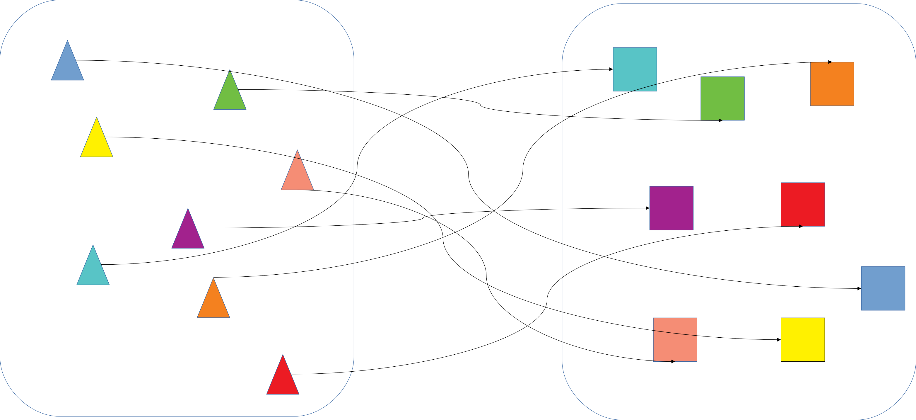
Математика для data science: Функции. Полиномы. Анализ

**Функция (отражение, оператор, преобразование)** - в математике соответствие между элементами двух множеств, установленное по такому правилу, что каждому элементу одного множества ставится в соответствие некоторый элемент из другого множества.

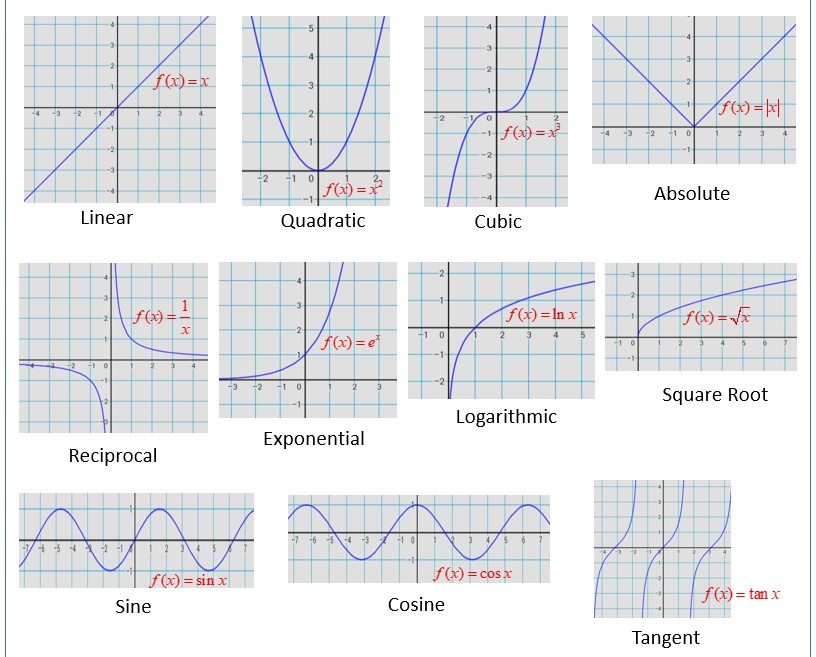
Множество В

Множество A



Если известно множество А и множество В, функция такое правило, которое каждому элементу множества А ставится в соответствие элемент множества В.

Виды функций:



В data science функции используются для построения нейросетей, анализа данных, для нахождения корреляций (корреляция – статистическая взаимосвязь двух и более случайных величин).

Выбор вида функции подбирает data scienсе.

Функция может быть представлена несколькими способами:

* математически (с помощью формулы)
* графически (график)
* таблично (с помощью таблицы)
* программно (python код)

**Программный способ представления**

Рассмотрим применение функции в контексте датасета о фильмах и актерах: функция — это некая модель, которая для каждого элемента множества актеров (множество А) подбирает элемент из множества фильмов (множество В).

Множество актеров и актрис

Множество фильмов

Сергей Бодров

Бригада

Сергей Безруков

Дети Арбата

Чулпан Хаматова

Функция

Плохая соседка

Анастасия Задорожная

Брат

Напишем код для функции, которая для каждого актера или актрисы из имеющегося множества будет находить фильм, в котором они играли. Для этого будем использовать Set представление в python.

*actors = set(['Сергей Бодров', 'Сергей Безруков', 'Чулпан Хаматова', 'Анастасия Задорожная'])*

Для написания функции отображения актеров в подпространство фильмов будем использовать словари (dictionary) в python

*dict\_actors = dict()*

*dict\_actors['Сергей Бодров'] = 'Брат'*

*dict\_actors['Сергей Безруков'] = 'Бригада'*

*dict\_actors['Чулпан Хаматова'] = 'Дети Арбата'*

*dict\_actors['Анастасия. Задорожная'] = 'Плохая соседка'*

В словаре: (множество А) фамилия и имя актера – ключ, (множество В) фильм, в котором снимался (лась) – значение. Словарь можно рассматривать как функцию для представления данных, где одному элементу из множества А, будет соответствовать один элемент из множества В.

Теперь для каждого актера в словаре dict\_actors хранится название фильма, в котором он/она участвовали.

Можно проверить, что все элементы множества actors отображаются в подпространство фильмов правильно.

***for*** *el* ***in*** *actors:* ||пробегаемся по всем актерам

*print(el + ': ' + dict\_actors[el])* ||печатаем фильм в котором участвовал актер

Сергей Бодров: Брат

Анастасия Задорожная: Плохая соседка

Чулпан Хаматова: Дети Арбата

Сергей Безруков: Бригада

В примере выше каждый актёр встречается только в одном фильме. Совсем иначе выглядит структура данных, если актёры встречаются в более чем одном фильме. В этом случае словарь необходимо заменить на таблицу словарей (хэш-таблицу). Хэш-таблица отличается от функции тем, что для одного аргумента (ключа) может быть несколько значений.

По сути создается словарь, где ключ - так же фамилия и имя актера, а значение – список фильмов, в которых тот снимался.

*dict\_actors\_2 = dict()*

*dict\_actors\_2['Сергей Бодров'] = ['Брат', 'Брат2']*

*dict\_actors\_2['Сергей Безруков'] = ['Бригада', 'Есенин','Кинопоэзия','Таинственная страсть']*

*dict\_actors\_2['Чулпан Хаматова'] = ['Дети Арбата', 'Гуд бай, Ленин!','Кинопоэзия', 'Таинственная страсть']*

*dict\_actors\_2['Анастасия Задорожная'] = ['Плохая соседка']*

Напечатаем фильмы, в которых участвовали актеры:

***for*** *el* ***in*** *actors:* ||пробегаемся по всем актерам

*films = ''* || создаем список фильмов в которых участвовал актер

***for*** *film* ***in*** *dict\_actors\_2[el]:* ||пробегаемся по словарю

*films += film + ', '*

*print(el + ': '+ films)* || печатаем все фильмы в которых участвовал актер

Если мы хотим проверить, в каких фильмах участвовали Сергей Бодров и Чулпан Хаматова совместно, достаточно проверить пересечение множества фильмов одного актера и другого:

**for** film **in** dict\_actors\_2['Чулпан Хаматова']: ||пробегаемся по всем фильмам одного актера

|| проверяем есть ли совпадения из списка второго актера с фильмами из списка первого актера

***if*** *film* ***in*** *dict\_actors\_2['Сергей Безруков']:*

*print(film)*

Кинопоэзия

Таинственная страсть

Заметьте, в данном случае мы использовали встроенную команду in, которая проверяет, имеется ли элемент в списке list.

Если дана вида: alt text.

Программный код будет выглядеть следующим образом:

***from******math******import*** *pow* || создаем библиотеку

***def*** *func(x):* || создаем функцию

*stepen = 1/4\*pow(x, 5) - pow(x, 4) + 1/4\*pow(x, 3) + 3/2\*pow(x, 2) +1* || вычисляем степень

*y = 1 - pow(2, stepen)* || вычисляем у равный 1 минус 2 в степени

***return*** *y*

**Графический способ представления функции**

Графически эта же функция будет представлена следующим образом.

Нарисуем график написанной функции:

*# создаем вектор x*

*x1 = np.linspace(-3.0, 3.0, 100)* || задали вектор, который имеет сто значений в интервале от -3 до 3

*# на основе вектора x строим значения для вектора y*

*y1 = [func(x1[i])* ***for*** *i* ***in*** *range(len(x1))]*

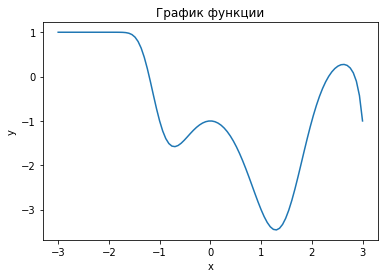
*plt.plot(x1, y1)* || вывод на экран

plt.title('График функции')

plt.ylabel('y')

plt.xlabel('x')

plt.show()



Если функция представлена несколькими формулами, например, для написания алгоритма для функции y=abs(x), легче всего разделить пространство аргументов на плоскости на 2 отрезка: -∞ до 0 включительно и от 0 до +∞:

alt text

*Напоминание! На графике функции различают точки, не включенные в график, с помощью "выколотых" точек (точка (0, 1) на графике). Точку, включенную в график, показывают увеличенного размера (точка (0, 2) на графике).*



*Что означает, что если значение х = 0, то значение у = 2,*

*при х равному* -∞ до 0 и 0 до +∞ у=1 (0 не включается)

код для данной функции и отрисовки графика функции будет выглядеть следующим образом:

x1 = np.linspace(-3.0, 3.0, 100) || создаем вектор из ста значений от -3 до 3

y1 = [1 for \_ in range(len(x1))] || считаем у

y1[len(x1)//2] = 2 || вычисление у для среднего х (средний х равен 0)

plt.scatter(x1, y1, marker='o')

plt.scatter(0, 1, s=100, marker='o') || рисуем исключённую точку

plt.scatter(0, 1, s=50, marker='o', c='white') || обозначаем что выколотая точка белого цвета и имеет размер 50

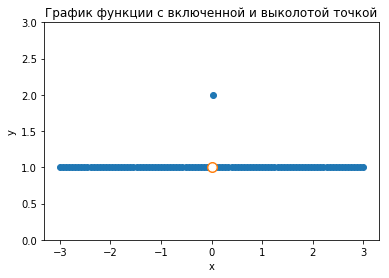
plt.title('График функции с включенной и выколотой точкой')

plt.ylabel('y')

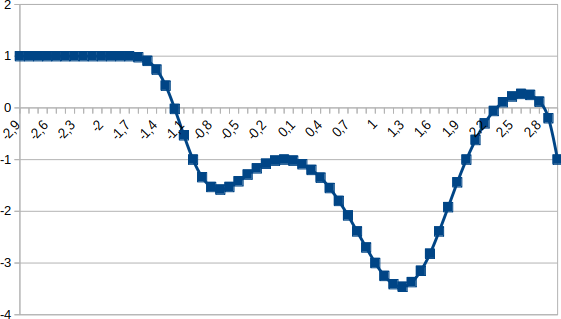
plt.xlabel('x')

plt.ylim(0, 3)

plt.show()



С помощью графического способа представления информации мы можем найти локальный минимум (максимум) функции (локальный минимум (максимум) – минимум(максимум) функции на некотором интервале, глобальный минимум (максимум) – минимум(максимум) на всем интервале значений функции)



Определение минимумов (максимумов) позволяет анализировать зависимость одного значения от другого. Например, зависимость курса валют от негативных слов в сети.

Полиноминальные функции или многочлены

Целая рациональная функция (также полиномиальная функция) — числовая функция одного действительного переменного вида:



где n-натуральное число.

На практике любую функцию можно представить в виде суммы нескольких полиномов. Примеры полиноминальной функции: линейная функция, квадратная функция, функция третьей степени и т.п.

Полиноминальная функция состоит из: чисел, умножения, сложения, переменной x

Для проверки является ли функция полиномиальной, необходимо ее упростить и проверить, является ли упрощенная функция стандартной формой полиномиальной функции. Пример:



Правило упрощения функции: раскрыть скобки, сложить переменные, упорядочить.

Одним из основных показателей полиномиальной функции является её **степень**. Степень полиномиальной функции - натуральное число n - наибольший показатель степени переменной x.



Степень полинома равна 6

**Степень полинома = 6.**

Степень определяет поведение функции при х стремящимся к бесконечности. Чем больше значение х, тем больше значении функции стремится к значению слагаемого с максимальной степенью.

Пример:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 2x^(3) | 8x^(2) | -13x | f(x) |
| 1 | 2 | 8 | -13 | -3 |
| 10 | 2,000 | 800 | -130 | 2,670 |
| 100 | 2,000,000 | 80,000 | -1,300 | 2,078,700 |
| 1,000 | 2,000,000,000 | 8,000,000 | -13,000 | 2,007,987,000 |

Можно заметить, что максимальное значение функции полинома стремится к значению третьей степени при большом значении x:

Аналогичную картину наблюдаем при отрицательных значениях x:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 2x^(3) | 8x^(2) | -13x | f(x) |
| -1 | -2 | 8 | 13 | 19 |
| -10 | -2,000 | 800 | 130 | -1,070 |
| -100 | -2,000,000 | 80,000 | 1,300 | -1,918,700 |
| -1,000 | -2,000,000,000 | 8,000,000 | 13,000 | -1,991,987,000 |

Таким образом, для достаточно больших положительных и отрицательных значениях x полиномиальная функция описывается слагаемым с максимальной степенью:



**Степень = 0: функция-константа**

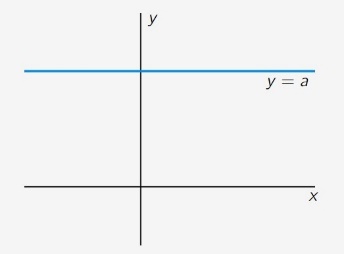
Формула: f(x)=a

График: горизонтальная линия

Пример програмного построения полинома: степень = 0.

*x1 = np.linspace(-3.0, 3.0, 100)*

*y1 = [3 for \_ in range(len(x1))]*

*plt.plot(x1, y1, '-o')*

*plt.title('График функции')*

*plt.ylabel('y')*

*plt.xlabel('x')*

*plt.show()*

**Степень = 1: линейная функция**

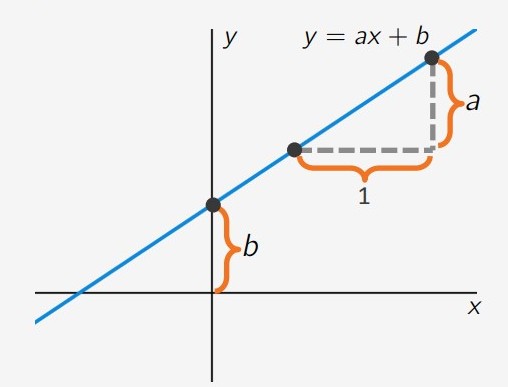
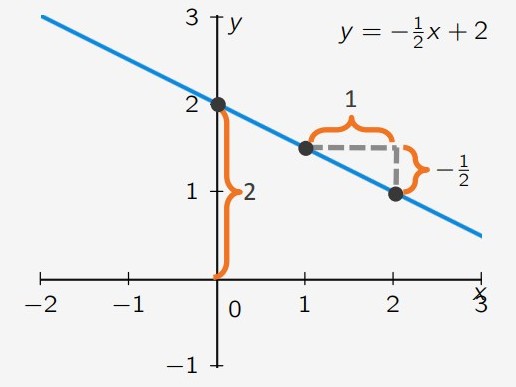
Формула: f(x)=ax+b

График: Прямая линия

a — коэффициент пропорциональности

b — смещение вдоль оси OY

От знака коэффициента будет зависеть направление прямой, от его значения будет зависеть угол наклон прямой относительно оси х.

Пример: f(x) = -1/2x + 2

a = -1/2

b = 2

**Степень = 2: квадратичная функция (многочлен второй степени)**

Формула:

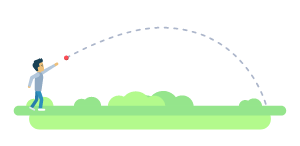


График: парабола

а – старший коэффициент

b — младший коэффициент

c — свободный коэффициент

Пример 1: движения тела, подброшенного вверх

Зависимость высоты от времени будет выражено формулой:

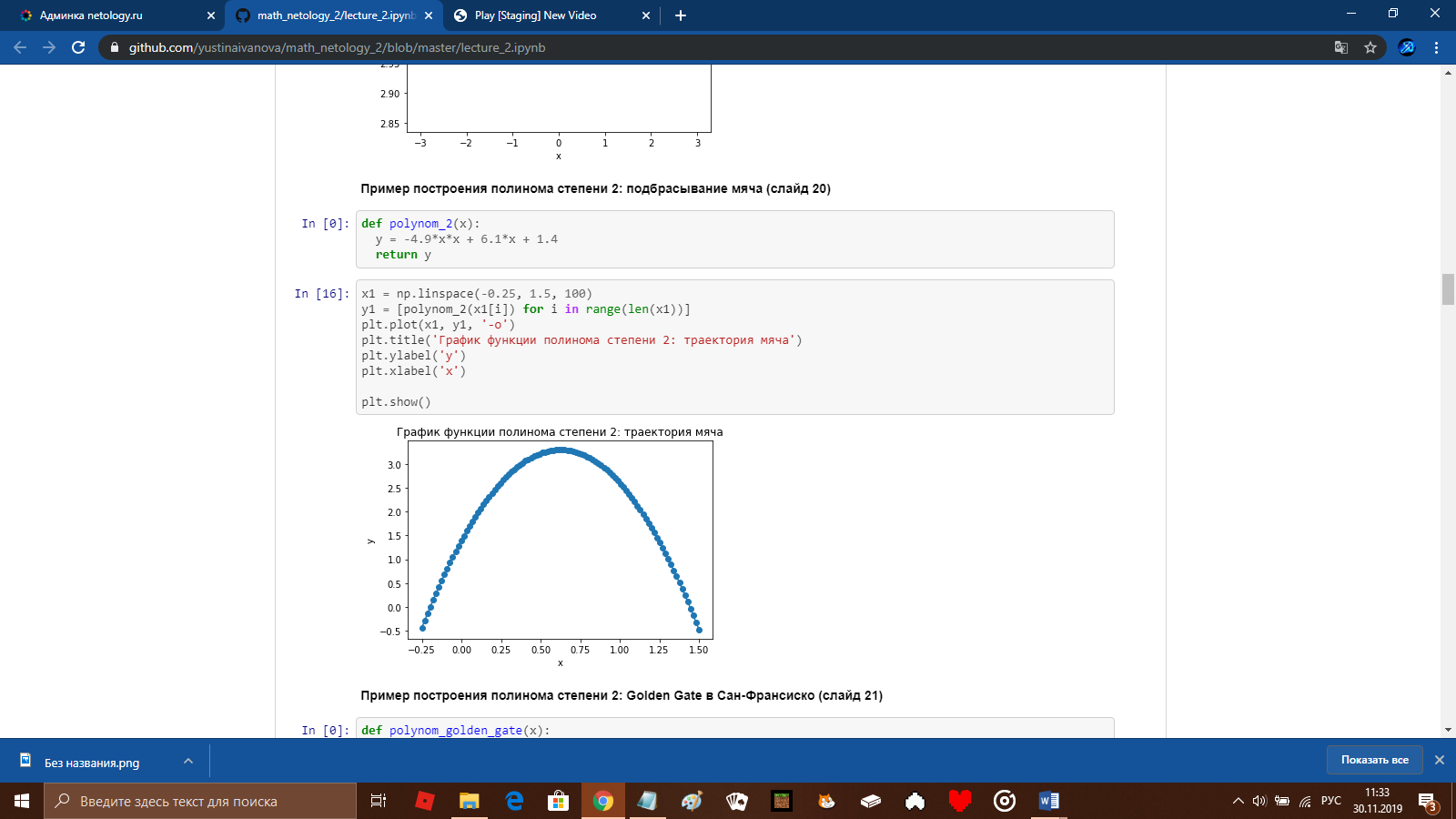


Зависимость траектории движения от удаленности расстояния выражено формулой:

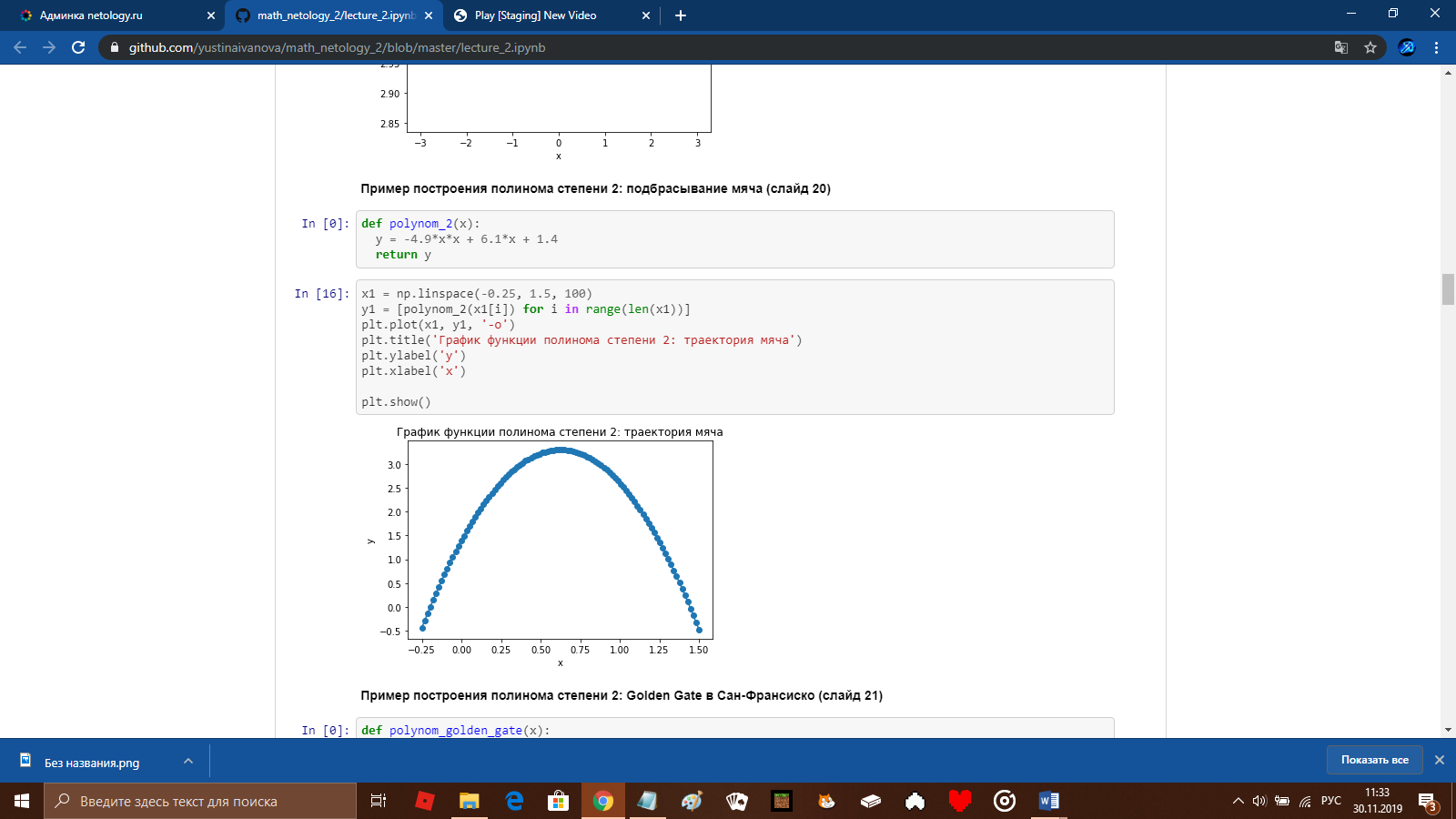


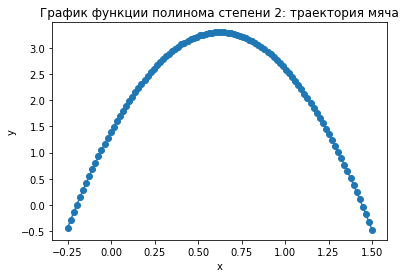
**Пример программного построения полинома степени 2: подбрасывание мяча**

Задаем функцию:



Создаем вектор из 100 значений, вычисляем значение y, строим график





Пример 2: Golden Gate в Сан-Фрасиско.

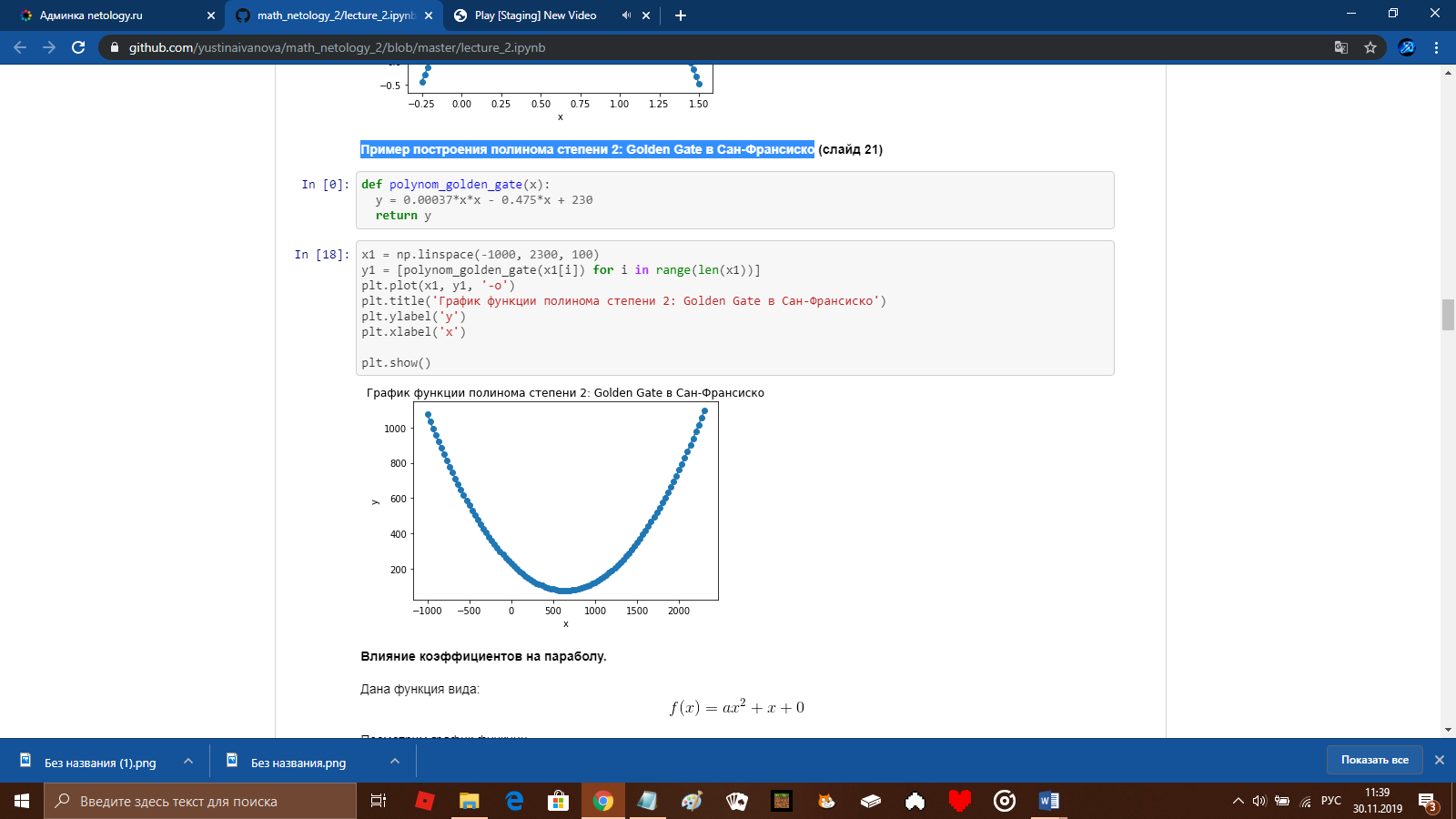


Зависимость высоты над уровнем моря от

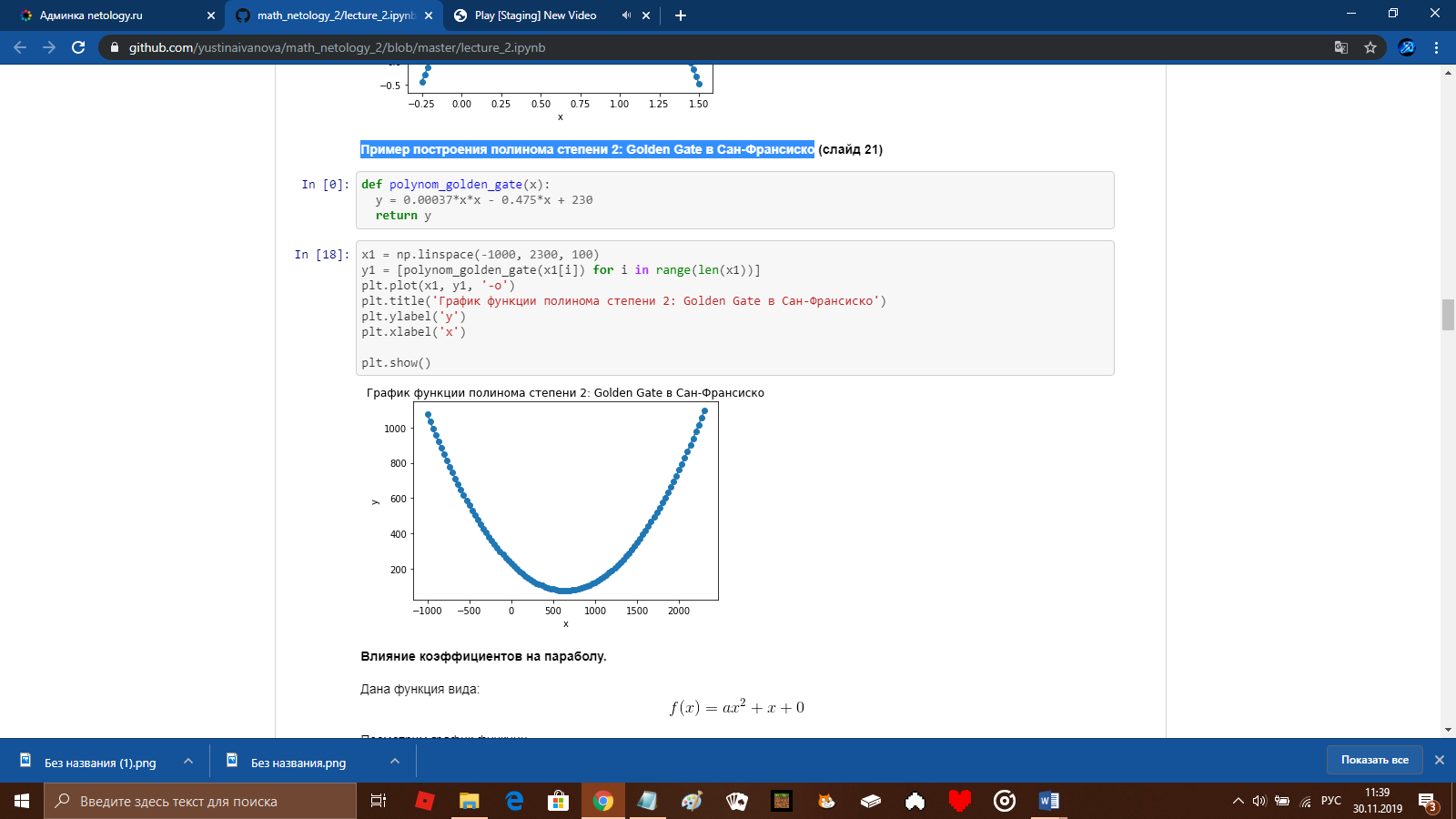
X — в метрах от левого столба

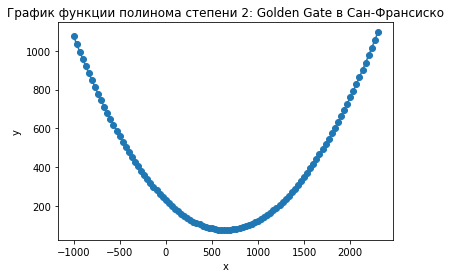
**Пример программного построения полинома степени 2: Golden Gate в Сан-Франсиско**

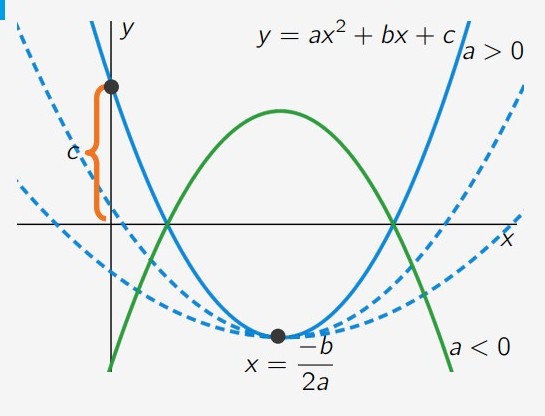
**Задаем функцию:**



Задаем сто точек на интервале от 1000 до 2300, считаем для них значение полинома, строим график





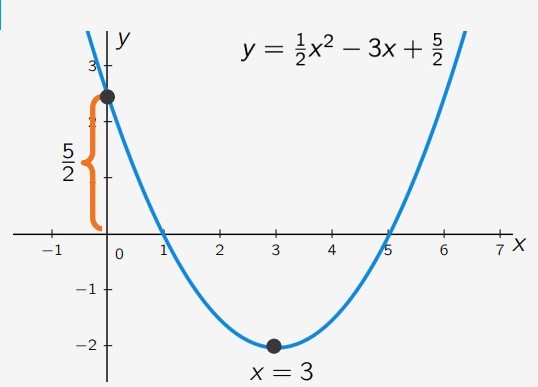
**Влияние коэффициентов на параболу**

Дана функция вида: f(x) = ax2 + bx + c

a – показывает направление ветвей параболы и её растянутость

b — угловой коэффициент в точке пересечения с осью ординат (x=0)

c — координата y при x=0

Пример:

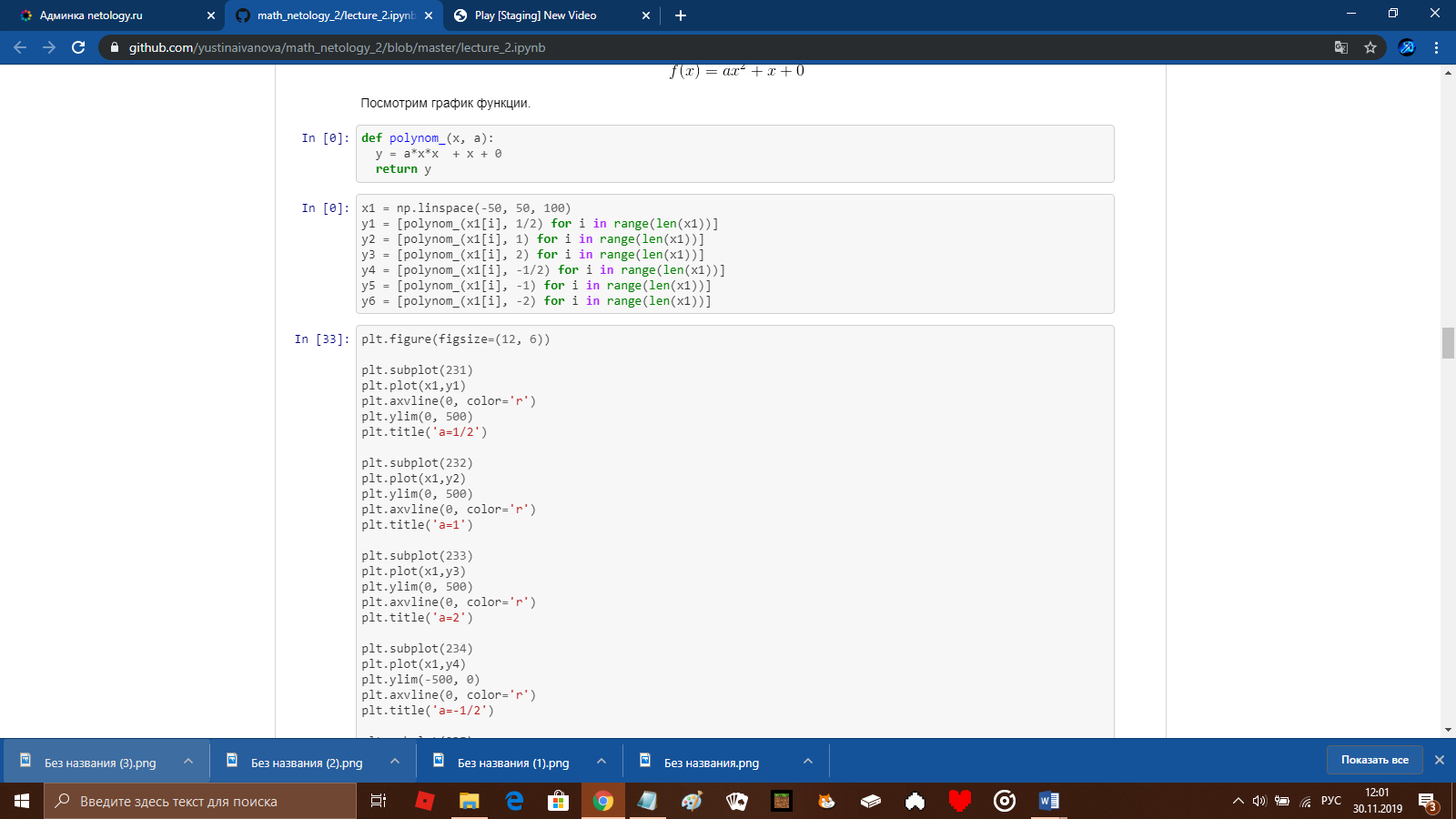
Дана функция 

Координаты вершины: 

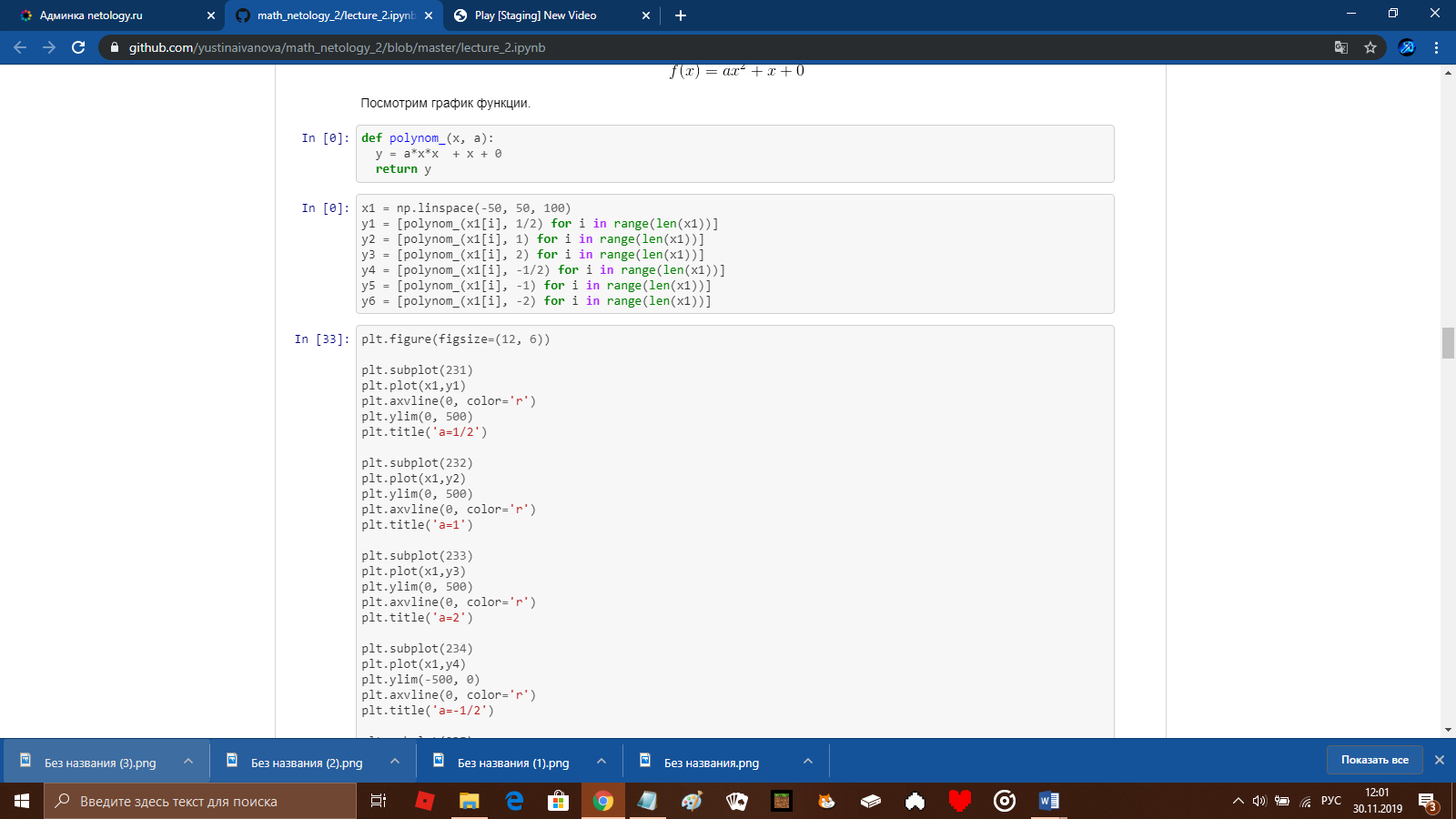
Координата y при x=0: у=5/2

Построим график функции вида: alt text, где коэффициент b = 1, с = 0, посмотрим зависимость функции от коэффициента а.

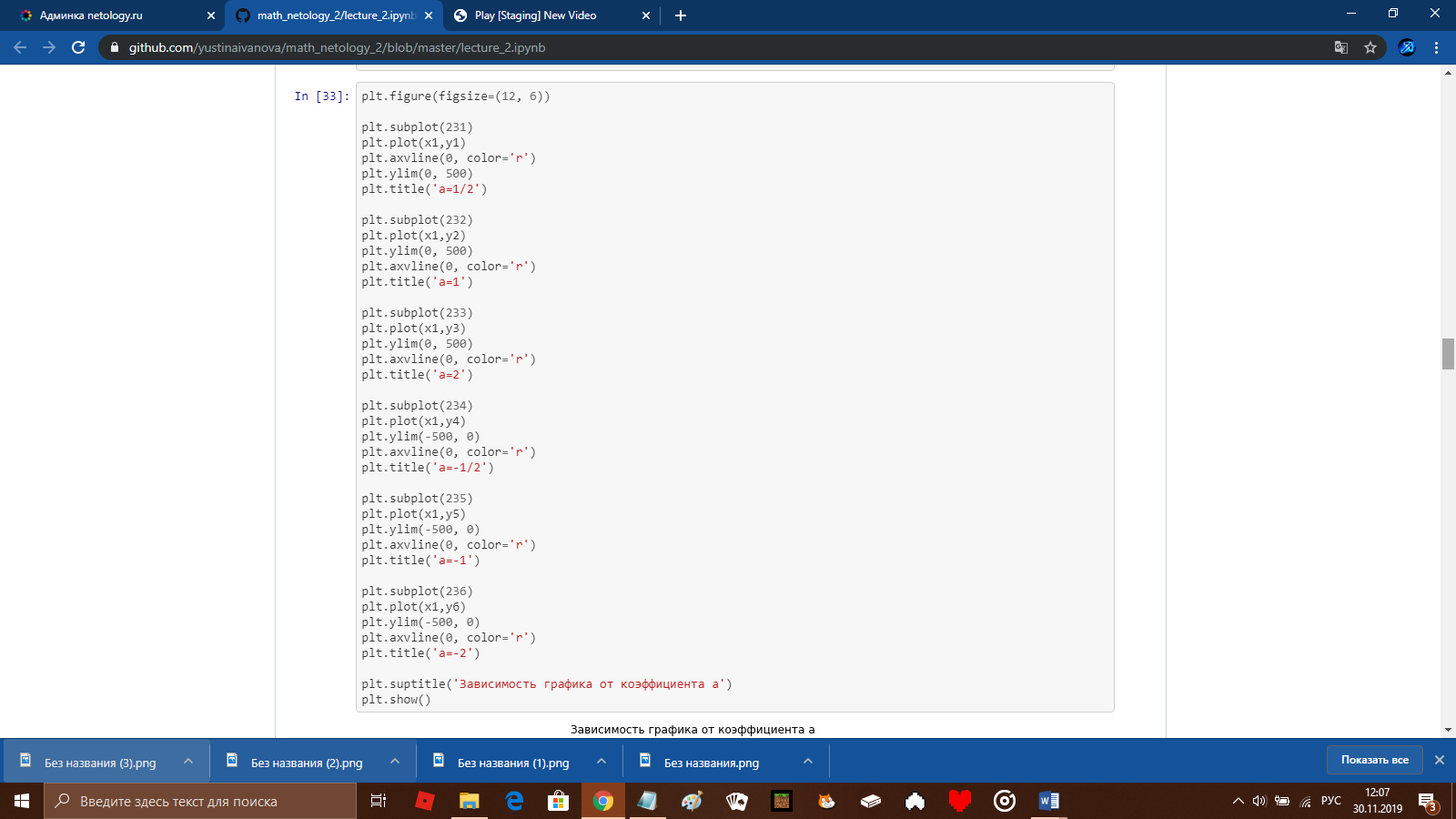
Задаем функцию:

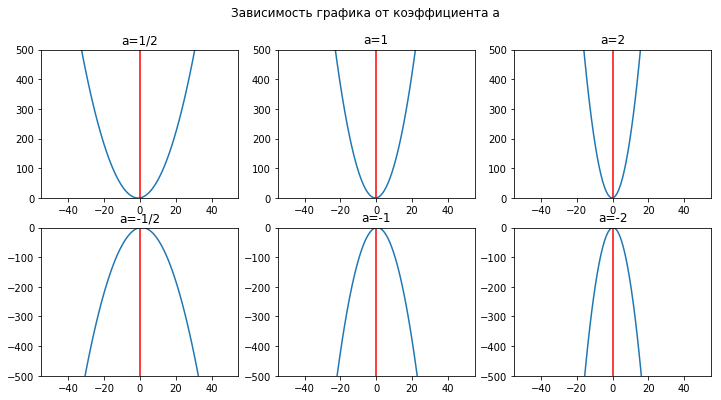


Задаем сто точек и для них 6 полиномов с разными коэффициентами а



Задаем построение графиков (2 ряда, 3 столбца и указываем по очереди какой график чертим: (plt.subplot(231), plt.subplot(232),plt.subplot(233), plt.subplot(234), plt.subplot(235), plt.subplot(2316)))   
для шести разных коэффициентов

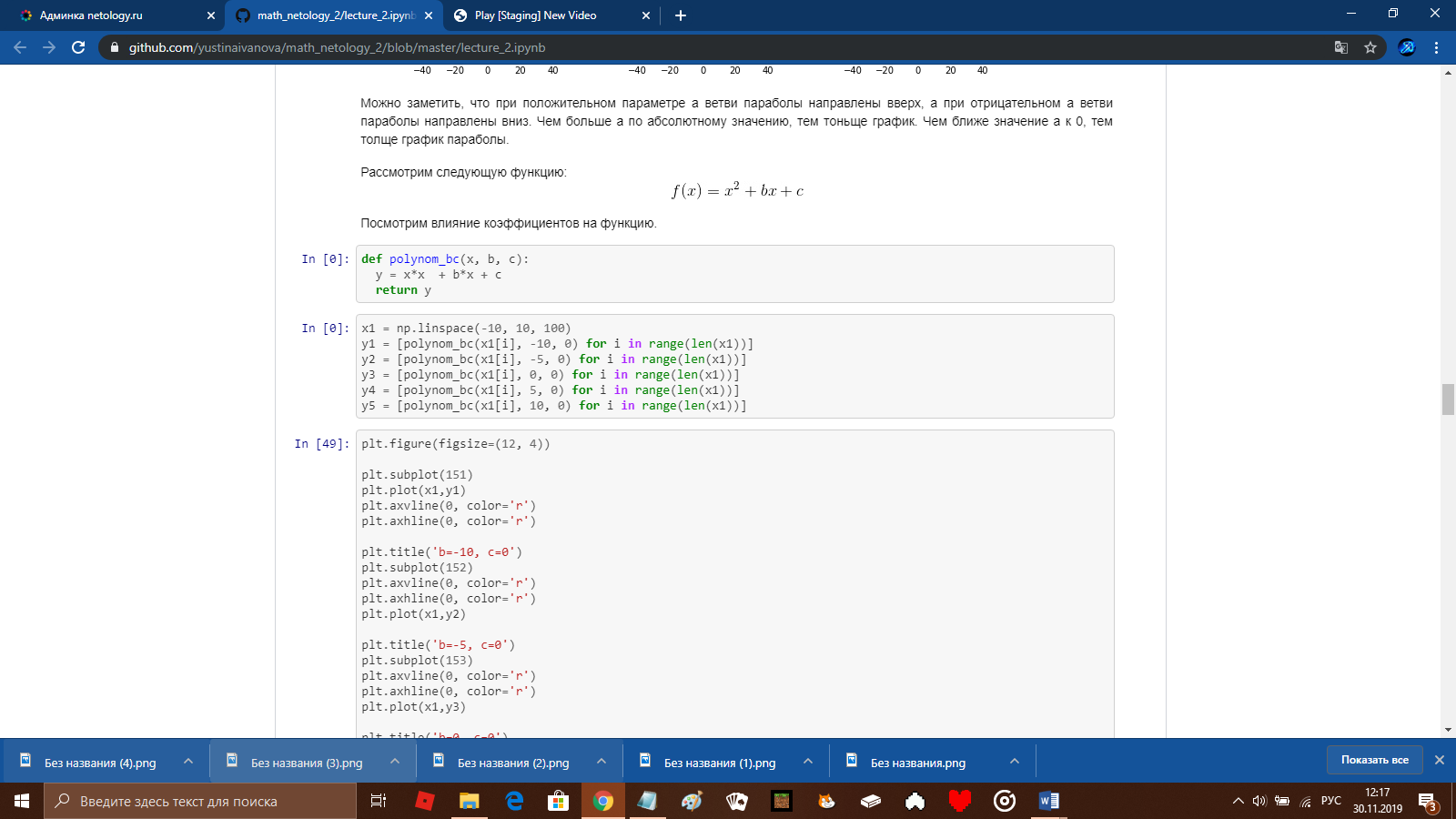




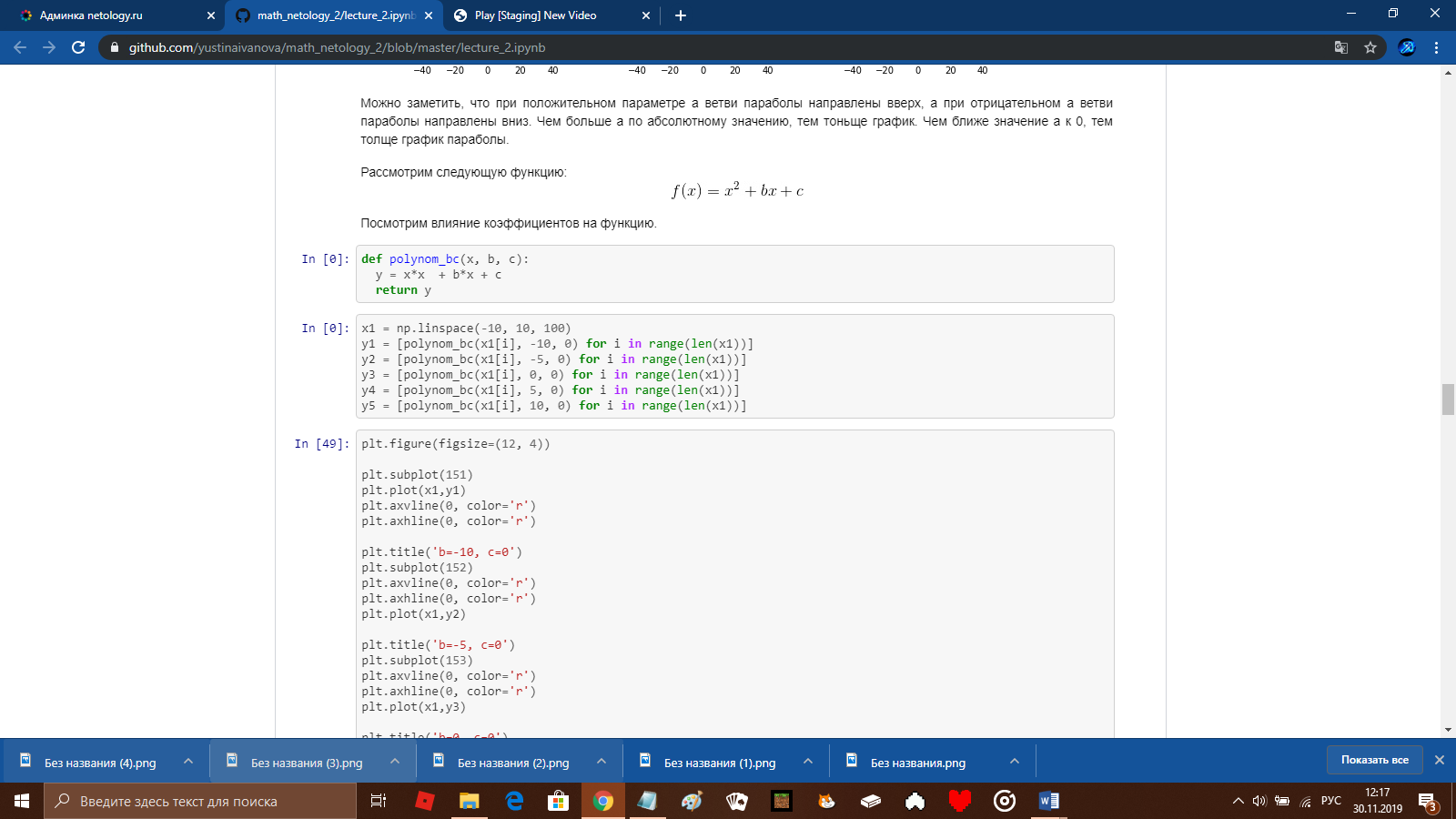
Можно заметить, что при положительном параметре a ветви параболы направлены вверх, а при отрицательном a ветви параболы направлены вниз. Чем больше a по абсолютному значению, тем тоньше график. Чем ближе значение a к 0, тем толще график параболы.

Рассмотрим влияние коэффициента b на вид функции, для этого возьмем функцию: alt text, где коэффициент а=1.

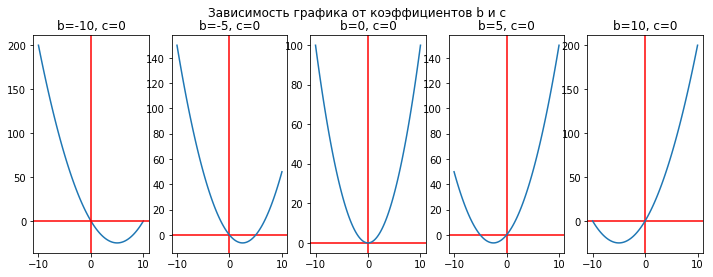
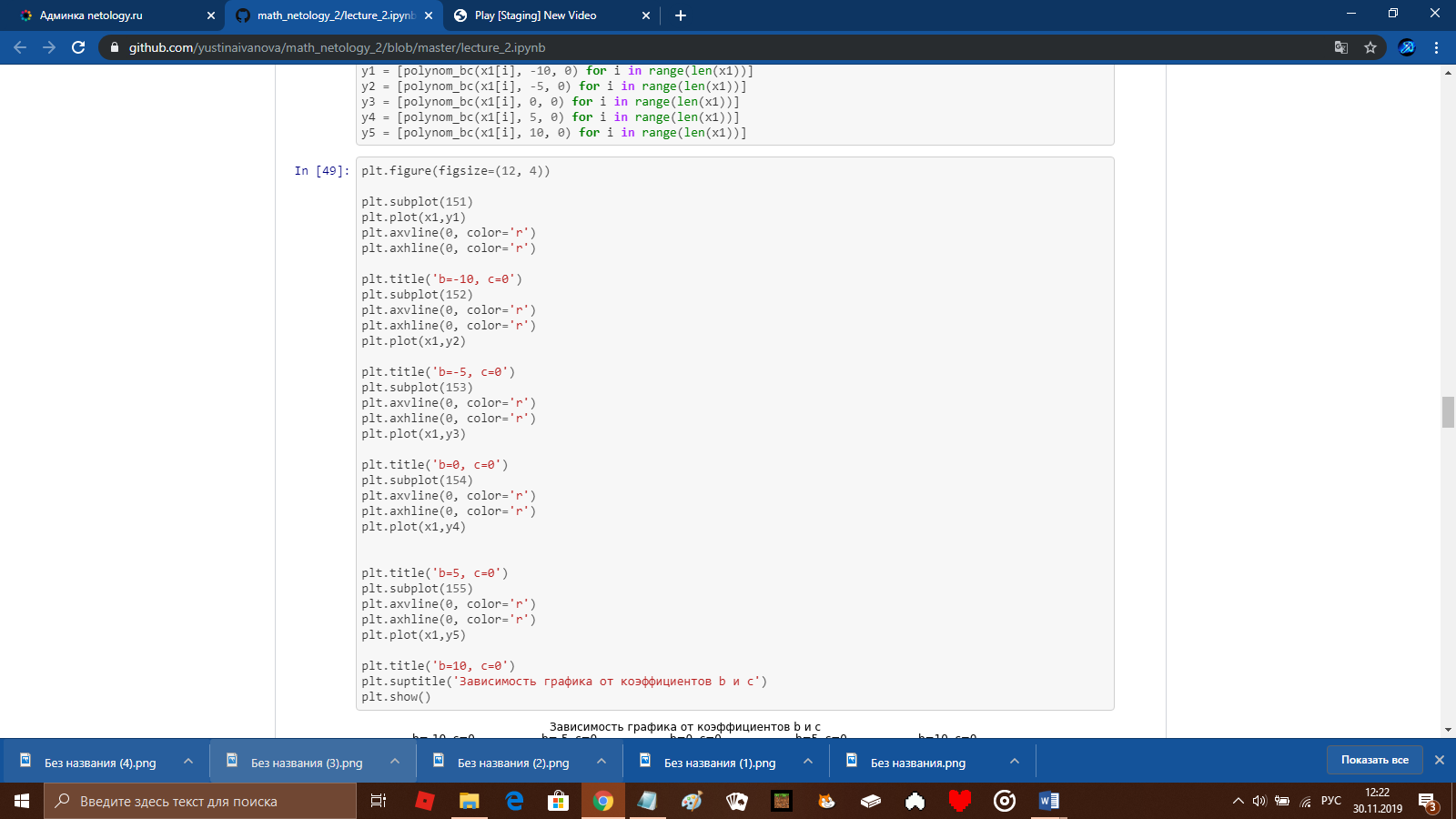
Задаем функцию:



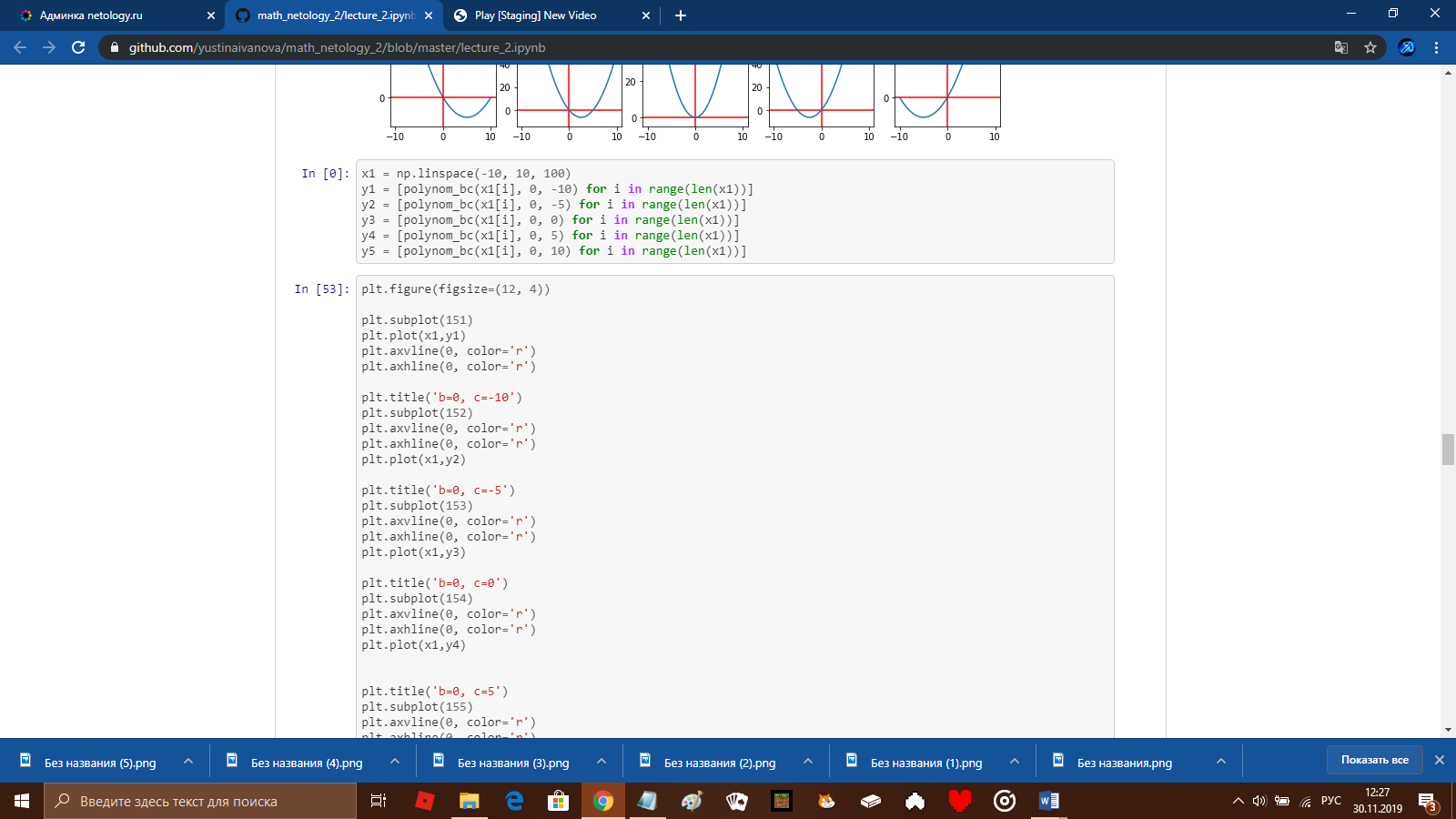
Вычислим значения для 5 функций с различными коэффициентами b и с=0 для ста значений х



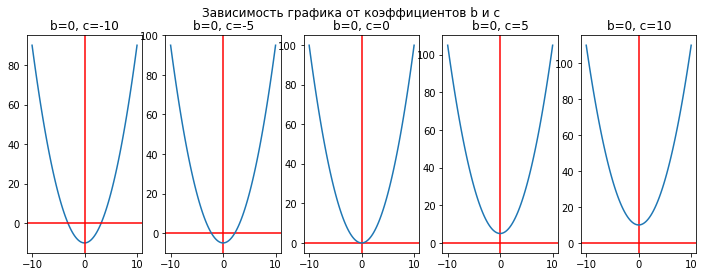
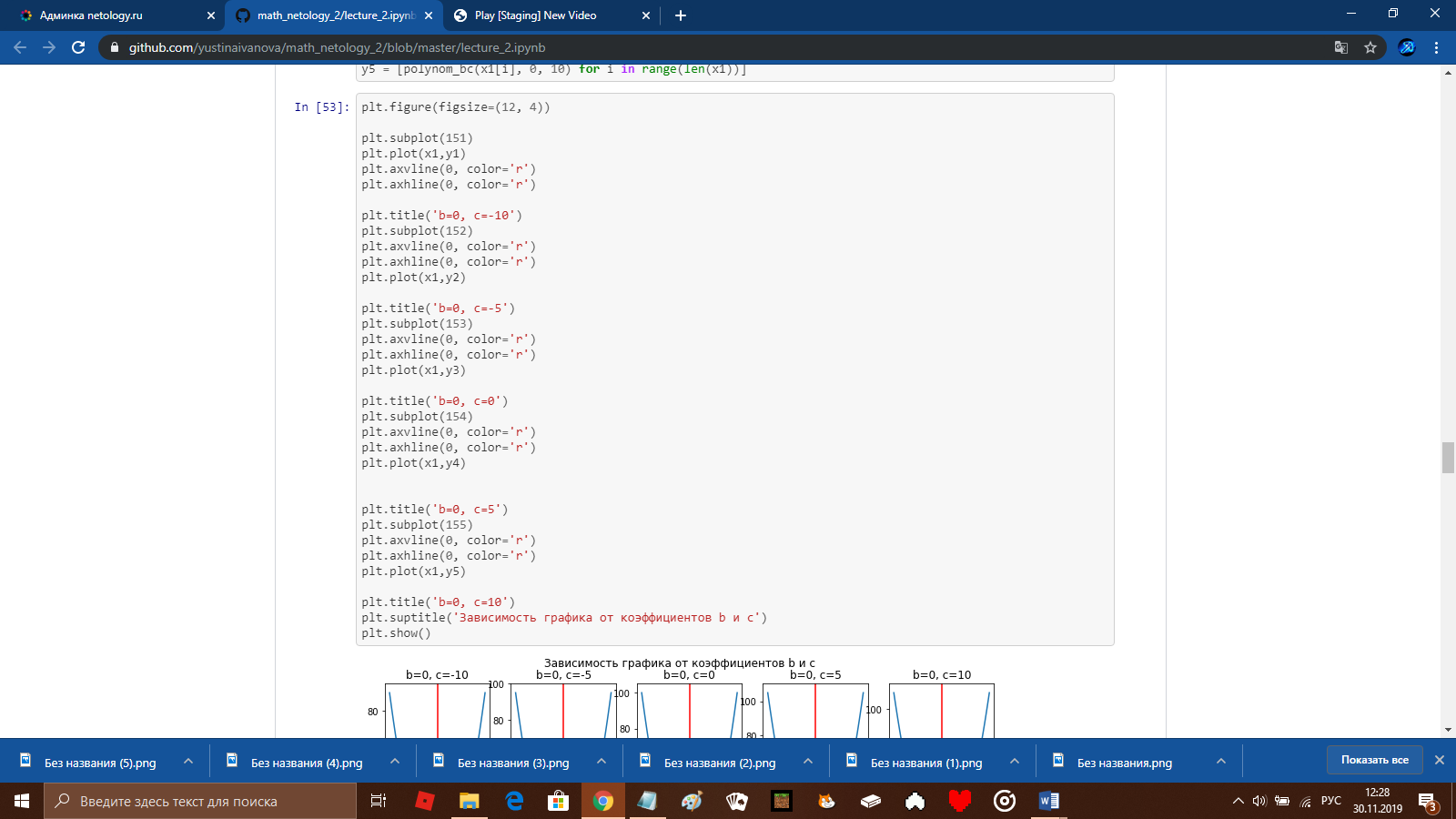
Построим 5 графиков в ряд



Вычислим значения для 5 функций с различными коэффициентами с и b=0 для ста значений х



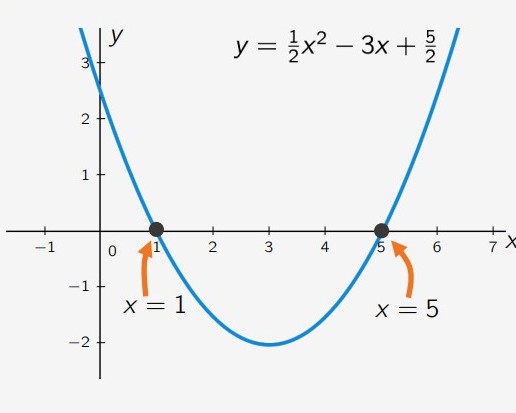
Построим 5 графиков в ряд



Можно заметить, что параметр c сдвигает вершину параболы по оси Oy на число, заданное c. Параметр b сдвигает вершину параболы по оси Ox.

**Квадратичная функция: факторизованное представление**

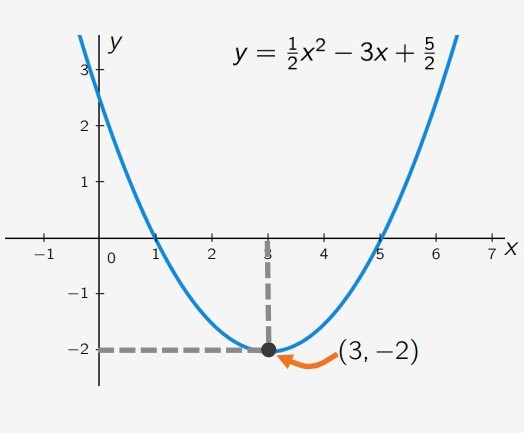
 Квадратичный [полином](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC) может быть представлен в виде:  , где p и q точки пересечения параболы с осью х.

Пример:



**Квадратичная функция: альтернативное представление**

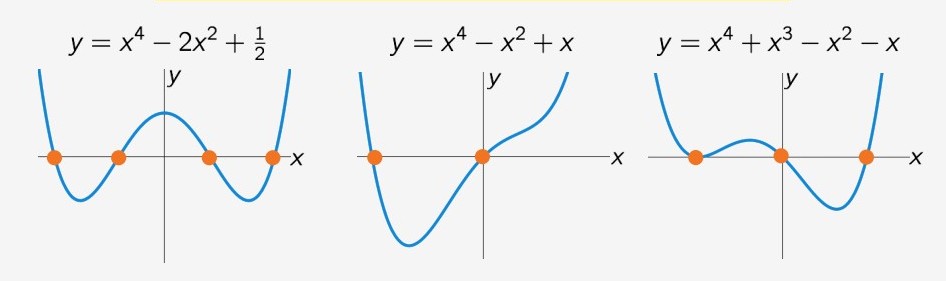
 при данном представлении квадратичной функции, вершина параболы будет иметь координаты (r, s).

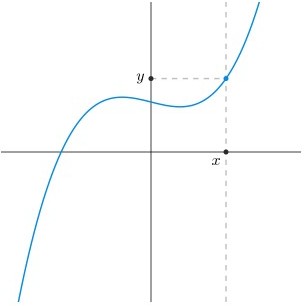
Пример:

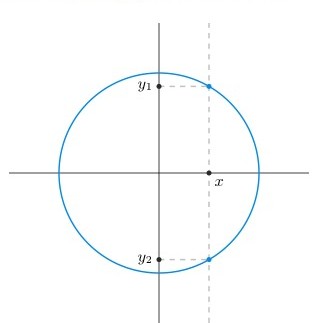


**Полиномы со степенью выше 2**

**!!! График полинома степени n пересекает ось OX не более чем n раз, где n — степень полинома**



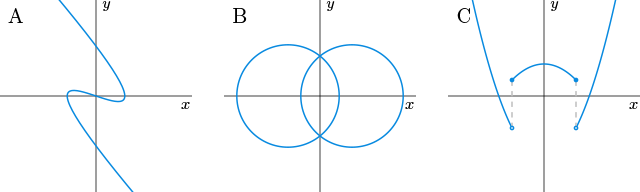
На данном графике для каждого аргумента функции существует только одно значение, поэтому можно сделать вывод, что данный график – это графическое представление полинома



На данном графике для каждого аргумента функции существует как минимум два значения - это графическое представление суммы полиномов

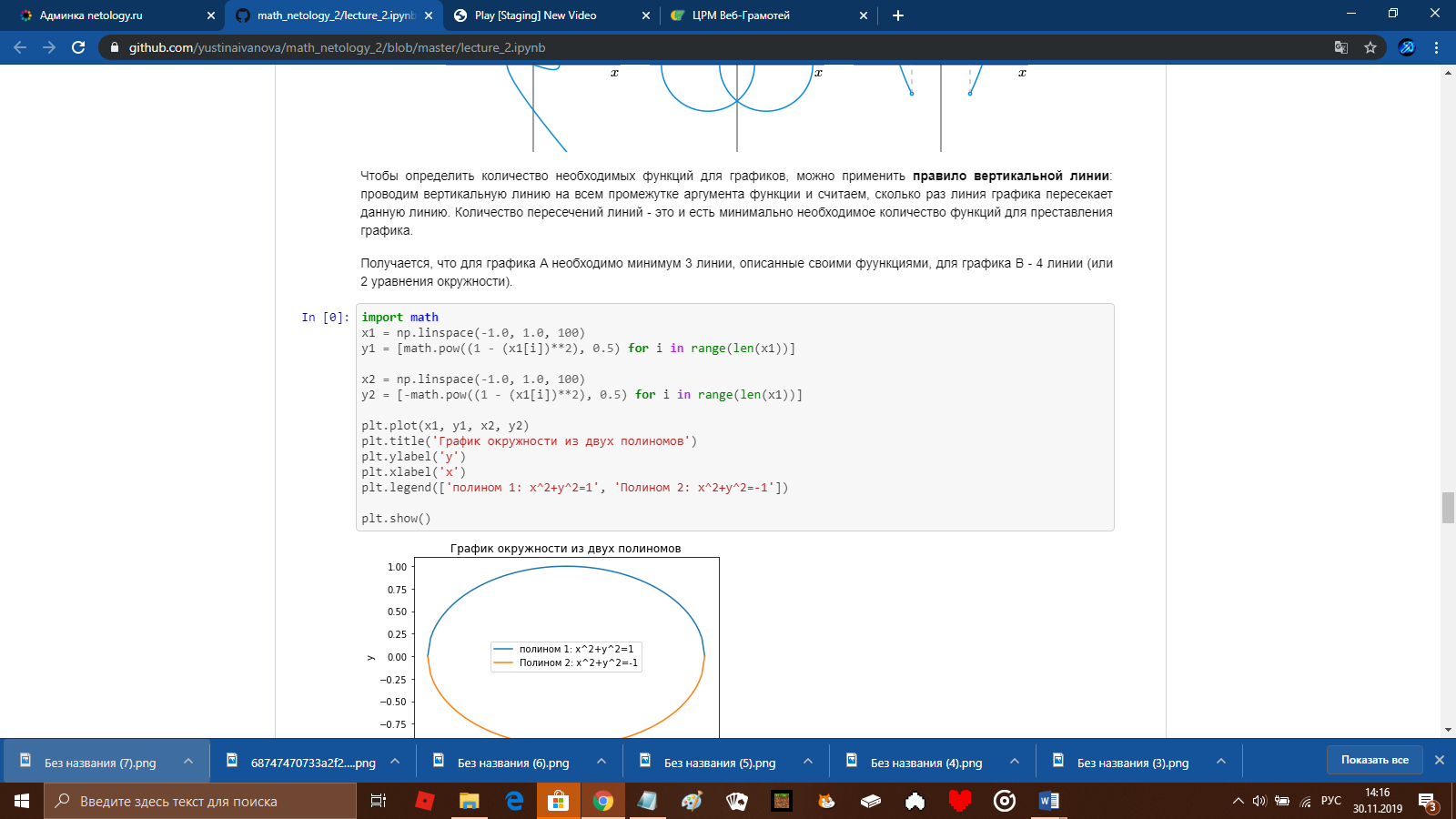
Любую функцию можно представить в виде суммы нескольких полиномов.

Допустим, имеются следующие графики:

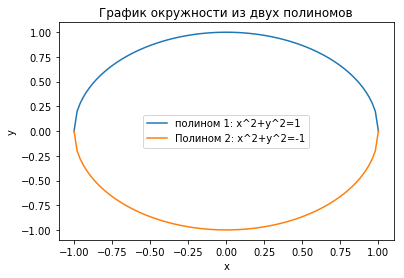


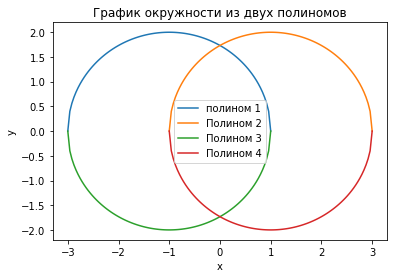
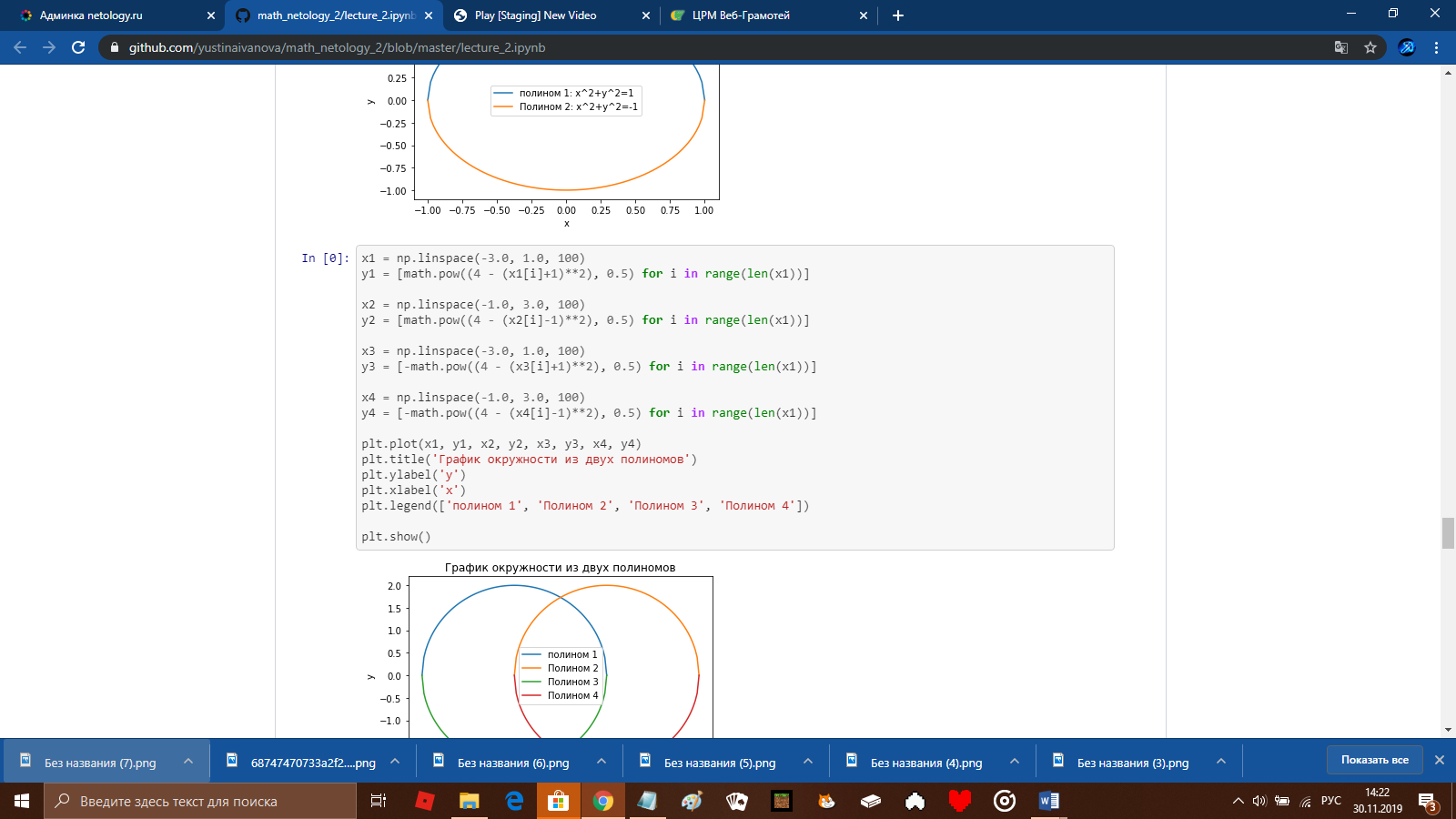
Чтобы определить количество необходимых функций для графиков, можно применить правило вертикальной линии: проводим вертикальную линию на всем промежутке аргумента функции и считаем, сколько раз линия графика пересекает данную линию. Количество пересечений линий - это и есть минимально необходимое количество функций для преставления графика.

Получается, что для графика A необходимо минимум 3 линии, описанные своими функциями, для графика B - 4 линии (или 2 уравнения окружности).

Пример построения окружности 

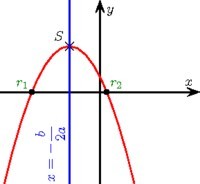
В данном примере для простоты используется формула окружности вида x2+y2 = R2, где R – радиус окружности (по факту необходимо представить окружность виде суммы двух полиномов второй степени ax2+bx+c и –ax2+bx+c и задать для х интервал)





Для построения моделей в data science часто используются полиномы второй степени.

**Вычисление вершины через дискриминант**

Для функции  дискриминант находится по формуле: .

Если D=0, то один корень уравнения, при D < 0 – корней нет (то есть нет пересечения с осью х)

Корни уравнения находятся по формулам:

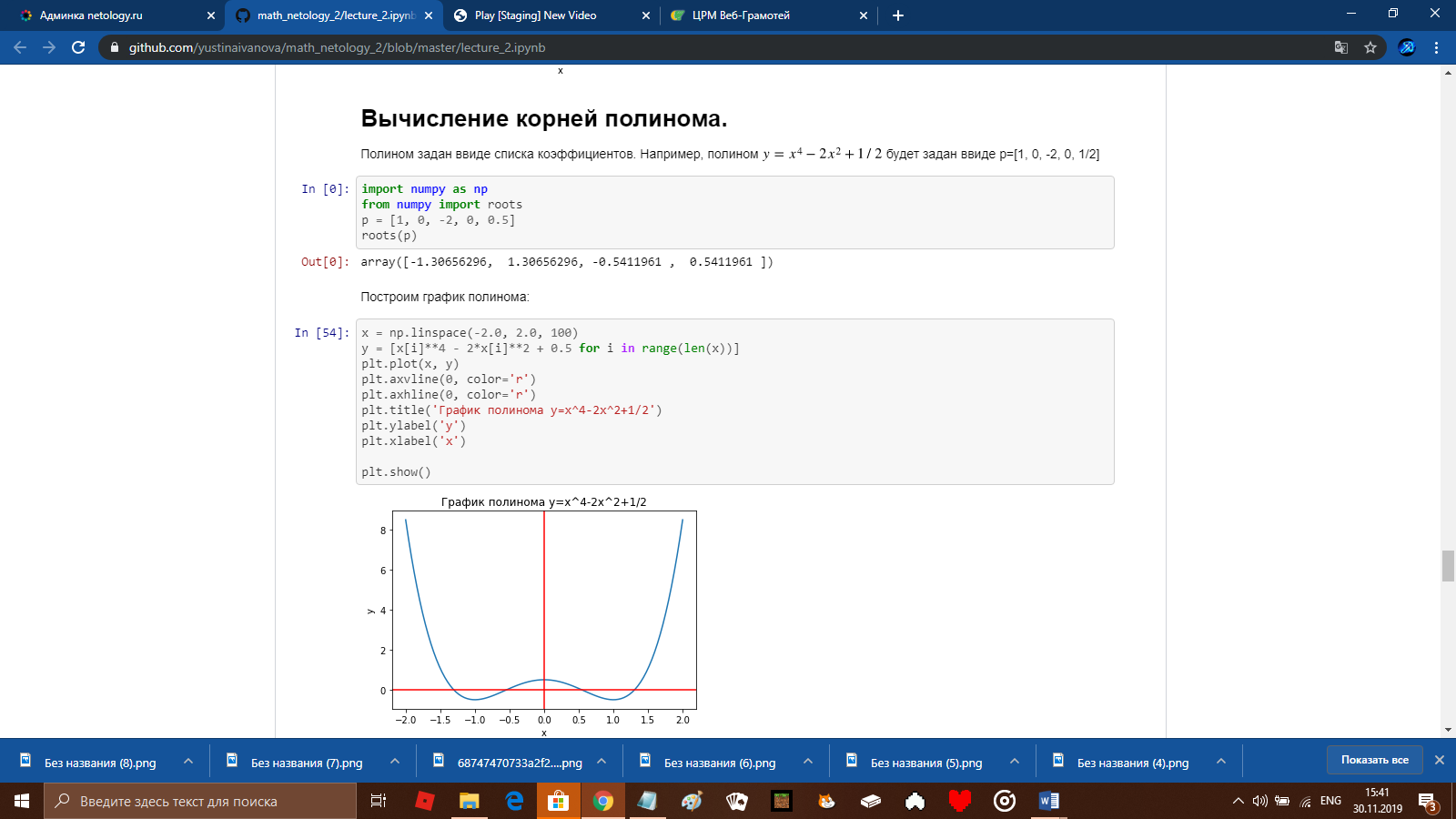
Соответственно координату х вершины параболы можно найти как: **** или 

Координата y вычисляется подстановкой x в уравнение.

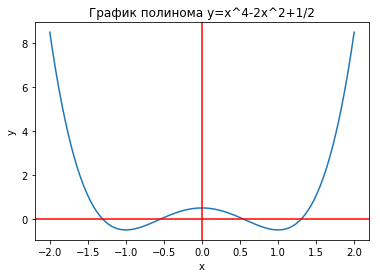
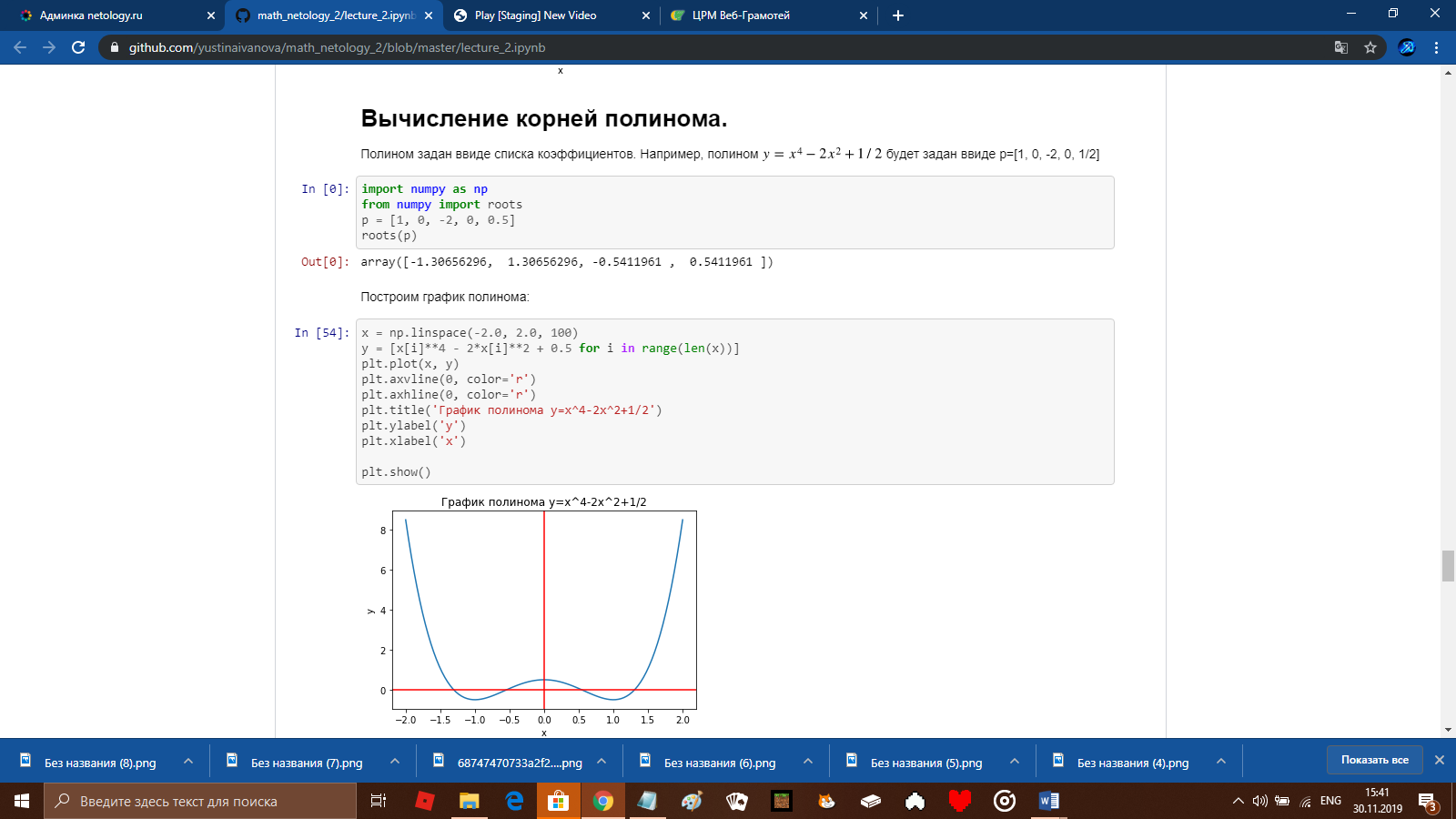
**Вершину можно вычислить програмно с помощью** numpy.roots(p), где p — коэффициенты полинома в виде списка

Пример:

Дан полином четвертой степени вида: y=x4-2x2+1/2 коэффициенты будут заданы в виде списка   
p=[1, 0, -2, 0, 1/2]



Строим график



Можно заметить, что корни, найденные в предыдущем примере, действительно совпадают с тем, что можно увидеть на графике.

**Вычисление вершины с помощью производной**

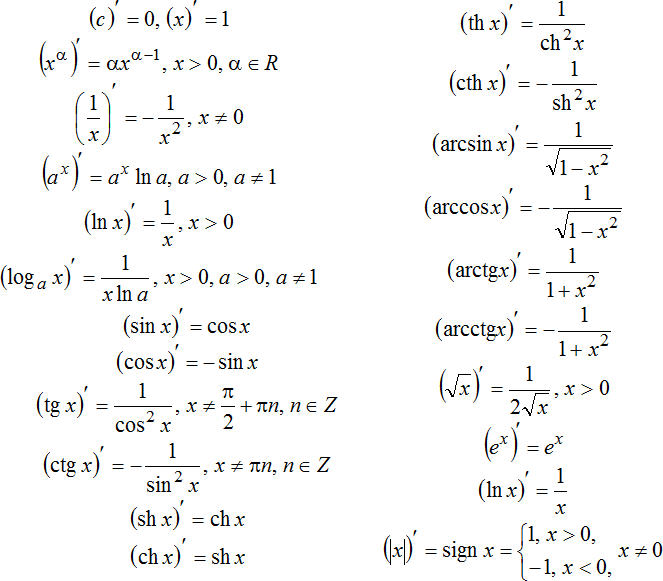
Для функции  производная находится по формуле:**.**

Производная в точке вершины равна 0: 

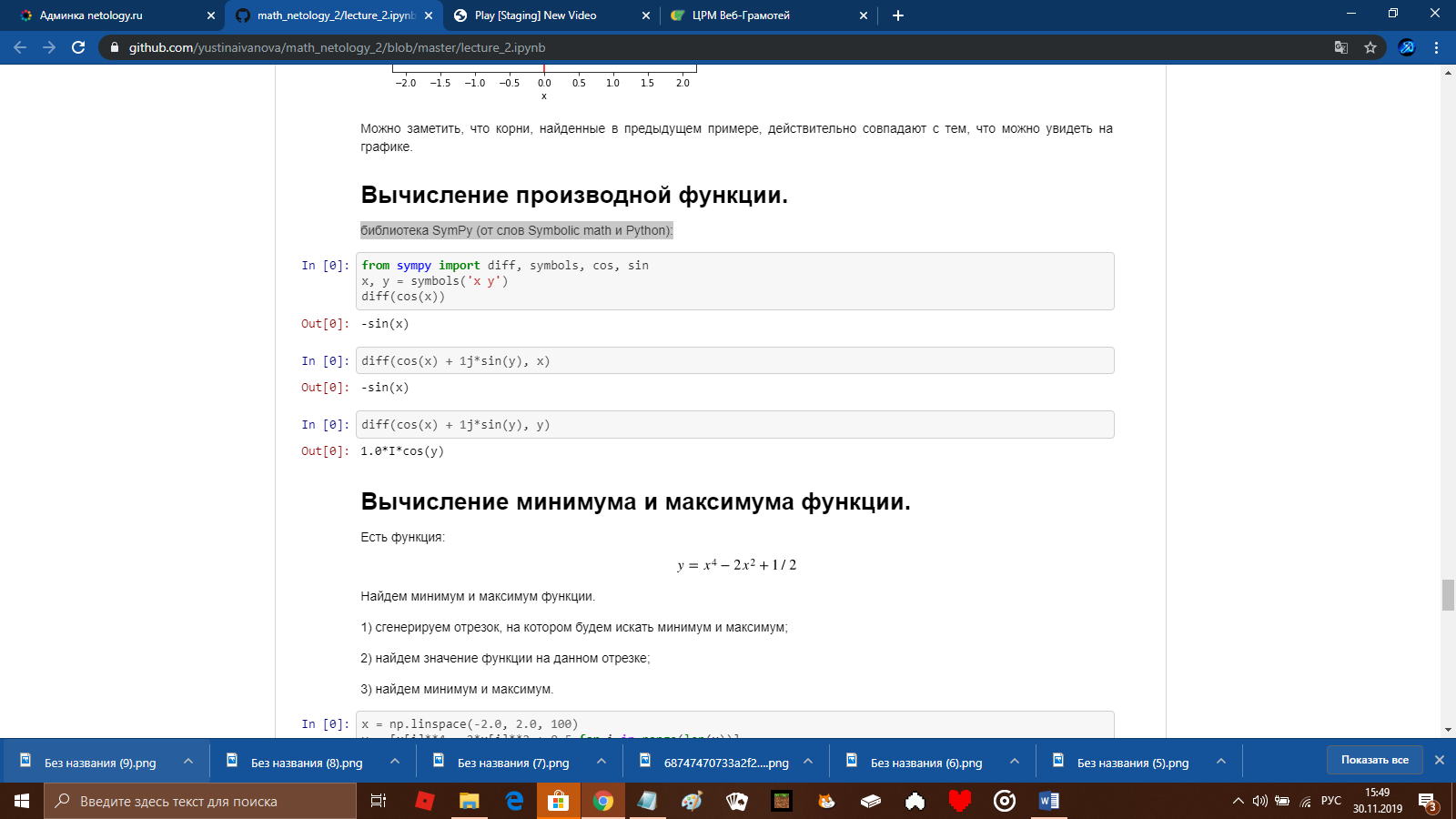
Координата x вершины параболы: 

Координата y вершины параболы вычисляется методом подстановки.

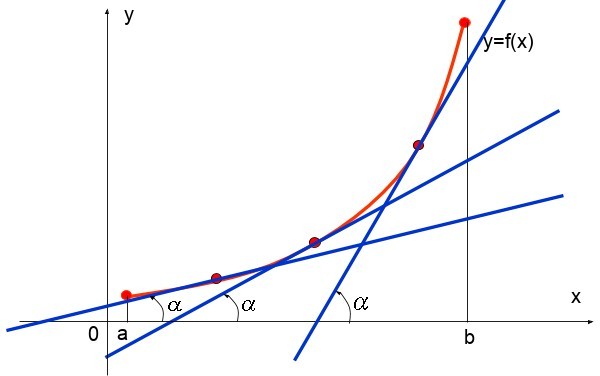
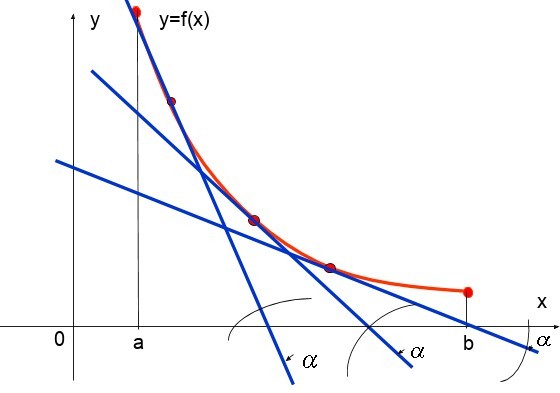
**Формулы вычисления производных различных функций**



**Программно вычислить производную можно с помощью библиотеки SymPy (от слов Symbolic math и Python):**



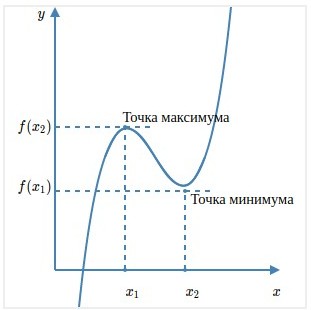
**Исследование функций с помощью производных**



Если f`(x)>0 на интервале, то функция возрастает на данном промежутке.

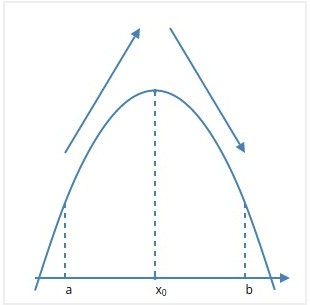
Если f`(x)<0 на интервале, то функция убывает на данном промежутке.

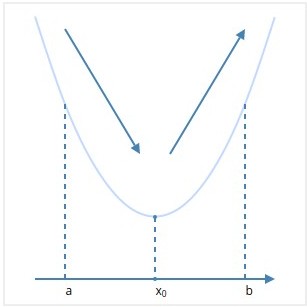
**Точки экстремума**

Точка экстремума — это точка максимума (минимума) минимума функции.

х0 является точкой минимума функции, если для всех f(x), достаточно близких точек верно неравенство 

х0 является точкой максимума функции, если для всех f(x), достаточно близких точек верно неравенство 

Если функция f(x) непрерывна на промежутке (a, b), возрастает на промежутке (a, x0) и убывает на промежутке (x0, b), то x0 является точкой максимума функции.



Если функция f(x) непрерывна на промежутке (a, b), убывает на промежутке (a, x0) и возрастает на промежутке (x0, b), то x0 является точкой минимума функции.

Пример программного способа нахождения максимума и минимума функции

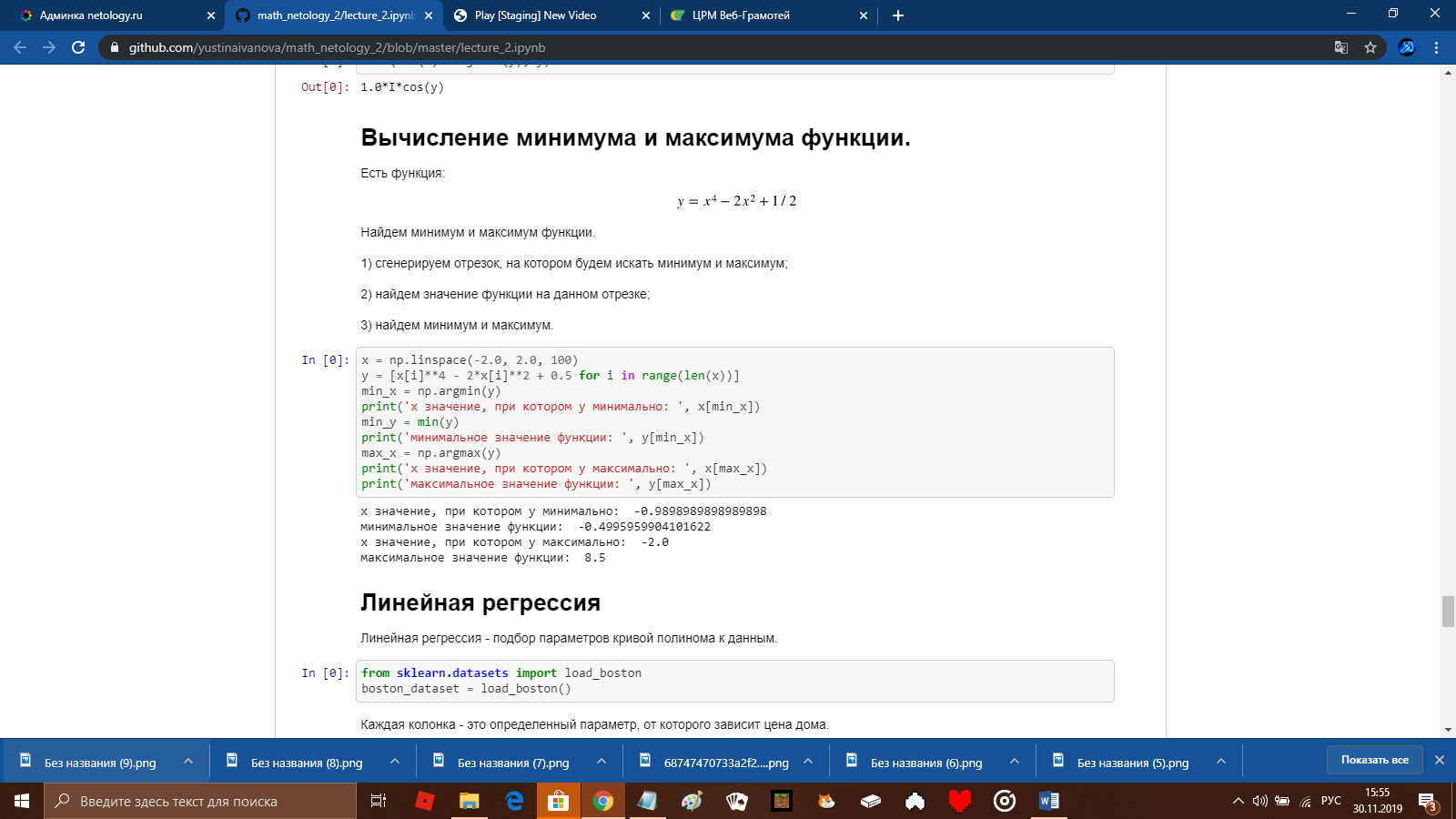
Дана функция: y=x4-2x2+1/2

Найдем минимум и максимум функции.

1) сгенерируем отрезок, на котором будем искать минимум и максимум;

2) найдем значение функции на данном отрезке;

3) найдем минимум и максимум.

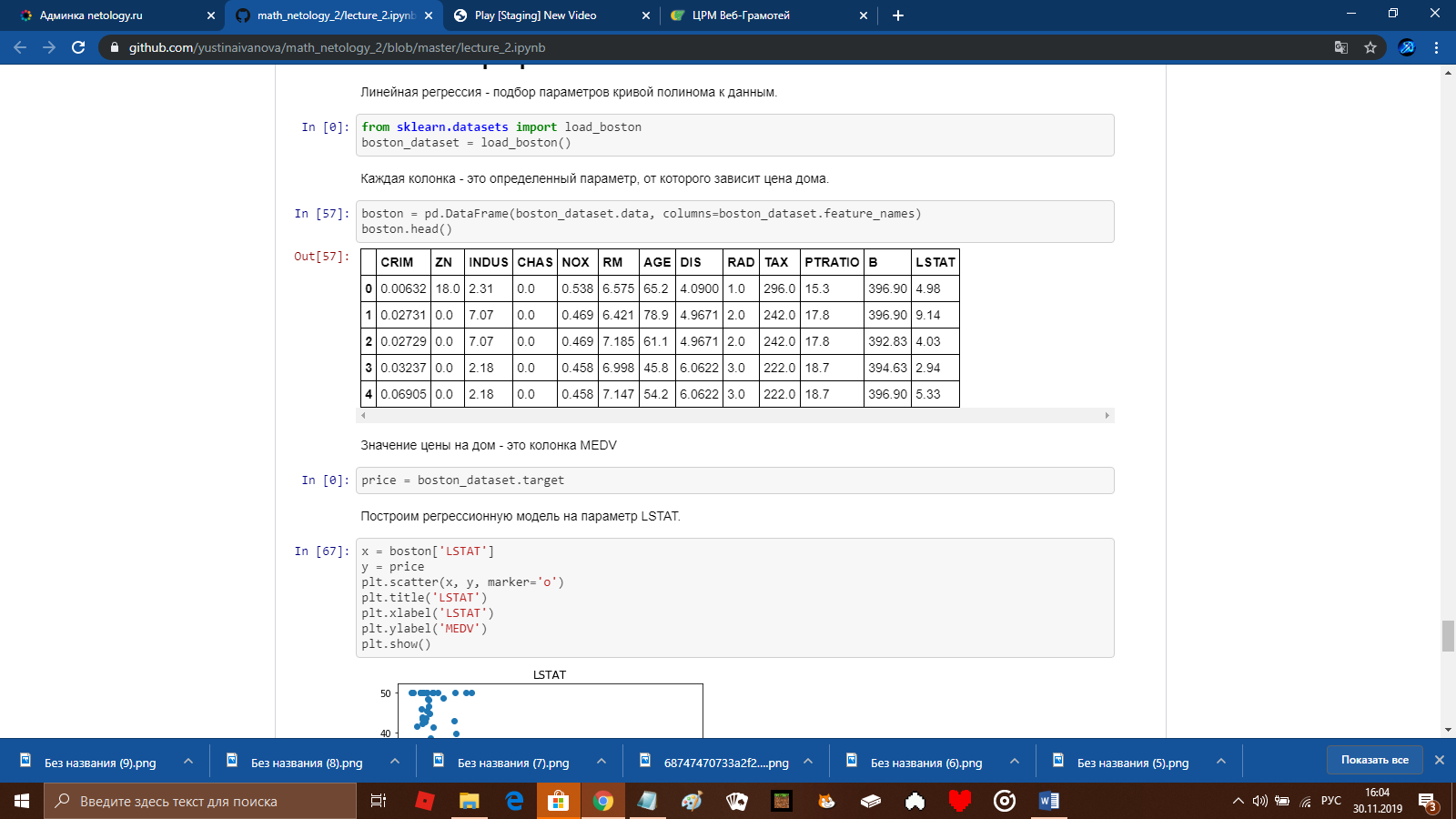


min\_x = np.argmin(y) – ищет номер минимального элемента

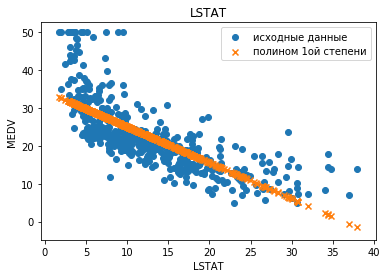
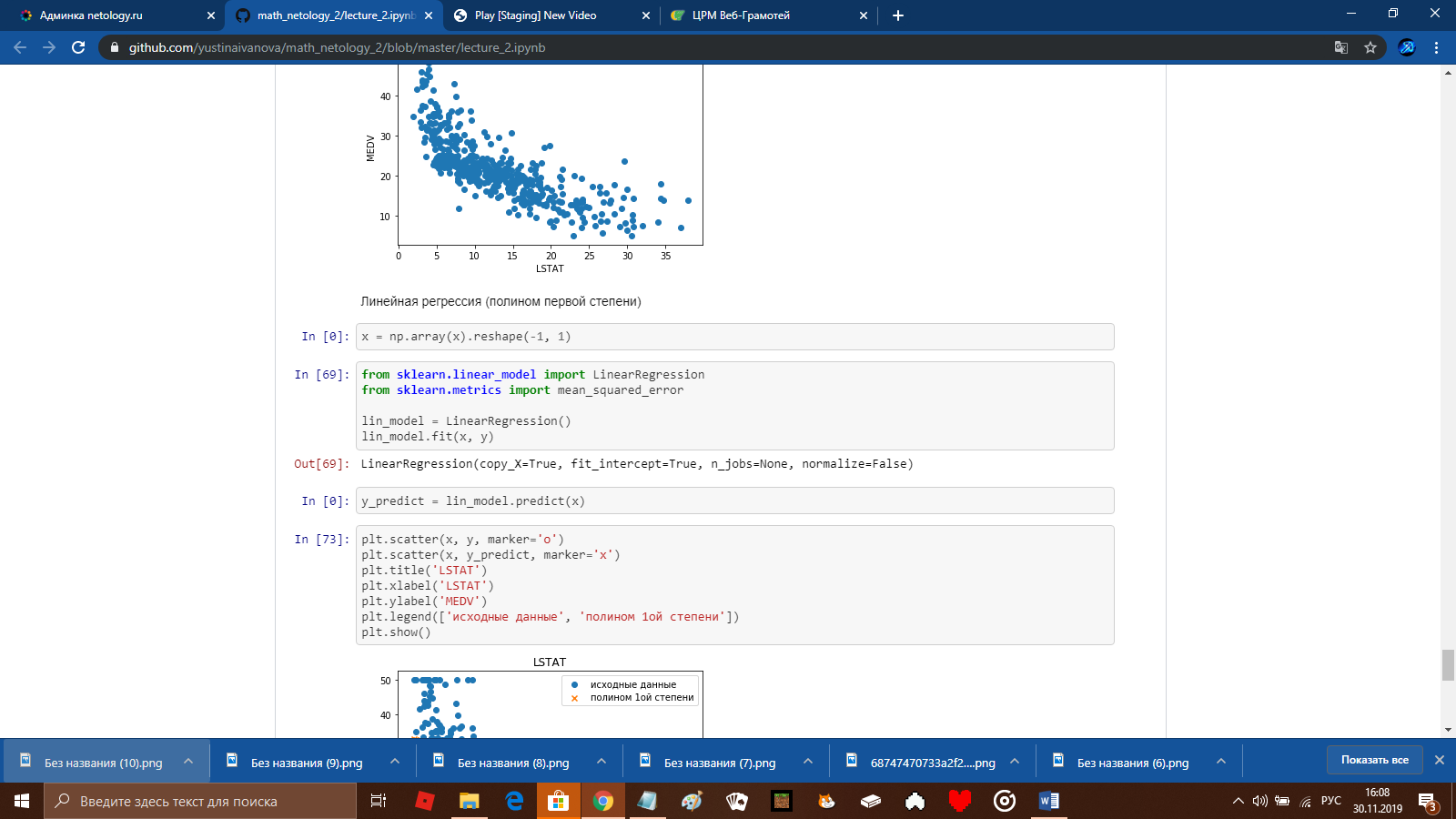
x[min\_x] – значение минимального элемента

**Линейная регрессия**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=98e40618beef97ffdcbdaa5518fcffd8b1472210&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f79757374696e616976616e6f76612f6d6174685f6e65746f6c6f67795f322f393865343036313862656566393766666463626461613535313866636666643862313437323231302f6c6563747572655f322e6970796e62&nwo=yustinaivanova%2Fmath_netology_2&path=lecture_2.ipynb&repository_id=218629146&repository_type=Repository#%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F-%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F)

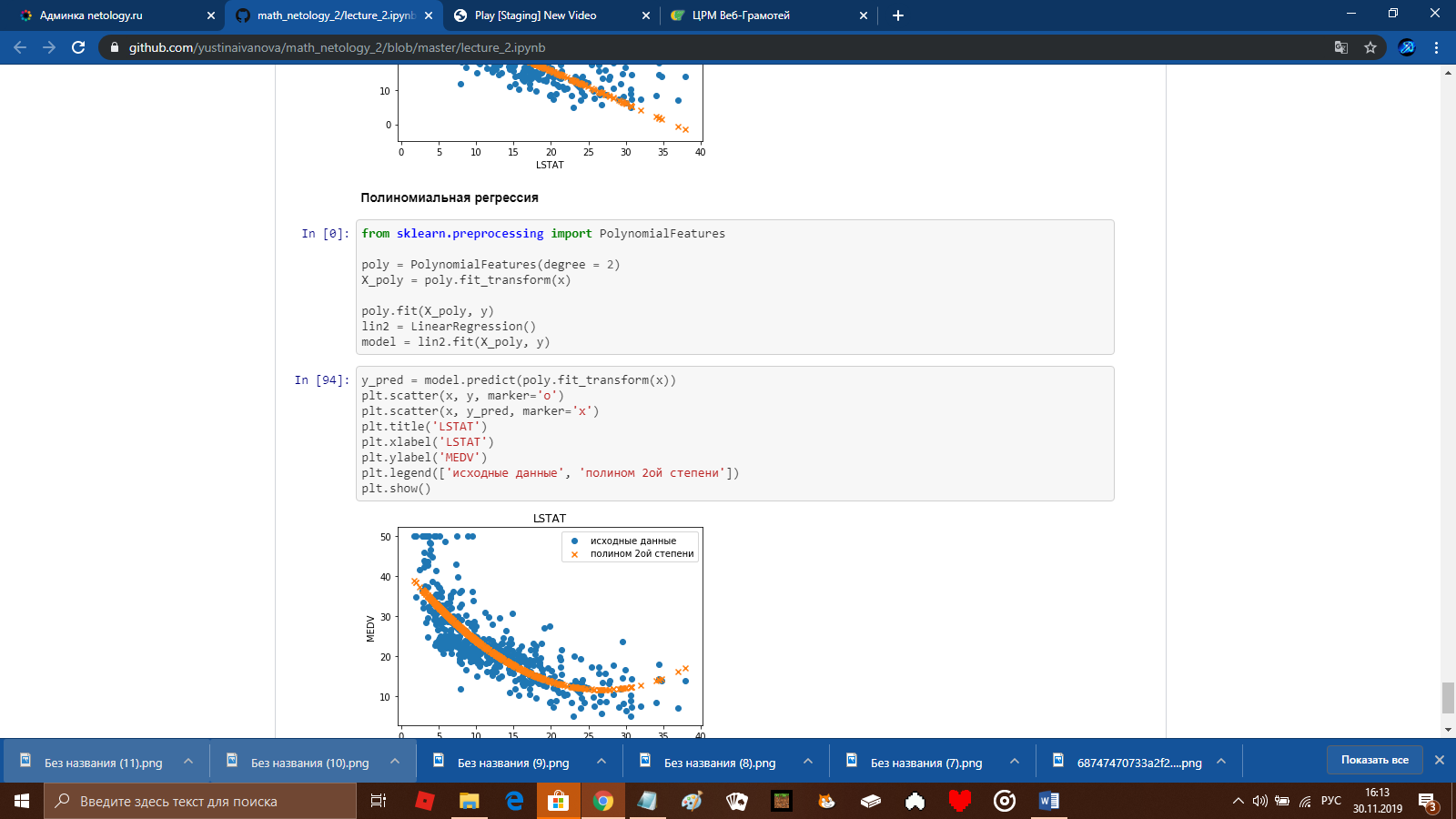
- метод восстановления зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости. Данный метод позволяет предсказывать значения зависимой переменной y по значениям независимой переменной x.

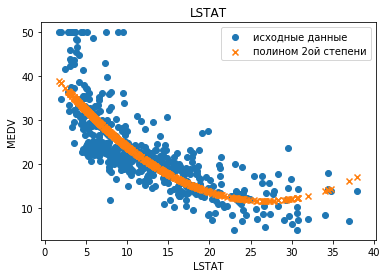
Необходимо подобрать такие параметры кривой полинома, которые описывали бы данные с наименьшей погрешностью. 





Можно заметить, что данная модель имеет большую погрешность.





Меняя значение degree (увеличивая степень полинома), можно подобрать наиболее подходящую модель, описывающую данные.