Оглавление

[**Часть 1. Обзор возможных путей реализации проекта** 2](#_Toc188115807)

[**Часть 2. Предварительная подготовка библиотек** 3](#_Toc188115808)

[**Часть 3. Написание программного кода** 4](#_Toc188115809)

[*3.1 Код, реализованный с помощью встроенных функций Python* 4](#_Toc188115810)

[*3.2 Код, реализованный с помощью библиотеки NumPy* 4](#_Toc188115811)

[*3.3 Код, реализованный с помощью библиотеки Matplotlib* 5](#_Toc188115812)

[**Приложение**  7](#_Toc188115813)

[**Список литературы** 8](#_Toc188115814)

# **Часть 1. Обзор возможных путей реализации проекта**

Для реализации построения графика квадратного уравнения я буду использовать язык программирования Python, так как он универсален, имеет простой синтаксис и множество библиотек. Чтобы решить задачу построения графика квадратного уравнения на Python необходимо использование сторонних библиотек. Примеры некоторых из них, позволяющих работать с математическими моделями и графиками:

* **Matplotlib** – основная библиотека для построения графиков:
  + Позволяет строить графики функций, настраивать внешний вид графиков (цвета, метки осей, сетку и т.д.).
  + Легко использовать для построения графика квадратной функции
* **NumPy** – для работы с числами и массивами:
  + Содержит многомерные массивы и матричные структуры данных
  + Необходима выполнения множества математических операций с массивами, удобна для работы с математическими вычислениями.
* **SymPy** – для работы с математическими выражениями:
  + Если нужно символьно решить квадратное уравнение или отобразить формулу.
  + Можно использовать для нахождения корней уравнения.

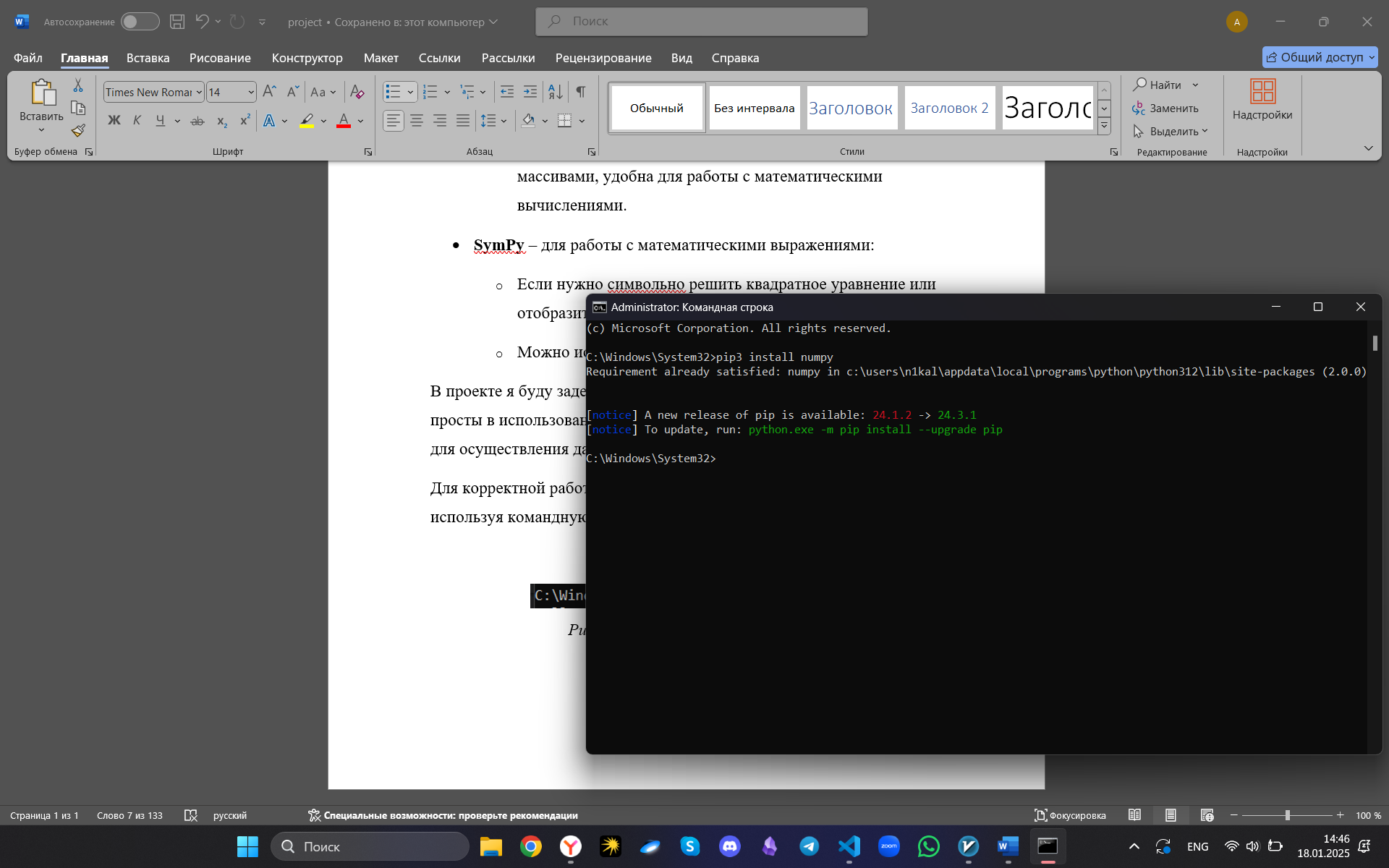
В проекте я буду задействовать библиотеки NumPy и Matplotlib, так как они просты в использовании и являются самыми популярными инструментами для осуществления данной задачи.

# **Часть 2. Предварительная подготовка библиотек**

Для корректной работы программы данные библиотеки нужно установить, используя командную строку или терминал среды разработки.



*Рис.1 Установка библиотеки Matplotlib*



*Рис.2 Установка библиотеки NumPy*

Для того, чтобы использовать данные библиотеки в проекте, их нужно импортировать при помощи команды import. Для удобства импортируем их с присвоением псевдонимов: «np» для NumPy, «plt» - для Matplotlib соответственно.

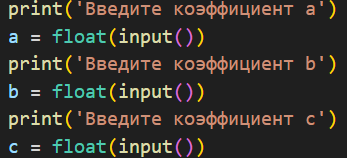


*Рис.3 Импортирование библиотек*

# **Часть 3. Написание программного кода**

## 3.1 Код, реализованный с помощью встроенных функций Python

Чтобы программа выводила график квадратного уравнения, необходимо ввести коэффициенты a, b и c, задающие функцию. Для этого выведем подсказку для удобства использования программы с помощью функции print(). Воспользуемся функцией input() для ввода коэффициентов и сразу переведем их значения в тип данных float для учета всех возможных вариантов задания уравнения.



*Рис.4 Ввод коэффициентов уравнения*

## *3.2 Код, реализованный с помощью библиотеки NumPy*

Необходимо задать множество иксов на графике (ось Ox) при помощи метода .linspace() и передать полученный массив переменной x. Аргументами .linspace() являются параметры start, stop, step, аналогично аргументам встроенной функции range(). Далее необходимо вычислить значение функции y для соответствующих x, рассчитаем их по формуле ax^2 + bx + c.



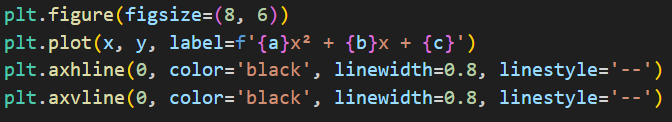
*Рис.5 Создание массива значений x и расчет соответствующих y*

## *3.3 Код, реализованный с помощью библиотеки Matplotlib*

Для вывода графика необходимо создать окно, с этим поможет справиться метод .figure(). В качестве его аргумента укажем figsize = (width, height), с числовыми параметрами вместо width и height, чтобы задать размер окна для графика в дюймах.

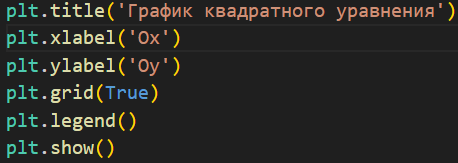
При помощи метода .plot() построим график в осях x и y, которые мы ранее рассчитали, добавим метку для легенды графика label в качестве аргумента, чтобы отобразить символьную запись квадратного уравнения.

Чтобы сделать график более читабельным и удобным для восприятия, сделаем оси Ox и Oy пунктирными с помощью метода .axhline() .axvline() для отрисовки вертикальной и горизонтальной прямой соответственно. В аргументах указано y = 0 и x = 0, color для задания цвета линии, linewidth для указания толщины линии и аргумент linestyle, отвечающий за пунктир.



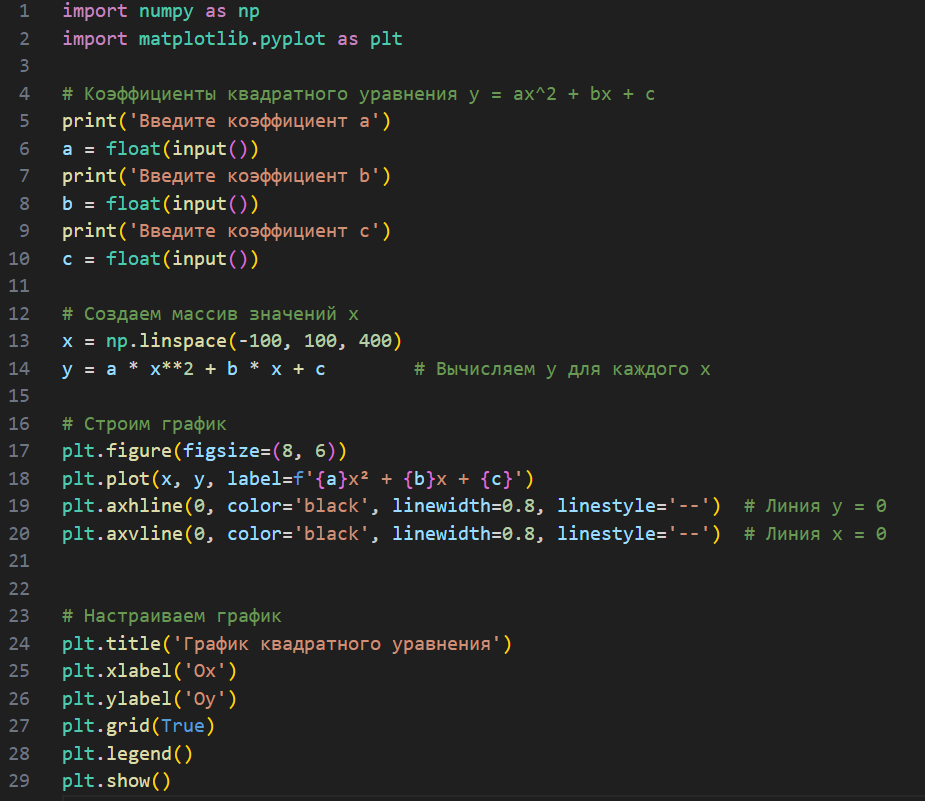
*Рис.6 Построение графика*

Далее сформируем заголовок графика, используя функцию .title(). Подпишем оси Ox и Oy при помощи функций .xlabel() и .ylabel() соответственно. Функция .grid() позволит отобразить сетку, при наведении мыши на которую можно увидеть значение x и y в любой точке графика. Функция .legend() отобразит легенду графика, а .show() выведет на экран настроенное ранее окно для графика.



*Рис.7 Отображение окна с графиком*

# **Приложение**



*Программный код проекта*

# **Список литературы**

Лутц М. Изучаем Python. Программирование на языке Python. — 5-е изд, 2022.

Matplotlib: документация [Электронный ресурс]. — URL: https://matplotlib.org/stable/contents.html (дата обращения: 18.01.2025).

NumPy: документация [Электронный ресурс]. — URL: <https://numpy.org/doc/stable/>

Герасимов А. Ю., Мартьянов А. В. Математика на Python: от простого к сложному, 2021.