**ПАВЛОВ ДАНИЛА ВЛАДИМИРОВИЧ**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON ПО ВИЗУАЛИЗАЦИИ ГРАФИКОВ ФУНКЦИИ, ЗАДАННЫХ ПОЛЬЗОВАТЕЛЕМ**

Диплом

|  |
| --- |
| Выпускная дипломная работа защищена  «     »  января 2024 г. |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Секретарь ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_ |

г. Казань

2024 г.

# ****1. Введение****

Представление данных на мониторе компьютера в графическом виде впервые было реализовано в середине 50-х годов для больших ЭВМ, применявшихся в научных и военных исследованиях. С тех пор графический способ отображения данных стал неотъемлемой принадлежностью подавляющего числа компьютерных систем, в особенности персональных. Графический интерфейс пользователя сегодня является стандартом “де-факто” для программного обеспечения разных классов, начиная с операционных систем.

Существует специальная область информатики, изучающая методы и средства создания и обработки изображений с помощью программно-аппаратных вычислительных комплексов, – компьютерная графика. Она охватывает все виды и формы представления изображений, доступных для восприятия человеком либо на экране монитора, либо в виде копии на внешнем носителе (бумага, кинопленка, ткань и прочее).

Без компьютерной графики невозможно представить себе не только компьютерный, но и обычный, вполне материальный мир. Визуализация данных находит применение в самых разных сферах человеческой деятельности.

В работе бухгалтеров и аналитиков постоянно встречается задача подведения итогов работы компании за определенный период: месяц, квартал, год и т.п. При этом получаются числовые таблицы порой весьма немаленькой величины. Окинув их взглядом, практически невозможно оценить степень успеха или провала компании за выбранный период. И, тем более, не видна динамика изменения отчетных показателей.

Если при этом приходится показывать отчетность вышестоящему руководству, то оно может не оценить большой набор чисел. В этом случае удобно показывать рисованные графики, построенные на основе исходных таблиц. Графическое представление данных позволяет быстро оценить соотношение нескольких величин.

В научно-исследовательской деятельности графики используются для емкого и доступного изображения выявленных закономерностей, изменений, воссоздания визуального представления о разработке или предложении.

Научная визуализация, использующая компьютерную графику, получила популярность как вполне созревшая. Основными приложениями были скалярные и векторные поля из компьютерного моделирования и опытных данных. Основными методами для визуализации двумерных (2D) скалярных полей является отображение цветом и рисование контурных линий. 2D-векторные поля визуализируются с помощью глифов и линий тока или методов линейной интегральной свёртки (LIC). 2D-тензорные поля часто преобразуется к векторному полю с помощью одного из двух собственных векторов для представления каждой точки поля, затем используется визуализация с помощью методов визуализации векторного поля.

Во время проведения анализа остро встает вопрос поведения той или иной функции при изменении ее параметров. Не многие сервисы предоставляют возможность изменять входные значения или дают возможность на ходу поменять параметр относительно которого строиться график.

Исходя из всего вышеперечисленного разработка приложения по визуализации графиков функции, заданных пользователем необходима во многих отраслях. Реализация данного ПО поможет в анализе различных зависимостей.

Данная тема была выбрана, так как часто приходится сталкиваться с большим объемом данных, которые необходимо скомпоновать и представить в читаемом виде. Кроме того, присутствует необходимость отобразить график при разных значениях параметров. Разработка приложения на Phyton, позволяет развить и отработать опыт создания интересных и разносторонних проектов. Кроме того, такие навыки востребованы на рынке труда, что открывает перспективы для карьерного роста и развития.

Определение цели и задач исследования:

Цель исследования: разработка приложения на языке программирования Python по визуализации графиков функции, заданных конечным пользователем.

Задачи исследования:

1. Обзор библиотек, позволяющих взаимодействовать с пользователем. Провести анализ существующих инструментов, выбрать наиболее подходящие под имеющуюся задачу.
2. Обзор библиотек для визуализации данных. Провести анализ существующих инструментов, выбрать наиболее подходящие под имеющуюся задачу.
3. Разработка алгоритма взаимодействия пользователя с приложением.
4. Создание приложения для отображения графиков по заданной функции и параметрам.
5. Определение дальнейшего направления развития программы.
6. Написание дипломной работы. Составить дипломную работу, включающую в себя введение, обзор литературы, алгоритм методологию и результаты исследования, анализ результатов, выводы и рекомендации.

Цели и задачи исследования направлены на получение практических и теоретических навыков, которые позволят создать приложение, в котором на постоянной основе пользователи смогут построить необходимый им график функции.

# ****2. Основные понятия****

**Обзор основных понятий в области визуализации данных:**

1. Фреймворк (Framework): Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений.
2. Графический интерфейс пользователя сокр. ГИП (Graphical User Interface (GUI)): Способ взаимодействия пользователя с компьютером с использованием графических элементов, таких как окна, кнопки и меню.

Чаще всего элементы интерфейса в GUI реализованы на основе метафор и отображают их назначение и свойства, что облегчает понимание и использование электронных устройств неподготовленными пользователями. Графический интерфейс пользователя является частью пользовательского интерфейса и определяет взаимодействие с пользователем на уровне визуализированной информации.

1. Прикладная программа (программное приложение, или сокращенно application, или приложение) — это компьютерная программа, предназначенная для выполнения определенной задачи, отличной от задачи, связанной с работой самого компьютера, обычно используемой конечными пользователями.
2. Функция — соответствие между двумя множествами, при котором каждому элементу одного множества соответствует единственный элемент другого.
3. Библиотека (от англ. library) — сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО). С точки зрения операционной системы (ОС) и прикладного ПО, библиотеки разделяются на динамические и статические. В контексте **визуализации данных** часто используются Matplotlib, Seaborn и Plotly.

Оглавление

[1. Введение 1](#_Toc182821421)

[2. Основные понятия 4](#_Toc182821422)

[3. Обзор библиотек 7](#_Toc182821423)

[4. Разработка приложения 24](#_Toc182821424)

[5. Направления развития 33](#_Toc182821425)

[6. Заключение 34](#_Toc182821426)

# 3. Обзор библиотек

3.1 Обзор библиотек графических пользовательских интерфейсов.

Фреймворки Python GUI упрощают разработку графических пользовательских интерфейсов (GUI) в приложениях на Python. Эти фреймворки предоставляют инструменты и библиотеки, которые помогают разработчикам создавать интуитивно понятные, визуально привлекательные программные интерфейсы, которые улучшат общее взаимодействие с пользователем. Далее рассмотрены несколько фреймворков Python GUI.  
 Pyside 2.

PySide2 — это официальный набор привязок Python для Qt 5, который позволяет разработчикам создавать кроссплатформенные графические интерфейсы для приложений. Он является частью проекта Qt for Python и предоставляет мощные инструменты для разработки десктопных приложений с богатым пользовательским интерфейсом.

Основные характеристики PySide2:

1. Кроссплатформенность: PySide2 позволяет создавать приложения, которые работают на различных операционных системах, включая Windows, macOS и Linux, с минимальными изменениями в коде.
2. Полный доступ к функциональности Qt: PySide2 предоставляет доступ ко всем возможностям Qt, включая графику, мультимедиа, сетевые операции и другие компоненты, что позволяет создавать сложные и функциональные приложения.
3. Простой в использовании: Библиотека предлагает интуитивно понятный API и хорошо структурированную документацию, что облегчает процесс разработки для новичков и опытных программистов.
4. Поддержка современных концепций: PySide2 поддерживает такие концепции, как сигналы и слоты, что упрощает обработку событий и взаимодействие между компонентами интерфейса.
5. Поддержка QML: PySide2 позволяет использовать QML — декларативный язык для проектирования интерфейсов, что дает возможность создавать динамичные и отзывчивые пользовательские интерфейсы.
6. Сообщество и ресурсы: PySide2 имеет активное сообщество и множество учебных материалов, примеров и форумов, что помогает разработчикам находить решения и делиться опытом.
7. Лицензирование: PySide2 распространяется под лицензией LGPL, что позволяет использовать библиотеку в коммерческих проектах без необходимости открывать исходный код.

PySide2 является отличным выбором для разработчиков, которые хотят создавать кроссплатформенные приложения с привлекательным графическим интерфейсом, используя возможности Qt и Python. Wax.

Wax — это библиотека для Python, предназначенная для разработки приложений с графическим интерфейсом на основе wxPython. Она предоставляет дополнительные инструменты и расширения, упрощая создание пользовательских интерфейсов и улучшая взаимодействие с wxPython.

Основные характеристики Wax:

1. Упрощение разработки: Wax предлагает более высокоуровневые абстракции и упрощает работу с элементами интерфейса, что снижает объем кода и ускоряет процесс разработки.

2. Поддержка wxPython: Wax строится на основе wxPython, что позволяет использовать все возможности этой библиотеки для создания кроссплатформенных приложений.

3. Гибкость и расширяемость: Wax предоставляет разработчикам возможность создавать кастомизированные виджеты и компоненты, что позволяет адаптировать интерфейс под специфические требования приложения.

4. Совместимость: Wax совместим с различными версиями wxPython, что позволяет использовать его в существующих проектах без необходимости значительных изменений.

5. Документация и ресурсы: Хотя Wax имеет меньшее сообщество по сравнению с другими библиотеками, доступна документация и примеры, которые помогают в изучении и использовании библиотеки.

Wax может быть полезен для разработчиков, которые хотят создать графические интерфейсы на Python, используя wxPython, но с меньшими усилиями и более удобным подходом к проектированию интерфейсов.

PyGame.

Pygame — это библиотека для Python, предназначенная для разработки видеоигр и мультимедийных приложений. Она предоставляет простые и удобные инструменты для работы с графикой, звуком и взаимодействием с пользователем.

Основные характеристики Pygame:

1. Простота использования: Pygame имеет интуитивно понятный интерфейс, что делает ее доступной как для новичков, так и для опытных разработчиков.

2. Кроссплатформенность: Pygame работает на различных операционных системах, включая Windows, macOS и Linux, что позволяет создавать игры, которые можно запускать на разных устройствах.

3. Поддержка 2D-графики: Библиотека предлагает функции для работы с изображениями, спрайтами и анимацией, что позволяет создавать красочные и динамичные игры.

4. Работа со звуком: Pygame поддерживает различные аудиоформаты, позволяя добавлять звуковые эффекты и музыку в игры.

5. Обработка событий: Библиотека позволяет легко управлять вводом с клавиатуры, мыши и других устройств, что делает взаимодействие с пользователем удобным и отзывчивым.

6. Сообщество и ресурсы: Pygame имеет активное сообщество и множество учебных материалов, что облегчает обучение и решение возникающих вопросов.

7. Расширяемость: Pygame можно интегрировать с другими библиотеками и инструментами Python, что позволяет создавать более сложные и функциональные проекты.

Pygame является отличным выбором для тех, кто хочет начать разработку игр на Python, так как она сочетает в себе простоту, мощные возможности и активную поддержку сообщества. Выбор библиотеки Tkinter для разработки графических интерфейсов в Python обоснован рядом факторов, которые делают ее привлекательной для разработчиков.

1. Простота и доступность: Tkinter является стандартной библиотекой Python, что исключает необходимость установки дополнительных пакетов. Это позволяет быстро начать разработку графических приложений.
2. Интуитивно понятный интерфейс: Библиотека предлагает простой и понятный API, что облегчает создание интерфейсов даже для начинающих разработчиков. Обширная документация и множество учебных материалов способствуют быстрому освоению.
3. Кроссплатформенность: Приложения, созданные с использованием Tkinter, работают на различных операционных системах (Windows, macOS, Linux), что позволяет разработать универсальные решения.
4. Широкий набор виджетов: Tkinter поддерживает множество графических элементов, таких как кнопки, текстовые поля, списки и меню, что дает возможность создавать многофункциональные интерфейсы.
5. Поддержка событий: Tkinter позволяет легко обрабатывать события, что делает интерфейсы интерактивными и отзывчивыми, улучшая пользовательский опыт.
6. Легкость интеграции: Библиотеку можно легко интегрировать с другими модулями и библиотеками Python, что расширяет возможности приложения.
7. Сообщество и ресурсы: существует большое количество ресурсов, примеров и активное сообщество, что помогает решать возникающие вопросы и находить решения в процессе разработки.

Эти факторы делают Tkinter оптимальным выбором для разработки графических приложений в Python, обеспечивая баланс между простотой использования и функциональностью.

3.2 Обзор библиотек для визуализации данных.

Визуализация данных — это метод, который позволяет специалистам по анализу данных преобразовывать сырые данные в диаграммы и графики, которые несут ценную информацию. Диаграммы уменьшают сложность данных и делают более понятными для любого пользователя.

Есть множество инструментов для визуализации данных, таких как Tableau, Power BI, ChartBlocks и другие, которые являются no-code инструментами. Они очень мощные, и у каждого своя аудитория. Однако для работы с сырыми данными, требующими обработки Python подойдет лучше всего.

В данной работе представлен обзор нескольких популярных библиотек для визуализации, таких как Matplotlib, Seaborn и Plotly. Для демонстрации примеров визуализации были выбраны два типа графиков:

* Линейный график – на нем продемонстрирована численность (млрд.) планеты Земля в зависимости от года
* Гистограмма – на ней показано распределение Гаусса

Matplotlib.

Matplotlib — библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой. Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

Библиотека написан и поддерживался в основном Джоном Хантером (англ. John Hunter) и распространяется на условиях BSD-подобной лицензии. Генерируемые в различных форматах изображения могут быть использованы в интерактивной графике, в научных публикациях, графическом интерфейсе пользователя, веб-приложениях, где требуется построение диаграмм.

Версия 2.1.1 — последняя стабильная — требует Python версии 2.7 или от 3.4 и выше и версию NumPy от 1.7.1 и выше.

Matplotlib построена на принципах ООП, но имеет процедурный интерфейс pylab, который предоставляет аналоги команд MATLAB.

Возможности Matplotlib.

Matplotlib является гибким, легко конфигурируемым пакетом, который вместе с NumPy, SciPy и IPython предоставляет возможности, подобные MATLAB. В настоящее время пакет работает с несколькими графическими библиотеками, включая wxWindows и PyGTK.

Пакет поддерживает многие виды графиков и диаграмм:

* Графики (англ. line plot)
* Диаграммы рассеяния (англ. scatter plot)
* Столбчатые диаграммы (англ. bar chart) и гистограммы (англ. histogram)
* Круговые диаграммы (англ. pie chart)
* Диаграммы стебель-листья (англ. stem plot)
* Контурные графики (англ. contour plot)
* Поля градиентов (англ. quiver)
* Спектральные диаграммы (англ. spectrogram)

Пользователь может указать оси координат, решетку, добавить надписи и пояснения, использовать логарифмическую шкалу или полярные координаты.

Несложные трёхмерные графики можно строить с помощью набора инструментов (toolkit) mplot3d. Есть и другие наборы инструментов: для картографии, для работы с Excel, утилиты для GTK и другие.

С помощью Matplotlib можно делать и анимированные изображения.

Набор поддерживаемых форматов изображений, векторных и растровых, можно получить из словаря FigureCanvasBase.filetypes. Типичные поддерживаемые форматы:

* Encapsulated PostScript (EPS)
* Enhanced Metafile (EMF)
* JPEG
* PDF
* PNG
* Postscript
* RGBA («сырой» формат)
* SVG
* SVGZ
* TIFF

Кроме того, на основе классов пакета можно создавать и другие модули. Например, для генерации искрографиков.

Далее приведены примеры программ по построению различных видов графиков с пояснениями и результатами выполнения рис.3.2.1-3.2.2.

График.

import matplotlib.pyplot as pl

x = [1974,1987,1999,2012,2022,2046]

y = [4,5,6,7,8,9]

pl.Figure(figsize=None,#Размер рисунка (ширина, высота) в дюймах: 2 числа с плавающей точкой, по умолчанию: :rc:`figure.figsize`

dpi=None,#dpi : значение с плавающей точкой, по умолчанию: :rc:`figure.dpi` Количество точек на дюйм.

facecolor=None,#Цвет фона :по умолчанию: :rc:`figure.facecolor`.

edgecolor=None,#Цвет кромки : по умолчанию: :rc:`figure.edgecolor

linewidth=0.0,#Толщина линии: число с плавающей точкой

frameon=None,#Флаг на отображение контура фигуры

subplotpars=None,#Подзаголовок фигуры

tight\_layout=None,#Флаг на механизм полной компоновки

constrained\_layout=None,#Флаг на механизм частичной компоновки

layout=None)#Механизм компоновки для позиционирования элементов графика, позволяющий избежать наложения элементов оформления осей (надписей, галочек и т.д.).

pl.title(label='Заголовок окна',

loc='center',#Выравнивание по краю

fontsize=14,#Размер шрифта

fontweight='regular')#Толщина шрифта

plt.xlabel('Подпись оси абсцисс')

plt.ylabel('Подпись оси ординат')

pl.figtext(0.2, 0.6,#Координаты текста в области окна

'Текст в области окна')

pl.plot(x,y, #Данные для графика

'r--o', #Формат вывода графика

label='Название графика')

pl.legend()# Метод для вывода легенды графика

pl.grid()# Метод для создания сетки

pl.show()# Метод для фиксации отображения графика



Рисунок 3.2.1. Пример линейного графика, построенного при помощи Matplotlib.

Гистограмма.

import matplotlib.pyplot as plt

import numpy as np

# Генерация случайного набора чисел

data = np.random.randn(1000)

# Построение гистограммы

plt.hist(data,#Значения

color='skyblue',#Цвет столбца

edgecolor='black')#Цвет окантовки столбца

# Оформление

plt.xlabel('Значение')

plt.ylabel('Количество')

plt.title('Гистограмма')

plt.show()

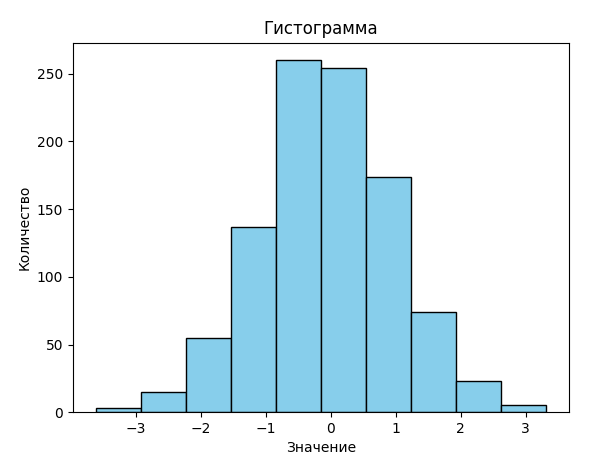


Рисунок 3.2.2. Пример гистограммы, построенной при помощи Matplotlib.

Seaborn

Seaborn — это библиотека для визуализации данных в Python, построенная на основе Matplotlib. Она предоставляет высокоуровневый интерфейс для создания красивых и информативных графиков, что облегчает анализ данных и их представление.

Основные характеристики Seaborn:

1. Упрощенная визуализация: Seaborn предлагает простой и интуитивно понятный синтаксис для создания сложных графиков, таких как тепловые карты, парные диаграммы и графики распределения.

2. Эстетика: Библиотека обеспечивает привлекательный стиль графиков по умолчанию, что позволяет создавать визуализации, которые выглядят профессионально без необходимости дополнительной настройки.

3. Интеграция с Pandas: Seaborn отлично работает с DataFrame из библиотеки Pandas, что упрощает работу с табличными данными.

4. Поддержка статистических функций: Seaborn включает встроенные функции для статистического анализа, что позволяет легко добавлять линии регрессии, доверительные интервалы и другие статистические элементы на графики.

5. Разнообразие графиков: Библиотека предлагает широкий спектр типов графиков, включая линейные, точечные, столбчатые и распределения, что помогает визуализировать данные с разных сторон.

6. Настраиваемость: Seaborn позволяет настраивать графики, изменяя цвета, размеры, стили и другие параметры, что дает разработчикам гибкость в создании уникальных визуализаций.

7. Легкость в использовании: Seaborn упрощает создание сложных визуализаций с минимальным количеством кода, что делает его идеальным инструментом для исследовательского анализа данных.

Seaborn является отличным выбором для аналитиков и исследователей, которые хотят быстро и эффективно визуализировать данные, используя Python.

Далее приведены примеры программ по построению различных видов графиков с пояснениями и результатами выполнения рис.3.2.3-3.2.4.

График.

import seaborn as sns

import matplotlib.pyplot as pl

import pandas as pd

# Данные

x\_y = {'year': [1974, 1987, 1999, 2012, 2022, 2046], 'people': [4, 5, 6, 7, 8, 9]}

# Создаем датафрейм

dframe = pd.DataFrame(x\_y)

# Строим график

sns.lineplot(x='year', y='people', data=dframe) # Строим график

pl.show()

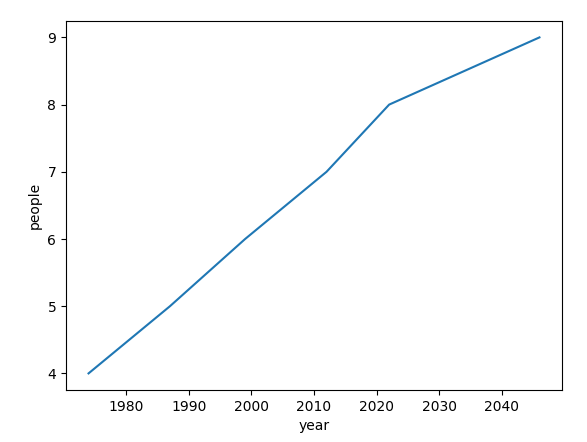


Рисунок 3.2.3. Пример линейного графика, построенного при помощи Seaborn

Гистограмма.

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# Генерация случайных данных с нормальным распределением

data = np.random.normal(loc=0, scale=1, size=1000)

# Создание гистограммы с помощью Seaborn

sns.histplot(data, bins=30, kde=True)

# Настройка заголовка и меток

plt.title('Гистограмма распределения Гауса')

plt.xlabel('Значение')

plt.ylabel('Частота')

# Отображение графика

plt.show()

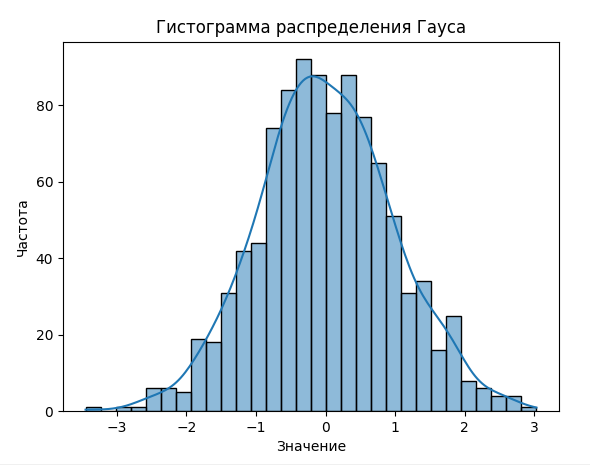


Рисунок 3.2.4. Пример гистограммы, построенной при помощи Seaborn.

Plotly.

Plotly — это библиотека для визуализации данных на Python, которая позволяет создавать интерактивные графики и приложения. Она предназначена для работы с различными типами данных и предоставляет мощные инструменты для создания публикаций и отчетов.

Основные характеристики Plotly:

1. Интерактивность: Plotly позволяет создавать интерактивные графики, которые можно масштабировать, перемещать и наводить курсор для получения дополнительной информации, что улучшает восприятие данных.

2. Широкий спектр графиков: Библиотека поддерживает множество типов графиков, включая линейные, столбчатые, круговые, 3D-графики, тепловые карты и многие другие.

3. Простота использования: Plotly предлагает интуитивно понятный интерфейс и хорошо документированные функции, что облегчает процесс создания визуализаций.

4. Кроссплатформенность: Графики, созданные с помощью Plotly, могут быть интегрированы в веб-приложения и работают на различных платформах, включая Jupyter Notebook, веб-браузеры и серверные приложения.

5. Поддержка Dash: Plotly предлагает Dash — фреймворк для создания веб-приложений на Python, который позволяет строить интерактивные интерфейсы для визуализации данных.

6. Статистические графики: Библиотека включает функции для создания сложных статистических графиков и визуализаций, что делает ее полезной для исследователей и аналитиков.

7. Совместимость с Pandas: Plotly легко интегрируется с библиотекой Pandas, что позволяет строить графики непосредственно из DataFrame.

8. Лицензирование: Plotly имеет как бесплатную, так и платную версии, что позволяет выбрать подходящий вариант в зависимости от потребностей проекта.

Plotly является отличным выбором для разработчиков и аналитиков, которые хотят создавать профессиональные и интерактивные визуализации данных, доступные для широкой аудитории.

Далее приведены примеры программ по построению различных видов графиков с пояснениями и результатами выполнения рис.3.2.5-3.2.6.

import plotly.graph\_objs as go

# Создаем объект фигуры

fig = go.Figure()

# Данные для построения

x = [1974, 1987, 1999, 2012, 2022, 2046]

y = [4, 5, 6, 7, 8, 9]

# Собираем точки и объединяем их линиями

fig.add\_trace(go.Scatter(x=x, y=y))

# Настройки окна

fig.update\_layout(legend\_orientation="h",

title="The number of the world's population by year",

xaxis\_title="year",

yaxis\_title="population",

margin=dict(l=0, r=0, t=30, b=0))

fig.show()

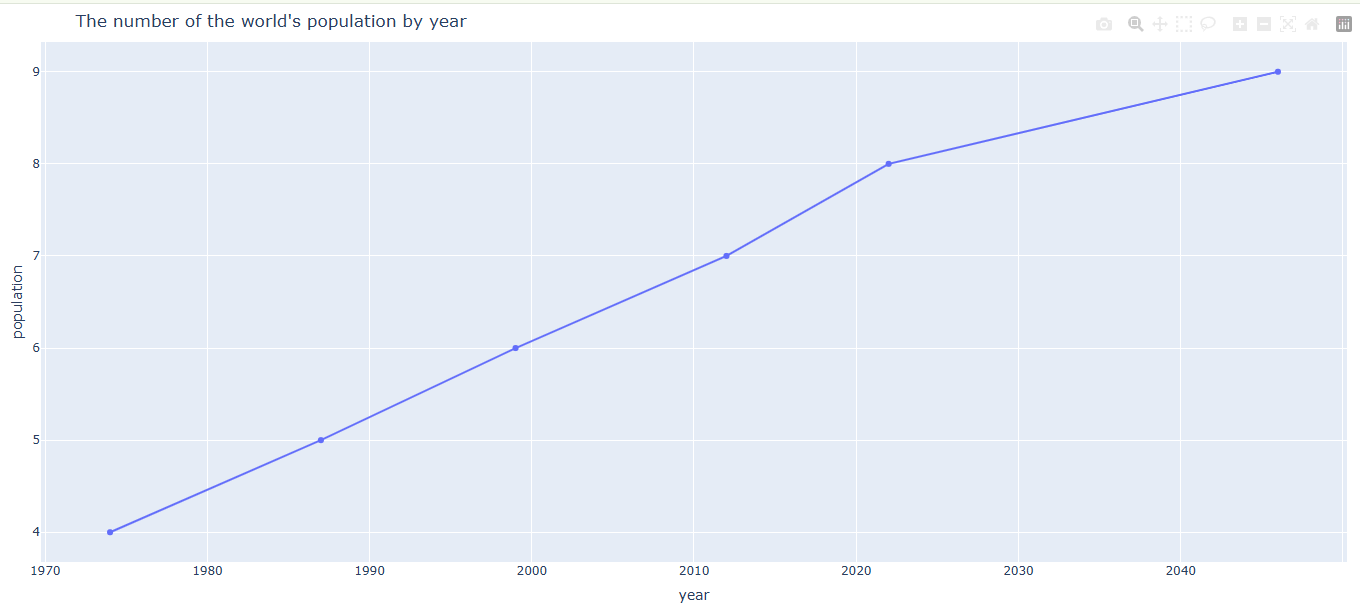


Рисунок 3.2.5. Пример линейного графика, построенного при помощи Plotly.

Гистограмма.

import numpy as np

import plotly.express as px

# Генерация случайных данных с нормальным распределением

data = np.random.normal(loc=0, scale=1, size=1000)

# Создание гистограммы с помощью Plotly

fig = px.histogram(data, nbins=30, title='Histogram of the Gaussian distribution')

# Отображение графика

fig.show()

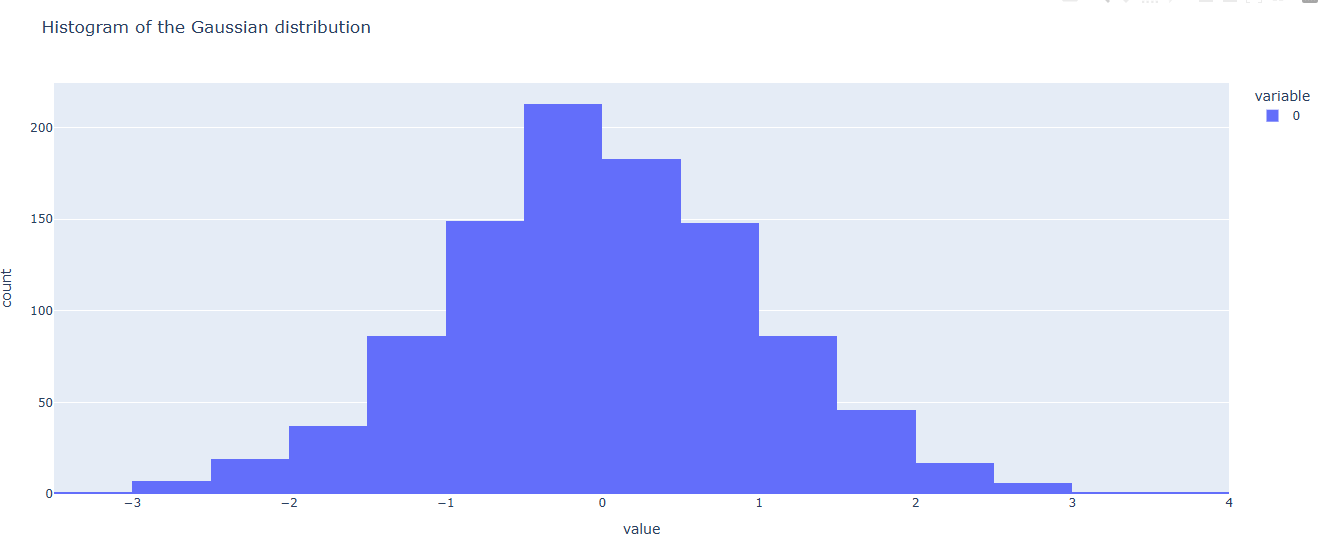


Рисунок 3.2.6. Пример гистограммы, построенной при помощи Plotly.

Выбор библиотеки Matplotlib для визуализации графиков в приложении обоснован несколькими причинами:

1. Широкие возможности: Matplotlib предлагает обширный набор инструментов для создания различных типов графиков, что позволяет легко визуализировать данные в разных форматах.
2. Гибкость: Библиотека предоставляет возможность настраивать графики на уровне деталей, включая цвета, стили линий, метки и аннотации, что дает возможность создавать уникальные визуализации.
3. Сообщество и поддержка: Matplotlib обладает большим и активным сообществом, а также обширной документацией и примерами, что облегчает решение возникающих вопросов и обучение.
4. Совместимость: Библиотека хорошо интегрируется с другими инструментами и библиотеками Python, такими как NumPy, Pandas и главное Tkinter, что делает ее идеальным выбором для аналитических приложений.
5. Простота использования: Основные функции библиотеки легко освоить, что позволяет быстро начать создание графиков, особенно для пользователей с базовыми знаниями Python.

Эти факторы делают Matplotlib надежным инструментом для визуализации данных и создания графиков в приложении, что позволяет эффективно представлять информацию пользователям.

# 4. Разработка приложения

В работе произведен обзор разных библиотек, из которых для написания проекта были выбраны следующие:

Основные:

* Matplotlib 3.9.2
* Tkinter
* Math

Для обзора библиотек:

* numpy 1.26.4
* matplotlib 3.9.2
* pandas 2.2.2
* plotly 5.22.0

На основе представленных библиотек было написано приложение, которое отображает график, заданный пользователем. Далее приведено описание программы.

На рис. 4.1, приведена структура проекта, который содержит следующие файлы и папки:

* Папки build и dist, были созданы на основе остального проекта и хранят исполняемый файл и данные для него.
* В папке images находятся изображения используемые в проекте
* В папке Matplot находятся файлы для создания графиков, используемых в обзоре библиотек
* В папке Plotly находятся файлы для создания графиков, используемых в обзоре библиотек
* В папке Seaborn находятся файлы для создания графиков, используемых в обзоре библиотек
* В паке menu находятся файлы панели меню:
  + about.py – выводит информацию о программе
  + help.py – выводит информацию “помощник”, о том как взаимодействовать с приложением
* Файл inp.py обрабатывает входные значения, принимает их или предает информацию о том, что входные данные не корректны. Так же реализован механизм, который позволяет строить графики математических функций
* Файл mplib.py строит график
* Файл out.py подготавливает данные для постройки графика
* Файл win\_menu.py создает панель меню на основном экране
* Файл window.py создает основное рабочее окно

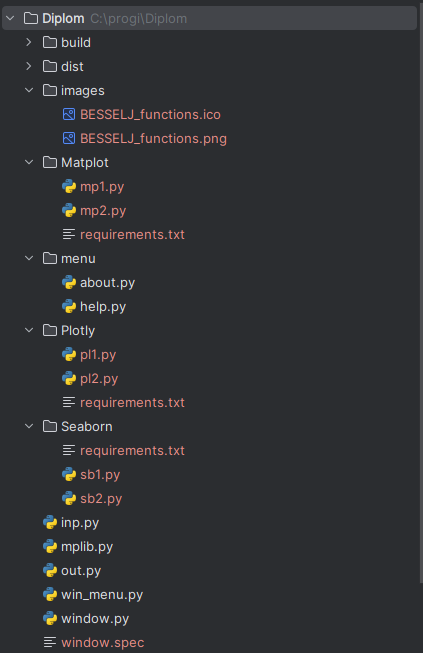


Рисунок 4.1. Структура проекта.

Далее подробно разберем логику работы приложения.

При запуске исходного фала window.exe открывается основное окно программы рис. 4.2.

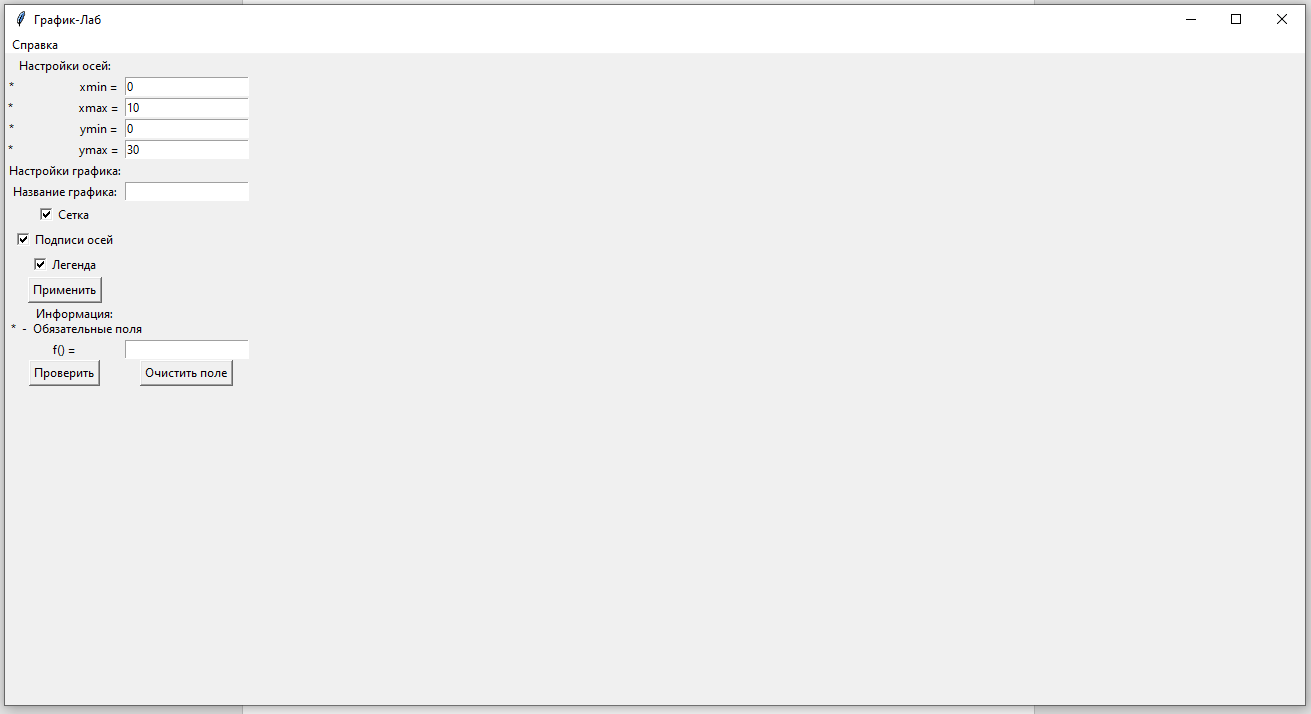


Рисунок 4.2. Основное окно программы.

В данном окне первый блок рис. 4.3, отведенный под настройки графика, а именно:

* xmin – минимальное значение абсциссы, от которого будет строиться график
* xmax - максимальное значение абсциссы, до которого будет строиться график
* ymin – минимальное значение по оси ординат, если точка выходит за данное значение, то она не будет отображена на графике
* ymax – максимальное значение по оси ординат, если точка выходит за данное значение, то она не будет отображена на графике
* Название графика – туда можно ввести любой текст, и он отобразиться как название графика
* Сетка - если данная функция активна, то на график добавится сетка
* Подписи осей – если данная функция активна, то на графике будут подписаны оси
* Легенда – если данная функция активна, то на графике будет выводится легенда
* Кнопка «Применить» - сохраняет настройки и выводит информационное сообщение. Если все настройки введены корректно, то появится сообщение «Настройки сохранены» иначе «Не все обязательные поля заполнены»

Примечание:

1. На рис. 4.3 приведены настройки по умолчанию, если изменение настроек не нужно можно просто проигнорировать данный блок.
2. В полях ввода xmin, xmax, ymin, ymax реализованы функции не позволяющие вводить туда нечисловые значения.

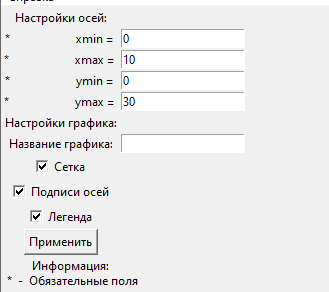


Рисунок 4.3. Блок настроек графика.

Во втором блоке рис.4.4 происходит ввод пользователем функции, ее проверка и запуск построения графика. Рассмотрим данный блок поближе:

* В первом окне пользователь может ввести функцию
* После чего необходимо нажать кнопку «Проверить». Если данные введены корректно, то в поле «Информация» отобразится введённая функция, иначе высветится информация «Упс, что то пошло не так... Проверьте функцию.»
* Если функция введена правильно, то после нажатия кнопки «Проверить», появится таблица nх3 (n – количество переменных в заданной функции)
  + В первом столбце находится название переменной
  + Во втором столбце можно ввести ее значение
  + В третьем столбце можно выбрать относительно какой переменной будет построен график
* Далее при нажатии на кнопку «Построить график» запускается механизм построения графика с учетом всех введенных параметров
* Если необходимо отобразить несколько графиков в одном окне, то нужно проделать все те же действия, но в конце нажать кнопку «Добавить график»

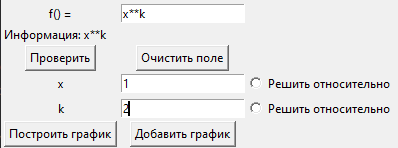


Рисунок 4.4. Блок ввода функции.

Примечание:

1. В полях ввода значений параметров, реализованы функции не позволяющие вводить туда нечисловые значения.
2. Можно выбрать только один параметр относительно которого будет строится график. При попытке выбрать несколько параметров будет произведено переключение между ними, что позволяет изменить выбор.
3. После выбора параметра окно ввода значения для него скрывается.
4. Если не был выбран не один параметр, то программа посчитает значение введенной функции и построит прямую с этим значением.

В результате выполнения программы, получается третий блок рис.4.5:

* В верхней части данного блока располагается построенный график
* В нижней стандартная панель NavigationToolbar2Tk, которая позволяет:
  + Домик – возвращает графика в окне к начальному положению
  + Стрелка влево - передвигает график в предыдущее положение
  + Стрелка в право – передвигает график в следующие положение
  + Крест из стрелок – позволяет передвигать график в окне в произвольное положение
  + Лупа – выделяет участок на графике и увеличивает его
  + Бегунки – данная кнопка открывает меню настроек окна и позволяет изменить его размеры

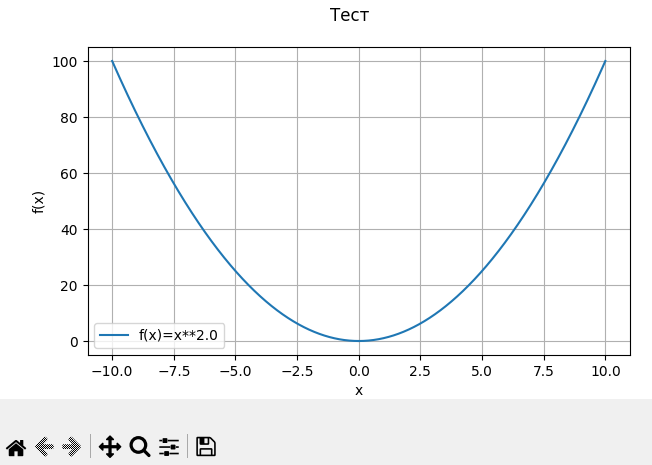


Рисунок 4.5. Блок с графиком.

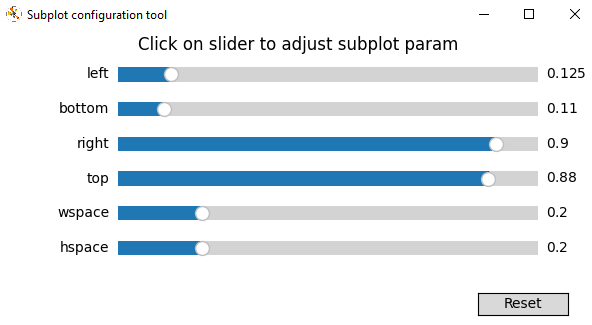


Рисунок 4.6. Настройка окна графика.

На рис. 7 приведен результат работа программы для построения трех графиков х^2, х^-1 и 2^k при заданных условиях табл.1

Таблица 1. Условия, введенные пользователем

|  |  |
| --- | --- |
| Параметр | Значение |
| Xmin | -10 |
| Xmax | 10 |
| Ymin | -30 |
| Ymax | 100 |
| Сетка | Включено |
| Подписи осей | Включено |
| Легенда | Включено |

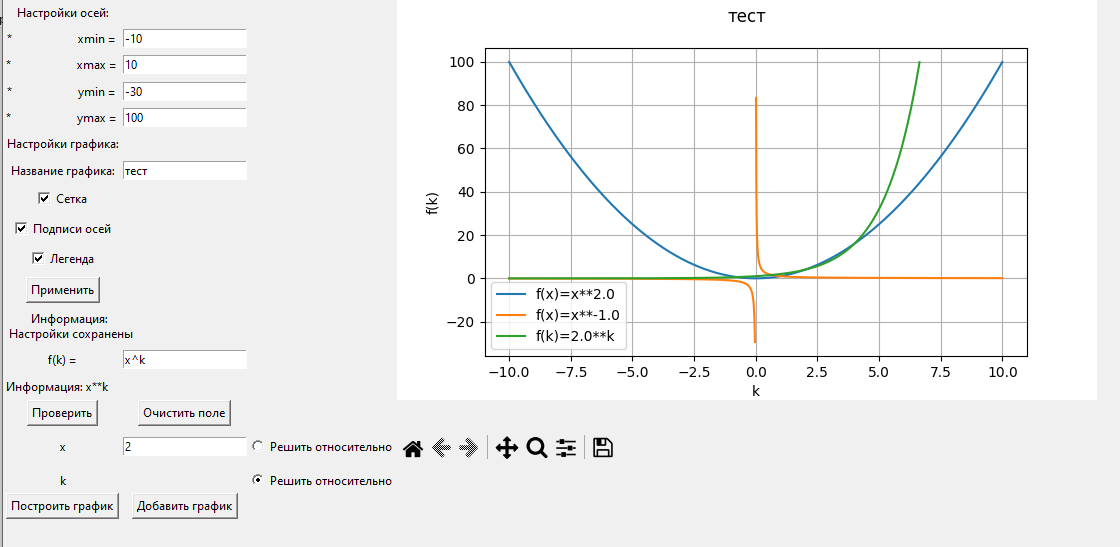


Рисунок 4.7. Пример работы программы

Примечание:

1. График функции f(k) – построен не до конца, он обрывается примерно в точке к=6,6. Это произошло по тому, что график ограничен сверху уровнем 100, а все остальные значения превышают этот уровень.
2. На рис. 7 показан результат разрыва графика (х^-1)(в стандартном исполнении matplotlib соединяет все точки)

Для удобства использования программы внедрено меню «Справка» рис.8, при выборе которого открываться подпункты «Помощь» и «О программе».

При выборе пункта «Помощь» открывается окно рис.9 с пошаговым описанием работы с приложением, а также описанием каретного ввода математических функций.

При выборе пункта «О программе» открывается окно рис.10 с общей информацией о приложении.

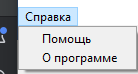


Рисунок 4.8. Общий вид меню.

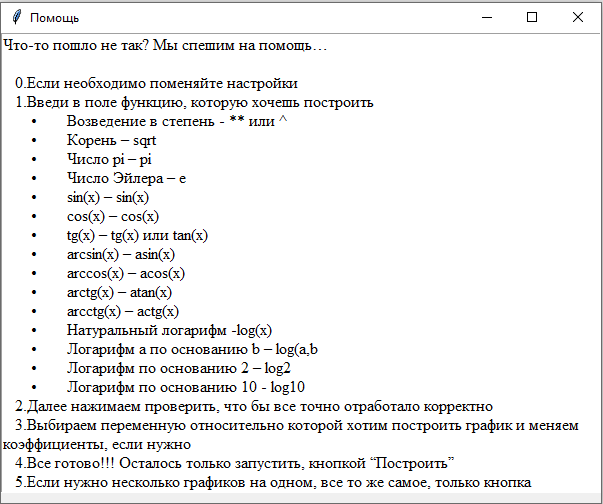


Рисунок 4.9. Окно «Помощь»

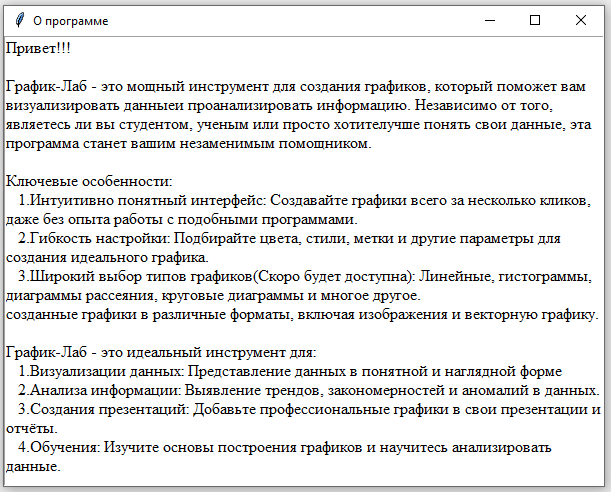


Рисунок 4.10. Окно «О программе»

# Направления развития

В этой главе сосредоточимся на направлениях развития приложения по созданию графиков, исследуя возможности для улучшения визуализации данных и повышения функциональности. Мы рассмотрим актуальные потребности пользователей и выделим ключевые особенности, которые смогут сделать наше приложение более удобным и эффективным. Особое внимание будет уделено интеграции новых инструментов и технологий, таких как интерактивные элементы и расширенные аналитические функции, что позволит пользователям легко и быстро создавать наглядные графики для анализа данных.

Основными пользователями приложений по построению графиков, являются люди, работающие с аналитикой. Исходя из этого можно было бы добавить в программу аналитические метрики.

Для расширения возможностей данной программы необходимо дополнить ее разными типами графиков, такими как гистограмма, круговая диаграмма, точечный график и 3D графика. А также возможностью загружать данные из других ресурсов.

Для удобства пользователей возможна переработка дизайна, в более интуитивно понятный и удобный с разработкой собственной цветовой палитры.

Резюмируя все вышеперечисленное, можно выделить три основных направления развития:

1. Добавление аналитических метрик
2. Добавление новых типов графиков
3. Доработка дизайна

# Заключение

В заключении нашей работы мы подводим итоги исследования программы для создания графиков, подчеркнув ее значимость в визуализации данных и упрощении аналитических процессов. Мы рассмотрели ключевые функциональные возможности, которые делают приложение удобным и эффективным для пользователей из различных областей. Также были выявлены направления для дальнейшего развития, включая интеграцию новых технологий и улучшение пользовательского интерфейса. Важно отметить, что успешная реализация предложенных улучшений позволит не только повысить конкурентоспособность приложения, но и значительно расширить его аудиторию. Таким образом, программа для создания графиков имеет все шансы стать незаменимым инструментом для анализа и представления данных в современном мире.