**Разработка калькулятора квадратных уравнений на Python**

Выполнил

ученик 10-4 класса

Ларин Илья

Руководитель

Ловыгина Надежда

Васильевна

**ОГЛАВЛЕНИЕ**

Введение………………………………………………………………………………………….3

I.Изучение истории…………………………………………………………………………...….4

1.1) История компьютеров………………………………………………………………4

1.2) История Python……………………………………………………………………...4

1.3) Почему Python………………………………………………………………………5

II.Библиотеки Python…………………………………………………………………………….7

2.1) Создание калькулятора……………………………………………………………..7

2.2) Используемые библиотеки…………………………………………………………7

Заключение……………………………………………………………………………………….9

Список литературы и интернет ресурсов……………………………………………………..10

**ВВЕДЕНИЕ**

В современном мире технологии играют важную роль в повседневной жизни. Компьютер одна из самых распространенных таких технологий, но его центр - процессор - является простым калькулятором, который только умеет вычислять числа в двоичной системе счисления, поэтому в своем пути изучения программирования я решил изучить историю создания вычислительных машин и сделать калькулятор.

Цель: Создать калькулятор квадратных уравнений на основе простейших алгоритмов и библиотек языка Python

Задачи:

1. Изучить историю создания компьютера
2. Изучить библиотеки Python, для создания графического интерфейса
3. Изучить библиотеки Python, облегчающие алгоритмизацию.
4. Написать программу для вычисления уравнений.
5. Создать удобный интерфейс

Данный проект представляет собой актуальное и полезное исследование, которое позволит изучить основы алгоритмизации. Написанный калькулятор может быть использован в повседневной жизни

**ИСТОРИЯ КОМПЬЮТЕРОВ**

Одно из первых устройств для счета, появившееся около 3000 лет до н.э. в Месопотамии, Китае и других древних цивилизациях был был **абак** (счеты). Он позволял выполнять простые арифметические операции с помощью костяшек на проволоке.

Первым прототипом настоящей вычислительной машины считается устройство, созданное Блезом Паскалем в 1642 году - Паскалина, которое могло выполнять простые арифметические операции. В дальнейшем, в XIX веке, Чарльз Бэббидж разработал концепцию "аналитической машины", которая закладывала основы для современных компьютеров, включая использование перфокарт для ввода данных.

В 1938 году немецкий инженер Конрад Цузе создал **Z1** — первую программируемую механическую машину. За ней последовали более совершенные модели, такие как **Z3**, которая стала первой полностью функциональной вычислительной машиной.

В 1943 году в США была разработана **ENIAC** — первая электронная вычислительная машина, использующая лампы и способная выполнять множество операций в секунду.

С изобретением транзисторов в конце 1940-х годов произошел настоящий прорыв в развитии компьютеров: они стали меньше по размеру и быстрее.

В 1970-х годах началась эра персональных компьютеров с появлением моделей, таких как **Altair** и **IBM PC**, который стал стандартом для многих компаний.

**ИСТОРИЯ PYTHON**

История языка программирования Python началась в конце 1980-х годов, когда голландский программист Гвидо ван Россум начал разработку нового языка в Центре математики и информатики (CWI) в Нидерландах. Ван Россум задумывал Python как преемника языка ABC, который был предназначен для обучения программированию, но имел свои ограничения. Он стремился создать язык, который был бы простым в использовании и при этом мощным, что позволило бы разработчикам легко решать различные задачи.

## Ранние годы и первая версия

Работа над Python началась в декабре 1989 года, и уже в феврале 1991 года была выпущена первая версия — Python 0.9.0. Эта версия включала основные функции, такие как обработка исключений, функции, модули и базовые типы данных, такие как списки, словари и строки. Ван Россум описывал модули как один из главных элементов программирования на Python, что способствовало модульности и повторному использованию кода.

## Официальные релизы

С ростом популярности языка в 1994 году была выпущена версия Python 1.0, которая добавила поддержку классов и систему модулей. В этом же году сформировалась группа обсуждения comp.lang.python, ставшая основным форумом для пользователей Python.

В октябре 2000 года вышла версия Python 2.0, которая включала множество новых функций, таких как полный сборщик мусора и поддержка Unicode. Эта версия также ознаменовала переход к более открытой модели разработки языка.

## Переход на Python 3

Долгожданная версия Python 3.0 была выпущена в декабре 2008 года после длительного тестирования. Она устранила многие недостатки предыдущих версий и ввела изменения, которые сделали язык более современным и удобным для разработчиков. Однако это также привело к потере обратной совместимости: программы, написанные для Python2, часто не работали в среде Python 3.

## Роль Гвидо ван Россума

Гвидо ван Россум оставался ключевой фигурой в развитии Python до июля 2018 года, когда он объявил о своем уходе с поста "великодушного пожизненного диктатора" проекта. После этого развитие языка стало полностью зависеть от сообщества разработчиков под управлением некоммерческой организации Python Software Foundation (PSF). PSF отвечает за выпуск новых версий языка, защиту интеллектуальных прав и организацию конференций.

После ухода из PSF ван Россум работал в компании Dropbox до октября 2019 года, а затем вернулся к разработке Python в Microsoft в ноябре 2020 года.

## Современное состояние

На сегодняшний день Python продолжает активно развиваться и остается одним из самых популярных языков программирования в мире. Он широко используется в различных областях — от веб-разработки до научных вычислений и машинного обучения. В последние годы были выпущены обновления, которые значительно улучшили безопасность и производительность языка.

Python стал языком выбора для многих разработчиков благодаря своей простоте и мощным возможностям. Сообщество вокруг языка продолжает расти, а новые версии приносят новые функции и улучшения, что делает его актуальным инструментом для решения современных задач программирования.

**ПОЧЕМУ PYTHON**

Калькулятор можно написать на любом языке программирования, но почему же мной был выбран именно Python? Главная причина, это то, что мы изучаем Python на уроках информатики в школе, а так же он предпочтителен для написания ЕГЭ, ещё у него есть ряд преимуществ, которые выделяют его на фоне других языков:

1. Python легок для изучения
2. Python имеет огромное сообщество разработчиков и обширную документацию, что упрощает поиск решений и примеров для реализации математических алгоритмов
3. Код на Python легко расширяется, что я использовал при создании проекта, добавив к калькулятору график функции
4. С использованием библиотек, таких как NumPy и SymPy, Python становится мощным инструментом для выполнения сложных математических операций, что ускоряет вычисление квадратных уравнений.

Из-за этих и некоторых других причин я выбрал именно Python для создания своего проекта.

**СОЗДАНИЕ КАЛЬКУЛЯТОРА**

Разработку калькулятора я начал со сбора информации в интернете, изучив историю создания компьютера, я написал простенькую функцию способную решить уравнение, но мне хотелось ускорить ее работу, поэтому я нашел в интернете библиотеку SymPy, которая способна облегчить выполнение задачи. Написав программу и создав для нее графический интерфейс при помощи стандартной библиотеки Tkinter, я понял, что создание калькулятора на Python оказалось слишком легким, а времени до сдачи проекта было еще много, поэтому воспользовавшись гибкостью языка и библиотекой Matplotlib, я добавил график функции введенного уравнения. Но программа несмотря на использовании SymPy все равно работала медленней чем хотелось, поэтому мне пришлось искать решение этой проблемы в интернете. Оказалось, что для черчения графика вычислялось очень много точек, все точки хранились в списке, что замедляло работу программы, поэтому я использовал библиотеку NymPy для создания одномерного массива, который работает гораздо быстрее списков.

**ИСПОЛЬЗОВАННЫЕ БИБЛИОТЕКИ**

**SymPy** — это мощная библиотека для символьных вычислений на языке Python. Она позволяет выполнять математические операции с символами, а не только с числами, что делает её полезной для аналитических вычислений, алгебраических преобразований, решения уравнений, дифференцирования, интегрирования и многого другого. SymPy является полностью написанной на Python и не требует внешних зависимостей

`symbols`: Используется для определения символических переменных в алгебраических выражениях.

`Eq`: Используется для создания символических уравнений.

`solve`: Используется для символического решения уравнений.

**Tkinter** — это стандартная библиотека Python для создания графических пользовательских интерфейсов (GUI). Она предоставляет инструменты для создания окон, кнопок, меток и других элементов интерфейса.

`tk.Tk()`: Инициализирует новое окно приложения Tkinter. Создает главное окно, в котором будут размещены все виджеты.

`root.title()`: Устанавливает заголовок главного окна приложения.

`root.geometry()`: Устанавливает размер главного окна приложения.

`tk.Label(root, text="...")`: Создает текстовую метку в указанном родительском окне (root), отображающую заданный текст.

`grid(row=..., column=..., padx=..., pady=...)`: Размещает виджет в сетке на указанной строке и столбце с дополнительными отступами (padx и pady) вокруг виджета.

`tk.Entry(root, width=...)`: Создает поле ввода (текстовое поле) в указанном родительском окне (root) с заданной шириной, позволяя пользователю вводить текст.

`tk.Button(root, text="...", command=...)`: Создает кнопку в указанном родительском окне (root) с заданным текстом. Параметр command указывает функцию, которая будет вызвана при нажатии на кнопку.

`tk.StringVar()`: Создает переменную Tkinter, которая может содержать строковое значение. Часто используется для управления значением полей ввода или меток.

`textvariable=...`: Связывает переменную Tkinter (например, tk.StringVar()) с виджетом, позволяя виджету отображать и обновлять значение переменной.

`state='readonly'`: Устанавливает состояние поля ввода как 'только для чтения', что означает, что пользователь не может напрямую изменять текст в поле.

`canvas.get\_tk\_widget().grid(...)`: Получает виджет Tkinter, связанный с объектом Matplotlib, и размещает его в сетке.

`root.iconbitmap()`: Устанавливает иконку для главного окна приложения, используя указанный файл.

`root.mainloop()`: Запускает цикл обработки событий Tkinter, который ожидает взаимодействия пользователя и обновляет графический интерфейс соответственно. Этот вызов необходим для того, чтобы окно приложения оставалось открытым.

**Matplotlib** — это одна из самых популярных и мощных библиотек для визуализации данных на языке Python. Она предоставляет широкие возможности для создания статических, анимированных и интерактивных графиков. Matplotlib часто используется в научных исследованиях, анализе данных, машинном обучении и других областях, где требуется визуальное представление данных.

`Figure`: Представляет собой фигуру, в которой можно рисовать графики.

`FigureCanvasTkAgg`: Интегрирует фигуры Matplotlib в приложение Tkinter, позволяя отображать графики внутри окна Tkinter..

`fig.add\_subplot(111)`: Добавляет подграфик в объект Figure, где 111 указывает на один подграфик в сетке 1x1.

**NumPy** — это библиотека для численных вычислений в Python. Она предоставляет поддержку массивов и матриц, а также набор математических функций для работы с этими структурами данных.

‘linespace’: Создает одномерный массив для диапазона значений x для построения графика квадратичной функции.

Эти библиотеки вместе позволяют приложению символически решать квадратные уравнения, создавать графический интерфейс для взаимодействия с пользователем и графически отображать квадратичную функцию в интерфейсе.

**ЗАКЛЮЧЕНИЕ**

Я изучил историю создания компьютера, нашел и изучил библиотеки Python, которые мне понадобятся при создании проектов.

После изучений я написал калькулятор квадратных уравнений, задаваемых пользователем, способный также выводить график функции этого уравнения.

Таким образом я узнал чем отличалось создание вычислительных машин раньше и сейчас, а также создал собственный калькулятор, который можно использовать в повседневной жизни.

**СПИСОК ЛИТЕРАТУРЫ И ИНТЕРНЕТ РЕСУРСОВ**

* <https://ru.wikipedia.org/wiki/Python>
* <https://www.sympy.org/en/features.html>
* <https://pythonru.com/biblioteki/sympy-v-python>
* <https://numpy.org/>
* https://skillbox.ru/media/code/biblioteka-numpy-vsye-chto-nuzhno-znat-novichku/