить кубик

Запусти программу, кликни по кнопке и посмотри, что получится. Программу легко изменить, чтобы имитировать кубик с 12 гранями или бросание монетки.



СОВЕТЫ ЭКСПЕРТА

Просто и понятно

Разрабатывая GUI, старайся не сбивать пользователя с толку, размещая на экране слишком много кнопок. Давай кнопкам понятные названия, чтобы было сразу ясно, для чего они нужны.

Цвета и координаты

Изображения на экране монитора компьютера состоят из маленьких цветных точек, которые называют пикселями. Чтобы нарисовать в программе картинку, компьютеру нужно объяснить, какого цвета должен быть каждый пиксель.

СМОТРИ ТАКЖЕ

- ❮ 154–155 Создание окон
- Рисование фигур 158–159 ❯
- Изменение рисунков 160–161 ❯

Выбор цвета

Важно указывать цвета способом, который понятен компьютеру. Для этого в Tkinter есть удобный инструмент.

1 Инструмент выбора цвета

Введи этот код в окне консоли, чтобы запустить инструмент Tkinter для выбора цвета.

Загружает все функции Tkinter

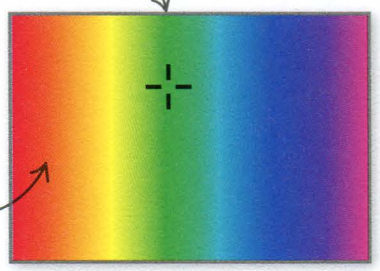
```
>>> from tkinter import *
>>> t = Tk()
>>> colorchooser.askcolor()
```

Американский вариант слова colour

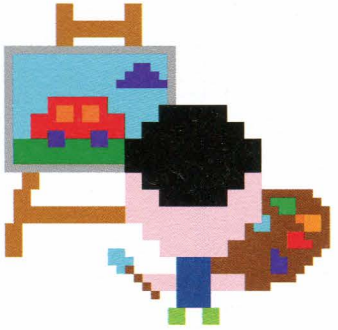
2 Выбираем цвет

Появится окно выбора цвета. Выбери нужный цвет и кликни по кнопке OK.

Чтобы выбрать цвет, кликни по нему



Это окно упрощает точный выбор цвета



3 Цветовые значения

После выбора цвета в окне консоли появится список чисел. Это значения красной, зеленой и синей составляющих, которые в смешении дают выбранный цвет.

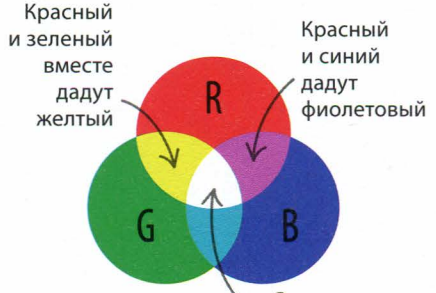
```
((60.234, 190.742, 52.203), '#3cbe34')
```

Красный — Зеленый — Синий — Код цвета в шестнадцатеричной системе (см. с. 182–183)

СОВЕТЫ ЭКСПЕРТА

Смешение цветов

Каждый пиксель экрана может светиться красным, зеленым и синим цветом. Смешивая эти цвета, можно получить любой другой цвет.



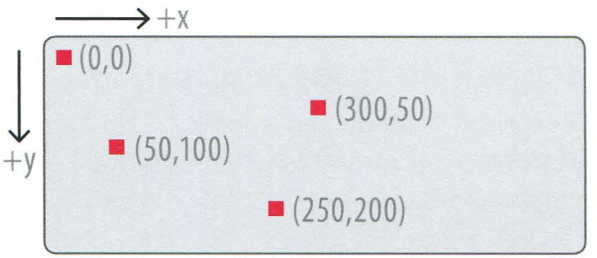
Рисование на холсте

Чтобы рисовать картинки в Python, нужно создать чистую область для рисования, которую называют холстом. Для указания, в каком месте холста рисовать, используются координаты X и Y.

СОВЕТЫ ЭКСПЕРТА

Координаты

В Tkinter X-координаты отсчитываются слева направо, а Y-координаты — сверху вниз. Точка (0, 0) — в верхнем левом углу холста.



1 Пишем графическую программу

Запусти этот код, чтобы создать окно и поместить в него холст. Затем программа будет случайным образом рисовать на холсте круги.

```
from random import *
from tkinter import *
size = 500
window = Tk()
canvas = Canvas(window, width=size, height=size)
canvas.pack()
while True:
    col = choice(['pink', 'orange', 'purple', 'yellow'])
    x0 = randint(0, size)
    y0 = randint(0, size)
    d = randint(0, size/5)
    canvas.create_oval(x0, y0, x0 + d, y0 + d, fill=col)
    window.update()
```

Загружает функции randint и choice из модуля random

Загружает функции Tkinter

Переменная size задает размеры холста

Создает в окне холст

Бесконечный цикл, чтобы программа рисовала круги без остановки

Выбирает из списка случайный цвет

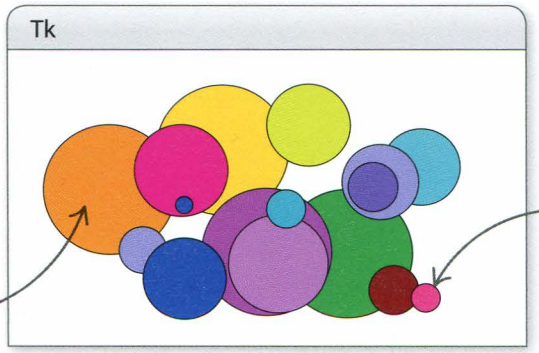
Рисует в случайном месте холста круг случайного размера

Рисует круг

Заполнить круг выбранным цветом (col)

2 Цветной холст

Запусти программу, и она будет рисовать на холсте круги.



У кругов случайный размер

Круги появляются в случайных координатах

Рисование фигур

Кроме добавления в GUI окон, кнопок и цветов, с помощью Tkinter можно рисовать на экране геометрические фигуры.

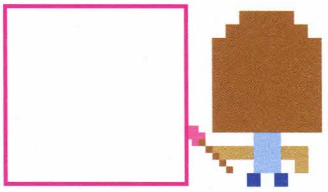
СМОТРИ ТАКЖЕ

Изменение рисунков **160–161 ›**

Реакция на события **162–163 ›**

Рисование простых фигур

С помощью прямоугольников и овалов можно изобразить множество вещей. Введи эти команды, рисующие фигуры на холсте.



```
>>> from tkinter import *
>>> window = Tk()
>>> drawing = Canvas(window, height=500, width=500)
>>> drawing.pack()
>>> rect1 = drawing.create_rectangle(100, 100, 300, 200)
>>> square1 = drawing.create_rectangle(30, 30, 80, 80)
>>> oval1 = drawing.create_oval(100, 100, 300, 200)
>>> circle1 = drawing.create_oval(30, 30, 80, 80)
```

Создает холст для рисования

Задает размер холста

Рисует прямоугольник

Задает (в координатах) позицию и размер прямоугольника

Нарисовав прямоугольник со сторонами одинаковой длины, получим квадрат

Рисует круг

Задает позицию и размер круга

Рисование по координатам

Чтобы указать, где именно рисовать фигуры, используют координаты. Первое число (X) обозначает расстояние на экране по горизонтали, а второе (Y) — по вертикали.

```
>>> drawing.create_rectangle(50, 50, 250, 350)
```

Имя холста

Координаты верхнего левого угла

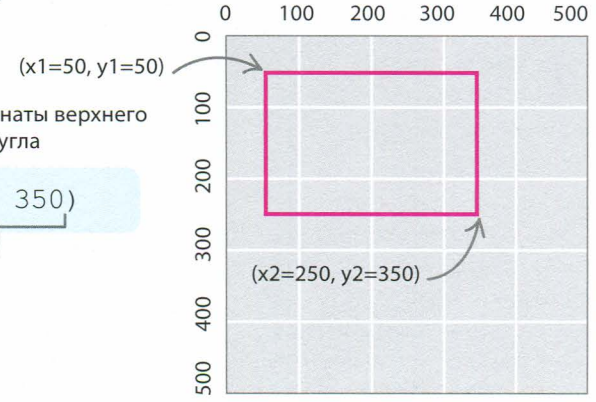
Координаты правого нижнего угла

△ **Задание координат**

Первые два числа задают координаты верхнего левого угла прямоугольника. Вторые два — правого нижнего угла.

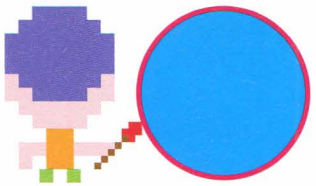
▽ **Координатная сетка**

Верхний левый угол прямоугольника — в координатах (50, 50). Нижний правый — в координатах (250, 350).



Добавим цвет

Если нужно рисовать разноцветные фигуры, в программе можно задавать цвет контура и его заполнения.



Рисует синий круг, обведенный по контуру красным

```
>>> drawing.create_oval(30, 30, 80, 80, outline='red', fill='blue')
```

Рисуем инопланетянина

Совмещая разные фигуры, можно изобразить практически что угодно. Вот инструкции по рисованию инопланетянина с помощью овалов, линий и треугольников.

1 Рисуем инопланетянина

Для каждой части инопланетянина нужно определить тип фигуры, размер, позицию на холсте и цвет. Каждой фигуре дается уникальный номер, который можно хранить в переменной.

```
from tkinter import *
window = Tk()
window.title('Alien')
c = Canvas(window, height=300, width=400)
c.pack()
body = c.create_oval(100, 150, 300, 250, fill='green')
eye = c.create_oval(170, 70, 230, 130, fill='white')
eyeball = c.create_oval(190, 90, 210, 110, fill='black')
mouth = c.create_oval(150, 220, 250, 240, fill='red')
neck = c.create_line(200, 150, 200, 130)
hat = c.create_polygon(180, 75, 220, 75, 200, 20, fill='blue')
```

Отображает в заголовке окна слово Alien

Создает холст

Рисует зеленый овал для туловища

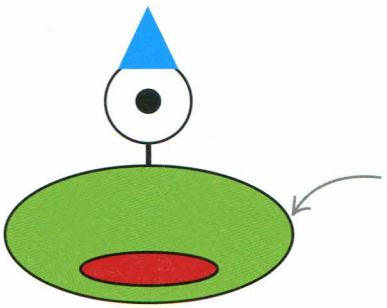
Рисует черную точку зрачка

Рисует овал для рта

Рисует синий треугольник для шляпы

2 Поздоровайся с инопланетянином

Запусти код, чтобы увидеть инопланетянина. У него зеленое туловище, красный рот и глаз на стебельке, и он носит синюю шляпу.



Готовый инопланетянин

Изменение рисунков

Изображение на холсте не обязано всегда быть одинаковым. Из программы можно менять его вид или передвигать его по экрану.

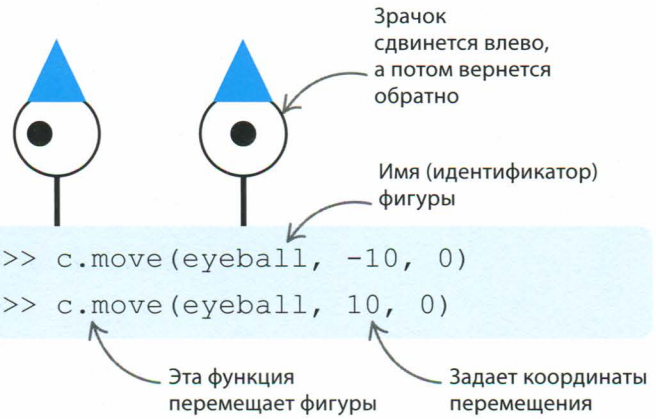
СМОТРИ ТАКЖЕ

❮ 158–159 Рисование фигур

Реакция на события 162–163 ❯

Перемещение фигур

Чтобы передвинуть фигуру по холсту, нужно указать компьютеру, что двигать (имя или число, обозначающее фигуру), и место, куда ее надо переместить.



◁ **Перемещение зрачка**
Введи этот код в окне консоли, чтобы сдвинуть зрачок влево, а потом вернуть обратно.

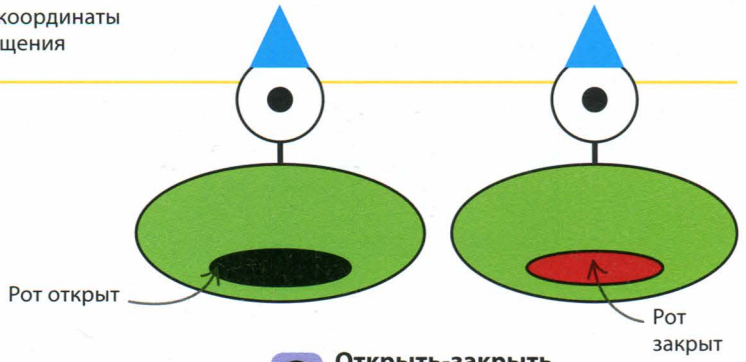
ЗАПОМНИ

Говорящие имена

Фигуры на холсте лучше называть понятными именами. На этой странице им даны имена вроде eyeball (зрачок) и mouth (рот), поэтому код легко читается.

Изменение цвета

Изменяя цвет овала, можно сделать так, будто рот открывается и закрывается.



1 Пишем код
Введи этот код, чтобы определить две функции, которые будут «открывать» и «закрывать» рот.

Функция itemconfig() меняет свойства уже нарисованных фигур

Открытый рот рисуем черным

```
def mouth_open():
    c.itemconfig(mouth, fill='black')
def mouth_close():
    c.itemconfig(mouth, fill='red')
```

Имя фигуры

Закрытый рот рисуем красным

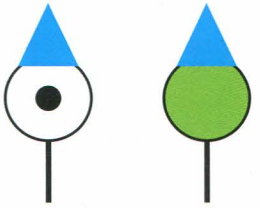
2 Открыть-закрыть
Чтобы открыть и закрыть рот инопланетянина, введи в окне консоли эти команды.

```
>>> mouth_open()
>>> mouth_close()
```

Введи эти команды, чтобы инопланетянин открыл и закрыл рот

Скрыть и показать

С помощью функции itemconfig() фигуры можно прятать. Если спрятать зрачок и через мгновение снова показать, создастся впечатление, что инопланетянин подмигнул.



◁ **Мигающий инопланетянин**
Чтобы инопланетянин подмигнул, нужно скрыть его зрачок и закрасить глаз зеленым.

1 **Определим функции для мигания**
Этот код определяет две функции для мигания.

Меняет цвет глаза на зеленый

Имя фигуры

```
def blink():
    c.itemconfig(eye, fill='green')
    c.itemconfig(eyeball, state=HIDDEN)
def unblink():
    c.itemconfig(eye, fill='white')
    c.itemconfig(eyeball, state=NORMAL)
```

Снова делает глаз белым

Прячет зрачок

Показывает зрачок

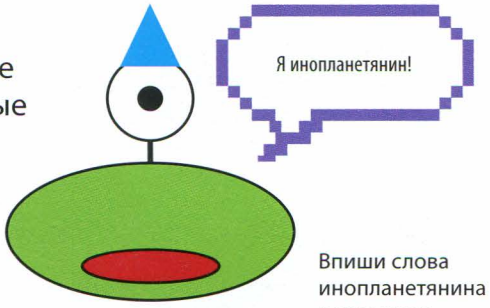
2 **Подмигивание**
Введи этот код в окне консоли, чтобы инопланетянин подмигнул.

```
>>> blink()
>>> unblink()
```

Функция unblink() снова показывает глаз со зрачком

Говорящий персонаж

Чтобы инопланетянин «заговорил», отобразим на экране сообщение. Можно сделать так, чтобы он говорил разные фразы в ответ на команды пользователя.



1 **Добавим текст**
Этот код добавляет к изображению инопланетянина текст и определяет функцию для похищения его шляпы.

Позиция текста на холсте

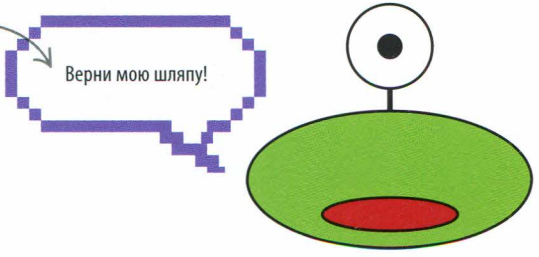
Впиши слова инопланетянина в кавычках

```
words = c.create_text(200, 280, text='I am an alien!')
def steal_hat():
    c.itemconfig(hat, state=HIDDEN)
    c.itemconfig(words, text='Give my hat back!')
```

Прячет шляпу

Когда шляпа исчезнет, инопланетянин потребует ее вернуть

Когда шляпа исчезнет, возникнет новое сообщение



2 **Крадем шляпу**
Введи этот код в окне консоли и смотри, что произойдет.

Введи, чтобы украсть шляпу

```
>>> steal_hat()
```

Реакция на события

При нажатии клавиш или перемещении мышки компьютер получает сигналы, которые называются событиями. Программы могут на эти события реагировать.

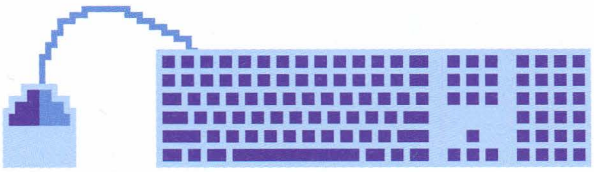
СМОТРИ ТАКЖЕ

❮ 158–159 Рисование фигур

❮ 160–161 Изменение рисунков

Имена событий

Устройства ввода, такие как мышка или клавиатура, могут генерировать множество разных событий. В Tkinter есть имена для каждого из них.



События мышки

- **<Button-1>** — Клик левой кнопкой мышки
- **<Button-3>** — Клик правой кнопкой мышки

События клавиатуры

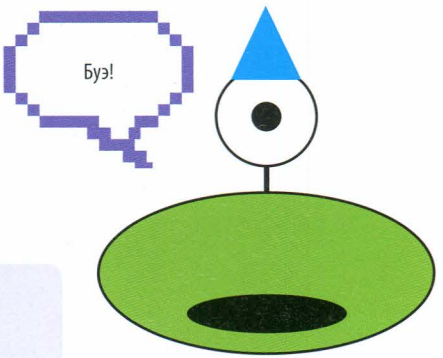
- **<Right>** — Нажата стрелка вправо
- **<Left>** — Нажата стрелка влево
- **<space>** — Нажата клавиша «пробел»
- **<Up>** — Нажата стрелка вверх
- **<Down>** — Нажата стрелка вниз
- **<KeyPress-a>** — Нажата клавиша A. Сюда можно подставлять разные буквы

События мышки

Чтобы программа реагировала на события мышки, привяжи к событию функцию. Определим функцию burp и привяжем ее к событию <Button-1>.

```
window.attributes('-topmost', 1)
def burp(event):
    mouth_open()
    c.itemconfig(words, text='Burp!')
c.bind_all('<Button-1>', burp)
```

- `window.attributes('-topmost', 1)` — Выводит окно Tkinter на передний план экрана
- `def burp(event):` — Определяет функцию burp
- `c.bind_all('<Button-1>', burp)` — Привязывает функцию burp к клику левой кнопкой мышки



△ **Рыгающий инопланетянин**
Нажми левую кнопку мышки, и инопланетянин «рыгнет» благодаря вызову функции burp («отрыжка»).

События клавиатуры

Также можно привязать функции к нажатиям клавиш на клавиатуре. Введи этот код, чтобы инопланетянин подмигивал при нажатии клавиш A и Z.

```
def blink2(event):
    c.itemconfig(eye, fill='green')
    c.itemconfig(eyeball, state=HIDDEN)
def unblink2(event):
    c.itemconfig(eye, fill='white')
    c.itemconfig(eyeball, state=NORMAL)
c.bind_all('<KeyPress-a>', blink2)
c.bind_all('<KeyPress-z>', unblink2)
```

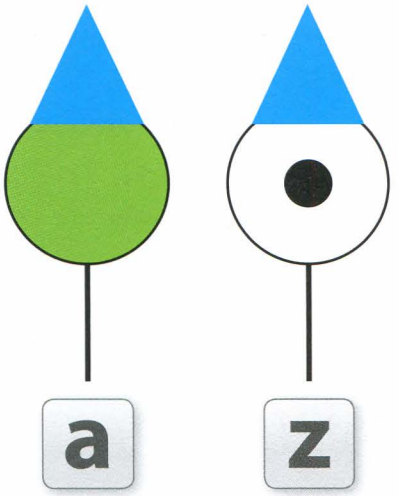
Делает глаз зеленым (закрытым)

Прячет зрачок

Показывает зрачок

Привязывает функции к событиям

Привязывает функцию unblink2 к клавише Z



△ **Инопланетянин, подмигни!**
Когда программа запущена, нажатие клавиши A закрывает глаз, а нажатие Z открывает его.

Перемещение с помощью клавиш

Нажатия клавиш можно использовать, чтобы перемещать фигуры. Этот код связывает стрелки на клавиатуре с движениями зрачка инопланетянина.

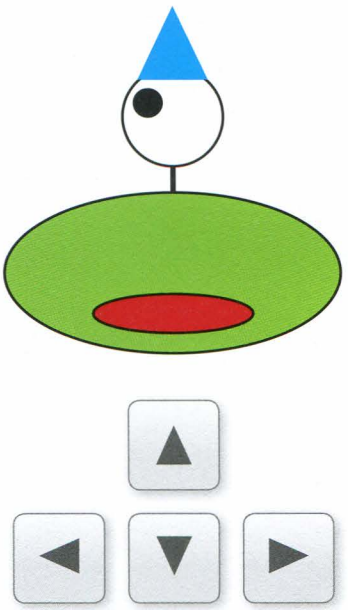
```
def eye_control(event):
    key = event.keysym
    if key == "Up":
        c.move(eyeball, 0, -1)
    elif key == "Down":
        c.move(eyeball, 0, 1)
    elif key == "Left":
        c.move(eyeball, -1, 0)
    elif key == "Right":
        c.move(eyeball, 1, 0)
c.bind_all('<Key>', eye_control)
```

Узнает имя нажатой клавиши

Зрачок движется вверх при нажатии стрелки вверх

Зрачок движется влево при нажатии стрелки влево

Привязывает функцию eye_control к нажатию любой клавиши



△ **Управление зрачком**
Зрачок двигается в сторону, соответствующую клавише со стрелкой.

ПРОЕКТ 7

Охотник за пузырями

Вооружись всеми усвоенными в этой главе знаниями, чтобы создать игру. Это большой проект, поэтому делай его по этапам и почаще сохраняй программу. Постарайся понять, как работает каждая часть кода, прежде чем переходить к следующему этапу. В результате получится игра, которую можно показать друзьям.

СМОТРИ ТАКЖЕ

- 154–155 Создание окон
- 156–157 Цвета и координаты
- 158–159 Рисование фигур

Цель игры

Перед тем как писать программу, подумай, какой должна быть игра и как она будет работать. Вот главные правила этой игры.

Игрок управляет подводной лодкой.

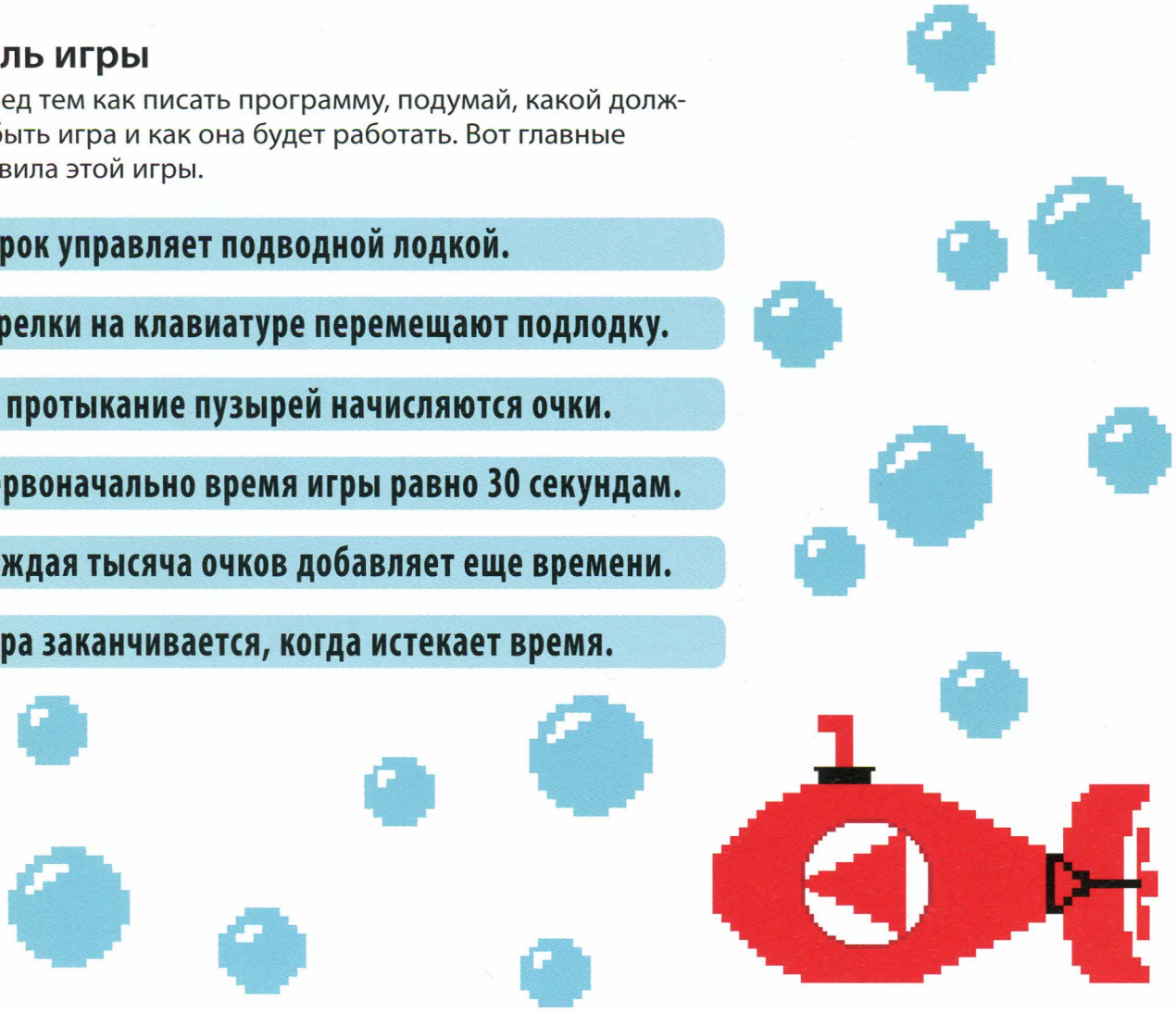
Стрелки на клавиатуре перемещают подлодку.

За протыкание пузырей начисляются очки.

Первоначально время игры равно 30 секундам.

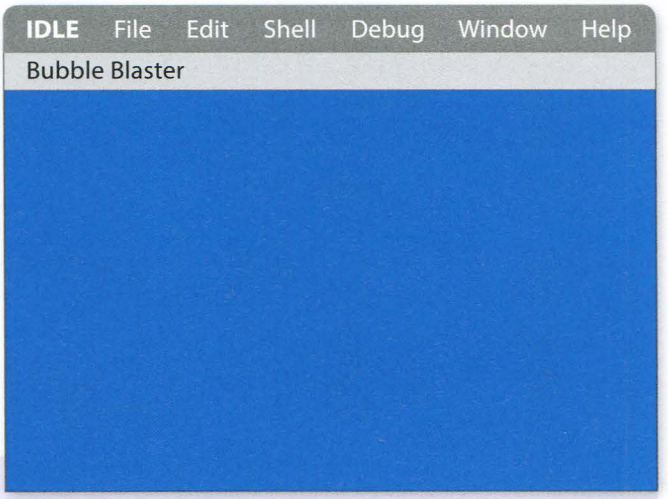
Каждая тысяча очков добавляет еще времени.

Игра заканчивается, когда истекает время.



Создадим окно игры и подлодку

Начнем с игрового экрана. Открой в IDLE новое окно программы и введи этот код, в котором создается окно игры и рисуется подводная лодка.



1 Используй Tkinter для создания интерфейса пользователя (GUI). Этот код создает главное окно игры.

```
from tkinter import *
HEIGHT = 500
WIDTH = 800
window = Tk()
window.title('Bubble Blaster')
c = Canvas(window, width=WIDTH, height=HEIGHT, bg='darkblue')
c.pack()
```

Загружает все функции Tkinter

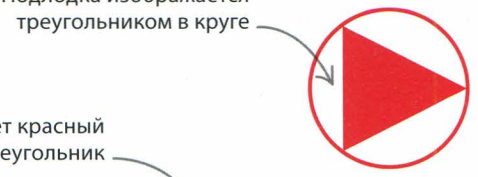
Задает размер окна

Задает название игры

Устанавливает темно-синий фон (море)

Создает холст

2 Подводная лодка будет изображаться простым значком, созданным с помощью функций рисования Tkinter. Введи и запусти этот код.




```
ship_id = c.create_polygon(5, 5, 5, 25, 30, 15, fill='red')
ship_id2 = c.create_oval(0, 0, 30, 30, outline='red')
SHIP_R = 15
MID_X = WIDTH / 2
MID_Y = HEIGHT / 2
c.move(ship_id, MID_X, MID_Y)
c.move(ship_id2, MID_X, MID_Y)
```

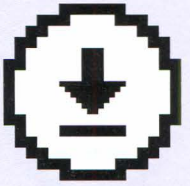
Рисует красный треугольник

Радиус (размер) подлодки

В переменных MID_X и MID_Y — координаты середины экрана

Рисует красный контур круга

Перемещает обе части подлодки в середину экрана



Не забудь сохранить свою работу.

ОХОТНИК ЗА ПУЗЫРЯМИ

Управление подлодкой

Следующий этап проекта — код, двигающий подлодку при нажатии стрелок на клавиатуре. Для этого определим так называемый обработчик событий — функцию, которая проверяет, какая клавиша была нажата, и передвигает подлодку.

3 Введи этот код, чтобы определить функцию под названием move_ship, которая передвигает подлодку при нажатии клавиш со стрелками. Запусти ее и проверь, как она работает.

```
SHIP_SPD = 10
def move_ship(event):
    if event.keysym == 'Up':
        c.move(ship_id, 0, -SHIP_SPD)
        c.move(ship_id2, 0, -SHIP_SPD)
    elif event.keysym == 'Down':
        c.move(ship_id, 0, SHIP_SPD)
        c.move(ship_id2, 0, SHIP_SPD)
    elif event.keysym == 'Left':
        c.move(ship_id, -SHIP_SPD, 0)
        c.move(ship_id2, -SHIP_SPD, 0)
    elif event.keysym == 'Right':
        c.move(ship_id, SHIP_SPD, 0)
        c.move(ship_id2, SHIP_SPD, 0)
c.bind_all('<Key>', move_ship)
```

Расстояние, на которое сдвинется подлодка при нажатии клавиши

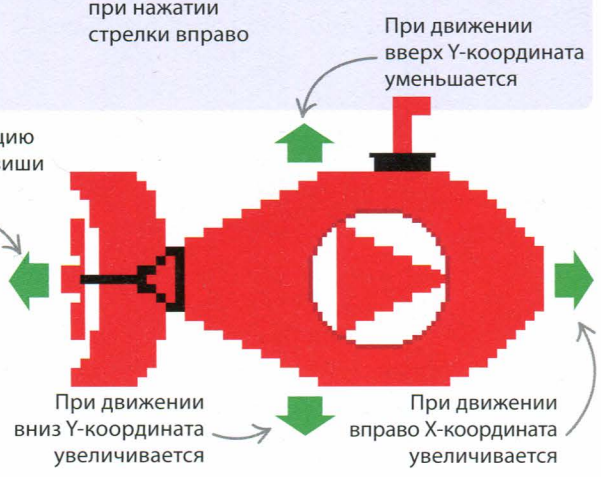
Двигает обе части подлодки вверх при нажатии стрелки вверх

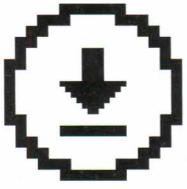
Выполнится при нажатии стрелки вниз (подлодка опускается)

Подлодка плывет влево при нажатии стрелки влево

Подлодка плывет вправо при нажатии стрелки вправо

Указывает, что надо вызывать функцию move_ship при нажатии любой клавиши





Не забудь сохранить свою работу.

▷ Как это работает

Функция move_ship двигает подлодку в разных направлениях. Увеличение координат X и Y соответствует движению подлодки вправо и вниз, а уменьшение — влево и вверх.

Пришло время пузырей

Теперь, когда подлодка может перемещаться, пора заняться пузырями, которые она будет протыкать. Пузыри должны быть разного размера и двигаться с разной скоростью.

4 Каждому пузырю нужно имя (чтобы программа могла различать их), размер и скорость.

```
from random import randint
bub_id = list()
bub_r = list()
bub_speed = list()
MIN_BUB_R = 10
MAX_BUB_R = 30
MAX_BUB_SPD = 10
GAP = 100
def create_bubble():
    x = WIDTH + GAP
    y = randint(0, HEIGHT)
    r = randint(MIN_BUB_R, MAX_BUB_R)
    id1 = c.create_oval(x - r, y - r, x + r, y + r, outline='white')
    bub_id.append(id1)
    bub_r.append(r)
    bub_speed.append(randint(1, MAX_BUB_SPD))
```

Создает пустые списки для хранения имени (bub_id), радиуса (bub_r) и скорости (bub_speed) каждого пузыря

Минимальный радиус пузыря — 10, максимальный — 30

Задает позицию пузыря на холсте

Выбирает случайный радиус пузыря — от минимального до максимального значения

Эта строка рисует пузырь

Добавляет имя, радиус и скорость пузыря в списки

СОВЕТЫ ЭКСПЕРТА

Списки пузырей

Для хранения свойств каждого пузыря служат три списка. Сначала они пустые, но после создания каждого пузыря его данные добавляются в списки. Каждый из них содержит разные данные:

bub_id: хранит имена пузырей, чтобы программа могла ими управлять;

bub_r: хранит радиус пузырей;

bub_speed: хранит скорости передвижения пузырей по экрану.

Не забудь сохранить свою работу.

ОХОТНИК ЗА ПУЗЫРЯМИ

Движение пузырей

Теперь у нас есть списки с именами, а также случайно выбранными радиусами и скоростями пузырей. Следующим этапом нужно написать код для движения пузырей по экрану.

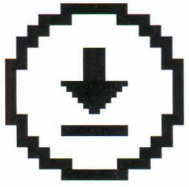
5 Эта функция перебирает все пузыри и по очереди двигает их.

```
def move_bubbles():
    for i in range(len(bub_id)):
        c.move(bub_id[i], -bub_speed[i], 0)
```

По очереди берет каждый пузырь из списка

Двигает пузырь по экрану в соответствии с его скоростью

6 Это главный цикл игры. Пока игра работает, он будет повторяться снова и снова. Попробуй его запустить!



Не забудь сохранить свою работу.

```
from time import sleep, time
BUB_CHANCE = 10
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    window.update()
    sleep(0.01)
```

Загружает нужные функции из модуля Time

Выбирает случайное число от 1 до 10

Если случайное число равно 1, создает новый пузырь (в среднем это 1 случай из 10, так что пузырей будет не слишком много)

Вызывает функцию move_bubbles

Обновляет окно, чтобы перерисовать пузыри в новых позициях

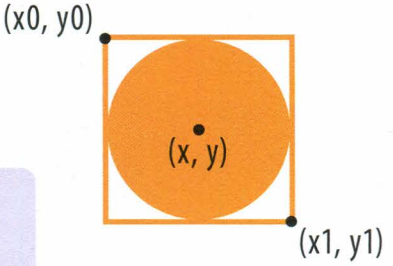
Замедляет игру, чтобы играть было не слишком сложно

7 Теперь создай функцию, которая по имени пузыря определяет, где он находится. Этот код нужно добавить в программу сразу после кода, введенного на шаге 5.

```
def get_coords(id_num):
    pos = c.coords(id_num)
    x = (pos[0] + pos[2])/2
    y = (pos[1] + pos[3])/2
    return x, y
```

Вычисляет X-координату середины пузыря

Вычисляет Y-координату середины пузыря



△ **Координаты пузырей**
Эта функция находит координаты середины пузыря, вычисляя точку между углами квадрата, в который вписан этот пузырь.

Пузыри должны лопаться

Игрок зарабатывает очки, протыкая пузыри, а значит, программа должна уметь убирать их с экрана. Для этого понадобятся функции, записанные ниже.

8 Эта функция убирает пузырь из игры. Она делает это, удаляя его из всех списков, а также с холста. Этот код надо добавить в программу сразу после кода, введенного на шаге 7.

```
def del_bubble(i):
    del bub_r[i]
    del bub_speed[i]
    c.delete(bub_id[i])
    del bub_id[i]
```

Удаляет пузырь, имя которого в *i*

Удаляет пузырь из списков радиусов и скоростей

Удаляет пузырь с холста

Удаляет пузырь из списка имен

9 Это код функции, которая удаляет пузыри, уплывшие за край холста. Добавь этот код сразу после кода, введенного на шаге 8.

```
def clean_up_bubs():
    for i in range(len(bub_id)-1, -1, -1):
        x, y = get_coords(bub_id[i])
        if x < -GAP:
            del_bubble(i)
```

Обратный цикл по списку пузырей — чтобы избежать ошибки работы for при удалении пузырей

Находит координаты пузыря

Если пузырь уплыл за экран, его нужно удалить, иначе он будет замедлять игру

10 Теперь дополни главный цикл игры (с шага 6) вызовами только что созданных функций. Запусти программу, чтобы убедиться, что в ней нет ошибок.

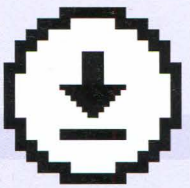
```
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    window.update()
    sleep(0.01)
```

Создает новый пузырь

Обновляет позиции всех пузырей

Удаляет пузыри, уплывшие за экран

Перерисовывает окно, чтобы отобразить все изменения



Не забудь сохранить свою работу.

ОХОТНИК ЗА ПУЗЫРЯМИ

Вычисляем расстояние между точками

В этой игре, как и во многих других, полезно знать расстояние между двумя объектами. Для расчета расстояний мы воспользуемся известной математической формулой.

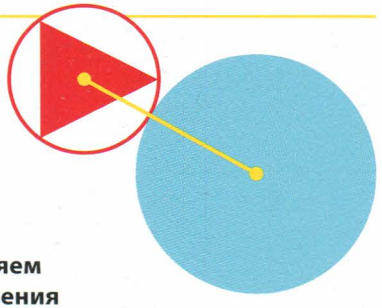
11 Эта функция вычисляет расстояние между двумя объектами. Добавь этот код сразу после кода, введенного на шаге 9.

```
from math import sqrt
def distance(id1, id2):
    x1, y1 = get_coords(id1)
    x2, y2 = get_coords(id2)
    return sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

- `from math import sqrt` — Загружает функцию sqrt из модуля Math
- `x1, y1 = get_coords(id1)` — Получает координаты первого объекта
- `x2, y2 = get_coords(id2)` — Получает координаты второго объекта
- `return sqrt((x2 - x1)**2 + (y2 - y1)**2)` — Возвращает расстояние между объектами

Пузыри лопаются

Игрок зарабатывает очки, протыкая пузыри. Большие пузыри и быстрые пузыри дают больше очков. Следующая часть кода на основе радиуса (расстояния от центра до края) каждого пузыря проверяет, не проткнут ли он.



▷ **Выявляем столкновения**

Если расстояние между центром подлодки и центром пузыря меньше, чем сумма их радиусов, значит, они столкнулись.

12 Если подлодка и пузырь столкнутся, программа должна убрать пузырь и обновить счет. Этот фрагмент кода нужно вставить сразу после кода, введенного на шаге 11.

```
def collision():
    points = 0
    for bub in range(len(bub_id)-1, -1, -1):
        if distance(ship_id2, bub_id[bub]) < (SHIP_R + bub_r[bub]):
            points += (bub_r[bub] + bub_speed[bub])
            del_bubble(bub)
    return points
```

- `points = 0` — В этой переменной хранится счет игры (набранные очки)
- `for bub in range(len(bub_id)-1, -1, -1):` — Этот цикл проходит по всем пузырям из списка (от последних к первым, чтобы избежать ошибок при удалении пузырей)
- `if distance(ship_id2, bub_id[bub]) < (SHIP_R + bub_r[bub]):` — Проверяет, не столкнулась ли подлодка с одним из пузырей
- `points += (bub_r[bub] + bub_speed[bub])` — Вычисляет, сколько очков полагается за этот пузырь, и прибавляет их к points
- `del_bubble(bub)` — Удаляет пузырь
- `return points` — Возвращает количество набранных очков

13 Дополни главный игровой цикл вызовами только что созданных функций. Убедись, что все части кода стоят на своих местах, их порядок важен. Запусти программу — пузыри должны лопаться при столкновении с подлодкой. Счет игры печатается в окне консоли.

```
score = 0
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    score += collision()
    print(score)
    window.update()
    sleep(0.01)
```

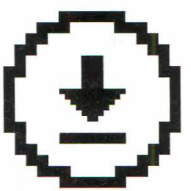
В начале игры установить количество очков в 0

Создает новые пузыри

Прибавляет очки за пузырь к общему счету

Печатает счет в окне консоли — позже мы будем выводить его как положено, в окне игры

Приостанавливает игру на короткий промежуток времени — попробуй убрать эту строку и посмотри, что произойдет

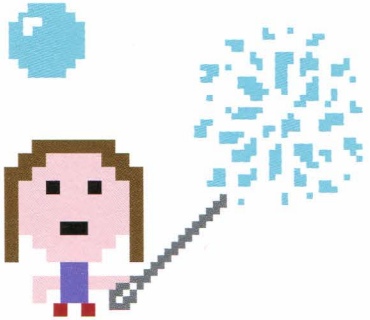


Не забудь сохранить свою работу.

СОВЕТЫ ЭКСПЕРТА

Сокращенная запись в Python

Код score += collision() — это короткая форма записи кода score = score + collision(), который складывает очки за столкновение и общий счет и заносит результат в общий счет. Это типичная программная конструкция, поэтому для нее и есть короткая форма записи. То же можно делать с вычитанием, например score - = 10 аналогично score = score -10.



ОХОТНИК ЗА ПУЗЫРЯМИ

Последние штрихи

Основные части игры теперь готовы. Остались последние детали — отображение счета и задание лимита времени, который будет уменьшаться, пока игра не закончится.

14 Добавь этот код после кода, введенного на шаге 12. Он отображает счет игрока и оставшееся время игры.

```
c.create_text(50, 30, text='TIME', fill='white' )
c.create_text(150, 30, text='SCORE', fill='white' )
time_text = c.create_text(50, 50, fill='white' )
score_text = c.create_text(150, 50, fill='white' )
def show_score(score):
    c.itemconfig(score_text, text=str(score))
def show_time(time_left):
    c.itemconfig(time_text, text=str(time_left))
```

Помещает рядом со временем и счетом подписи TIME («время») и SCORE («счет»), чтобы игрок понимал, что эти цифры означают

Создает надписи для оставшегося времени и количества очков

Отображает счет

Отображает оставшееся время

15 Теперь надо задать ограничение по времени и количество очков, за которое игроку дается призовое время. Этот фрагмент кода нужно добавить прямо перед главным игровым циклом.

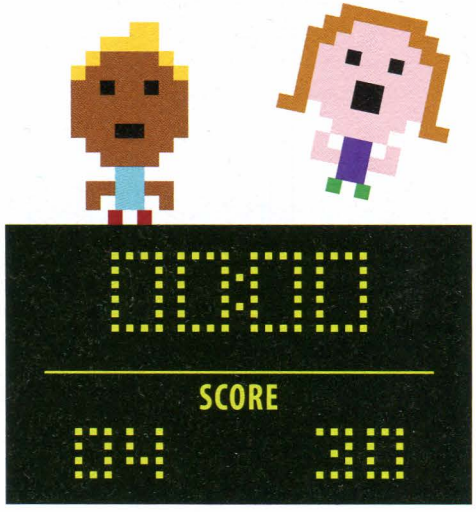
```
from time import sleep, time
BUB_CHANCE = 10
TIME_LIMIT = 30
BONUS_SCORE = 1000
score = 0
bonus = 0
end = time() + TIME_LIMIT
```

Загружает функции из модуля Time

В начале игры лимит времени равен 30 секундам

Определяет, когда игроку дается дополнительное время (за выигрыш в 1000 очков)

Сохраняет время окончания игры в переменной end



△ **Панель счета**

Панели счета — хороший наглядный способ показать игроку его достижения.

16 Добавь в главный цикл вызовы новых функций отображения счета и времени.

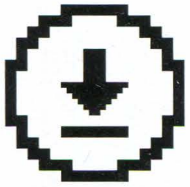
```
#MAIN GAME LOOP
while time() < end:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    score += collision()
    if (int(score / BONUS_SCORE)) > bonus:
        bonus += 1
        end += TIME_LIMIT
    show_score(score)
    show_time(int(end - time()))
    window.update()
    sleep(0.01)
```

Повторяет главный игровой цикл вплоть до окончания игры

Вычисляет, когда игроку нужно добавить призовое время

print(score) заменен на show_score(score), чтобы счет отображался в главном окне игры

Показывает оставшееся время



Не забудь сохранить свою работу.

17 И наконец, добавь надпись GAME OVER («Конец игры»), которую нужно показать, когда истечет время. Вставь этот код в самый конец своей программы.

```
c.create_text(MID_X, MID_Y, \
    text='GAME OVER', fill='white', font=('Helvetica',30))
c.create_text(MID_X, MID_Y + 30, \
    text='Score: '+ str(score), fill='white')
c.create_text(MID_X, MID_Y + 45, \
    text='Bonus time: '+ str(bonus*TIME_LIMIT), fill='white')
```

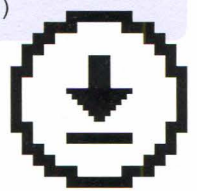
Помещает надпись в середину экрана

Задает шрифт. Шрифт Helvetica хорошо подходит для крупных надписей

Показывает набранные очки

Выставляет белый цвет текста

Показывает, сколько призового времени заработал игрок

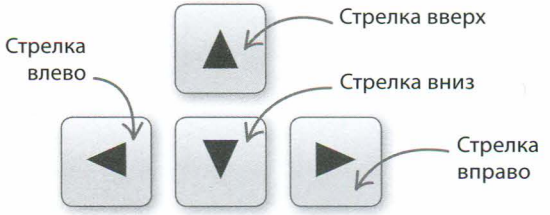


Не забудь сохранить свою работу.

ОХОТНИК ЗА ПУЗЫРЯМИ

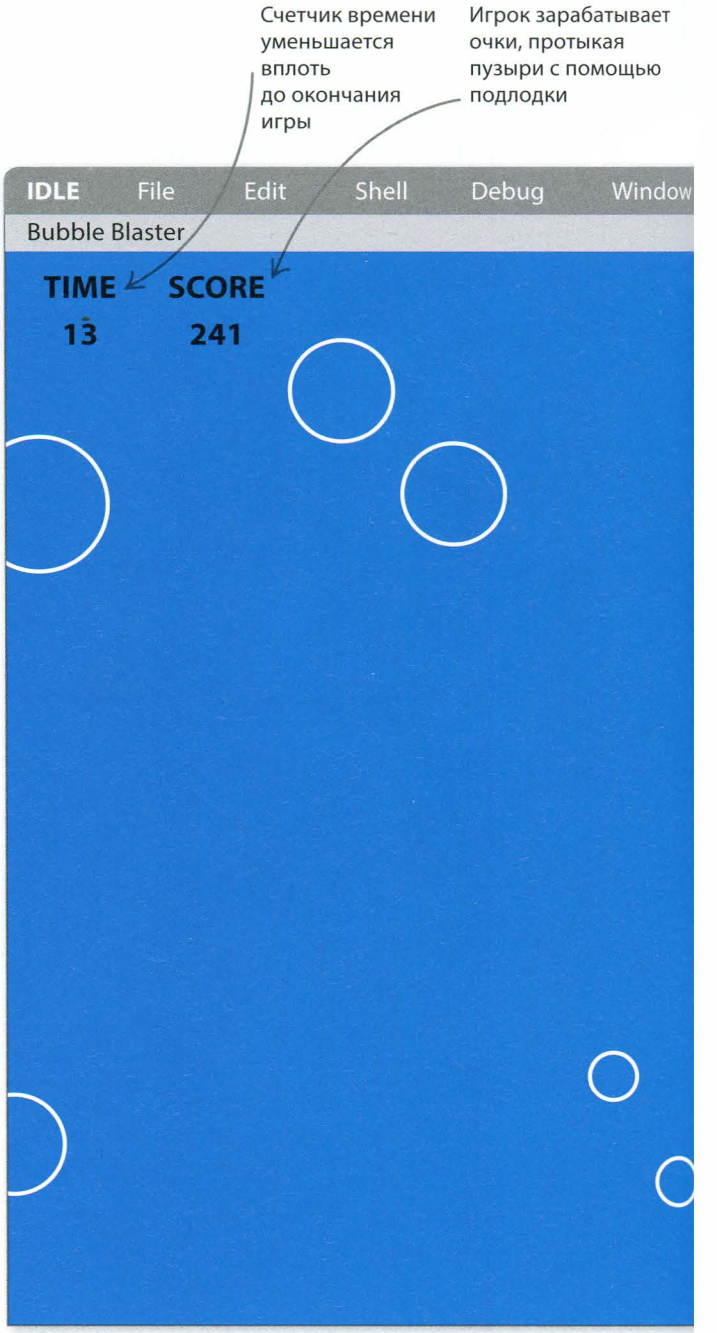
Пора играть

Молодец! Игра «Охотник за пузырями» готова, и в нее можно поиграть. Попробуй запустить ее. Если что-то работает не так, вспомни советы по отладке и внимательно просмотри код с предыдущих страниц: все ли введено верно?



△ **Управление**

Подлодка управляется клавишами-стрелками. Можно доработать программу, чтобы она поддерживала другие способы управления.



СОВЕТЫ ЭКСПЕРТА

Доработка игры

Все компьютерные игры начинаются с основной идеи. Потом в них играют, их тестируют, настраивают и дорабатывают. Это первая версия игры. Вот некоторые предложения по ее усовершенствованию.

Сделай игру сложнее, изменив лимит времени и количество очков, за которое дается призовое время.

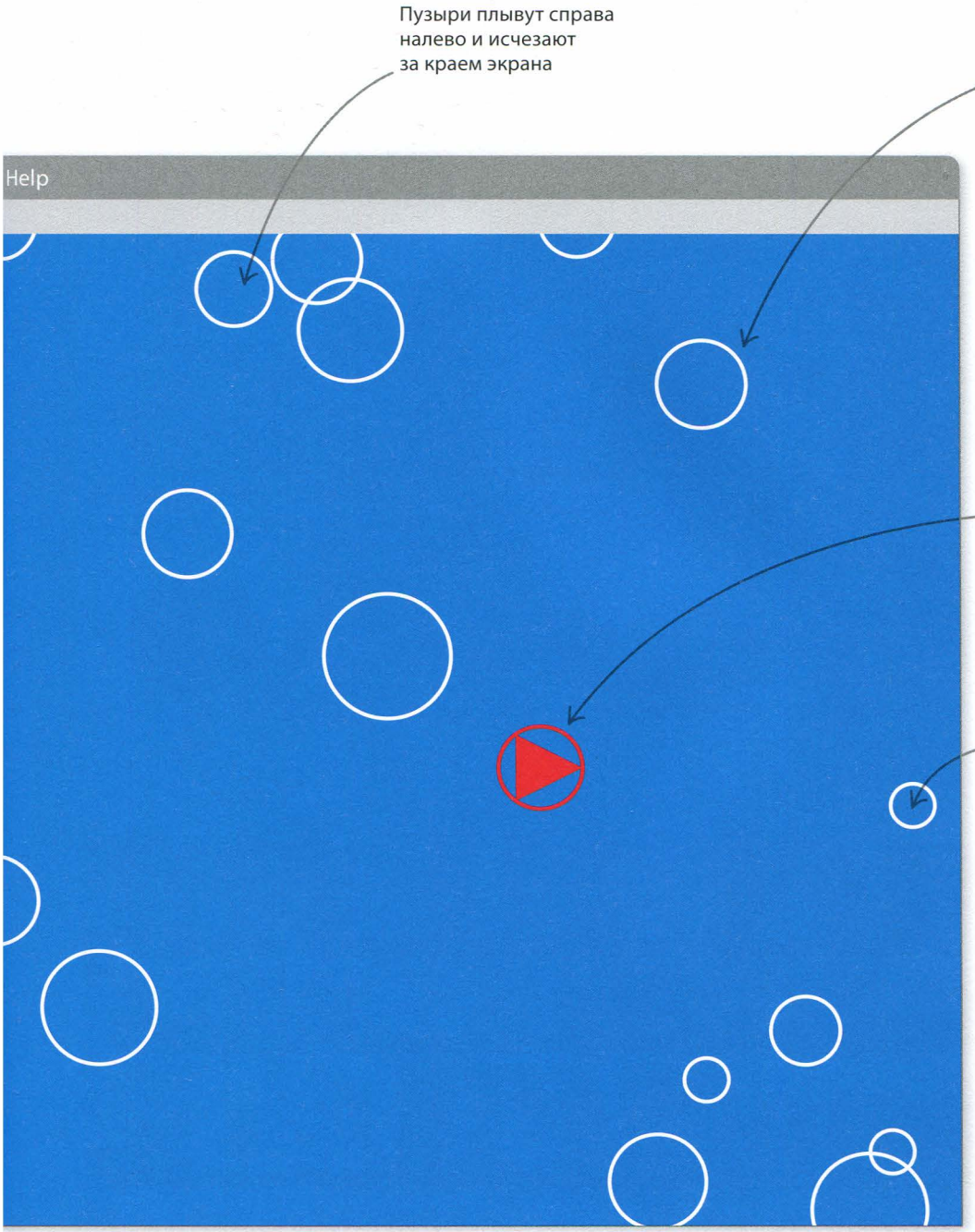
Выбери другой цвет для подводной лодки.

Более детально изобрази подлодку.

Добавь специальный вид пузырей, увеличивающих скорость подлодки.

Добавь бомбу, которая устраняет все пузыри при нажатии на пробел.

Добавь экран с лучшими результатами игры.



◁ Суперподлодка

Теперь ты можешь показать игру своим друзьям. Играйте по очереди, чтобы посмотреть, кто наберет больше очков. Потом покажи друзьям код игры и объясни, как он работает.

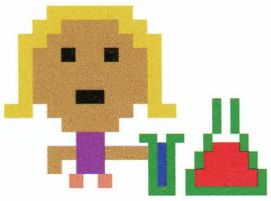
Что дальше?

СМОТРИ ТАКЖЕ
❮ 152–153 Библиотеки
Компьютерные игры 204–205 ❯

Теперь, разобравшись с Python-проектами в этой книге, ты уже на пути к тому, чтобы стать отличным программистом. Вот некоторые идеи по поводу того, чем еще заняться на Python и как развивать навыки программирования дальше.

Экспериментируй

Исследуй примеры кода из этой книги. Меняй их, добавляй новые возможности и не бойся ничего испортить! Помни, что Python — это мощный профессиональный язык программирования, на котором можно писать самые разные программы.



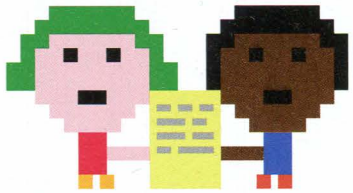
Составляй свои библиотеки

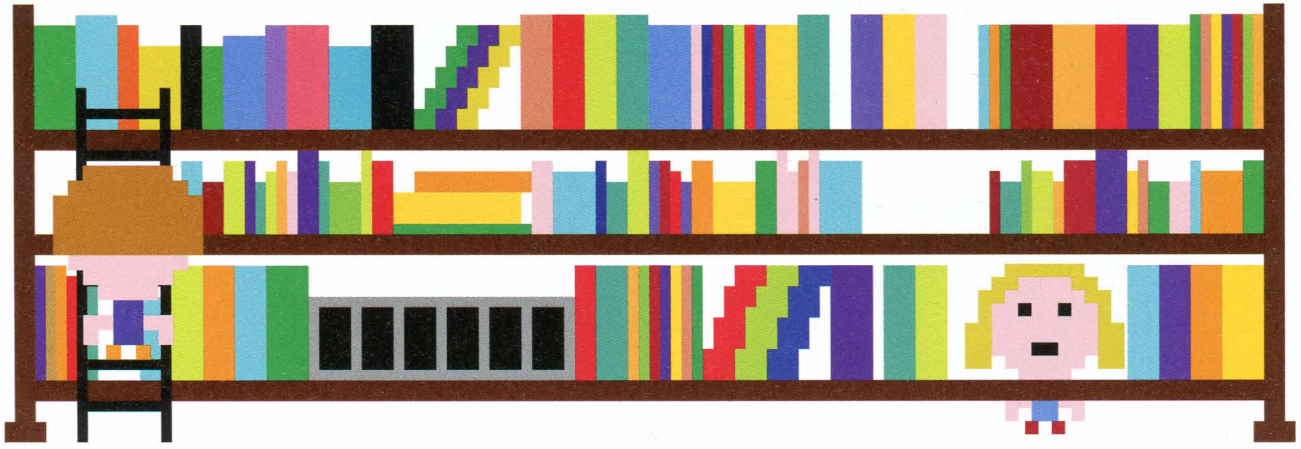
Программисты любят использовать код повторно и делиться своими достижениями. Создай собственную библиотеку полезных функций и выложи ее в интернете. Знать, что другие программисты пользуются твоим кодом, очень приятно. Ты можешь написать что-нибудь столь же полезное, как Tkinter или Turtle!

ЗАПОМНИ

Исследуй чужой код

Изучай интересные программы и библиотеки, написанные другими людьми: читай код и комментарии к нему, постарайся понять, как он работает и почему написан именно так, — это улучшит твои навыки программирования. Кроме того, ты познакомишься с библиотеками, которые можно использовать в своих программах.





Пиши игры на Python

Ты можешь создавать собственные игры на Python. В библиотеке Pygame, которая доступна в интернете, есть множество упрощающих разработку игр функций и инструментов. Начни с простых игр и переходи к более сложным.

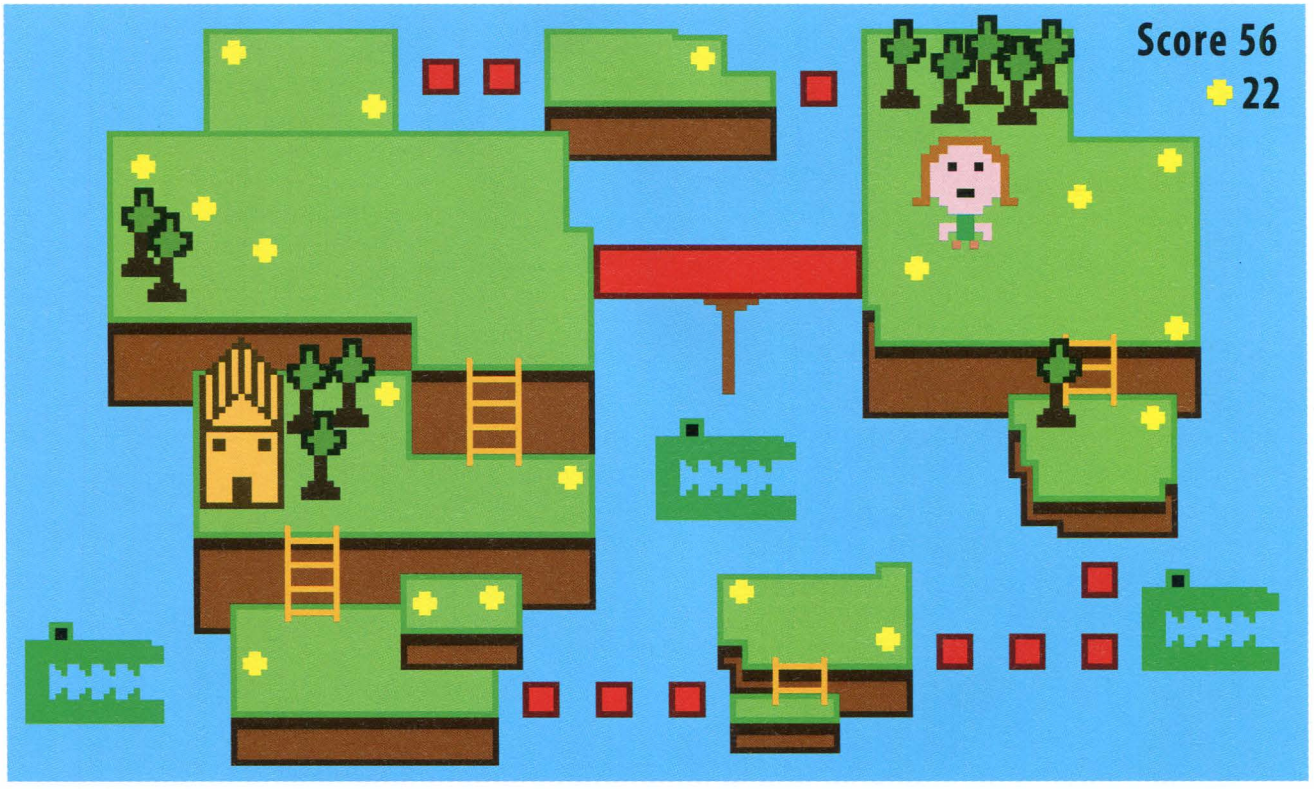
СОВЕТЫ ЭКСПЕРТА

Разные версии Python

Код, который ты встретишь в книгах или в Сети, может быть написан для разных версий Python. Хоть версии и очень похожи, не исключено, что в код придется вносить небольшие поправки.

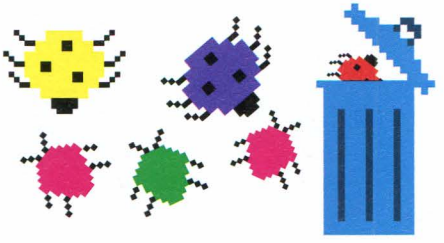
```
print 'Hello World'
```
Python 2

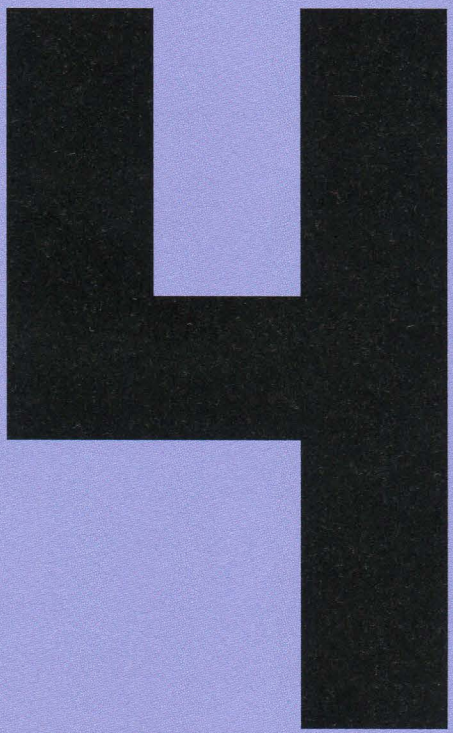
```
print('Hello World')
```
Python 3



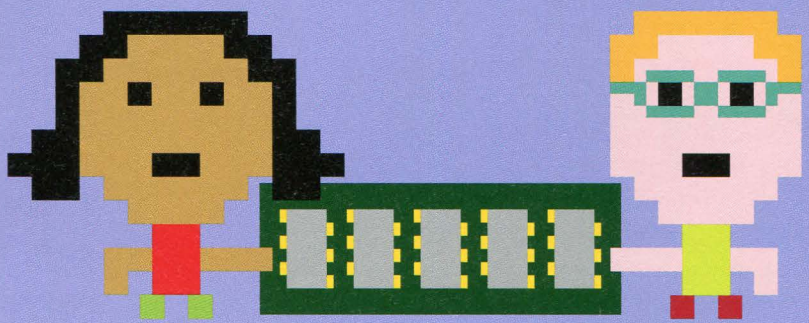
Отлаживай свой код

Отладка — важная часть программирования. Не сдавайся, если программа не работает. Вспомни, что компьютеры делают лишь то, что им сказано, просмотри код и выясни, что с ним не так. Порой ошибку найти проще, если просматривать код вместе с другим программистом.





Устройство компьютеров



Внутри компьютера

Самые первые компьютеры были простыми счетными машинами. В принципе, с тех пор мало что изменилось — компьютеры принимают входные данные (ввод), выполняют вычисления и возвращают ответ (вывод).

СМОТРИ ТАКЖЕ

Хранение данных в файлах 192–193 ❯

Интернет 194–195 ❯

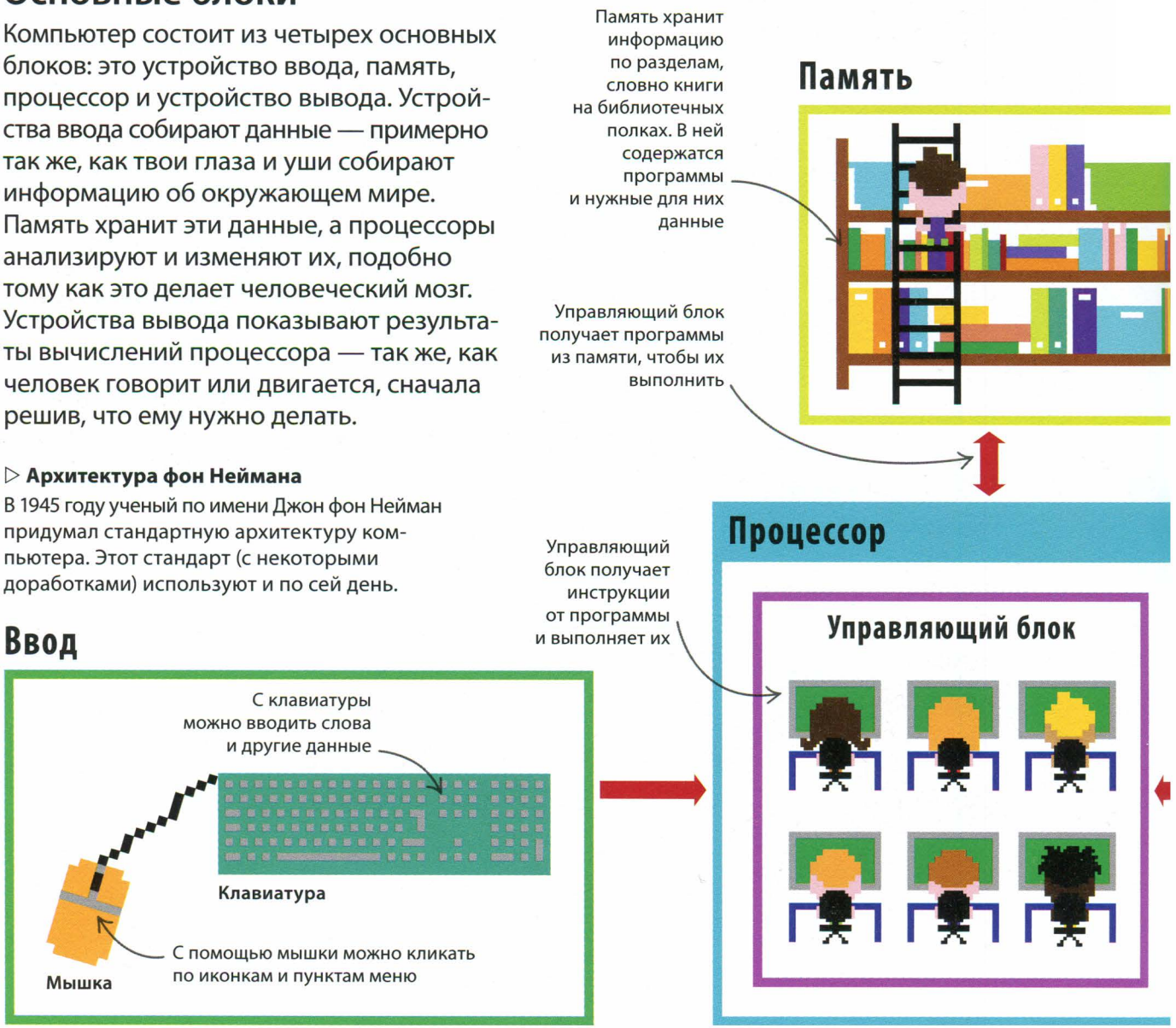
Мини-компьютеры 214–215 ❯

Основные блоки

Компьютер состоит из четырех основных блоков: это устройство ввода, память, процессор и устройство вывода. Устройства ввода собирают данные — примерно так же, как твои глаза и уши собирают информацию об окружающем мире. Память хранит эти данные, а процессоры анализируют и изменяют их, подобно тому как это делает человеческий мозг. Устройства вывода показывают результаты вычислений процессора — так же, как человек говорит или двигается, сначала решив, что ему нужно делать.

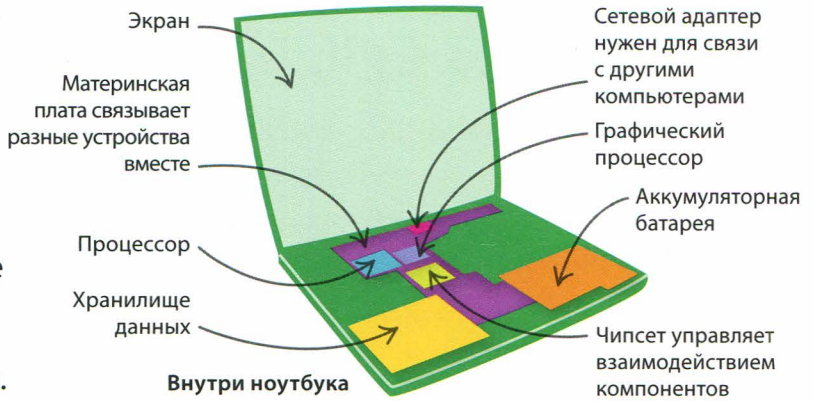
▷ **Архитектура фон Неймана**

В 1945 году ученый по имени Джон фон Нейман придумал стандартную архитектуру компьютера. Этот стандарт (с некоторыми доработками) используют и по сей день.

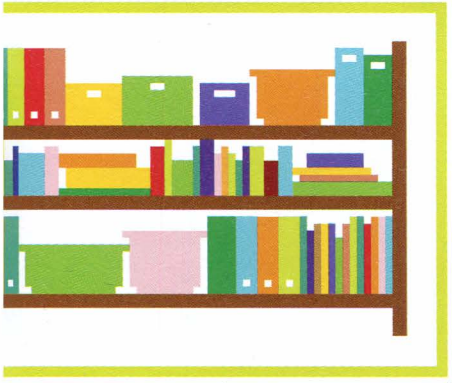


Компьютерное железо

Железо — это физические детали компьютера. Компьютер состоит из множества устройств, которые работают сообща. По мере того как производители делают компьютеры все меньше, добавляя им все больше возможностей, детали тоже становятся меньше, выделяют меньше тепла и потребляют меньше энергии.



Внутри ноутбука

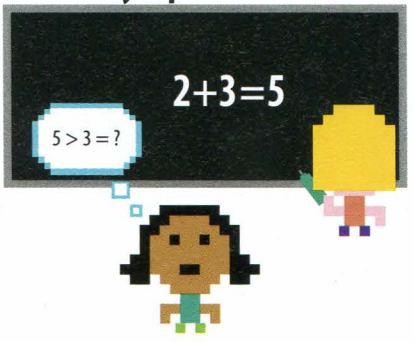


Арифметически-логический блок получает из памяти данные для вычислений

Процессор состоит из двух блоков: первый выполняет инструкции, а второй производит вычисления

Арифметически-логическое устройство (АЛУ) выполняет нужные для программы вычисления

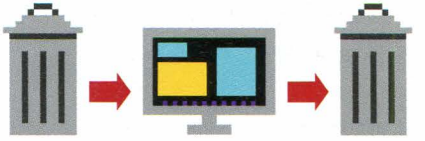
Арифметически-логическое устройство



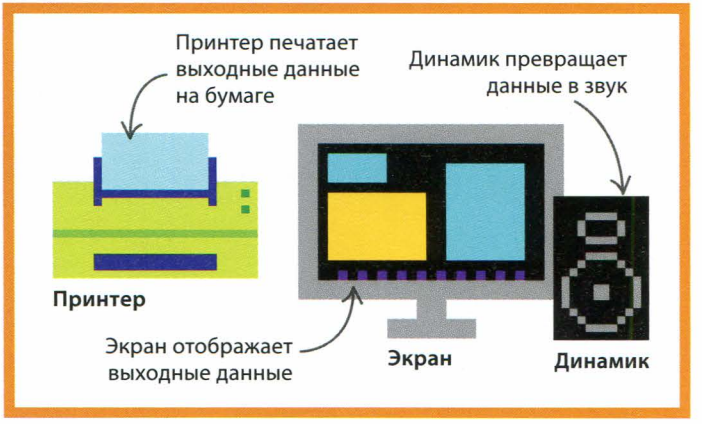
СЛЕНГ

GIGO

«Мусор туда, мусор обратно» (англ. сокращ. GIGO: garbage in, garbage out) — фраза, означающая, что даже лучшие программы выдают чепуху, если передать им неверные входные данные.



Вывод



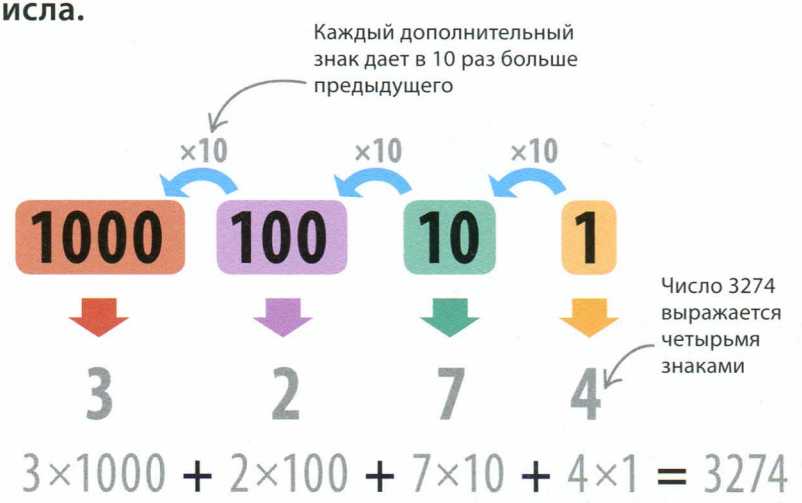
Двоичная система

СМОТРИ ТАКЖЕ
Символы и коды 184–185 ›
Логические вентили 186–187 ›

Как компьютеры выполняют сложные вычисления, если все, что они понимают, — это электрические сигналы? Двоичная система счисления используется, чтобы преобразовывать сигналы в числа.

Что такое основание?

Основание системы счисления — это количество значений, которые можно получить, используя только один числовой знак. Каждый дополнительный знак увеличивает количество значений кратно основанию.



▷ Десятичная система
Десятичная система — самая популярная система счисления, которая использует основание 10. С помощью одного знака она может выразить 10 значений, с помощью двух — 100, трех — 1000.

Двоичный код

В основе своей компьютеры понимают только два значения: «включение» и «выключение» электрического сигнала, а значит, работают с числовым основанием, равным двойке, — с двоичными числами. Каждый знак числа — это или 1, или 0, и каждый следующий знак дает вдвое больше значений, чем предыдущий.

▷ 1 и 0
Провод с включенным сигналом — это 1. Провод, где сигнал выключен, — это 0.

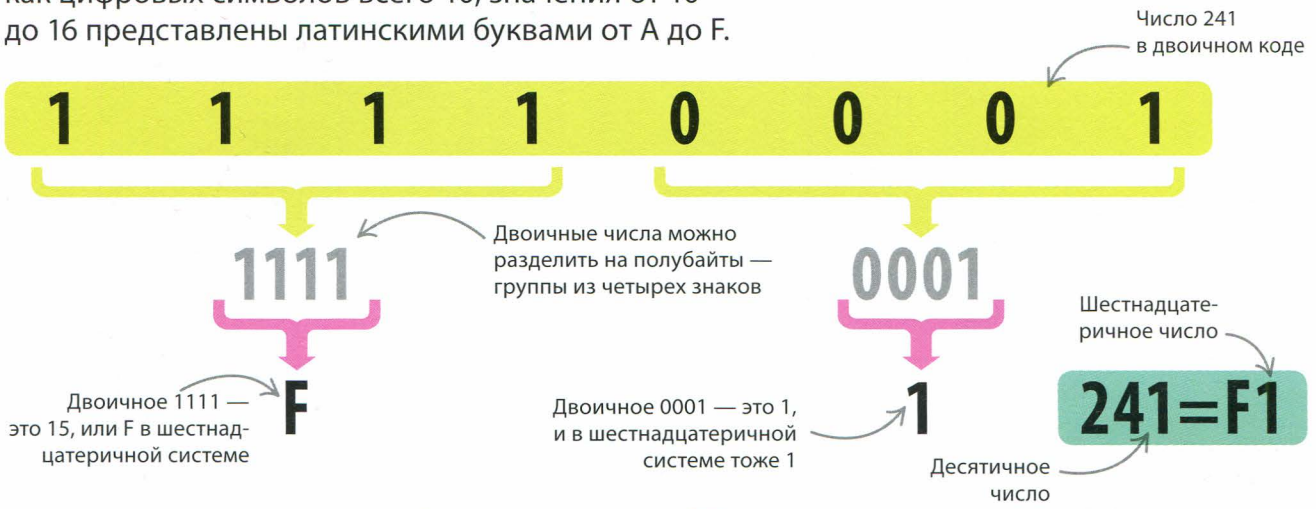
◁ Двоичное число
Диапазон значений удваивается с каждым новым знаком.

Шестнадцатеричная система

В программах часто используют числовое основание 16 из-за его связи с компьютерным двоичным кодом. Так как цифровых символов всего 10, значения от 10 до 16 представлены латинскими буквами от A до F.

▽ **Полубайты**

Полубайт состоит из четырех двоичных знаков, и его можно представить одним шестнадцатеричным знаком.



▽ **Сравнение числовых оснований**

Из этой таблицы видно, что в шестнадцатеричной системе меньшим количеством знаков можно выразить больше значений.

РАЗНЫЕ ОСНОВАНИЯ

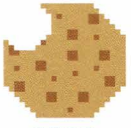
Десятичное	Двоичное	Шестнадцатеричное
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

ЗАПОМНИ

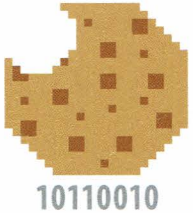
Биты, полубайты, байты

Двоичный знак называют битом, и это наименьшее значение в информатике. Из битов можно составить полубайты и байты. Килобит — это 1024 бита. Мегабит — 1024 килобита.

Биты: каждый бит — это один двоичный знак: 1 или 0.



Полубайты: полубайт состоит из 4 бит, что соответствует одному шестнадцатеричному знаку.



Байты: байт состоит из 8 бит, или двух шестнадцатеричных знаков. Это дает диапазон значений от 0 до 255 (от 00 до FF).

Символы и коды

Двоичный код служит для преобразования чисел в электрические сигналы. Но как бы компьютер выразил в двоичном коде слова и символы с этой страницы?

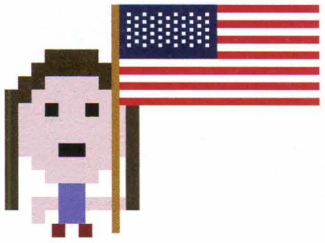
СМОТРИ ТАКЖЕ

‹ 180–181 Внутри компьютера

‹ 182–183 Двоичная система

ASCII

Самые первые компьютеры хранили символы по-разному, каждая модель — своим уникальным способом, и все было хорошо до тех пор, пока не понадобилось переносить данные с компьютера на компьютер. Тогда для кодирования символов была разработана общая система под названием ASCII.



▷ Таблица ASCII

В системе ASCII каждому заглавному и каждому строчному символу алфавита присвоено десятичное значение. Числовые значения соответствуют также знакам препинания и другим символам, например пробелу.

▷ ASCII и двоичный код

Поскольку каждому символу соответствует число, для хранения в компьютере это число нужно преобразовать в двоичный код.

R = 82 = 1010010

r = 114 = 1110010

▽ ASCII в Python

Преобразовывать символы между ASCII и двоичным кодом можно в большинстве языков, включая Python.

Эта команда печатает символ, ASCII-значение и двоичное значение для каждой буквы в строке Sam

```
>>> name = 'Sam'
>>> for c in name:
        print(c, ord(c), bin(ord(c)))

S 83 0b1010011
a 97 0b1100001
m 109 0b1101101
```

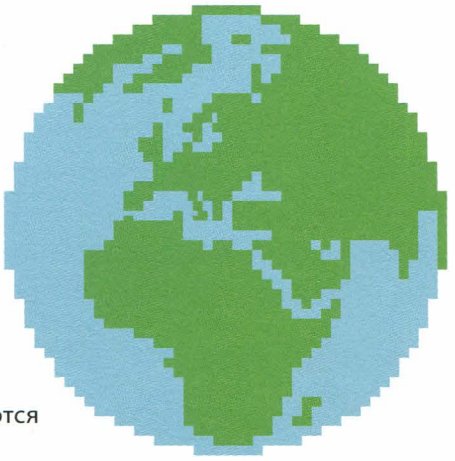
Результат. Двоичные числа помечены символами 0b

ASCII

32	ПРОБЕЛ	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	СТЕРЕТЬ

Юникод

Вскоре после того как компьютеры стали обмениваться данными, начали проявляться ограничения системы ASCII. Тогда был разработан новый стандарт «Юникод», способный выразить каждый из тысяч символов сотен мировых языков.



▷ **Международный код**
«Юникод» подходит для всех языков в мире. Например, арабские символы располагаются в диапазоне 0060–06FF.

▷ **Символы в «Юникоде»**
В «Юникоде» символы представлены шестнадцатеричными значениями — последовательностями букв и цифр (см. с. 182–183). У каждого символа есть свой код. В «Юникод» добавляют все новые и новые символы, и некоторые из них довольно необычны, например значок зонтика.


2602

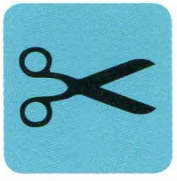

2EC6


08A2


0036


0974


004D


2702


A147

ЗАПОМНИ

Шестнадцатеричный код

При записи шестнадцатеричных чисел используется основание 16. Значения от 0 до 9 записывают десятичными цифрами, а от 10 до 15 — буквами от A до F. Каждому шестнадцатеричному числу соответствует двоичное значение.

Юникод-значение буквы «ё» в шестнадцатеричном коде

То же значение в двоичном коде

ё = 00EB = 11100111

▽ **«Юникод» в Python**
«Юникод» поможет отображать специальные символы в Python — для этого введи в Python-строку значение символа в «Юникоде».

Символы «\u» перед шестнадцатеричным кодом обозначают, что это «Юникод»

```
>>> 'Zo\u00EB'
'Zoë'
```

Код будет преобразован в букву «ё»

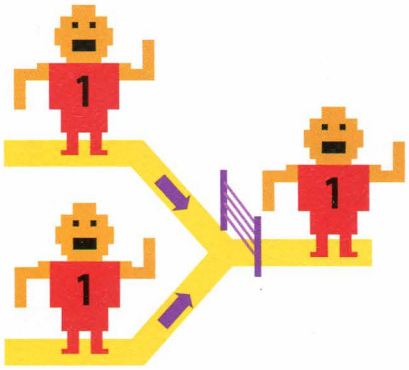
Логические вентили

СМОТРИ ТАКЖЕ
‹ 180–181 Внутри компьютера
‹ 182–183 Двоичная система

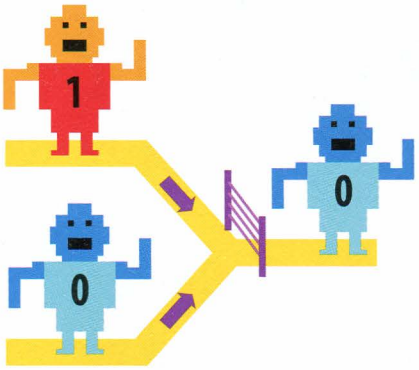
Компьютеры могут использовать электрические сигналы не только для кодирования чисел и символов, но и для принятия решений — с помощью таких устройств, как «логические вентили». Есть четыре основных типа логических вентилей: «И», «НЕ», «ИЛИ» и «Исключающее ИЛИ».

Вентиль «И»

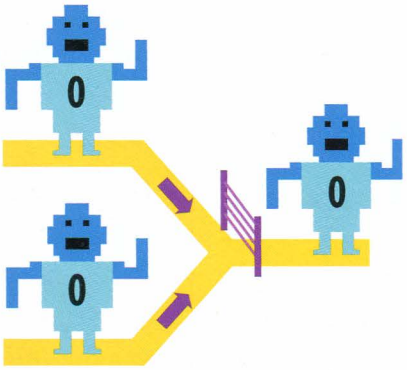
Вентили принимают один или несколько входных сигналов и на основе простого правила выдают результирующий сигнал. Вентили «И» включают выходной сигнал (1), только если включены оба входных сигнала (1 *и* 1).



△ **Входы 1 и 1 = выход 1**
Оба входных сигнала включены — вентиль «И» выдает сигнал «включено».



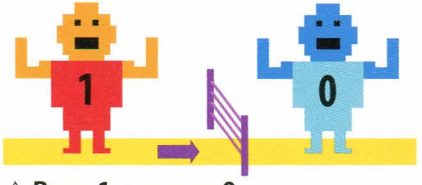
△ **Входы 1 и 0 = выход 0**
Если один вход включен, а другой выключен, выходной сигнал выключен.



△ **Входы 0 и 0 = выход 0**
Вентиль «И» выдает сигнал «выключено», если выключены оба входных сигнала.

Вентиль «НЕ»

Эти вентили «переворачивают» входной сигнал: «включено» превращается в «выключено», а «выключено» — во «включено». Вентили «НЕ» также называют инверторами.



△ **Вход 1 = выход 0**
Вентиль «НЕ» превращает «включено» в «выключено» и наоборот.

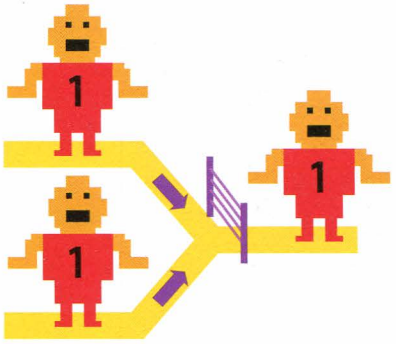
В ЖИЗНИ

Джордж Буль (1815–1864)

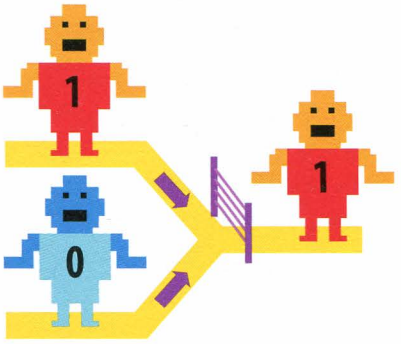
Джордж Буль — английский математик, разработавший систему решения логических задач, на основе которой и были созданы логические вентили. Раздел математики, имеющий дело со значениями, которые могут быть только «истинными» или «ложными» (положительными или отрицательными), назван в его честь булевой логикой.

Вентиль «ИЛИ»

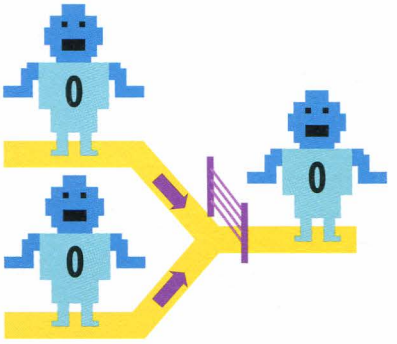
Вентиль «ИЛИ» выдает сигнал «включено», если включен любой из входов или оба сразу.



△ **Входы 1 и 1 = выход 1**
Два сигнала «включено» дают на выходе «включено».



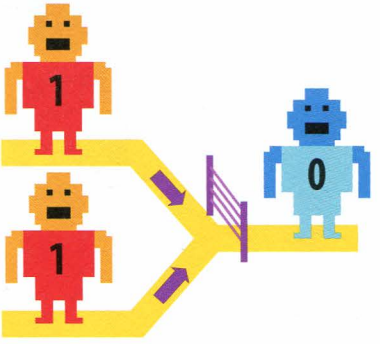
△ **Входы 1 и 0 = выход 1**
Один сигнал «включено» и один «выключено» все еще дают «включено».



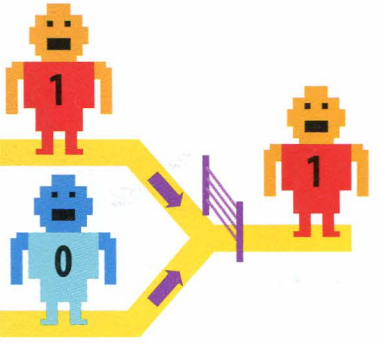
△ **Входы 0 и 0 = выход 0**
Вентиль «ИЛИ» выдает «выключено», только если выключены оба входа.

Вентиль «Исключающее ИЛИ»

Этот тип вентиля выдает «включено», когда один из входов включен, а другой выключен. Два включенных или два выключенных входа дадут «выключено». Эти вентили также называют вентилями XOR (Exclusive OR gate).



△ **Входы 1 и 1 = выход 0**
Два «включено» дают на выходе «выключено».

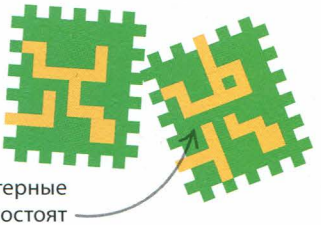


△ **Входы 1 и 0 = выход 1**
Выход «включен», только если входные сигналы различны.

СОВЕТЫ ЭКСПЕРТА

Устройство компьютерных схем

Совмещая эти базовые логические вентили, можно создавать схемы для выполнения множества сложных операций. Например, соединив вентиль «И» с вентилем XOR, мы получим схему для сложения двух битов данных. Соединив два вентиля «ИЛИ» с двумя вентилями «НЕ» и закольцевав их, получим схему для хранения бита данных. Даже самые сложные компьютеры состоят из миллиардов миниатюрных логических схем.



Компьютерные микросхемы состоят из множества логических схем

Процессоры и память

В компьютере есть микросхемы разных типов. Так, микросхема процессора выполняет программы, а микросхемы памяти хранят данные для мгновенного доступа к ним.

СМОТРИ ТАКЖЕ

‹ 180–181 Внутри компьютера

‹ 186–187 Логические вентили

Процессор

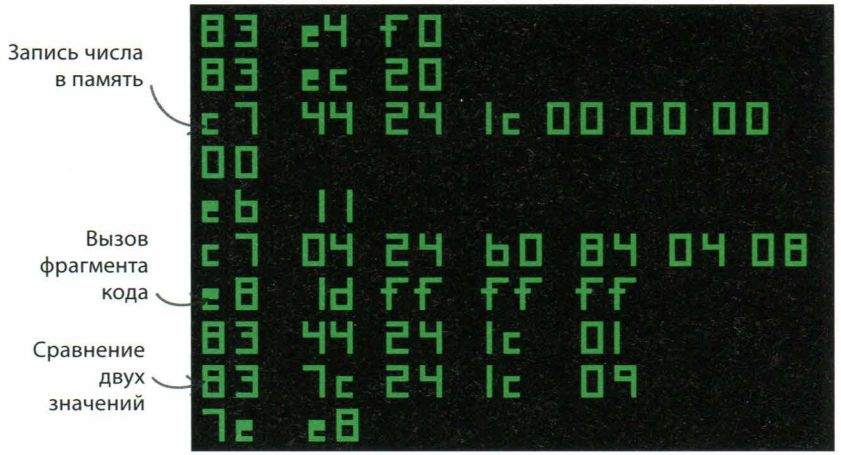
Процессоры состоят из миниатюрных электрических схем, напечатанных на кремниевых пластинах (материале, похожем на стекло). Маленькие переключатели — транзисторы — используются для создания простых логических вентилей, из которых состоят сложные схемы. Все программы на твоем компьютере выполняются такими схемами.

◁ Схемы в процессоре
Схемы синхронизируются тактовыми импульсами, подобно тому как дирижер управляет игрой музыкантов в оркестре.

Машинный код

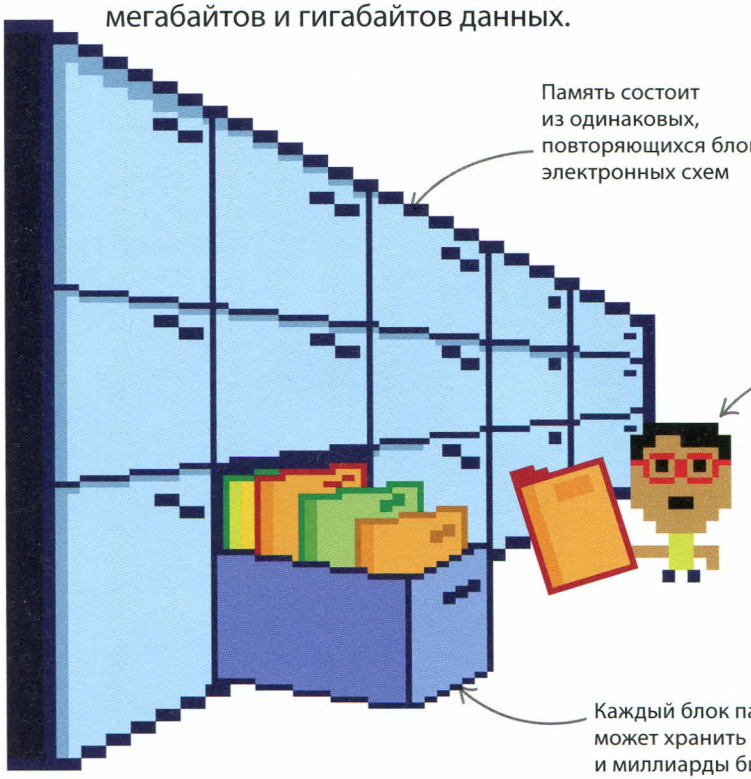
Процессоры понимают лишь набор инструкций, называемый машинным кодом. Из простых инструкций сложения, вычитания, хранения данных и других можно составлять сложные программы.

▷ Что такое машинный код?
Машинный код — это просто числа, поэтому программисты используют языки вроде Python, которые в итоге преобразуются в машинный код.



Память

Микросхемы памяти тоже напечатаны на кремниевых пластинах. Несколько логических вентилей формируют схему-защелку. Каждая защелка хранит один бит (простейший элемент данных с двоичным значением 1 или 0). Из множества таких защелок получаются хранилища мегабайтов и гигабайтов данных.



СЛЕНГ

ОЗУ (RAM)

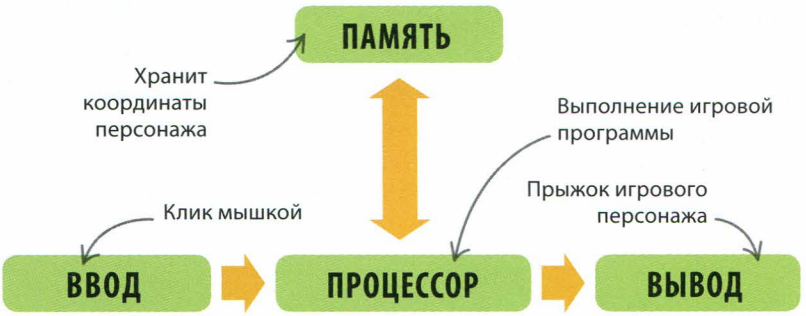
Память часто называют словом ОЗУ (сокращение от «оперативное запоминающее устройство» — англ. Random Access Memory) — это означает, что можно быстро обратиться к любой ее части, в отличие от устаревших типов памяти с поочередным доступом от начала к концу.

◁ Программы и данные
Программы непрерывно считывают, записывают и обновляют данные в памяти.

ЗАПОМНИ

Обработка информации

Процессор и память в сочетании с устройствами ввода и вывода — это все, что необходимо компьютеру. Например, пользователь игровой программы вводит координаты с помощью мышки, процессор производит расчеты, обращаясь при этом к памяти, и выдает результат — игровой персонаж на экране делает прыжок.



Необходимые программы

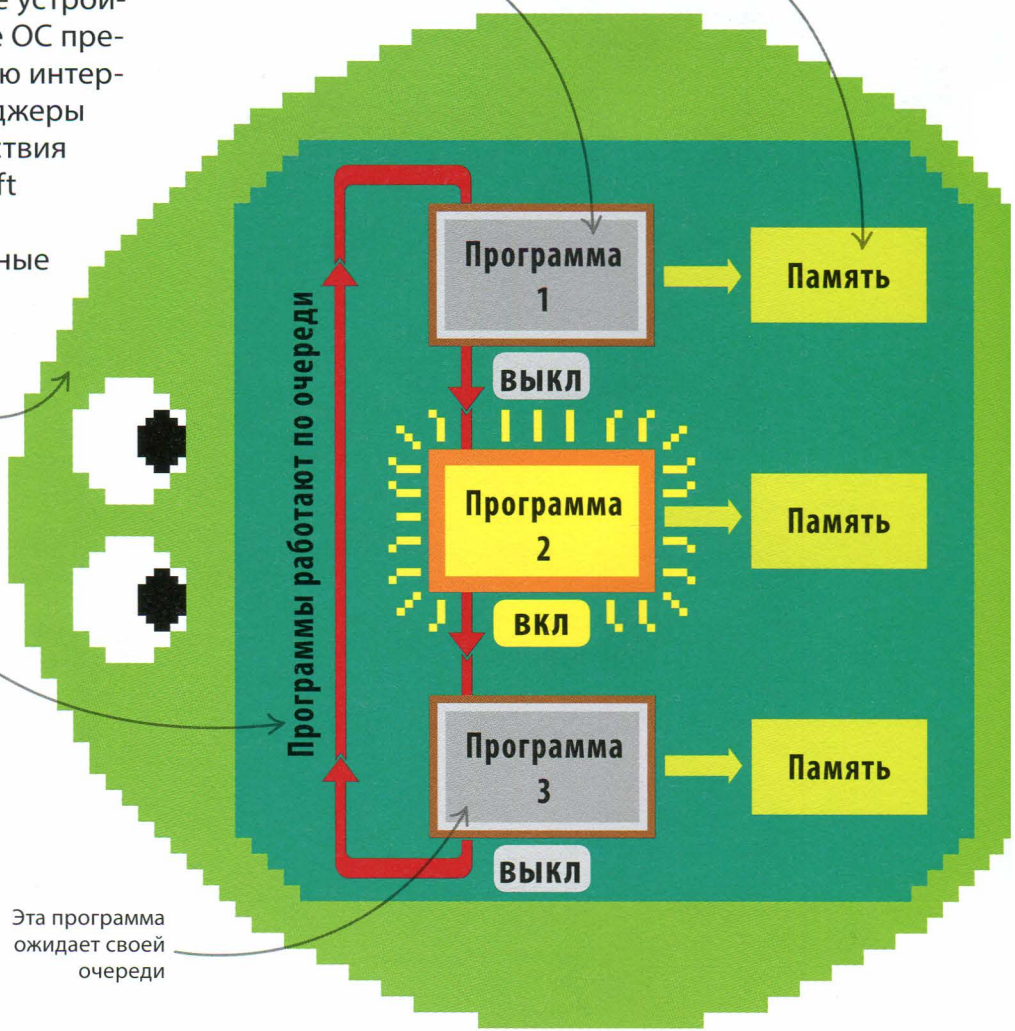
Есть программы, которые необходимы каждому компьютеру для правильной работы. Одни из наиболее важных программ — операционные системы, компиляторы и интерпретаторы.

СМОТРИ ТАКЖЕ

- ❮ 180–181 Внутри компьютера
- ❮ 182–183 Двоичная система
- ❮ 188–189 Процессоры и память

Операционная система

Операционная система (ОС) управляет ресурсами компьютера. Она определяет, каким программам разрешено выполнение, время их работы и какие устройства им доступны. Также ОС предоставляет пользователю интерфейсы (например, менеджеры файлов) для взаимодействия с компьютером. Microsoft Windows и Mac OS X — популярные операционные системы.



▷ Как это работает

Процессорное время делится на порции. Программа получает свою порцию, и, если она не успела завершиться за это время, ОС приостанавливает ее и переходит к следующей программе.

Компиляторы и интерпретаторы

Языки для составления программ (такие, как Python) называют «языками высокого уровня». Процессор не понимает этих языков — для их преобразования в понятный процессору низкоуровневый язык (машинный код) служат компиляторы и интерпретаторы.

△ **Компилятор**
Компиляторы создают машинный код, который можно сохранить и впоследствии запустить.

▷ **Интерпретатор**
Интерпретаторы одновременно преобразуют код и выполняют его.

ОС выполняет функцию моста между программами, которые нужно запустить, и компьютерным железом

Хранение данных в файлах

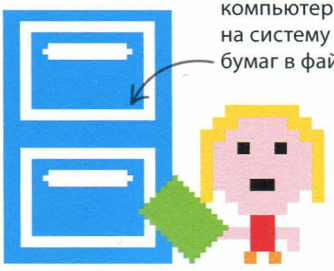
В компьютерной памяти можно хранить не только числа и символы, но и множество других данных, включая музыку, изображения и видео. Но как хранятся эти данные и где их искать?

СМОТРИ ТАКЖЕ

- ❮ 182–183 Двоичная система
- ❮ 188–189 Процессоры и память
- ❮ 190–191 Необходимые программы

Как хранятся данные?

Когда данные нужно отложить на потом, их записывают в файл. Этому файлу можно дать имя, по которому его будет легко найти снова. Файлы хранят на жестких дисках, на флешках и даже в интернете, так что данным ничего не грозит, даже если компьютер выключен.



Файловая система компьютера похожа на систему хранения бумаг в файлах

СОВЕТЫ ЭКСПЕРТА

Размер файла

Файлы — это блоки двоичных данных (бит).

Размер файла измеряют в таких единицах, как:

байты (B)

1 B = 8 бит (например, 10011001)

килобайты (KB)

1 KB = 1024 B

мегабайты (MB)

1 MB = 1024 KB = 1 048 576 B

гигабайты (GB)

1 GB = 1024 MB = 1 073 741 824 B

терабайты (TB)

1 TB = 1024 GB = 1 099 511 627 776 B

▽ **Информация о файле**

Файл — это не только хранимые данные, но и его свойства, которые содержат все нужные для ОС сведения о файле.

Сделай правый клик по файлу, чтобы увидеть его свойства, такие как тип, расположение и размер

СВОЙСТВА ФАЙЛА

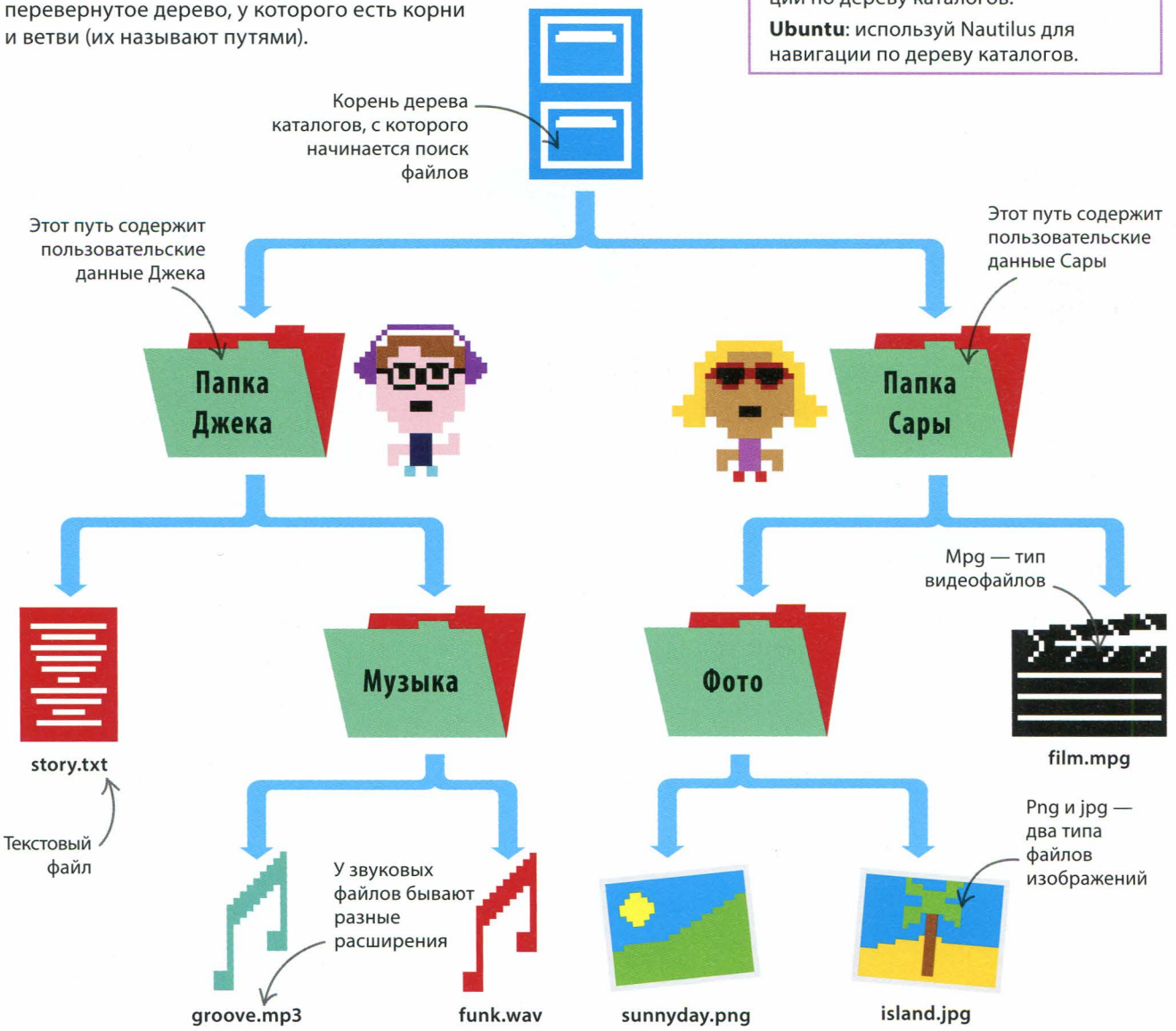
Имя файла должно быть запоминающимся — **имя**	groove
К какому типу принадлежит файл, обычно три символа — **расширение (тип файла)**	mp3
Программа, способная обработать данные файла — **чем открывать**	Music Player
Расположение файла на компьютере — **полный путь к файлу**	/Users/Jack/Music
Размер файла (см. табличку слева) — **размер**	50 MB

Каталоги

Файлы легче найти, если они грамотно рассортированы. С этой целью файлы можно группировать в каталогах, которые также называют папками. Часто удобно, чтобы каталоги были вложены один в другой, образуя дерево каталогов.

▽ **Дерево каталогов**

Когда в каталогах есть другие каталоги, получается структура, напоминающая перевернутое дерево, у которого есть корни и ветви (их называют путями).



СОВЕТЫ ЭКСПЕРТА

Работа с файлами

Программа «Файловый менеджер» помогает искать файлы и каталоги. Файловый менеджер в каждой операционной системе свой.

Windows: используй «Проводник Windows» для навигации по дереву каталогов.

Mac OS: используй Finder для навигации по дереву каталогов.

Ubuntu: используй Nautilus для навигации по дереву каталогов.

Интернет

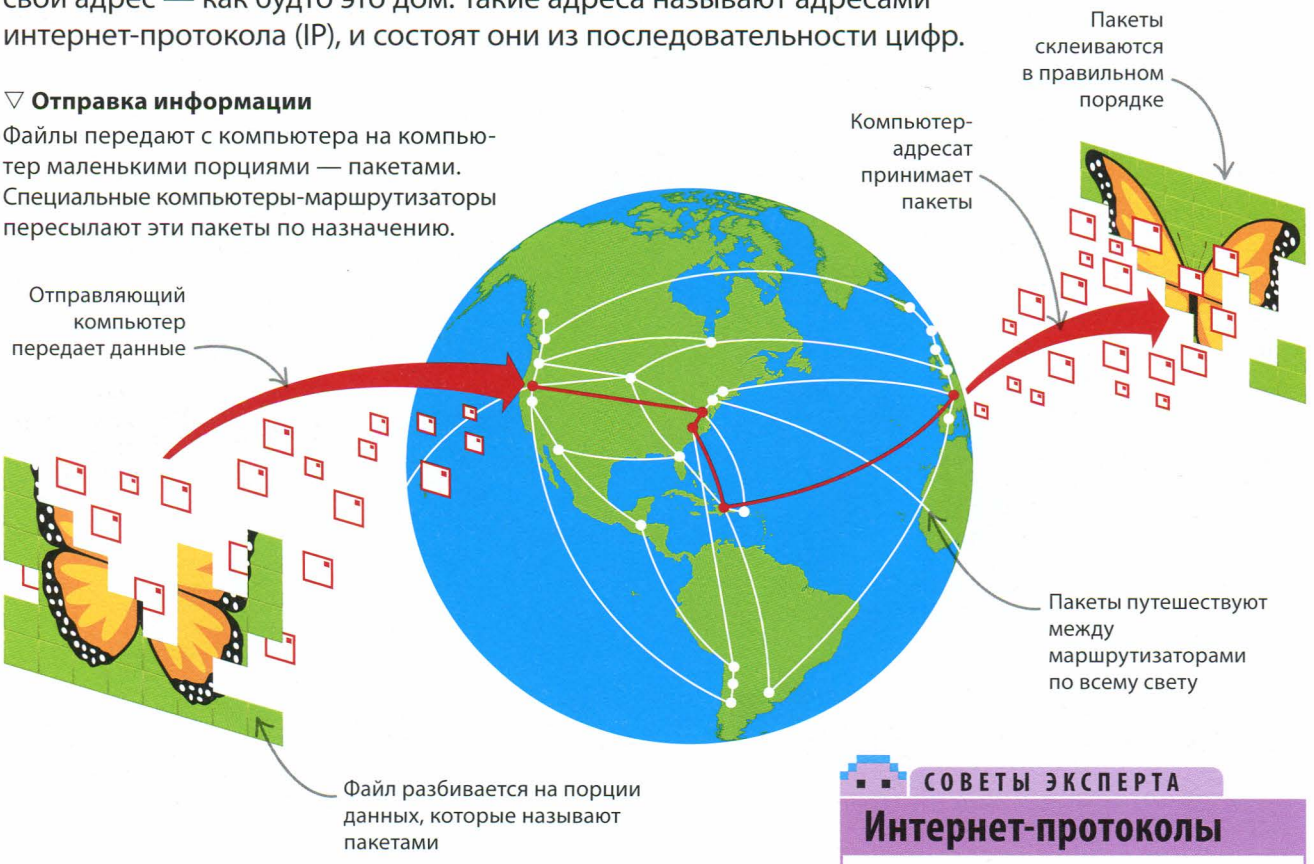
Интернет — это сеть, связывающая компьютеры по всему миру. Когда компьютеров так много, нужны специальные технологии организации данных.

СМОТРИ ТАКЖЕ
❮ 182–183 Двоичная система
❮ 192–193 Хранение данных в файлах

IP-адреса

Каждый подключенный к интернету компьютер или телефон имеет свой адрес — как будто это дом. Такие адреса называют адресами интернет-протокола (IP), и состоят они из последовательности цифр.

▽ **Отправка информации**
Файлы передают с компьютера на компьютер маленькими порциями — пакетами. Специальные компьютеры-маршрутизаторы пересылают эти пакеты по назначению.



◁ **Адреса**
Каждый пакет данных помечается IP-адресами отправителя и получателя. Доменные имена, вроде dk.com, тоже преобразуются в IP-адреса.

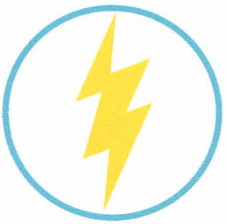
СОВЕТЫ ЭКСПЕРТА

Интернет-протоколы

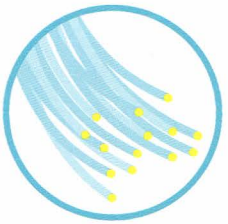
Протокол — это список правил. Интернет-протоколы содержат правила, определяющие размеры пакетов и их структуру. Интернет-устройства должны следовать этим правилам, чтобы успешно обмениваться данными.

Передача данных

Перед отправкой пакеты нужно преобразовать в двоичные сигналы (нули и единицы), которые можно передавать на большие расстояния. Для этого в каждом интернет-устройстве есть сетевой адаптер. Разные адаптеры передают данные разными способами.



△ **Электричество**
Для передачи нулей и единиц в виде электрических сигналов разной силы используют медные провода.



△ **Свет**
Специальные стекловолокна (оптоволоконные кабели) служат для передачи данных в виде световых импульсов.

△ **Радиоволны**
Разные типы радиоволн используются для беспроводной передачи нулей и единиц.

Порты

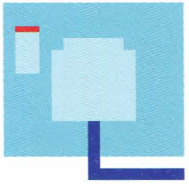
Так же, как письмо отправляют конкретному человеку, живущему по указанному адресу, пакеты можно отправлять конкретной программе, работающей на устройстве. Для обращения к программам используют номера, которые называются портами. У некоторых программ есть специально выделенные для них порты, например веб-браузер всегда принимает пакеты через порт 80.

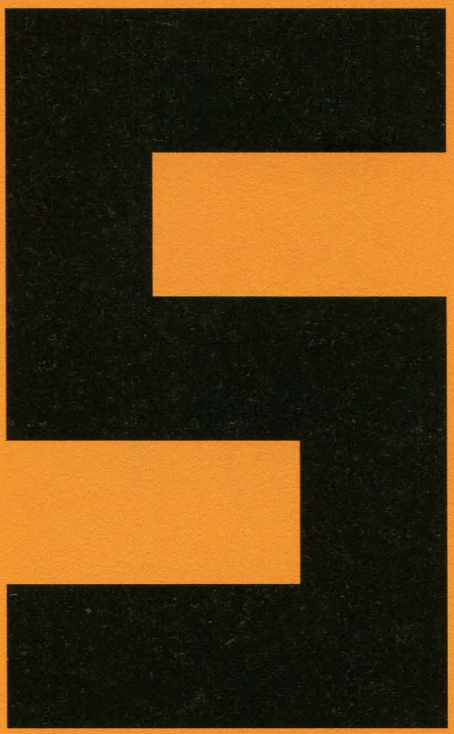
▽ **Номера портов**
Номера портов находятся в диапазоне от 0 до 65535 и делятся на три группы: общеизвестные, зарегистрированные и частные.

СОВЕТЫ ЭКСПЕРТА

Сокеты

Комбинация IP-адреса и порта известна как сокет (например, 192.168.0.1:2222). Сокеты позволяют программам обмениваться данными через интернет, что необходимо, например, для онлайн-игр.





Программирование в реальном мире

Компьютерные языки

Существуют тысячи языков программирования. Какой из них выбрать, зависит от многих факторов — к примеру, назначения программы и модели компьютера, на котором она должна работать.

СМОТРИ ТАКЖЕ

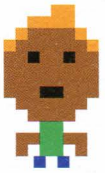
Компьютерные игры **204–205 ❯**

Мобильные приложения **206–207 ❯**

Популярные языки программирования

Некоторые языки обрели известность как особенно подходящие для создания того или иного вида программ. Вот как выглядит программа Hello World! («Привет, мир!») на нескольких популярных языках программирования.

```
#include <stdio.h>
main()
{
    printf("Hello World!");
}
```



△ **C**

Один из самых известных языков, C часто используется для программирования электронных устройств.

```
#include <iostream>
int main()
{
    std::cout << "Hello World!" << std::endl;
}
```

△ **C++**

Основан на C с добавлением новых возможностей. Подходит для создания быстрых программ, например приставочных игр.

```
#import <stdio.h>
int main(void)
{
    printf("Hello World!");
}
```

△ **Objective-C**

Основан на C, с новыми возможностями. Стал известен как язык для программирования компьютеров Apple Mac и iOS-устройств.

```
alert('Hello World!');
```

△ **JavaScript**

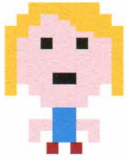
Служит для создания программ, работающих в веб-браузере, — вроде простых игр и интерактивных сайтов.

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

△ **Java**

Универсальный язык, работающий почти на всех компьютерах. На нем часто пишут программы для ОС Android.

```
<?php
echo "Hello World!";
?>
```



◁ **PHP**

Работает на сетевых серверах, обычно используется для создания интерактивных веб-сайтов.

Языки из прошлого

Многие языки, которые были популярны 20–30 лет тому назад, теперь устарели, хотя на некоторых системах их используют и сейчас. По нынешним меркам эти языки часто считаются сложными в использовании.

BASIC

Разработанный в 1964 году в Дартмутском колледже в США, BASIC был очень популярен как язык для первых домашних компьютеров.

Fortran

Разработанный в 1954 году в фирме IBM, Fortran чаще всего используется для математических расчетов на больших компьютерах. Его до сих пор применяют в прогнозировании погоды.

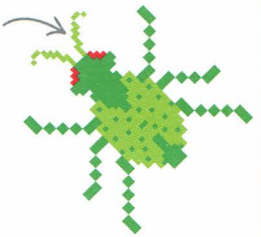
COBOL

Разработанный в 1959 году комитетом экспертов, COBOL до сих пор используется во многих банковских и бизнес-системах.

В ЖИЗНИ

Ошибка 2000 года

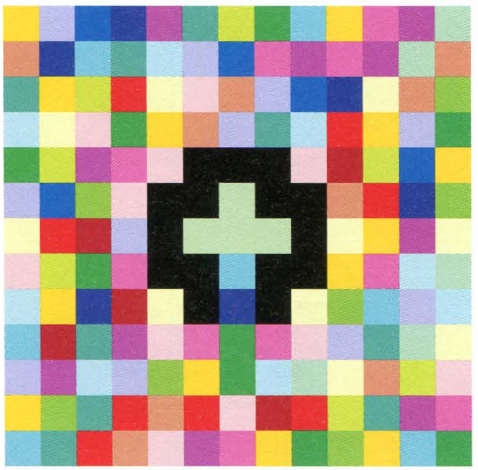
Во многих программах на устаревших языках вроде COBOL год был представлен двузначным числом (например, 99 вместо 1999). «Ошибка 2000 года» грозила системам при наступлении 2000 года, ведь тогда год в датах сбросился бы в 00.



Чтобы предотвратить «ошибку 2000 года», пришлось обновить компьютеры по всему миру

Экзотические языки

Среди тысяч разнообразных языков программирования некоторые были придуманы для особых, необычных целей.



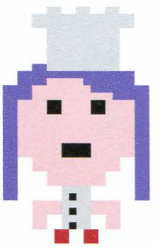
△ **Piet**

Программы на языке Piet выглядят как абстрактные картины. Это изображение — программа Hello World! («Привет, мир!»).

```
('&%:9]!~}|z2Vxwv-,POqponl$Hjig%eB@@>a=<M:9[p6tsl1TS/
QlOj)L(l&%$""Z~AA@UZ=RvttT`R5P3m0LEDh,T*?(b&`$#87[]{W
```

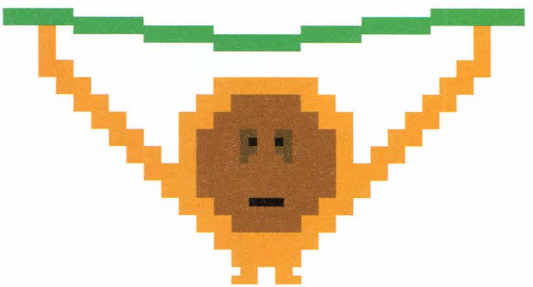
△ **Malbolge**

Язык Malbolge разработан так, чтобы программировать на нем было невозможно. Первый код на Malbolge появился лишь через два года после создания языка и был сгенерирован другой программой.



△ **Chef**

Программы на языке Chef выглядят как кулинарные рецепты! Впрочем, по ним вряд ли получится приготовить что-то съедобное.



△ **Ook!**

Язык Ook! разработан для орангутанов и содержит лишь три команды: Ook, Ook! и Ook? Из них можно составить еще шесть команд, таких как «Ook! Ook» или «Ook? Ook!».

Звезды программирования

Изо дня в день миллионы программистов по всему миру развивают компьютерную индустрию, но порой кто-нибудь делает особенно важный шаг. Вот несколько наиболее знаменитых программистов.

СМОТРИ ТАКЖЕ

‹ 18–19 Как стать программистом

Компьютерные игры **204–205 ›**

Ада Лавлейс

Страна: Великобритания

Годы: 1815–1852

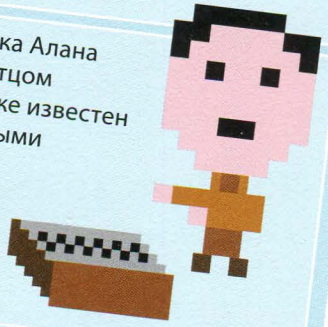
Известность: Ада Лавлейс считается первым программистом в истории. В 1843 году она опубликовала первую программу для аналитической машины Чарльза Бэббиджа (прообраз компьютера). Она также предложила способы представления символов в виде чисел.

Алан Тьюринг

Страна: Великобритания

Годы: 1912–1954

Известность: Математика Алана Тьюринга называют отцом информатики. Он также известен своими революционными работами по взлому немецких шифров во Вторую мировую войну.



Грейс Хоппер

Страна: США

Годы: 1906–1992

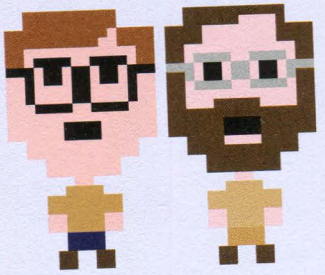
Известность: Грейс Хоппер создала первый в мире компилятор (транслятор понятной человеку программы в машинный код). Грейс была не только ученым, но и контр-адмиралом флота США!

Билл Гейтс и Пол Аллен

Страна: США

Годы: Гейтс родился в 1955-м, Аллен — в 1953-м

Известность: В 1970-х Билл Гейтс и Пол Аллен основали компанию Miscosoft. Они придумали ряд фантастически популярных программ, таких как Miscosoft Windows и Microsoft Office.

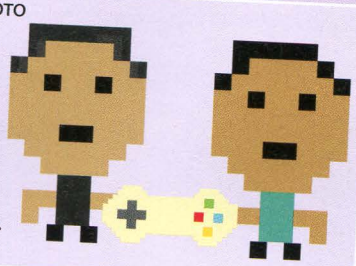


Гумпей Ёкои и Сигэру Миямото

Страна: Япония

Годы: Ёкои — 1941–1997, Миямото родился в 1952-м

Известность: Ёкои и Миямото работали на Nintendo — компанию, выпускающую видеоигры. Ёкои придумал приставку Game Boy, а Миямото — ряд легендарных игр, например «Супер Марио».



Тим Бернерс-Ли

Страна: Великобритания

Годы: родился в 1955-м

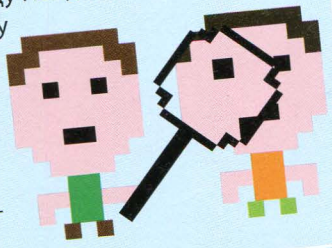
Известность: Работая в CERN (знаменитый исследовательский центр в Швейцарии), Тим Бернерс-Ли изобрел бесплатную для всех сеть интернет. В 2004 году королева Елизавета Вторая произвела его в рыцари.

Ларри Пейдж и Сергей Брин

Страна: США

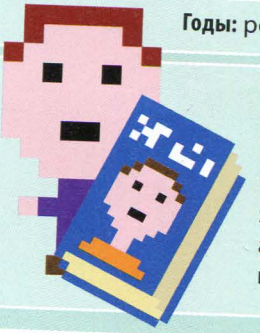
Годы: оба родились в 1973-м

Известность: В 1996 году Пейдж и Брин начали работу над проектом, который стал поисковиком Google. Их методы эффективного поиска произвели революцию в интернете.



Марк Цукерберг

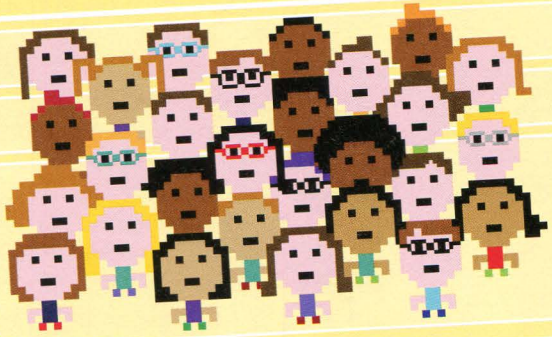
Страна: США



Годы: родился в 1984-м

Известность: Цукерберг запустил социальную сеть Facebook в 2004 году, еще учась в колледже. Теперь это компания-миллиардер, а Цукерберг — один из богатейших людей мира.

Движение открытого программного обеспечения



Страна: все страны

Годы: основано в 1970-м

Известность: Движение открытого ПО — это всемирное сообщество разработчиков, считающих, что программы должны быть бесплатными и доступными для всех. В рамках движения разработано много важных проектов, например операционная система GNU/Linux и Wikipedia.

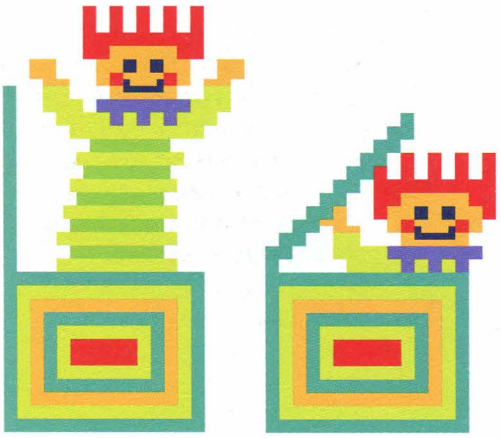
Трудолюбивые программы

СМОТРИ ТАКЖЕ
❮ 180–181 Внутри компьютера
❮ 192–193 Хранение данных в файлах

Компьютеры и программы стали привычной частью быта. Изо дня в день сложные программы, созданные для решения невероятно трудоемких задач, помогают людям в их делах.

Упаковка файлов

Почти все файлы, которые передаются по сети интернет, тем или иным способом упакованы (сжаты). Программа упаковки выявляет и удаляет избыточные данные, чтобы в файле осталась только полезная информация.

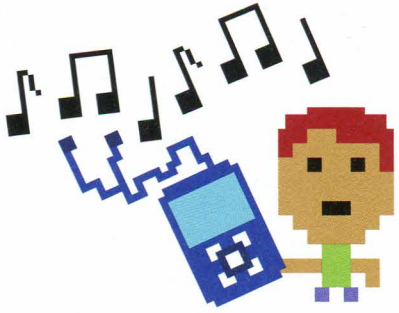


◁ **Сжатие данных**
Упаковка файлов напоминает запихивание попрыгунчика на пружинке в шкатулку, чтобы игрушка занимала меньше места.

В ЖИЗНИ

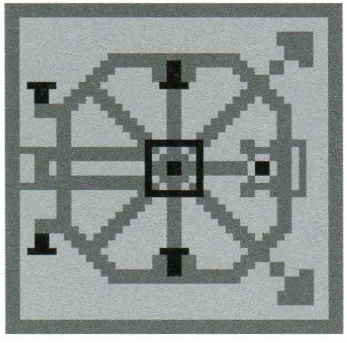
Звуковые файлы

Если бы не упаковка файлов, в музыкальный плеер помещалось бы от силы несколько песен. Сжатие звуковых данных позволяет закачать в обычный недорогой смартфон тысячи музыкальных треков.



Секретные шифры

Когда ты вводишь пароль на сайте, делаешь покупки или отправляешь сообщения в интернете, специальные программы шифруют твои личные данные, чтобы злоумышленник, перехватив их, ничего не смог понять. Глобальные банковские системы целиком зависят от программ шифрования данных.

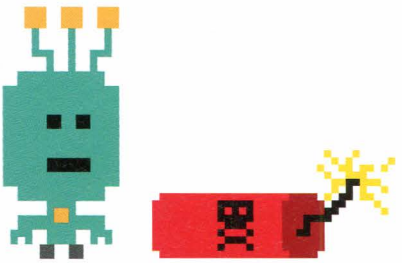


◁ **Криптография**
Криптография — это наука о шифрах. Сложные математические алгоритмы шифровки и расшифровки личных данных помогают обезопасить их от злоумышленников.

Искусственный интеллект

«Умными» бывают не только компьютерные игры. Искусственный интеллект (ИИ) используется в медицине, а также для управления роботами в опасных для людей условиях, например в зонах военных действий или природных катастроф.

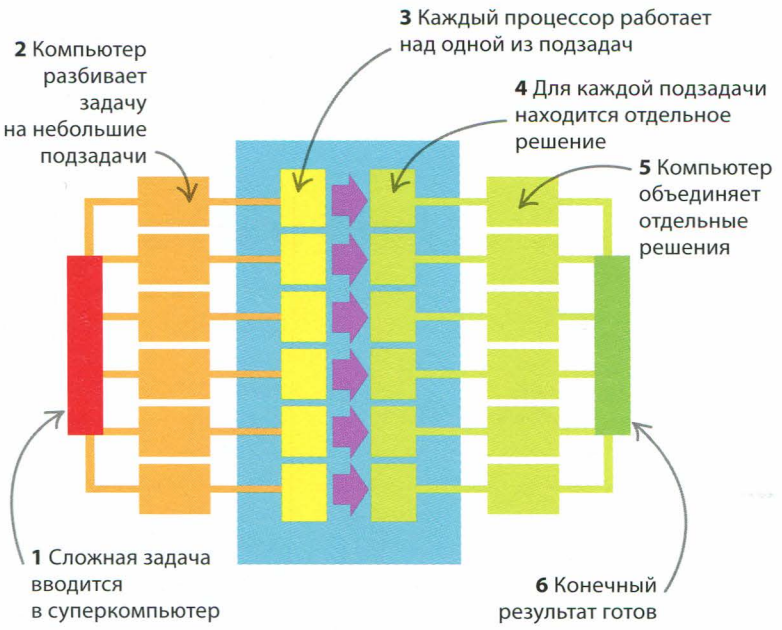
△ **Медицина**
Программные системы анализируют громадные объемы медицинской информации, соотносят ее с данными о пациенте и затем предлагают диагноз.



△ **Разминирование**
Используя «умных» роботов для разминирования мест, из которых эвакуированы люди, можно спасти жизни множеству солдат.

Суперкомпьютеры

Суперкомпьютеры, производящие расчеты движения космических объектов, состоят из тысяч процессоров, которые используют общие данные и быстро обмениваются информацией. Получается компьютер, способный выполнять миллионы операций в секунду.

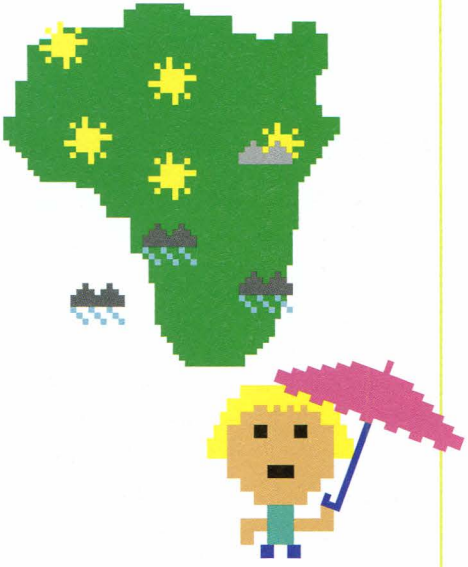


△ **Как это работает**
Задачи разбиваются на небольшие подзадачи, каждой из которых одновременно занимаются разные процессоры. Затем результаты объединяются в общее решение.

В ЖИЗНИ

Прогноз погоды

Предсказывать погоду очень сложно. Суперкомпьютеры работают с громадными объемами данных, которые нужны для построения прогноза. Каждый процессор суперкомпьютера рассчитывает погоду для небольшого участка карты. Затем результаты объединяются в общий прогноз.



Компьютерные игры

СМОТРИ ТАКЖЕ

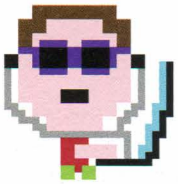
❮ **200–201** Звезды программирования

Мобильные **206–207** ❯ приложения

Что нужно для создания современной видеоигры? Все компьютерные игры делаются по разным рецептам из одних и тех же ингредиентов. Как правило, успешные игры выпускают целые команды разработчиков, а не программисты-одиночки.

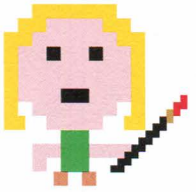
Кто делает компьютерные игры?

Даже простые игры на твоем телефоне, возможно, созданы большими группами людей. Чтобы игра вышла удачной и стала популярной, требуется внимание к деталям на каждом этапе ее разработки — для этого и нужно много людей с различными навыками.



△ **Программист**
Программисты пишут код, благодаря которому игра работает, однако для этого необходимо участие остальных членов команды.

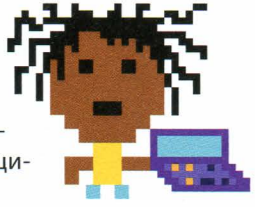
◁ **Дизайнер уровней**
Архитекторы виртуального мира, дизайнеры уровней создают интересные для прохождения игровые уровни.



△ **Дизайнер графики**
Игровые уровни и персонажи должны выглядеть привлекательно. Дизайнеры графики отвечают за внешний вид всех объектов в игре.

△ **Сценарист**
Для современных игр интересный сюжет важен не менее, чем для книг или фильмов. Сценаристы придумывают все игровые истории и всех персонажей.

▷ **Тестировщик**
Может показаться, что играть в игры целыми днями — это сказка, а не работа, однако тестировщики зачастую снова и снова проходят один и тот же уровень, выискивая ошибки.

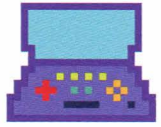


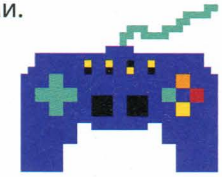
◁ **Звукоинженер**
Хорошей игре (так же, как и хорошему фильму) для создания настроения нужна качественная музыка и звуковые эффекты.

СЛЕНГ

Игровые консоли

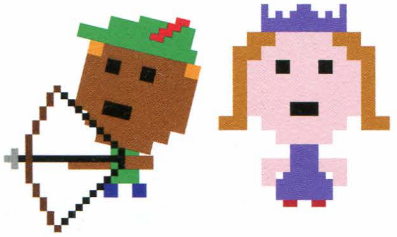
Игровая приставка, или консоль, — это специальная модель компьютера, созданная для запуска игр. Консоли, такие как PS4 и Xbox One, обычно оснащены мощными графическими и звуковыми процессорами, что позволяет делать игры более реалистичными.



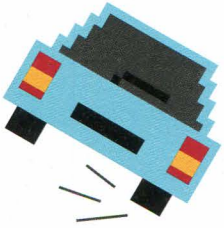


Элементы игры

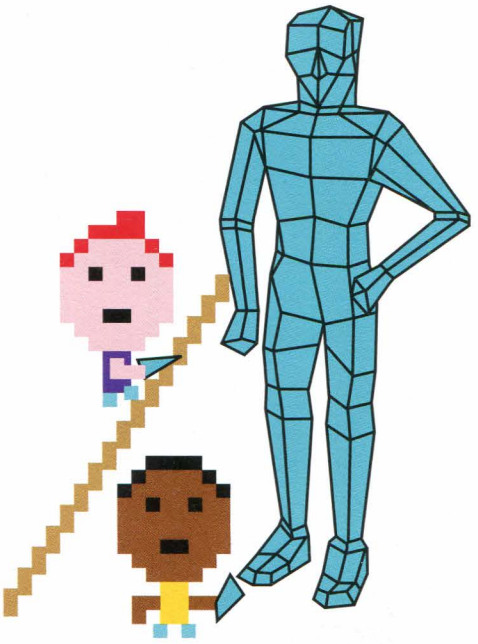
Наиболее типичные игровые элементы часто компонуют в виде «игровых движков». Движки представляют собой готовую основу для быстрого создания новых игр.



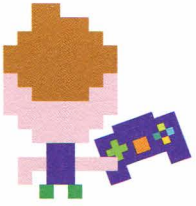
▷ Сюжет и логика игры
Каждой игре нужен хороший сюжет и некая цель, например спасение принцессы. Грамотно проработанная игровая логика не даст игроку заскучать.



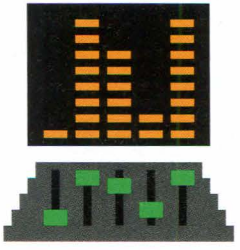
◁ Игровая физика
Чтобы игра была достоверной, нужно воссоздать в ее виртуальном мире законы мира реального, например гравитацию и реакцию на столкновения.



△ Графика
Чем более реалистичными становятся игры, тем сложнее их графика. Движения людей, дым и воду очень трудно изобразить достоверно.



▷ Управление
Хорошей игре нужно удобное и понятное управление. Удачное управление заставляет игрока забыть о том, что он нажимает на кнопки.



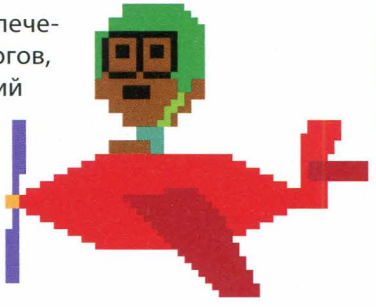
▷ Звук
Все голоса, звучащие в игре, должны быть сначала записаны. Также нужно создать фоновую музыку и сопровождающие звуковые эффекты.

△ Искусственный интеллект
Люди часто играют вместе с компьютерными игроками или против них. Искусственный интеллект позволяет компьютерным персонажам вести себя так, будто они живые.

В ЖИЗНИ

Серьезные игры

Игры нужны не только для развлечения. В обучении летчиков, хирургов, солдат и людей других профессий используют игры. В некоторых коммерческих фирмах сотрудники даже играют в стратегические игры, чтобы улучшить навыки планирования.



Мобильные приложения

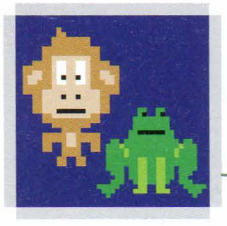
Мобильные телефоны открыли для программистов целый мир возможностей. Для лучшего взаимодействия с пользователем мобильные приложения могут подключаться к датчику движения или запрашивать географические координаты.

СМОТРИ ТАКЖЕ

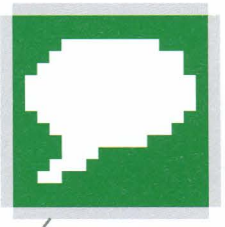
- ❮ 190-191 Необходимые программы
- ❮ 198–199 Компьютерные языки
- ❮ 204–205 Компьютерные игры

Что такое приложение?

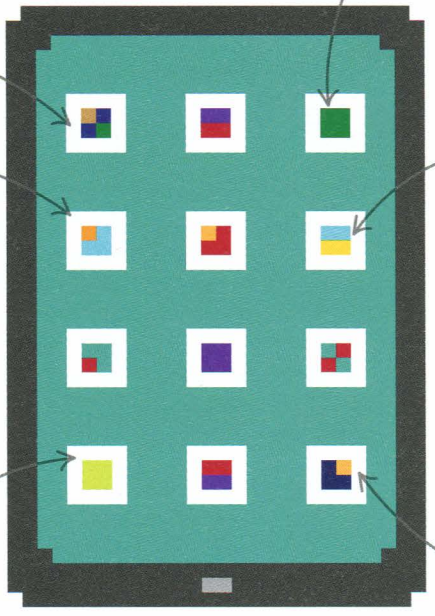
Мобильными приложениями, или просто приложениями, называют программы для мобильных устройств: смартфонов, планшетов и даже «умных» часов. Есть много видов приложений для выполнения разных задач.



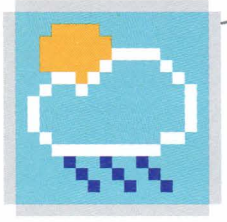
◁ **Игры**
Для мобильных устройств написано множество игр, от простых головоломок до динамичных приключенческих боевиков.



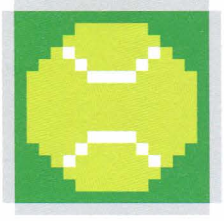
◁ **Социальные сети**
Приложения соцсетей позволяют общаться с друзьями, где бы они ни находились, — делиться мыслями, фотографиями, музыкой и видеозаписями.



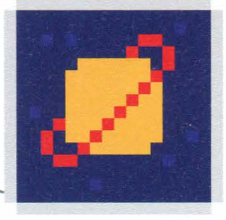
△ **Путешествия**
На основе географических координат и отзывов других пользователей приложения для путешественников выдают рекомендации по местным ресторанам, отелям и развлечениям.



△ **Погода**
Мобильные приложения показывают точный прогноз погоды для той местности, где ты находишься, и позволяют узнать погоду в любой точке земного шара.



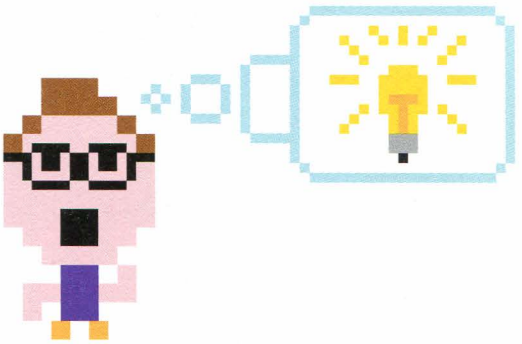
◁ **Спорт**
С помощью приложений можно вести план тренировок, а также следить за результатами спортивных соревнований.



△ **Образование**
С помощью обучающих приложений дети могут осваивать счет и грамоту, а взрослые — изучать иностранные языки.

Как создать приложение

Перед тем как создать новое приложение, нужно ответить на ряд вопросов. Что оно будет делать? На каких устройствах работать? Как оно будет взаимодействовать с пользователем? Ответив на эти вопросы, можно шаг за шагом создавать приложение.



1 Придумай концепцию

Концепция нового приложения должна подходить для мобильных устройств. Это может быть новая идея или развитие уже существующей.

Mac

Android

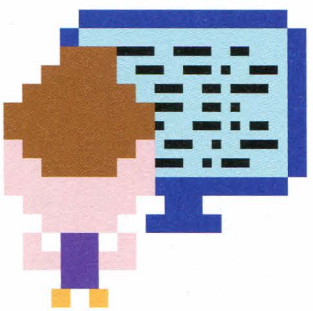
Windows

2 Какая операционная система?

Будет ли приложение предназначено для какого-то одного вида устройств? Есть инструменты, позволяющие написать приложение и затем адаптировать его к разным операционным системам.

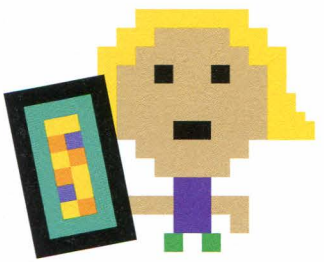
3 Научись создавать приложения

Теперь нужно освоить язык программирования и приобрести другие навыки, необходимые для создания качественного приложения. В этом помогут обучающие ресурсы в интернете.



4 Напиши программу

Чтобы создать хорошее приложение, нужно время. Простую работающую версию можно написать и за пару недель, но, чтобы приложение имело успех, на его разработку придется потратить хотя бы несколько месяцев.



5 Протестируй приложение

Если в приложении полно ошибок, оно быстро надоест пользователям. Чтобы этого не произошло, полезно сделать тесты частью кода, а также попросить друзей и родных протестировать приложение перед его выпуском.

Программирование для интернета

СМОТРИ ТАКЖЕ

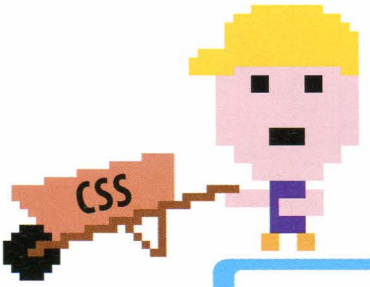
❮ 198–199 Компьютерные языки

Использование JavaScript 210–211 ❯

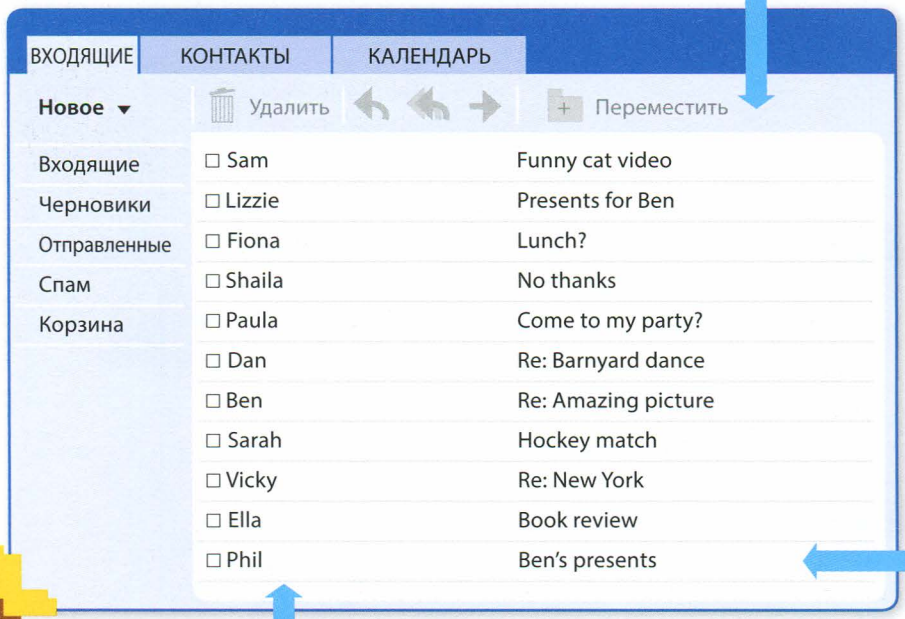
Интернет-сайты создают с помощью языков программирования, таких как Python. Один из самых важных языков — JavaScript, который позволяет делать сайты интерактивными.

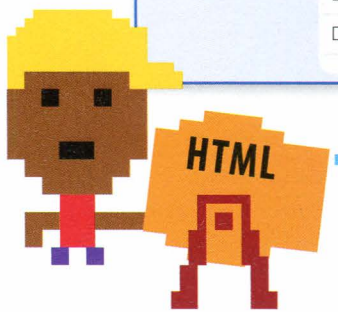
Как устроена страница сайта

Большинство интернет-страниц написано на нескольких разных языках. К примеру, сайт интернет-почты сделан с помощью CSS, HTML и JavaScript. JavaScript позволяет сайту реагировать на клики мышкой сразу, без перезагрузки страницы.



◁ **CSS**
Язык стилей CSS управляет цветом, шрифтами и внешним видом страницы.





◁ **HTML**
Язык разметки HTML позволяет описать основную структуру страницы, задав области для текста и изображений.

▷ **JavaScript**
JavaScript позволяет изменять страницу в ответ на действия пользователя. Например, при клике по заголовку письма появится его текст.

HTML

При открытии сайта интернет-браузер скачивает HTML-файл и запускает его, чтобы отобразить страницу. Если ты хочешь увидеть, как это происходит, набери код (справа) в окне программы IDLE (см. с. 92–93) и сохрани его в файл с расширением .html. Если сделать двойной клик по этому файлу, откроется окно браузера с надписью Hello World!

```
<html>
 <head>
  <title>The Hello World Window</title>
 </head>
 <body>
  <h1>Hello World in HTML</h1>
  <p>Hello World!</p>
 </body>
</html>
```

Блоки текста помечаются тегами (специальными метками HTML). Этот тег задает заголовок страницы

Тегами и выделяются параграфы в тексте

Этот тег показывает, где заканчивается HTML-код

Попробуем JavaScript

JavaScript понимают все современные браузеры. Код JavaScript обычно помещают внутрь HTML-кода, так что в следующем примере использованы сразу два языка. Блок JavaScript-кода помечается тегами <script>.

1 Введи JavaScript-код

В IDLE открой окно программы и введи этот код. Внимательно проверь его — в случае ошибки ты не увидишь ничего, кроме чистой страницы.

```
<script>
alert("Hello World!");
</script>
```

Тегом <script> помечается JavaScript-код

Команда alert отображает окно с сообщением

2 Сохрани файл

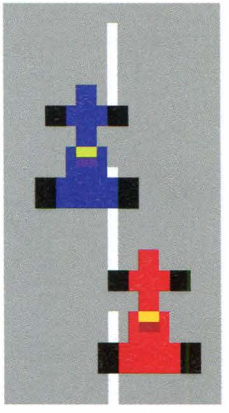
Сохрани файл под именем test.html, чтобы система поняла, что это HTML-код, а не файл Python. Сделай по файлу двойной клик, чтобы запустить его.

Не забудь задать файлу расширение «.html»

СОВЕТЫ ЭКСПЕРТА

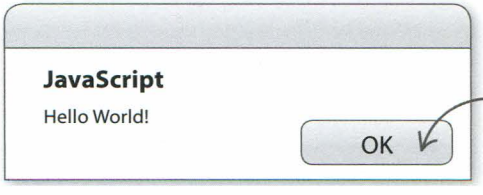
Игры на JavaScript

JavaScript настолько хорош для интерактивных сайтов, что на нем можно писать игры — от простых головоломок до динамичных гонок и стрелялок. Эти игры будут запускаться в любом современном браузере, не требуя установки. Кроме того, на JavaScript можно создавать веб-приложения вроде интернет-почты или календарей.



3 Появится всплывающее окошко

Запустится браузер, а затем появится окно с сообщением Hello World! Чтобы закрыть его, кликни по кнопке OK.



На JavaScript можно создавать интерактивные элементы, например кнопки

Использование JavaScript

JavaScript хорош для создания встроенных в HTML программ, «оживляющих» сайты, чтобы пользователь мог взаимодействовать с элементами страниц. JavaScript напоминает Python, однако он лаконичнее и сложнее в изучении.

СМОТРИ ТАКЖЕ

122–123 Циклы в Python

162–163 Реакция на события

208–209 Программирование для интернета

Ввод данных

JavaScript-код, как и код на Python, может запросить данные от пользователя, например во всплывающем окне. Эта программа предлагает пользователю ввести свое имя и в ответ здоровается с ним.

Создает окно запроса и помещает введенный ответ в переменную

Это сообщение будет показано в окне запроса имени

1 Запрос данных

В этом примере имя пользователя сохраняется в переменной. В IDLE открой окно программы, введи этот код и сохрани файл с расширением «.html».

```
<script>
var name = prompt("Please enter your name");
var greeting = "Hello " + name + "!";
document.write(greeting);
</script>
```

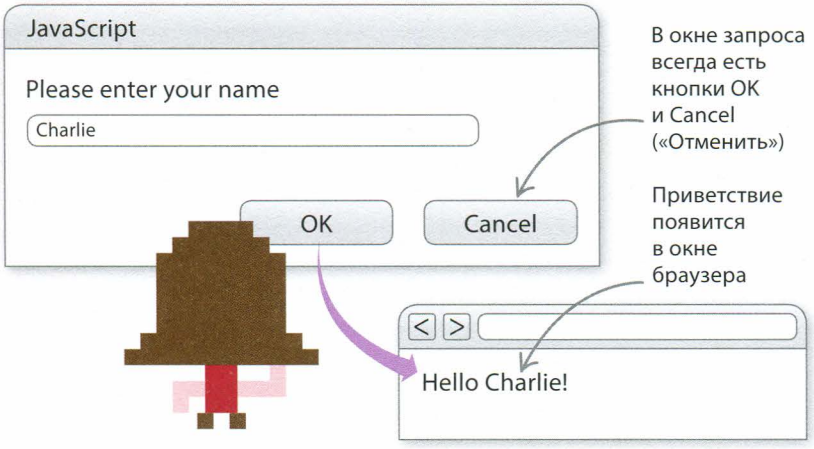
В конце каждой строки JavaScript-кода должна стоять точка с запятой

Тег </script> показывает, где заканчивается JavaScript-код

Отображает приветствие

2 Появится окно запроса

Сделай двойной клик по HTML-файлу, чтобы открыть его в браузере. Введи в окне запроса свое имя и нажми OK, чтобы увидеть приветствие.



В окне запроса всегда есть кнопки OK и Cancel («Отменить»)

Приветствие появится в окне браузера

СОВЕТЫ ЭКСПЕРТА

Вводи код внимательно

Программируя на JavaScript, всегда проверяй, верно ли введен код. В случае ошибки браузер пропустит весь блок JavaScript и отобразит чистую страницу без каких-либо предупреждений. Если это произошло, еще раз проверь код.

События

Событие — это любое действие в программе, например клик мышкой или нажатие на клавишу. Фрагмент кода, реагирующий на событие, называют обработчиком события. Обработчики событий используются в JavaScript очень часто и могут запускать различные функции, добавляющие страницам интерактивности.

1 Введи код

В этом примере событие (клик по кнопке) запускает функцию, которая печатает скороговорку. В IDLE введи этот код в окне программы и сохрани с расширением «.html».

```
<button onclick="tonguetwist()">Say this!</button>
<script>
 function tonguetwist()
 {
  document.write("She sells seashells");
 }
</script>
```

Имя функции

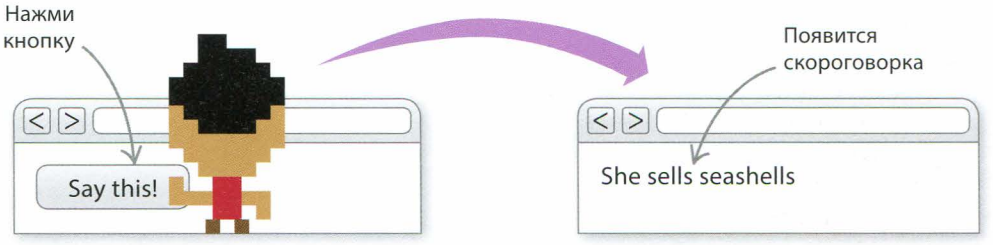
HTML-код связывает нажатие кнопки с функцией

Блок кода выделяется фигурными скобками — это работает так же, как отступы в Python

Код JavaScript-функции

2 Запусти программу

Сделай двойной клик по файлу, чтобы запустить программу в окне браузера.



Циклы в JavaScript

Цикл — это блок кода, который выполняется повторно. Использовать циклы гораздо удобнее, чем вводить одни и те же команды несколько раз.

1 Код цикла

Как и в Python, команда for в JavaScript служит для задания цикла. Часть кода, которую нужно повторять, заключают в фигурные скобки. В этом примере создается простой счетчик, значение которого возрастает на 1 при каждом повторе цикла.

```
<script>
for (var x=0; x<6; x++)
{
 document.write("Loop count: "+x+"<br>");
}
</script>
```

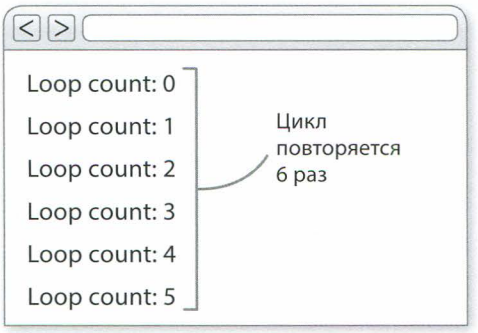
Тег <script> обозначает начало JavaScript-кода

Здесь создается счетчик с именем *x* и начальным значением 0, который увеличивается на 1 при каждом повторе

Печатает сообщение «Loop count:» и значение счетчика

2 Результат

Сохрани файл с расширением «.html» и запусти его. Цикл будет повторяться до тех пор, пока *x* меньше 6 (выражение «*x* < 6» в коде). Чтобы увеличить число повторов, измени число после «<» на большее.



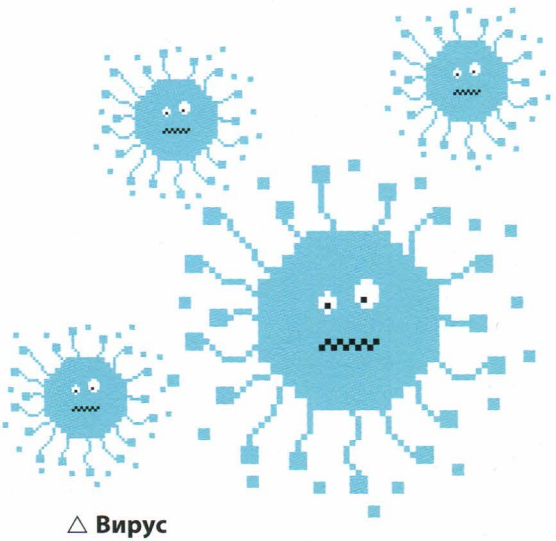
Зловредные программы

СМОТРИ ТАКЖЕ
❮ 194–195 Интернет
❮ 202–203 Трудолюбивые программы

Кроме игр и полезных приложений, существуют программы, созданные для того, чтобы красть твои данные или портить компьютер. Часто они кажутся безобидными, так что и не заметишь, как станешь их жертвой.

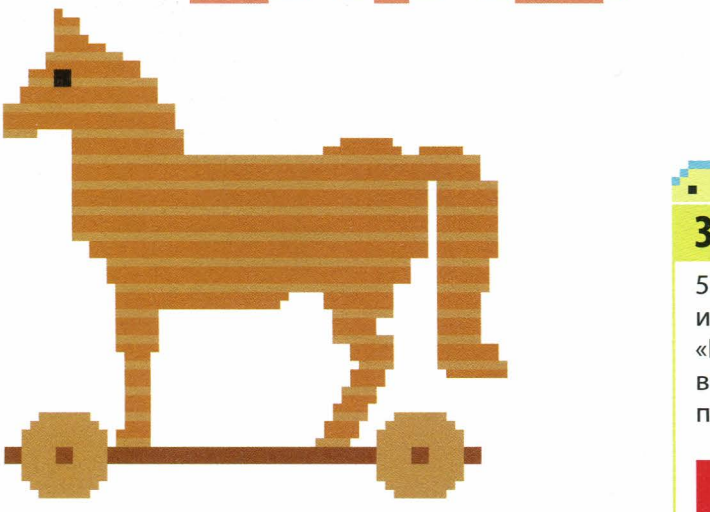
Вредоносное ПО

Программы, которые делают что-то исподтишка или без разрешения, называют вредоносным программным обеспечением (ПО). Хоть это и противозаконно, есть множество программ, готовых тайком проникнуть на твой компьютер.



△ **Вирус**
Так же как вирусы в теле человека, эти программы воспроизводят сами себя снова и снова. Распространяются они по электронной почте или другими способами передачи файлов между компьютерами.

▷ **Червь**
Червь — вредоносная сетевая программа, которая перебирается с компьютера на компьютер. Черви могут засорить и замедлить сеть — самый первый червь практически парализовал интернет в 1988 году.

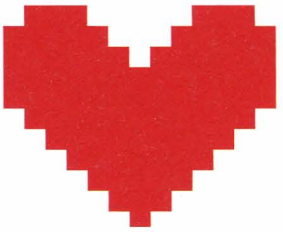


△ **Троянские кони**
Вредоносное ПО, притворяющееся безобидной программой, называют троянским конем (трояном). Название идет из Античности, когда во время войны греки подарили троянцам огромного деревянного коня. Внутри коня прятались солдаты, и это помогло грекам выиграть войну.

В ЖИЗНИ

Знаменитый червь

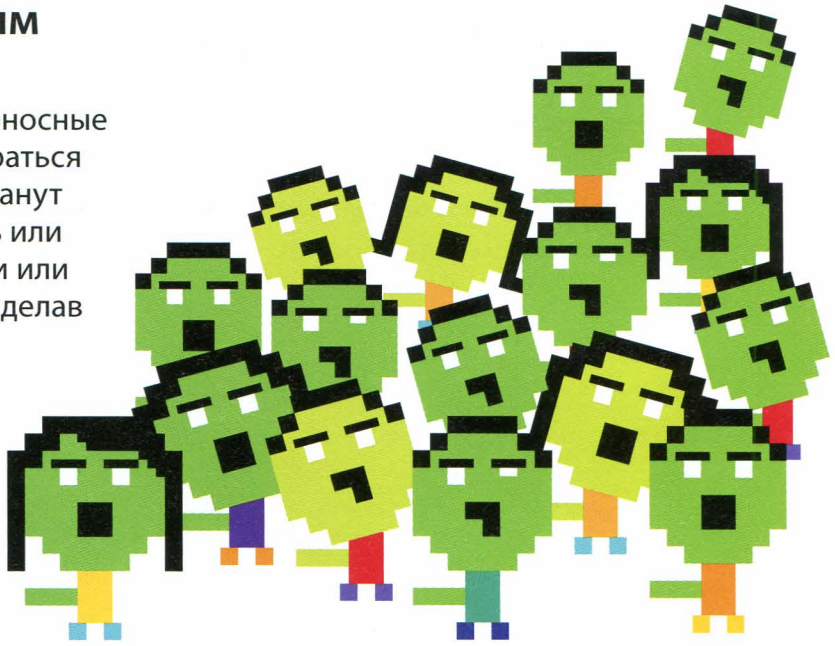
5 мая 2000 года филиппинские пользователи интернета получили письма с заголовком «ILOVEYOU» («ялюблютебя»). К письму под видом признания в любви прилагался повреждающий файлы червь.



◁ **ILOVEYOU**
Червь быстро проник на компьютеры по всему миру. Сумма нанесенного им ущерба составила более 20 миллиардов долларов.

Что нужно вредоносным программам

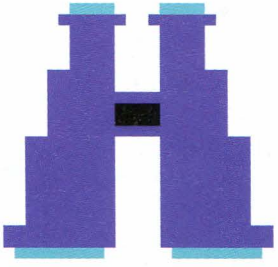
Вирусы, черви и трояны — вредоносные программы, которые хотят пробраться на твой компьютер, но что они станут делать потом? Они могут удалить или испортить файлы, украсть пароли или управлять твоим компьютером, сделав его частью зомби-ботнета.



▷ Зомби-ботнет
Ботнет — это группа зараженных компьютеров, которые по указу злоумышленников рассылают рекламу или перегружают сайты запросами, чтобы вывести их из строя.

Хорошие программы спешат на помощь

К счастью, люди не беззащитны перед вредоносным ПО. Серьезные компании работают над программами для борьбы с червями, вирусами и троянами. Среди таких программ наиболее известны антивирусы и файерволы.



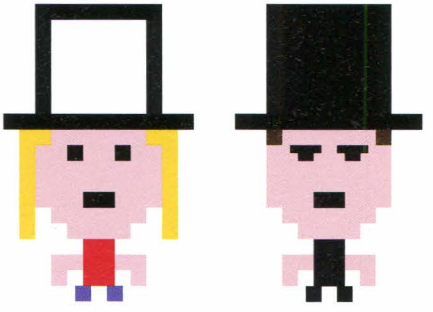
△ Антивирусы
Антивирусные программы ищут вредоносное ПО, сканируя файлы на компьютере и анализируя их содержимое на предмет подозрительного кода.

△ Файерволы
Файерволы стараются оградить компьютер от вредоносных программ, анализируя все приходящие из интернета данные.

СЛЕНГ

Хакеры

Программистов, которые изучают и создают вредоносное ПО, называют хакерами. Тех, кто пишет вредоносные программы, зовут черными хакерами, а тех, кто пишет программы для борьбы с вирусами, червями и троянами, — белыми хакерами.



Белый хакер Черный хакер

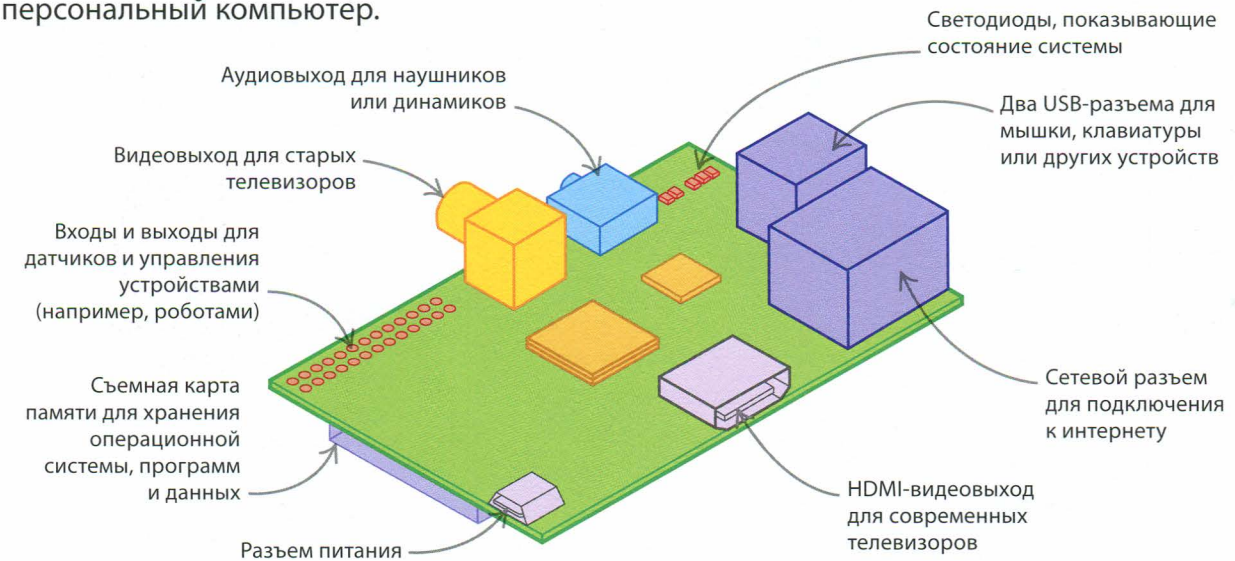
Мини-компьютеры

СМОТРИ ТАКЖЕ
❮ 180–181 Внутри компьютера
❮ 202–203 Трудолюбивые программы

Компьютеры бывают большими и дорогими. Однако немало и дешевых, очень компактных моделей, и благодаря этим свойствам их можно использовать новыми, интересными способами.

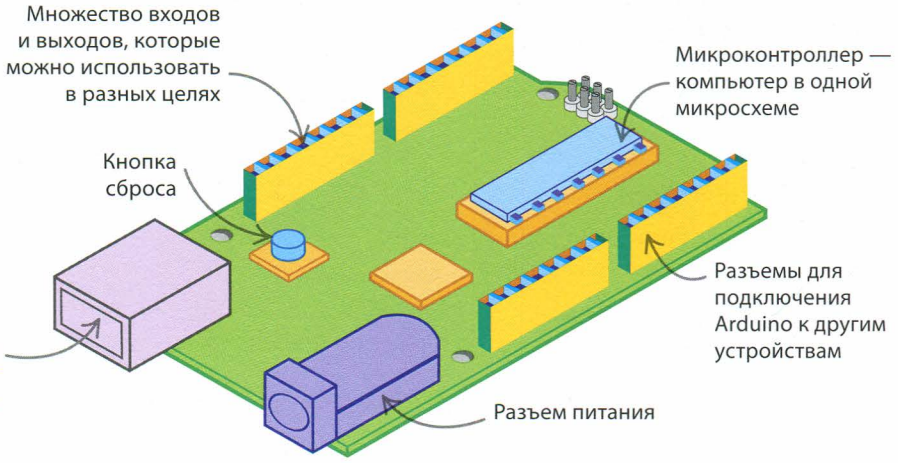
Raspberry Pi

Pi — компьютер размером с кредитную карту, разработанный для обучения информатике. Для своих размеров он очень мощный и может выполнять те же программы, что и обычный персональный компьютер.



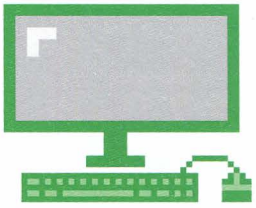
Arduino

Arduino дешевле, чем Pi, однако он менее мощный. Часто на его основе конструируют самодельные электронные приборы или роботов. Arduino может взаимодействовать с программой на Scratch.



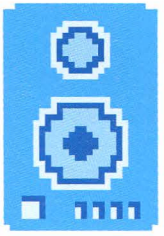
Применение мини-компьютеров

Благодаря широким возможностям по подключению устройств мини-компьютеры можно использовать множеством способов. Вот лишь несколько вариантов.



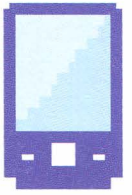
△ **Компьютер**

Подключи клавиатуру, мышку и монитор — получится персональный компьютер.



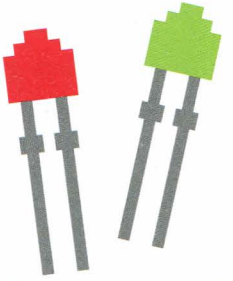
△ **Звук**

Подключи пару колонок и отправляй на них музыку по сети.



△ **Мобильные телефоны**

Подключи компьютер к интернету с помощью мобильного телефона.

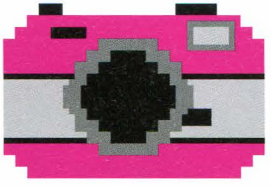


△ **Самодельные приборы**

Подключая простые электронные устройства, создавай приборы или роботов.

△ **Телевизор**

Подключи телевизор — получится медиацентр для просмотра фильмов и картинок.



△ **Видеокамера**

Подключив простую видеокамеру, сделай из нее веб-камеру.

△ **USB**

Подключи USB-накопитель, чтобы делиться файлами по сети.

△ **Карта памяти**

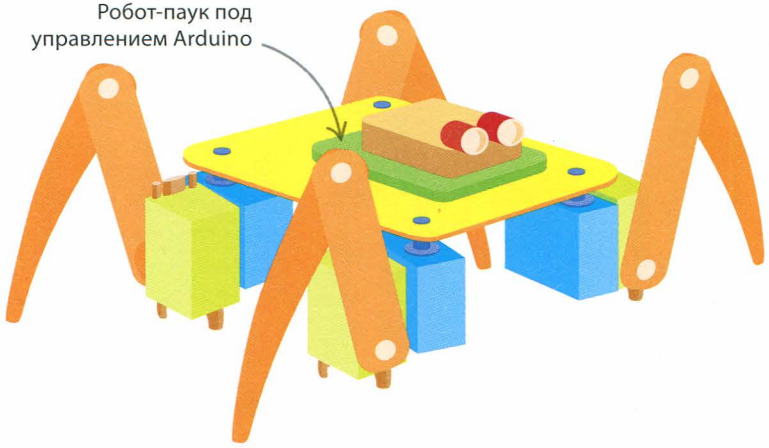
Меняй программы на мини-компьютере простой заменой карты памяти.

В ЖИЗНИ

Самодельные роботы

Из-за маленького размера, веса и низкой цены мини-компьютеры все чаще используют для создания различных роботов, таких как:

- **метеозонды**, которые собирают атмосферные данные;
- **мини-автомобили**, обходящие препятствия с помощью ультразвука;
- **манипуляторы**, способные захватывать и перемещать предметы.



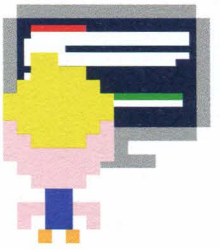
Стань знатоком программирования

СМОТРИ ТАКЖЕ
❮ 176–177 Что дальше?
❮ 214–215 Мини-компьютеры

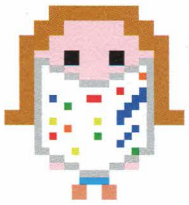
Секрет мастерства в том, чтобы программировать с удовольствием. Будь это хобби или карьера — до тех пор, пока тебе весело и интересно, ничто не остановит твое развитие.

Как развить навыки программирования

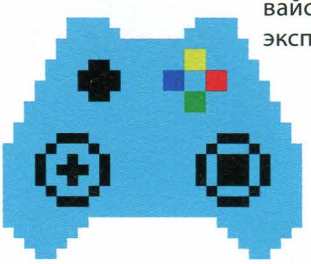
Навыки программирования улучшаются с опытом — как при игре на пианино или занятиях спортом. На то, чтобы стать экспертом, можно потратить годы, но дорога будет легкой, если она в радость. Вот несколько советов, как стать знатоком программирования.



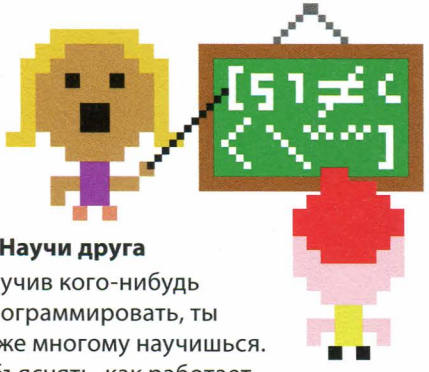
△ **Программируй больше**
Говорят, что повторение — мать учения, и это правда. Чем больше кодов ты пишешь, тем выше твое мастерство. Не останавливайся, и скоро ты станешь экспертом.



◁ **Больше любопытства**
Изучай сайты и книги по программированию, читай чужой код. Так ты узнаешь приемы и трюки, доходить до которых самостоятельно пришлось бы годами.

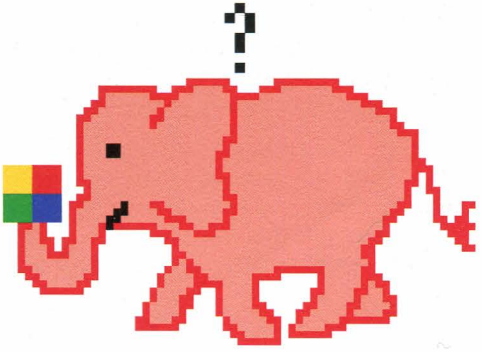


△ **Заимствуй идеи**
Встретив хорошую программу, подумай, какие идеи и приемы из нее можно использовать в своем коде. Даже лучшие программисты занимаются копированием чужих идей и их усовершенствованием.



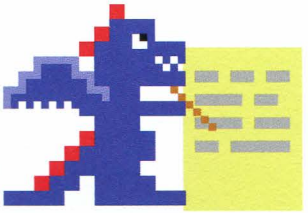
▷ **Научи друга**
Научив кого-нибудь программировать, ты тоже многому научишься. Объяснять, как работает код, — хороший способ проверить свои знания.

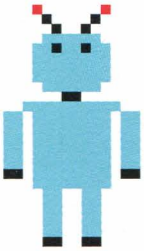
▷ **Тренируй мозг**
Твой ум как мышца — если его упражнять, он станет сильнее. Развивай мышление с помощью логических головоломок и математических задач, играй в интеллектуальные игры.



▷ Тестируй свой код

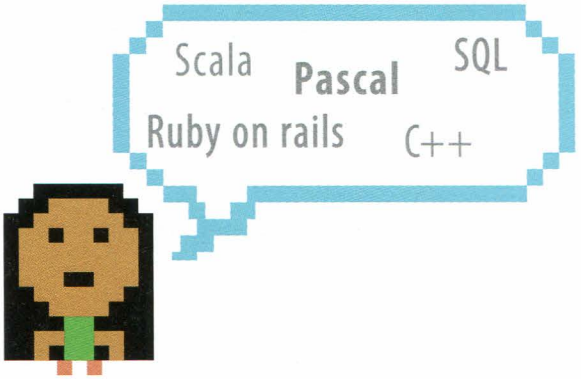
Проверяй программу, вводя некорректные значения и наблюдая, насколько она устойчива к ошибкам. Попробуй усовершенствовать свою программу или даже переписать чужую — так ты узнаешь чужие секретные приемы.





◁ Построй армию роботов

Ты можешь подключать свой компьютер к различным устройствам, от светодиодов до роботов. Выясняя, как завоевать мир, ты узнаешь много нового и интересного.

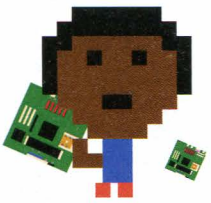


△ Изучай новые языки

Стань полиглотом. С каждым новым языком программирования ты узнаешь больше о языках, которые тебе уже известны (или ты думаешь, что известны). Бесплатные версии большинства языков можно найти в интернете.

▷ Разбери компьютер на части

Разбери старый компьютер, чтобы посмотреть, как он устроен, — в нем не так уж много деталей. Только сперва убедись, что никто не возражает! А лучше собери собственный компьютер и запусти на нем свои программы.



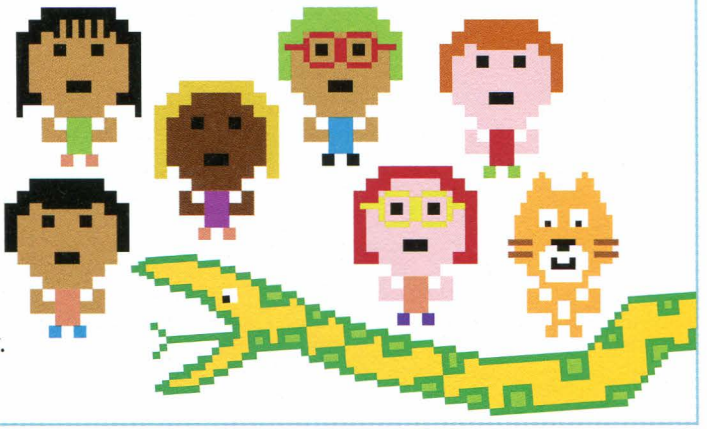
▷ Выиграй приз

Почему бы тебе не поучаствовать в соревнованиях по программированию? Их проводится множество, самой разной сложности. Международные соревнования вроде Google's Code Jam — самые сложные, но есть и варианты попроще.

ЗАПОМНИ

Получай удовольствие!

Программирование похоже на разгадывание головоломок. Это сложно и увлекательно, и ты часто будешь заходить в тупик — порой это огорчает. Но появятся и успехи — когда ты, найдя решение, с восторгом и гордостью посмотришь на свой работающий код. Чтобы не терять интереса, выбирай задачи, которые тебе по плечу, — проект быстро наскучит, если он слишком прост или неподъемно сложен. Не бойся экспериментировать и идти против правил, доверься любопытству. Но главное — получай удовольствие!



Глоссарий

ASCII
«Американский стандарт обмена информацией» — способ представления символов в двоичном коде.

GPU
Графический процессор для показа изображений на экране компьютера.

GUI
Графический интерфейс пользователя — видимые элементы управления программой, с которыми можно взаимодействовать, например кнопки и окна.

IP-адрес
Последовательность чисел — уникальный адрес компьютера в интернете.

Random
Функция, выдающая непредсказуемые результаты. Полезна при создании игр.

Run (запуск)
Команда запуска программы.

Алгоритм
Набор пошаговых инструкций по выполнению задачи — например, в виде компьютерной программы.

Байт
Элемент цифровых данных, состоящий из восьми бит.

Библиотека
Набор функций, который можно повторно использовать в разных проектах.

Бит
Двоичное значение — 0 или 1. Наименьший элемент цифровых данных.

Булево выражение
Вопрос, на который есть лишь два возможных ответа, например «истина» и «ложь».

Ввод
Данные, которые вводят в компьютер с клавиатуры, микрофона, мышки и так далее.

Вентиль
Логические вентили используются в компьютерах для принятия решений. Они принимают на входе один или несколько сигналов и на основе простого правила генерируют выходной сигнал. Примеры вентилей — «И», «ИЛИ», «НЕ».

Ветвь
Место в программе, где может быть выбран один из двух путей выполнения.

Вещественное
Число с плавающей десятичной точкой.

Вирус
Вредоносная программа, которая создает свои копии, чтобы заражать другие компьютеры.

Вредоносное ПО
Программы, созданные специально, чтобы повредить компьютер или украсть из него данные.

Вывод
Данные, которые программа показывает пользователю.

Вызов
Обращение к функции из кода программы.

Выполнение
См. Run (запуск).

Графика
Элементы на экране, которые не являются текстом, например картинки, иконки и символы.

Данные
Информация, например в виде текста, символов и чисел.

Двоичный код
Способ записи чисел и других данных, при котором используются только нули и единицы.

Железо
Физические детали компьютера, которые можно увидеть и потрогать, такие как провода, клавиатура, экран монитора.

Индекс
Номер элемента в списке. В Python индекс первого элемента равен 0, второго — 1 и так далее.

Интерфейс
Механизм взаимодействия с программой или устройством.

Каталог
Место для упорядоченного хранения файлов.

Компьютерная сеть
Способ связи между двумя или более компьютерами.

Контейнерный тип
Тип данных в программе, предназначенный для хранения нескольких других элементов данных.

Кортеж
Список элементов, разделенных запятыми, заключенный в круглые скобки.

Машинный код
Простейший язык, который понимает компьютер. Программы на других языках программирования должны быть преобразованы в него, чтобы их можно было запустить.

Модуль
Блок кода — отдельная часть программы.

Оператор (statement)
Наименьшая завершенная инструкция языка программирования.

Операция (operator)
Символ, выполняющий определенное действие, например «+» (сложение) или «–» (вычитание).

ОС
Операционная система (ОС) — среда для выполнения других программ, связывающая их с физическими устройствами.

Отладка
Поиск и исправление ошибок в программе.

Отладчик
Программа для поиска ошибок в других программах.

Ошибка
Неполадка в коде программы, из-за которой программа выполняется некорректно.

Память
Компьютерная микросхема для хранения данных.

Переменная
Именованное место для хранения данных, которые можно изменять.

Порт
Число, обозначающее адрес определенной программы на компьютере.

Пошаговый режим
Способ выполнения программы, когда инструкции выполняются по одной и можно на каждом шаге проверить, правильно ли работает программа.

Программа
Набор инструкций, указывающих компьютеру, как выполнять задачу.

Процессор
Компьютерная микросхема, выполняющая программы.

Сервер
Компьютер, хранящий файлы, доступные другим компьютерам по сети.

Синтаксис
Правила, определяющие, как нужно писать программу, чтобы она работала.

Событие
Действие, на которое программа может отреагировать, например нажатие клавиши или клик мышкой.

Сокет
Сочетание IP-адреса и порта, позволяющее программам общаться между собой в сети интернет.

Софт
Программы, запущенные на компьютере и управляющие его работой.

Спрайт
Графический объект, который можно перемещать.

Строка
Последовательность символов, может содержать цифры, буквы и знаки (например, запятые).

Троян
Вредоносная программа, которая, чтобы обмануть пользователя, притворяется другой программой.

Упаковка (сжатие)
Способ уменьшения объема данных, чтобы они занимали меньше места.

Файл
Набор данных, хранящихся под общим именем.

Функция
Фрагмент кода, выполняющий часть общей задачи.

Хакер
Человек, который взламывает компьютерные системы. Белые хакеры ищут проблемы безопасности, чтобы их исправить. Черные хакеры незаконно вторгаются в компьютерные системы, чтобы повредить их или извлечь какую-то выгоду.

Целое
Число без десятичной точки и не записанное в виде дроби.

Цикл
Часть программы, которая выполняется повторно (чтобы не вводить одни и те же команды несколько раз).

Шестнадцатеричная система
Система исчисления с основанием 16, где значения от 10 до 15 представлены латинскими буквами от A до F.

Шифрование
Способ кодирования данных, чтобы они были доступны лишь определенному кругу людей.

Юникод
Универсальная кодировка текста, в которой можно записать тысячи разных букв и знаков.

Язык программирования
Язык, на котором можно давать инструкции компьютеру.

Алфавитный указатель

Жирным шрифтом выделены основные страницы по теме.

Благодарности

DORLING KINDERSLEY благодарит: Вики Шот, Мэнди Ири, Сандру Перри и Таништу Чакраборти за помощь в оформлении; Оливию Стэнфорд за помощь в редактировании; Кэролайн Хант за чтение корректуры; Хелен Питерс за алфавитный указатель; Эдама Бакенбери за техническую поддержку.

DORLING KINDERSLEY INDIA благодарит: Канику Миттал за помощь в оформлении; Павана Кумара за помощь в подготовке к выпуску; Салони Сингх за контроль над работой редакторов.

Scratch разработан Lifelong Kindergarten Group в MIT Media Lab, см.: http://scratch.mit.edu