Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
**дисциплины**
**«Искусственный интеллект и машинное обучение»**
**Вариант 9**

Выполнил:
Кравчук Мирослав Витальевич
2 курс, группа ИТС-б-о-23-1,
11.03.02«Инфокоммуникационные технологии и системы связи», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

**Тема: Основы работы с библиотекой matplotlib**

**Цель:** исследовать базовые возможности библиотеки matplotlib языка программирования Python

**Ссылка на GitHub**: https://github.com/miron2314/DLab-3.git

**Порядок выполнения работы:**

1.Изучил теоретический материал работы.

2.Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования.

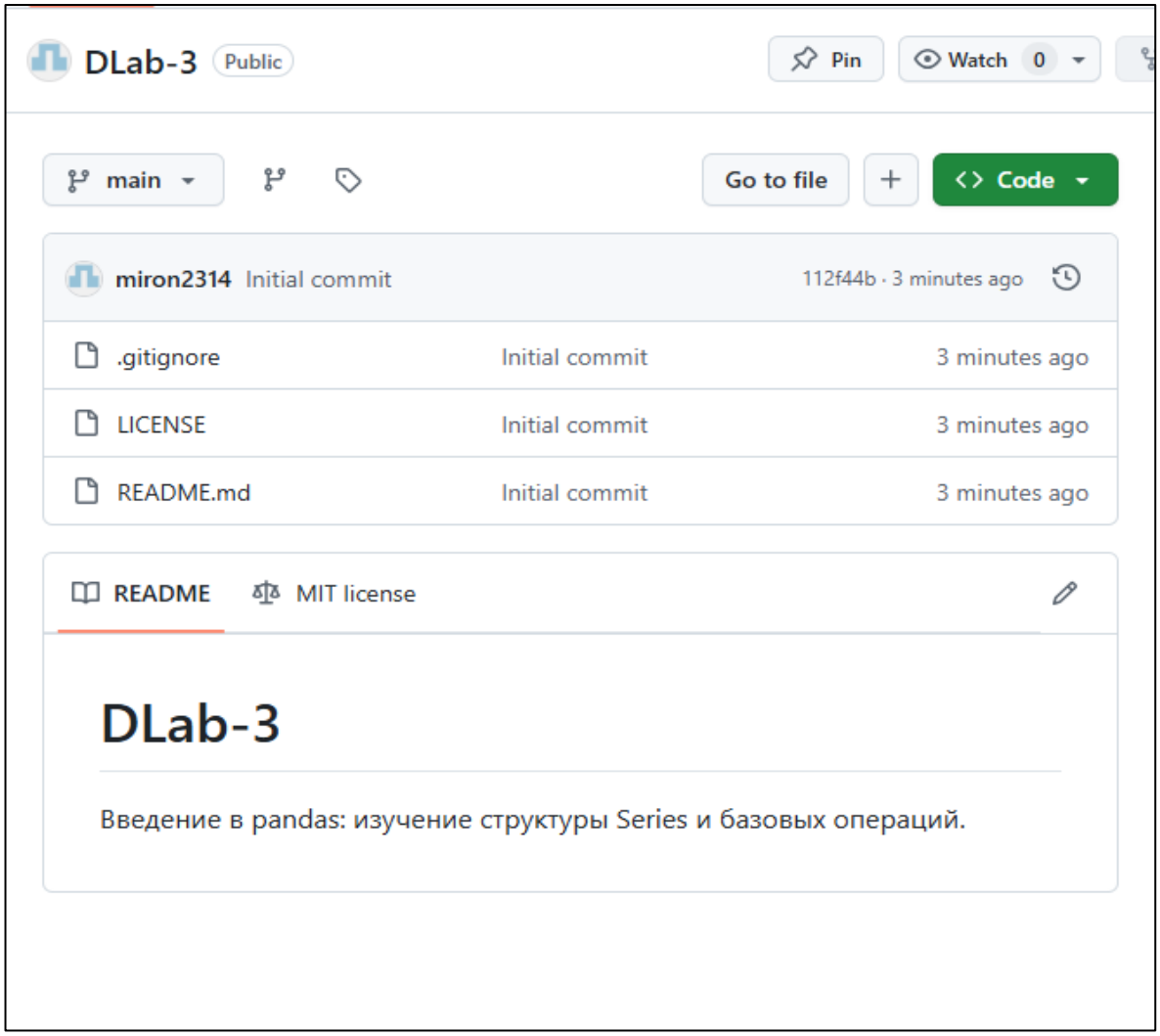


Рисунок 1. Репозиторий

3.Выполнил клонирование репозитория.

```
PS C:\Users\USER> git clone https://github.com/miron2314/DLab-3.git
Cloning into 'DLab-3'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
Receiving objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
PS C:\Users\USER>
```

Рисунок 2. Клонирование

4.Проработал примеры работы.

```python
#Построение графика
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 50)
y = x
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x786615cb8290>]
```

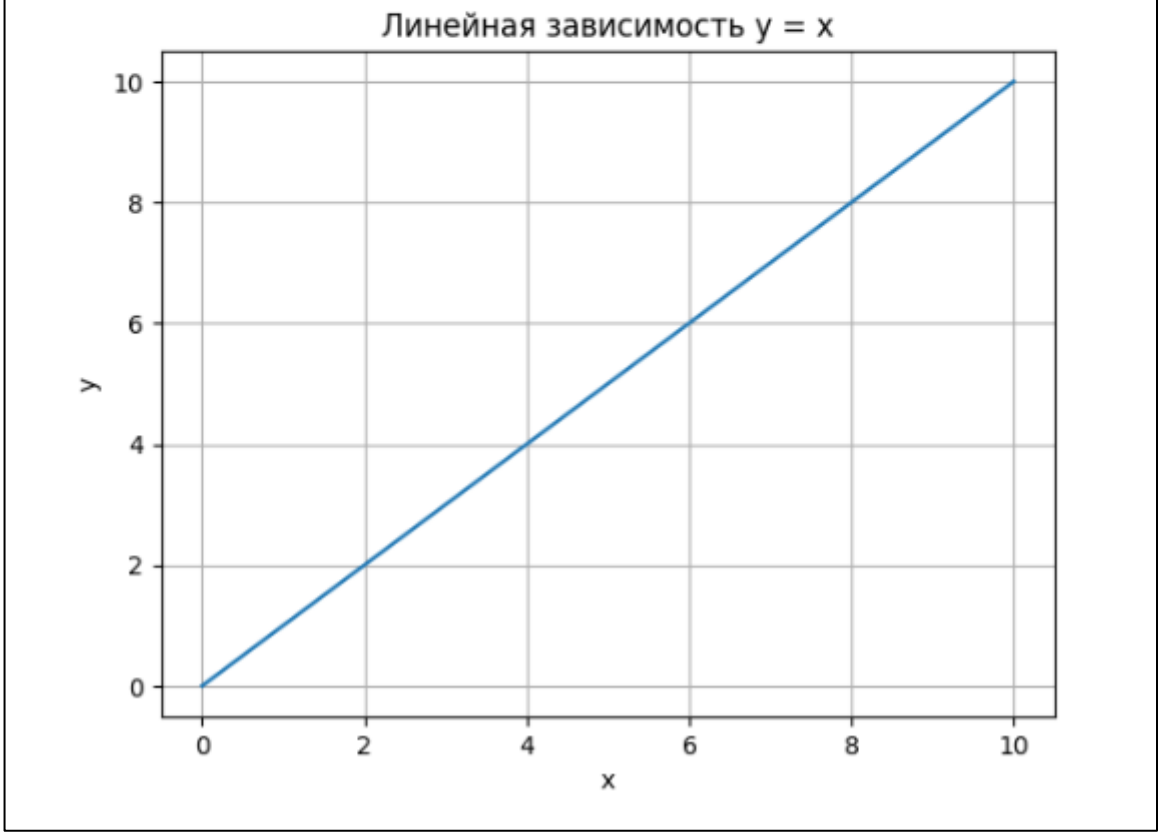


Рисунок 3. Проработка примеров

```python
#Несколько графиков на одном поле
import matplotlib.pyplot as plt

# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = x**2

# Построение графика
plt.title("Зависимости: y1 = x, y2 = x^2")  # заголовок
plt.xlabel("x")  # ось абсцисс
plt.ylabel("y")  # ось ординат
plt.grid()  # включение отображения сетки
plt.plot(x, y1, label='y1 = x')  # построение графика для y1
plt.plot(x, y2, label='y2 = x^2')  # построение графика для y2
plt.legend()  # добавление легенды
plt.show()  # отображение графика
```

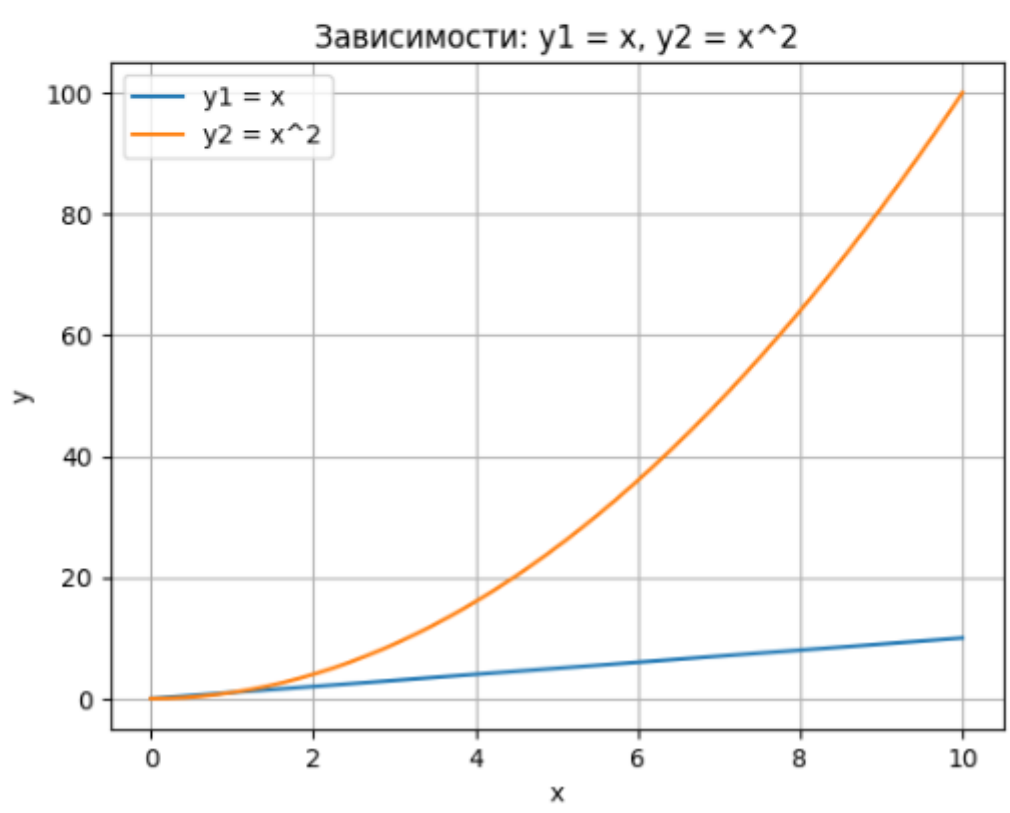


Рисунок 4. Проработка примеров

```python
#Несколько разделенных полей с графиками
import matplotlib.pyplot as plt

# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# Построение графиков
plt.figure(figsize=(9, 9))
plt.subplot(2, 1, 1)
plt.plot(x, y1) # построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.ylabel("y1", fontsize=14) # ось ординат
plt.grid(True) # включение отображение сетки
plt.subplot(2, 1, 2)
plt.plot(x, y2) # построение графика
plt.xlabel("x", fontsize=14) # ось абсцисс
plt.ylabel("y2", fontsize=14) # ось ординат
plt.grid(True)
```

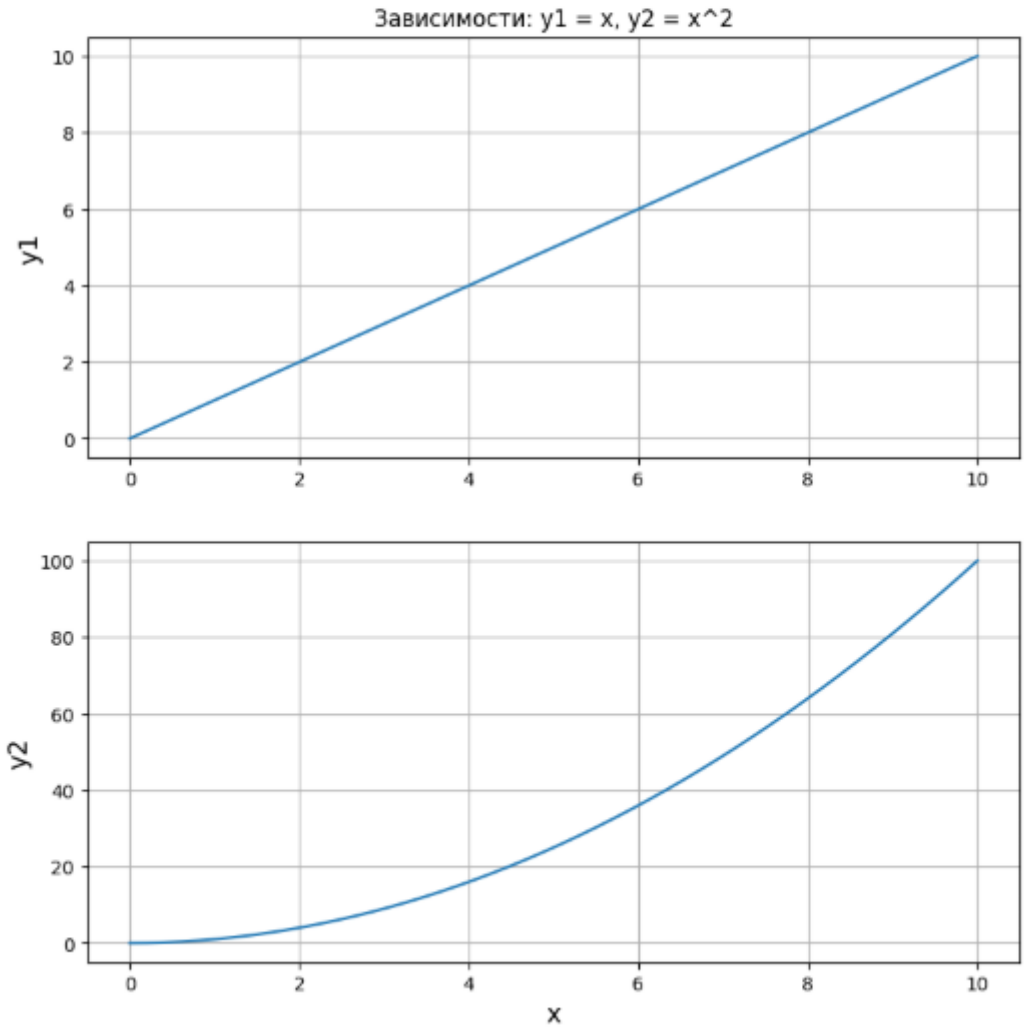


Рисунок 5. Проработка примеров

```python
#Построение диаграммы
import matplotlib.pyplot as plt

fruits = ["apple", "peach", "orange", "bannana", "melon"]
counts = [34, 25, 43, 31, 17]
plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```

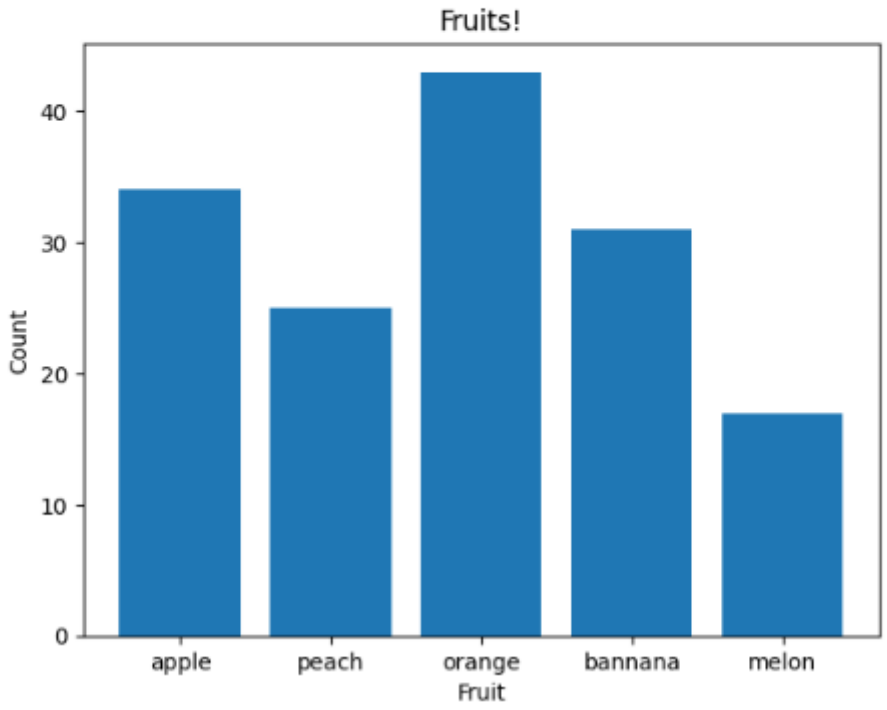


Рисунок 6. Проработка примеров

```
[ ] import matplotlib.pyplot as plt

    x = [1, 5, 10, 15, 20]
    y1 = [1, 7, 3, 5, 11]
    y2 = [4, 3, 1, 8, 12]
    plt.figure(figsize=(12, 7))
    plt.plot(x, y1, 'o-r', alpha=0.7, label="first", lw=5, mec='b', mew=2,
    ms=10)
    plt.plot(x, y2, 'v-.g', label="second", mec='r', lw=2, mew=2, ms=12)
    plt.legend()
    plt.grid(True)
```

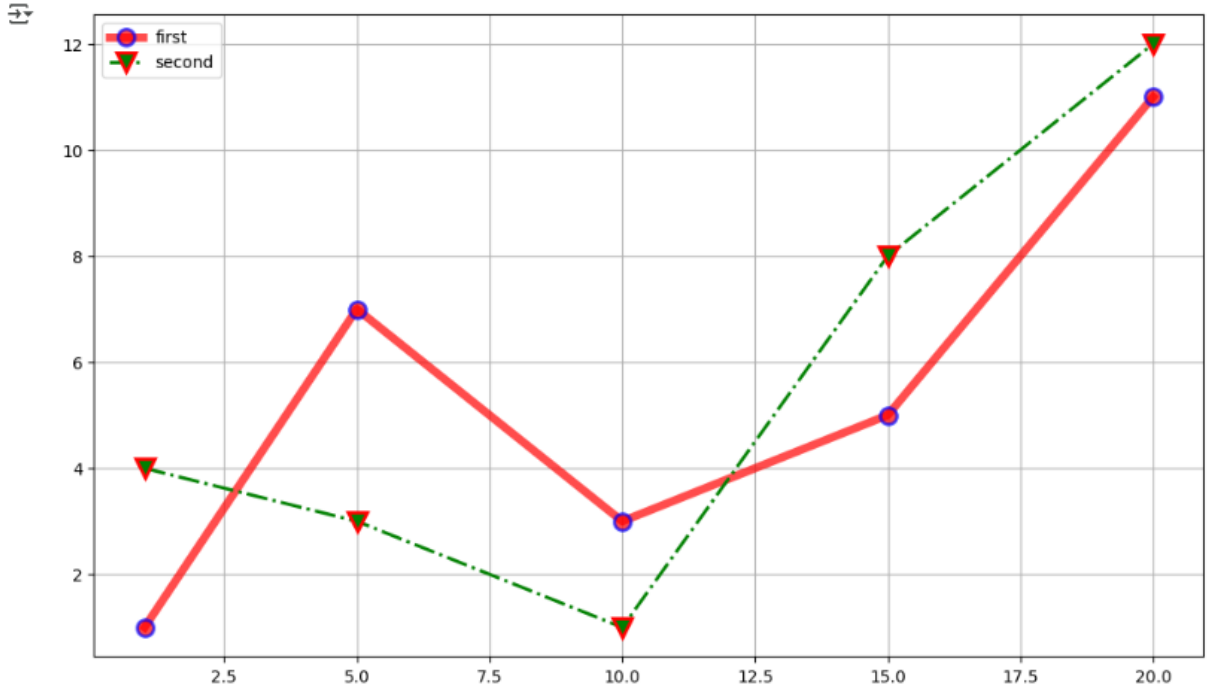


Рисунок 7. Проработка примеров

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)
plt.plot(x, y, c = "r")
plt.fill_between(x, y)
```

```
<matplotlib.collections.FillBetweenPolyCollection at 0x78661546f950>
```

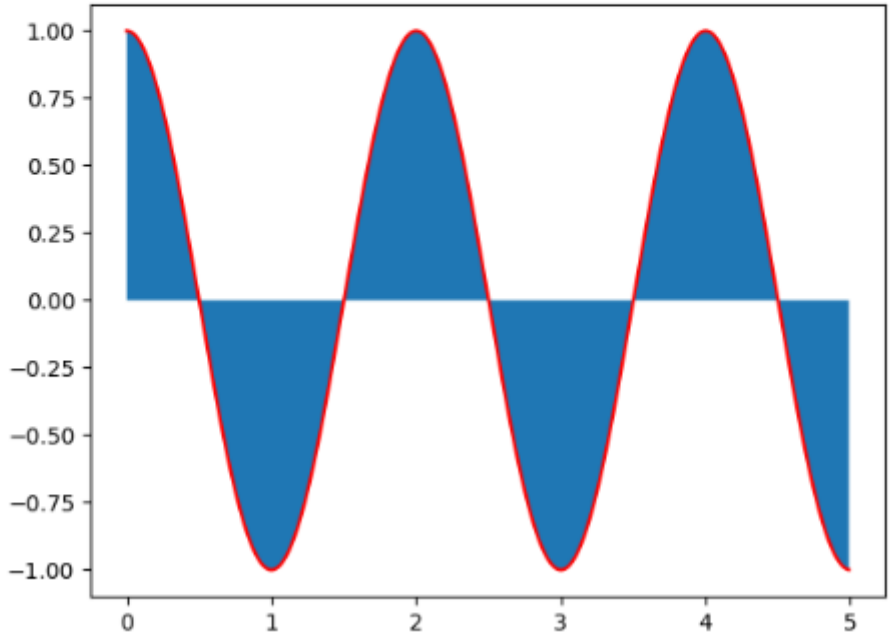


Рисунок 8. Проработка примеров

```
x = [1, 2, 3, 4, 5, 6, 7]
y = [7, 6, 5, 4, 5, 6, 7]
plt.plot(x, y, marker="o", c="g")
```

```
[<matplotlib.lines.Line2D at 0x786615551d10>]
```

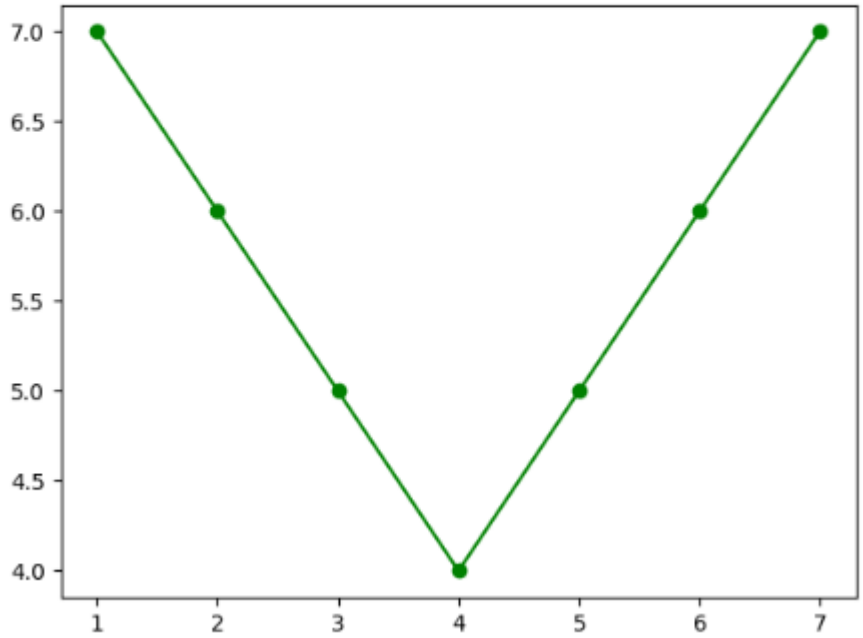


Рисунок 9. Проработка примеров

```
import numpy as np

x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)
plt.plot(x, y, marker="o", c="g")
```

```
[<matplotlib.lines.Line2D at 0x786615710550>]
```

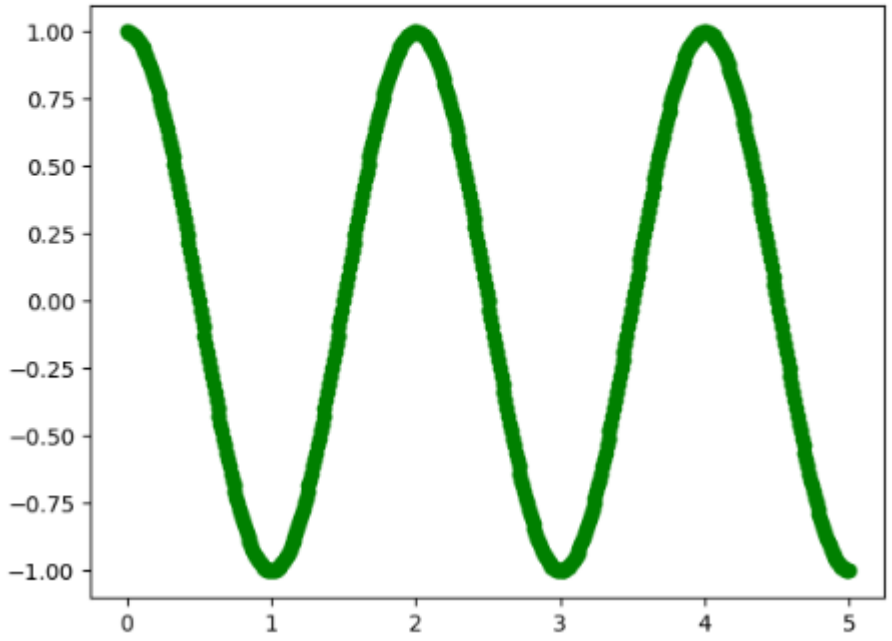


Рисунок 10. Проработка примеров

```python
#Круговые диаграммы
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BMV", "AUDI", "Jaguar"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis("equal")
```

```
(np.float64(-1.0999964468631331),
 np.float64(1.0999998308030063),
 np.float64(-1.0999996500566958),
 np.float64(1.0999997141645377))
```

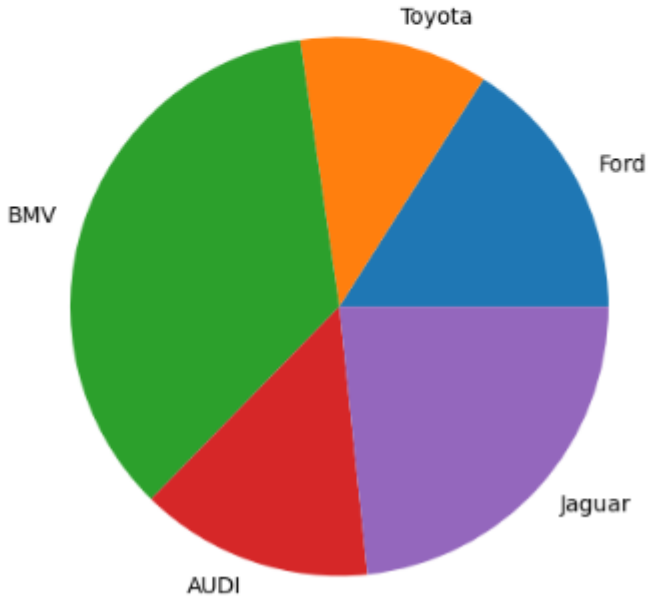


Рисунок 11. Проработка примеров

```
als = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BMV", "AUDI", "Jaguar"]
explode = (0.1, 0, 0.15, 0, 0)
fig, ax = plt.subplots()
ax.pie(vals, labels=labels, autopct='%1.1f%%', shadow=True,
explode=explode, wedgeprops={'lw':1, 'ls':'--','edgecolor':"k"},
rotatelabels=True)

ax.axis("equal")
```

```
(np.float64(-1.2541693795448878),
 np.float64(1.199144954675332),
 np.float64(-1.1017809084520842),
 np.float64(1.137406140467696))
```

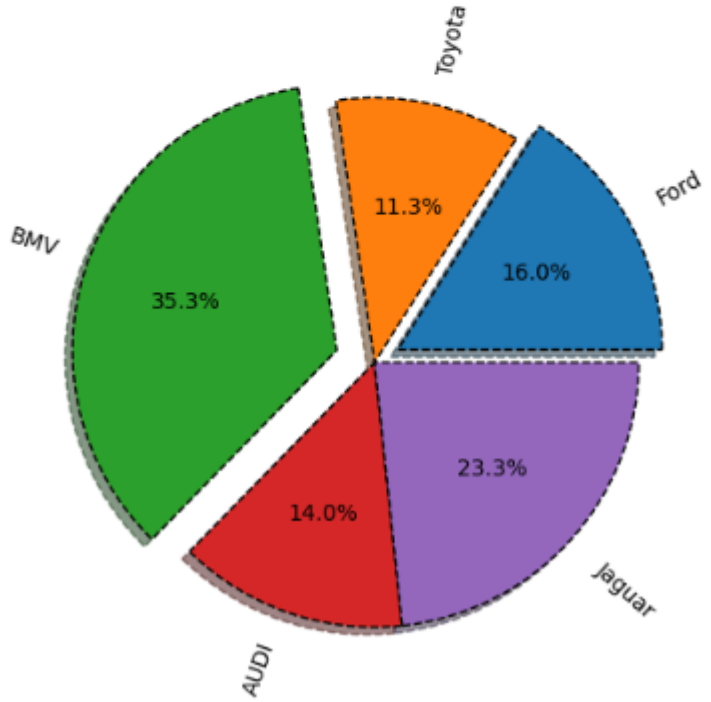


Рисунок 12. Проработка примеров

```
fig, ax = plt.subplots()
offset=0.4
data = np.array([[5, 10, 7], [8, 15, 5], [11, 9, 7]])
cmap = plt.get_cmap("tab20b")
b_colors = cmap(np.array([0, 8, 12]))
sm_colors = cmap(np.array([1, 2, 3, 9, 10, 11, 13, 14, 15]))
ax.pie(data.sum(axis=1), radius=1, colors=b_colors,
wedgeprops=dict(width=offset, edgecolor='w'))
ax.pie(data.flatten(), radius=1-offset, colors=sm_colors,
wedgeprops=dict(width=offset, edgecolor='w'))
```

```
([<matplotlib.patches.Wedge at 0x78661591a690>,
  <matplotlib.patches.Wedge at 0x7866161d2550>,
  <matplotlib.patches.Wedge at 0x7866161d2850>,
  <matplotlib.patches.Wedge at 0x7866161dbb10>,
  <matplotlib.patches.Wedge at 0x7866161da790>,
  <matplotlib.patches.Wedge at 0x786616278c10>,
  <matplotlib.patches.Wedge at 0x7866162787d0>,
  <matplotlib.patches.Wedge at 0x7866161a3150>,
  <matplotlib.patches.Wedge at 0x7866161da990>],
 [Text(0.6463143432704803, 0.13370777719657054, ''),
  Text(0.4521935101701442, 0.48075048555358085, ''),
  Text(0.040366619514524194, 0.6587644010030974, ''),
  Text(-0.3454229285629237, 0.5623904341496336, ''),
  Text(-0.6578039143572915, 0.053796005950489834, ''),
  Text(-0.4898743406367375, -0.44229303678186227, ''),
  Text(-0.1204957005984727, -0.6489073787046062, ''),
  Text(0.3901135986261199, -0.5323639546090425, ''),
  Text(0.6332654137702297, -0.18594331319630678, '')])
```

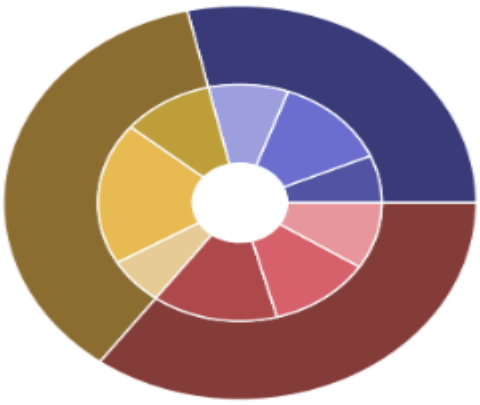

Рисунок 13. Проработка примеров

```
np.random.seed(19680801)
data = np.random.randn(25, 25)
plt.imshow(data)
```

```
<matplotlib.image.AxesImage at 0x78661596bfd0>
```

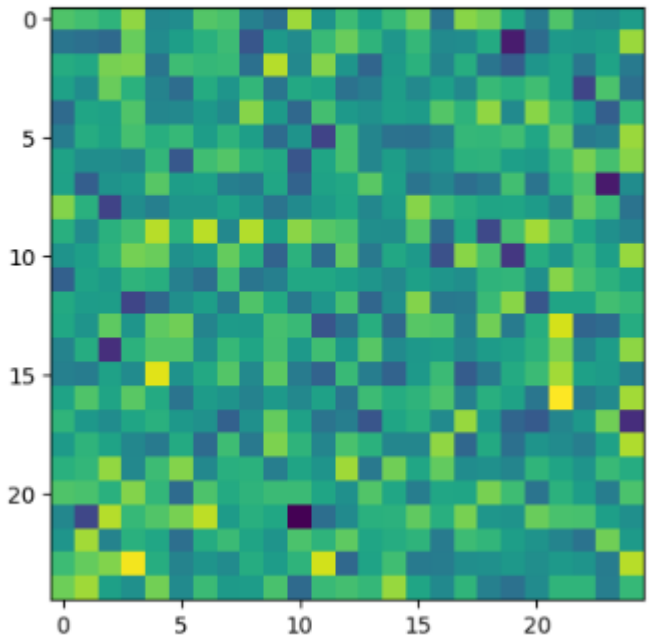


Рисунок 14. Проработка примеров

```
np.random.seed(123)
data = np.random.rand(5, 7)
plt.pcolormesh(data, cmap='plasma', edgecolors="k", shading='flat')
```

```
<matplotlib.collections.QuadMesh at 0x7866153b8c50>
```

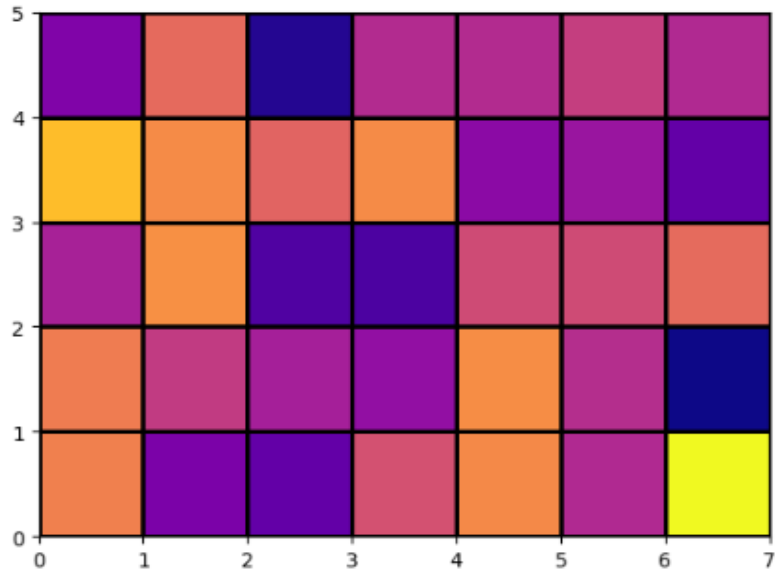


Рисунок 15. Проработка примеров

```
x = np.linspace(-np.pi, np.pi, 50)
y = x
z = np.cos(x)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='parametric curve')
```

```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x7866156a8fd0>]
```

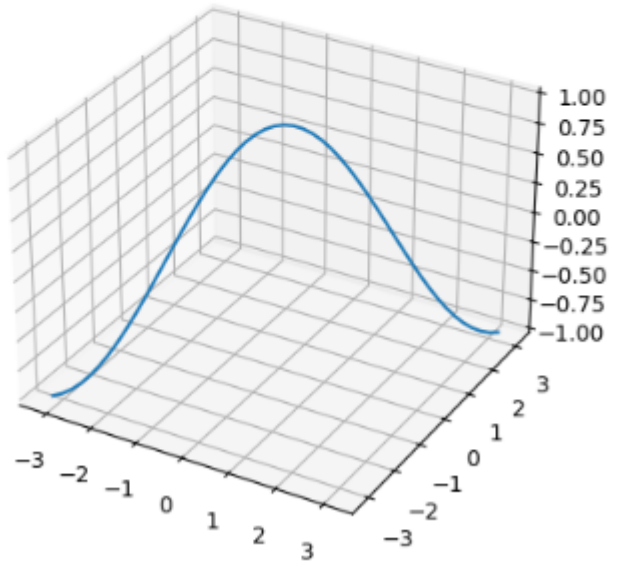


Рисунок 16. Проработка примеров

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
ax.legend()
```

```
<ipython-input-36-df3be398868f>:8: UserWarning: No artists with labels f
  ax.legend()
<matplotlib.legend.Legend at 0x7866154b21d0>
```

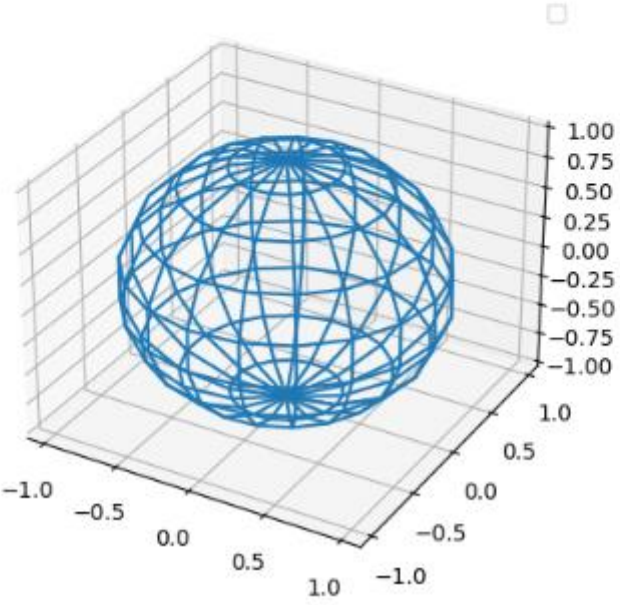


Рисунок 17. Проработка примеров

```
u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
x = np.cos(u)*np.sin(v)
y = np.sin(u)*np.sin(v)
z = np.cos(v)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='inferno')
ax.legend()
```

```
<ipython-input-37-8b200c6d2455>:8: UserWarning: No artists w
  ax.legend()
<matplotlib.legend.Legend at 0x786615d42b90>
```

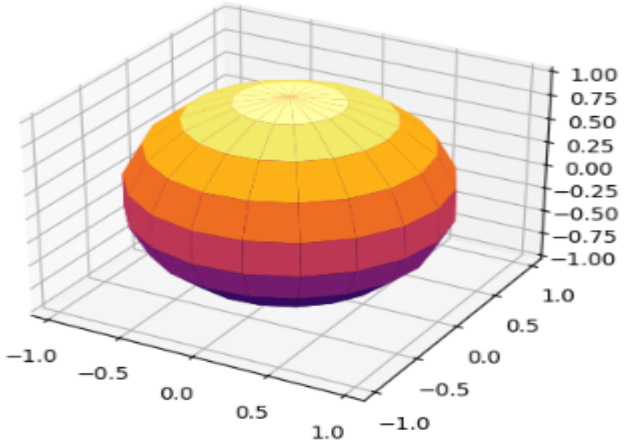


Рисунок 18. Проработка примеров

5.Выполнил практические задания.

**Задание 1.** Построение простого графика.

Напишите код, который строит график функции y = x2 на интервале [−10, 10]. Добавьте заголовок, подписи осей и сетку.

**Листинг кода:**

```
#Задание 1
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-10, 10)
y = x ** 2
plt.title(r"График для функции $y=x^2$")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.plot(x, y)
plt.show()
```

## 1.Построение простого графика

*Напишите код, который строит график функции y = x2 на интервале [−10, 10].*

*Добавьте заголовок, подписи осей и сетку.*

```
#Задание 1
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10, 10)
y = x ** 2
plt.title(r"График для функции $y=x^2$")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.plot(x, y)
plt.show()
```

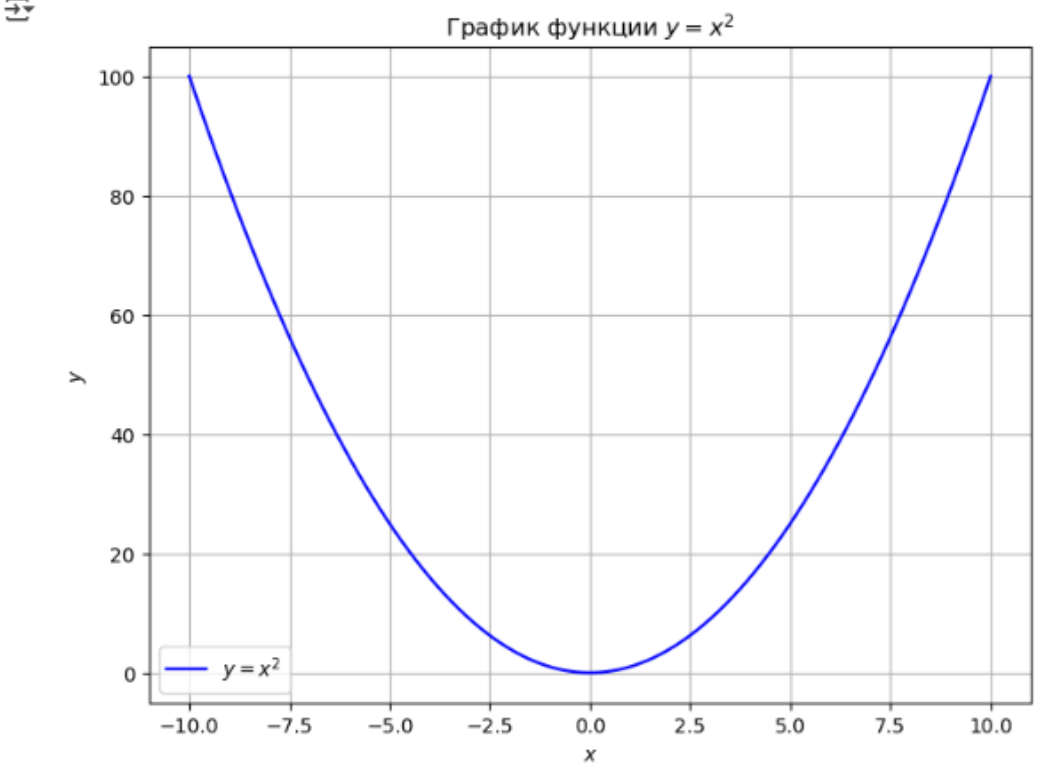


Рисунок 19. Выполнение задания 1

**Задание 2.** Настройка стилей и цветов.

Постройте три линии на одном графике:

y = x (синяя, пунктирная линия),

y = x2 (зеленая, штрихпунктирная линия),

y = x3 (красная, сплошная линия). Добавьте легенду и сделайте оси одинакового масштаба

**Листинг кода:**

```
#Задание 2
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(1, 3, 100)
y1 = x
y2 = x**2
y3 = x**3
plt.title(r"Графики для задания 2: $y=x$, $y=x^2$, $y=x^3$", fontsize=14)
plt.xlabel("x")
plt.ylabel("y")
plt.plot(x, y1, color = 'blue', linestyle = ':', label = r'$y=x$')
plt.plot(x, y2, color = 'green', linestyle = '-.', label = r'$y=x^2$')
plt.plot(x, y3, color = 'red', linestyle = '-', label = r'$y = x^3$')
plt.legend()
plt.axis('equal')
plt.show()
```

```python
#Задание 2
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(1, 3, 100)
y1 = x
y2 = x**2
y3 = x**3

plt.title(r"Графики для задания 2: $y=x$, $y=x^2$, $y=x^3$", fontsize=14)
plt.xlabel("x")
plt.ylabel("y")

plt.plot(x, y1, color = 'blue', linestyle = ':', label = r'$y=x$')
plt.plot(x, y2, color = 'green', linestyle = '-.', label = r'$y=x^2$')
plt.plot(x, y3, color = 'red', linestyle = '-', label = r'$y = x^3$')

plt.legend()

plt.axis('equal')

plt.show()
```

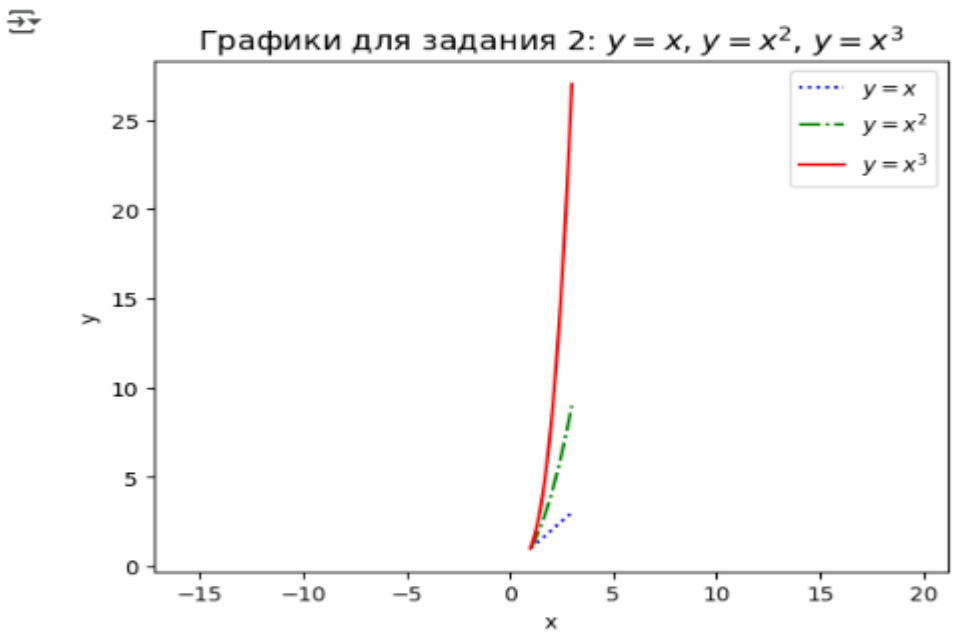


Рисунок 20. Практическое задание 2

**Задание 3.** Использование различных типов графиков

Сгенерируйте 50 случайных точек и постройте диаграмму рассеяния (scatter plot), где цвет точек зависит от их координаты по оси x, а размер точек зависит от координаты по оси y.

**Листинг кода:**

```
#Задание 3
import numpy as np
import matplotlib.pyplot as plt
```

```
x = np.random.randint(0, 10, 50)
y = np.random.randint(0, 10, 50)
colors = x
razmer = y * 20
plt.title('Задание 3')
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(x, y, s = razmer, c = colors)
plt.show()
```

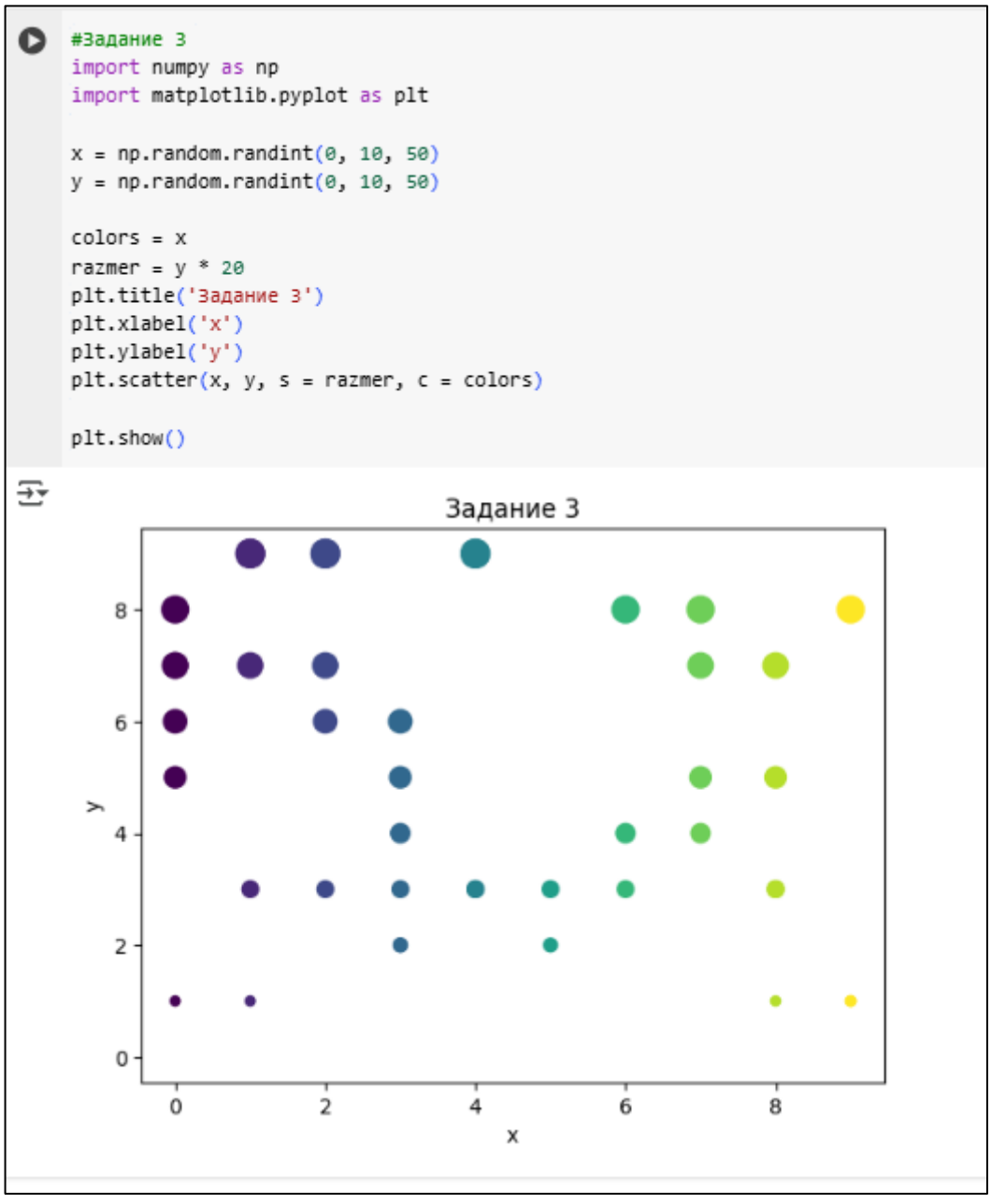


Рисунок 21. Практическое задание 3

**Задание 4.** Гистограмма распределения.

Сгенерируйте 1000 случайных чисел из нормального распределения с параметрами μ = 0, σ = 1 и постройте их гистограмму с 30 бинами. Добавьте вертикальную линию в среднем значении.

**Листинг кода:**

```
#Задание 4
import numpy as np
import matplotlib.pyplot as plt
mu = 0
sigma = 1
numbers = np.random.normal(mu, sigma, 1000)
sredn = np.mean(numbers)
plt.hist(numbers ,bins = 30, label = 'Гистограмма рассеивания')
plt.axvline(sredn, color = 'red', label = 'Среднее значение')
plt.title("Задание 4")
plt.xlabel('Значения')
plt.ylabel('Плотность вероятности')
plt.legend()
plt.show()
```

```python
#Задание 4
import numpy as np
import matplotlib.pyplot as plt

mu = 0
sigma = 1

numbers = np.random.normal(mu, sigma, 1000)
sredn = np.mean(numbers)

plt.hist(numbers ,bins = 30, label = 'Гистограмма рассеивания')
plt.axvline(sredn, color = 'red', label = 'Среднее значение')

plt.title("Задание 4")
plt.xlabel('Значения')
plt.ylabel('Плотность вероятности')

plt.legend()

plt.show()
```

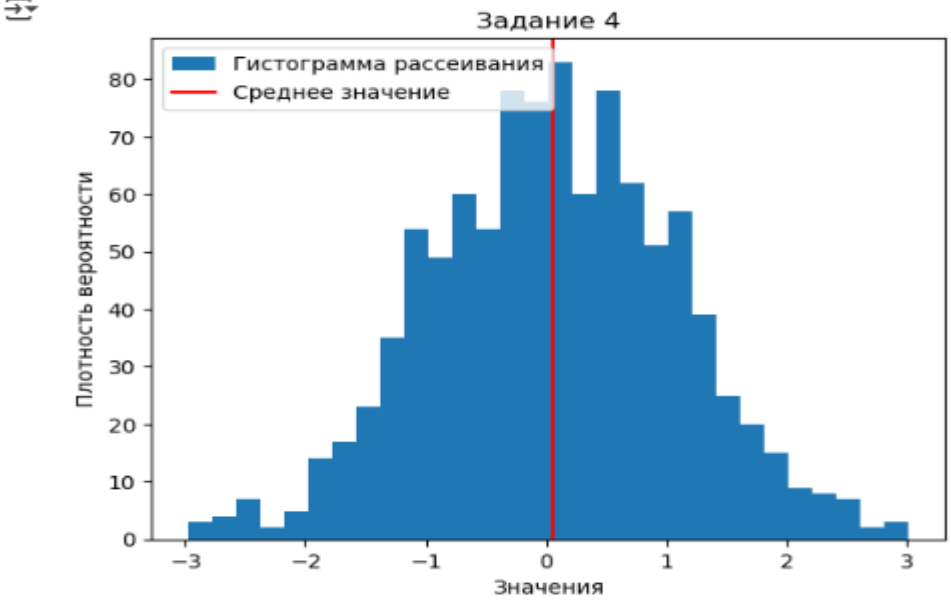


Рисунок 22. Практическое задание 4

**Задание 5.** Столбчатая диаграмма

Создайте столбчатую диаграмму, которая показывает количество студентов, получивших оценки:

"Отлично" — 20 человек,

"Хорошо" — 35 человек,

"Удовлетворительно" — 30 человек,

"Неудовлетворительно" — 15 человек.

Добавьте подписи к осям и заголовок.

**Листинг кода:**

```
#Задание 5
import matplotlib.pyplot as plt
ycheniki = [20, 35, 30, 15]
ocenki = ['Отлично', 'Хорошо', 'Удовлетворительно', 'Неудовлетворительно']
plt.figure(figsize=(9,4))
plt.title('Количество полученных оценок')
plt.xlabel('Полученная оценка')
plt.ylabel('Кол-во учеников')
plt.bar(ocenki,ycheniki, color='gray')
plt.show()
```

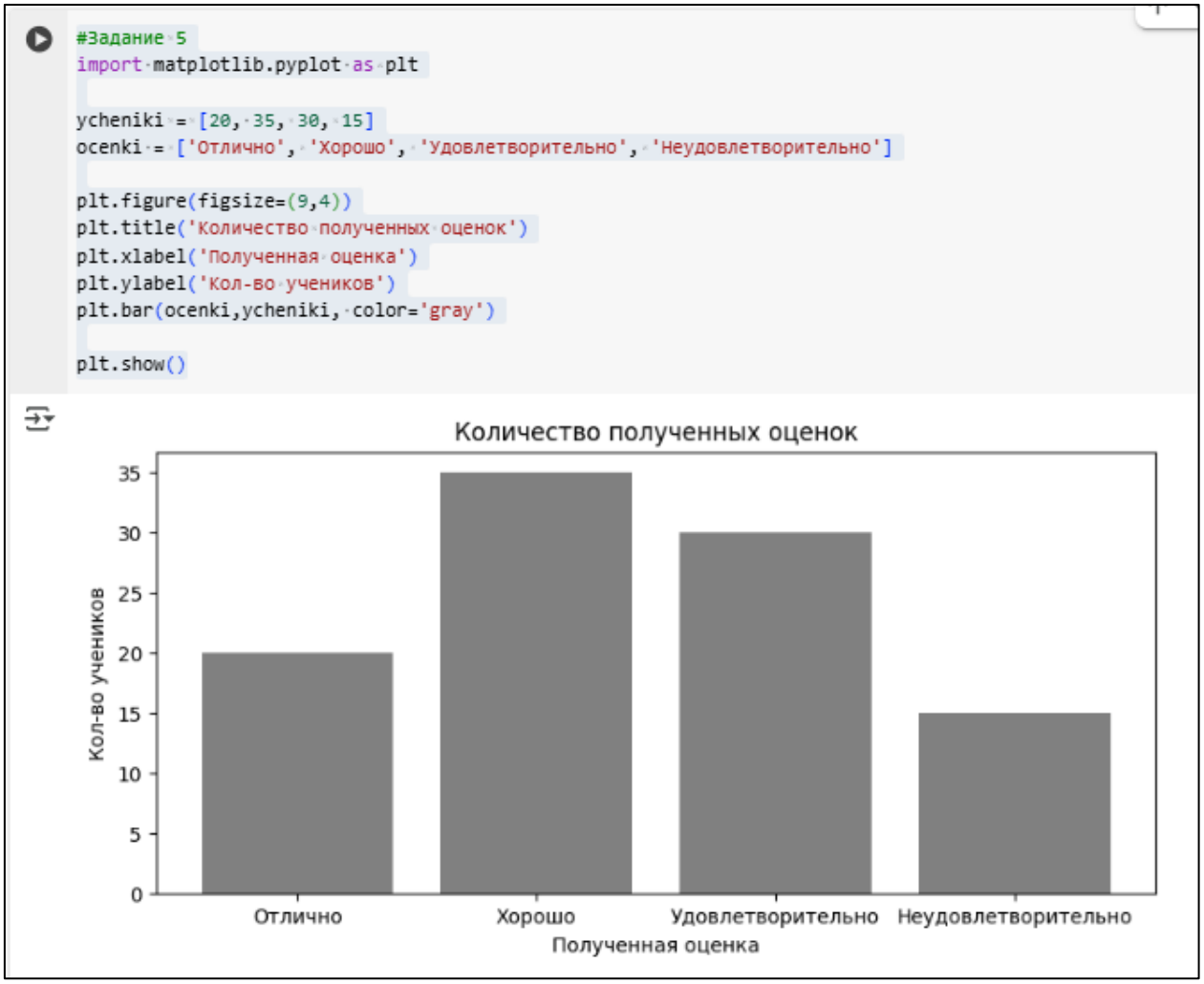


Рисунок 23. Практическое задание 5

**Задание 6.** Круговая диаграмма Используя данные предыдущей задачи, постройте круговую диаграмму с процентными подписями секторов.

**Листинг кода:**

```
#Задание 6
import matplotlib.pyplot as plt
ycheniki = [20, 35, 30, 15]
mylabels = ['Отлично', 'Хорошо','Удовлетворительно', 'Неудовлетворительно']
plt.figure(figsize=(9,4))
plt.title('Количество полученных оценок')
plt.pie(ycheniki, labels=mylabels, autopct='%1.1f%%')
plt.show()
```

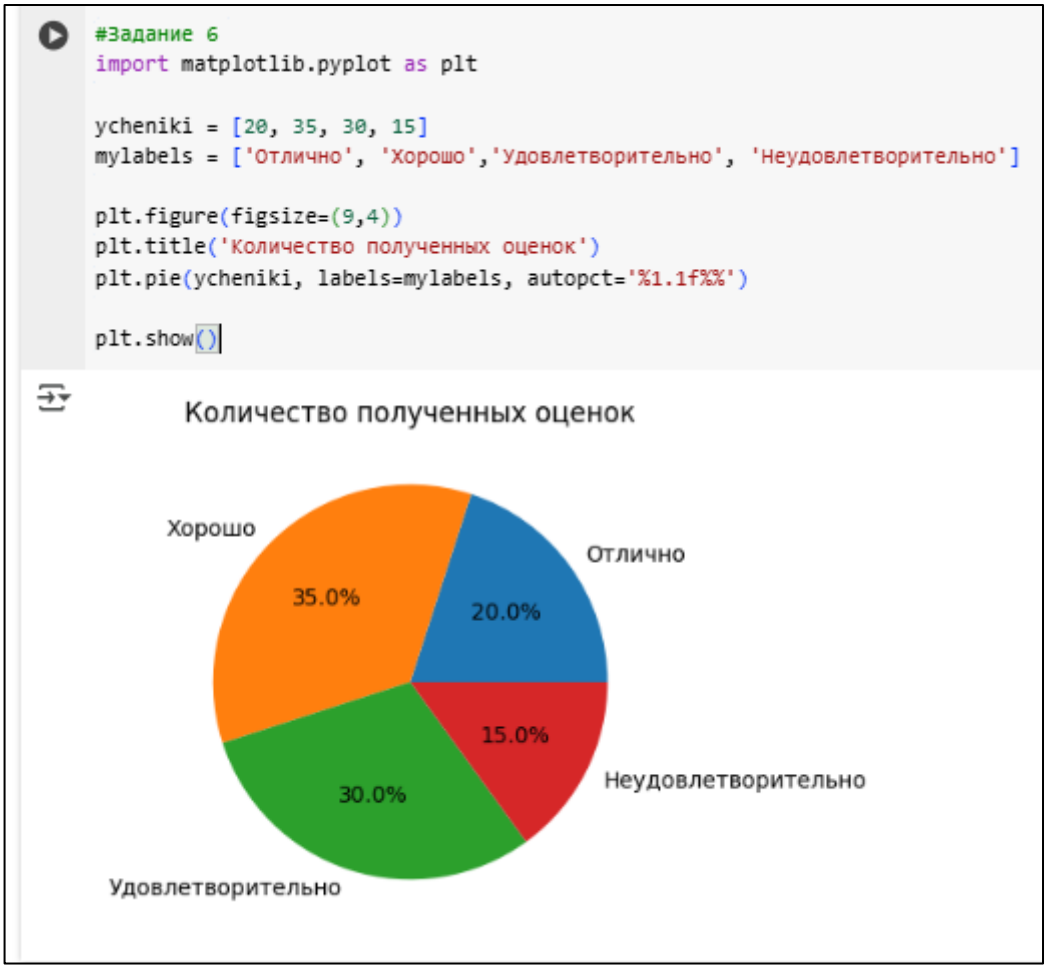


Рисунок 24. Практическое задание 6

**Задание 7.** Трехмерный график поверхности

Используя mpl_toolkits.mplot3d , постройте 3D-график функции z = sin(√x2 + y2) на сетке значений x, y в диапазоне [−5, 5].

**Листинг кода:**

```
#Задание 7
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
x = np.linspace(-5, 5)
y = np.linspace(-5, 5)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='viridis')
plt.title(r'3D график функции $z = \sin(\sqrt{x^2+y^2})$', fontsize=14)
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.show()
```

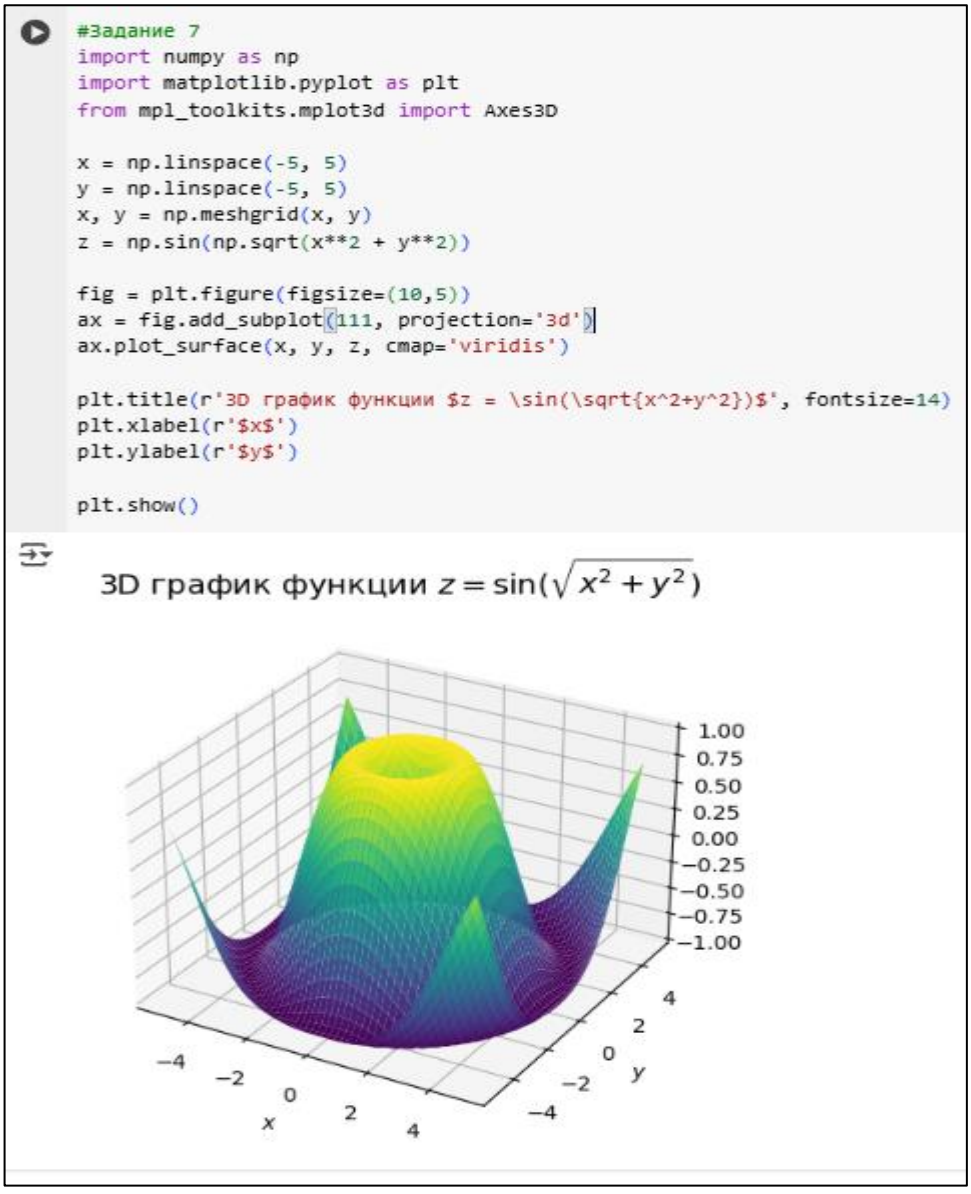


Рисунок 25. Практическое задание 7

**Задание 8** Множественные подграфики(subplots)

Постройте четыре графика в одной фигуре (2x2 сетка):

1. Линейный график y = x

2. Парабола y = x2

3. Синус y = sin(x)

4. Косинус y = cos(x)

Добавьте заголовки к каждому подграфику.

**Листинг кода:**

```
#Задание 8
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
x = np.linspace(-10, 10, 100)
y1 = x
y2 = x**2
y3 = np.sin(x)
y4 = np.cos(x)
fig, axs = plt.subplots(2, 2, figsize=(10,5))
axs[0,0].plot(x, y1)
axs[0,0].set_xlabel('x')
axs[0,0].set_ylabel('y')
axs[0,0].set_title(r'Линейный график $y=x$')
axs[0,1].plot(x, y2, color='red')
axs[0,1].set_xlabel('x')
axs[0,1].set_ylabel('y')
axs[0,1].set_title(r'Парабола $y=x^2$')
axs[1,0].plot(x, y3, color='orange')
axs[1,0].set_xlabel('x')
axs[1,0].set_ylabel('y')
axs[1,0].set_title(r'Синус $y=sin(x)$')
axs[1,1].plot(x, y4, color = 'green')
axs[1,1].set_xlabel('x')
axs[1,1].set_ylabel('y')
axs[1,1].set_title(r'Косинус $y=cos(x)$')
plt.tight_layout()
plt.show()
```

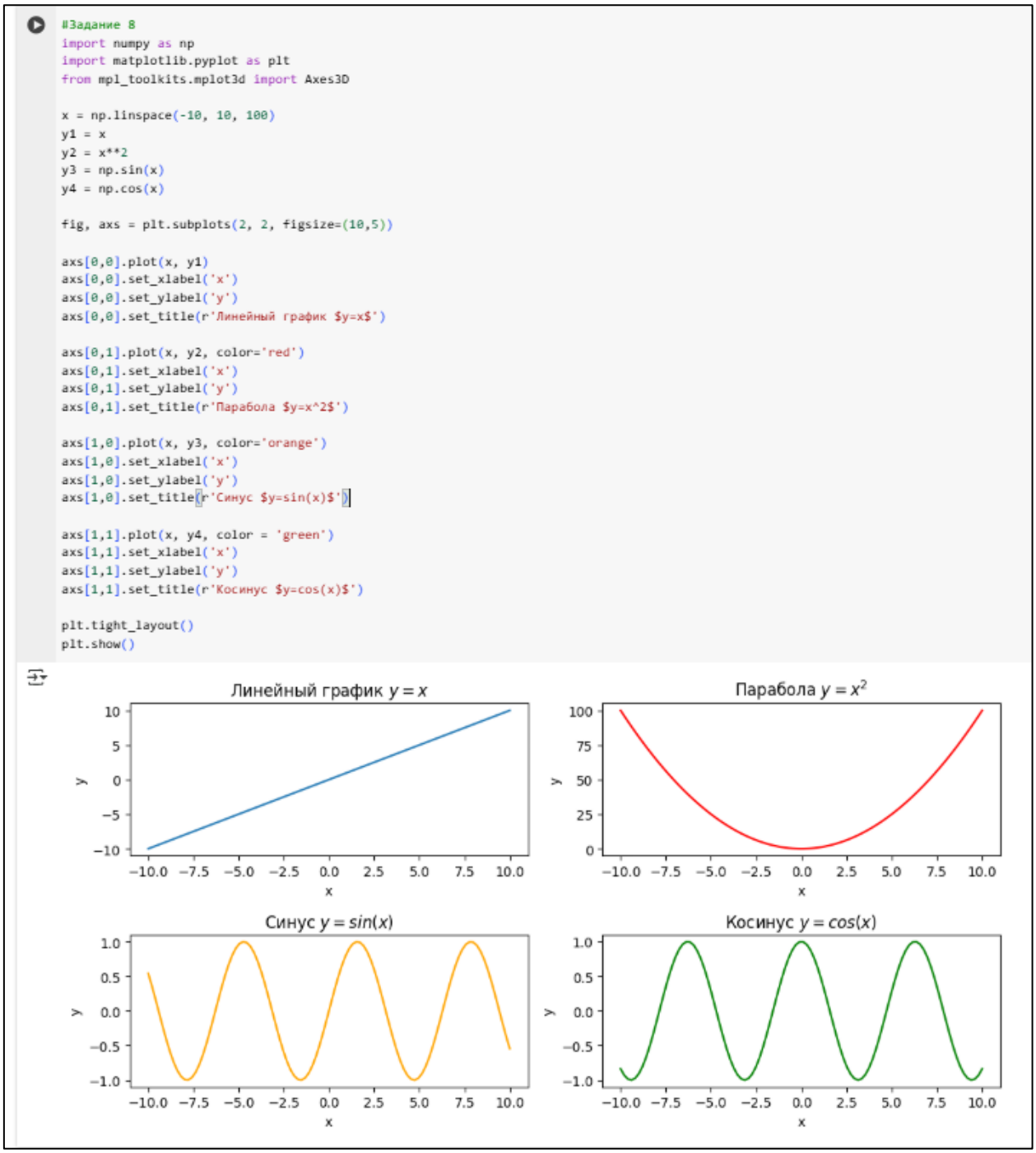


Рисунок 26. Практическое задание 8

**Задание 9.** Тепловая карта (imshow)

Создайте случайную матрицу 10 × 10 с элементами от 0 до 1 и визуализируйте её как тепловую карту с цветовой шкалой.

**Листинг кода:**

```
#Задание 9
import numpy as np
import matplotlib.pyplot as plt
```

```
matrix = np.random.rand(10, 10)
plt.imshow(matrix, cmap='viridis', interpolation='nearest')
plt.colorbar()
plt.title('Тепловая карта случайной матрицы 10x10')
plt.show()
```

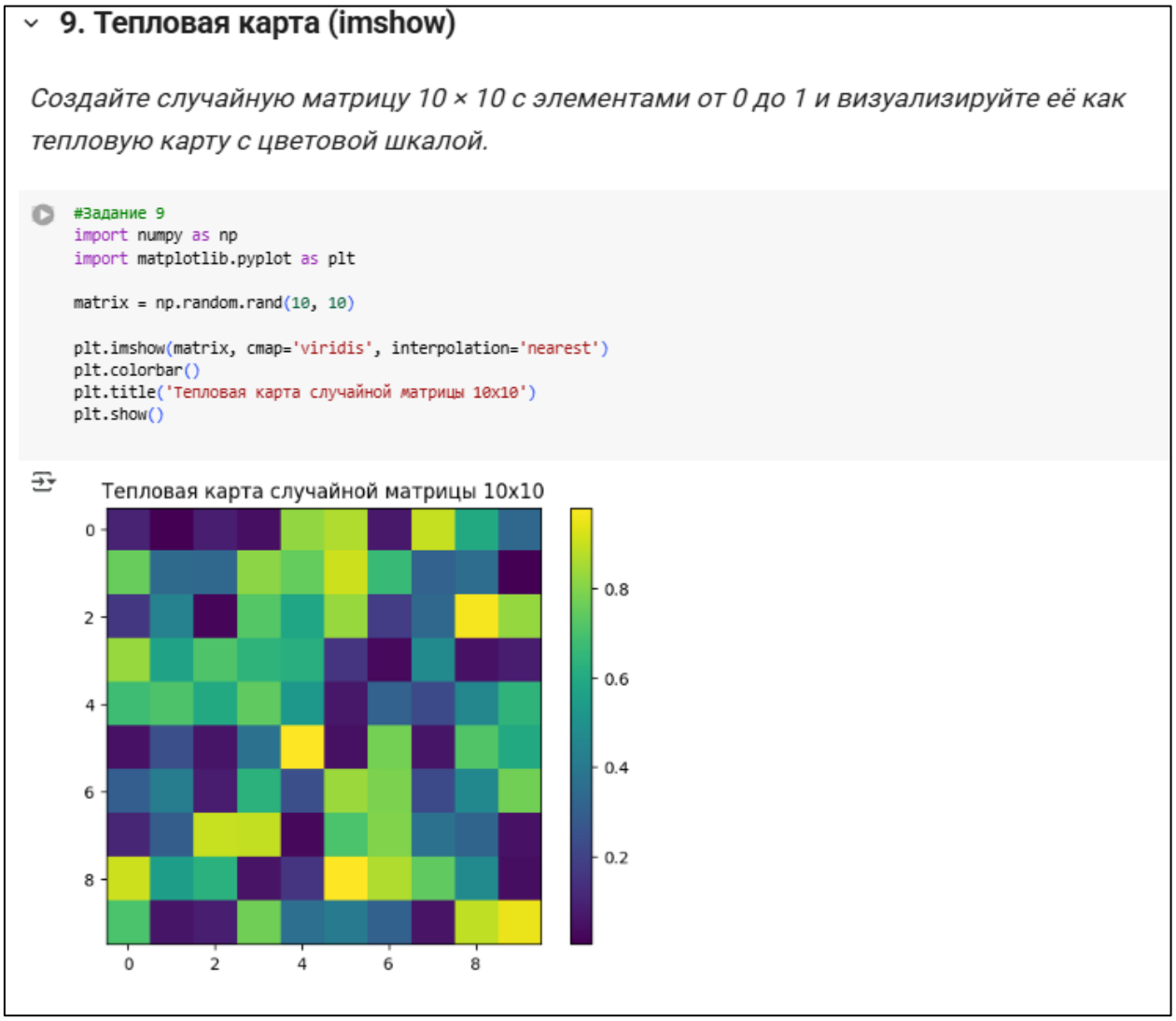


Рисунок 27. Практическое задание 9

6. Выполнил индивидуальное задание.

**Вариант 9**

**Индивидуальное задание №1**

Измеряли количество отжиманий,выполняемых спортсменом в течение тренировки:

Время(минуты): [0, 5, 10, 15, 20, 25, 30]

Количество отжиманий: [10, 25, 40, 50, 55, 57, 58]

Добавьте заголовок и измените стиль графика на пунктирный.

**Листинг кода:**

```
#Индивидуальное задание 1. Вариант 9
import matplotlib.pyplot as plt
time = [0, 5, 10, 15, 20, 25, 30]
pushups = [10, 25, 40, 50, 55, 57, 58]
plt.figure(figsize=(8, 6))
plt.plot(time, pushups, '--o')
plt.title('Динамика количества отжиманий в ходе тренировки')
plt.xlabel('Время (минуты)')
plt.ylabel('Количество отжиманий')
plt.show()
```

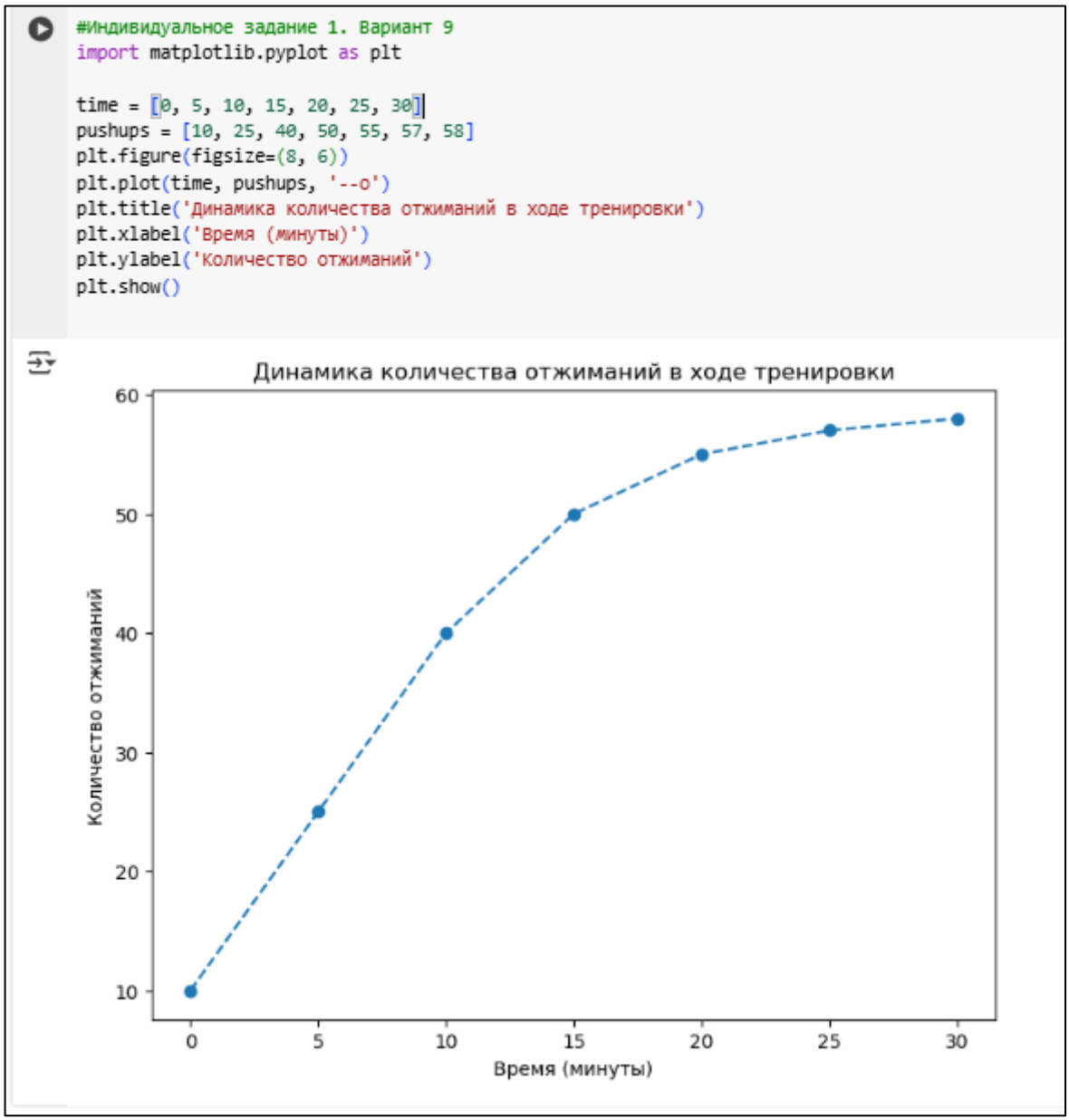


Рисунок 28. Индивидуальное задание №1

**Индивидуальное задание №2**

Распределение скачиваний среди категорий приложений:

Категории: ['Игры', 'Соцсети', 'Образование', 'Музыка', 'Финансы']

Скачивания (млн): [50, 40, 30, 20, 10]

Добавьте цветовую градацию от более популярных к менее популярным.

**Листинг кода:**

```
#Индивидуальное задание 2. Вариант 9
import matplotlib.pyplot as plt
categories = ['Игры', 'Соцсети', 'Образование', 'Музыка', 'Финансы']
downloads = [50, 40, 30, 20, 10]
colors = plt.cm.Reds(df['Скачивания (млн)'] / max(downloads))
fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.bar(df['Категория'], df['Скачивания (млн)'], color=colors)
plt.title('Распределение скачиваний среди категорий приложений')
plt.ylabel('Количество скачиваний (млн)')
plt.show()
```

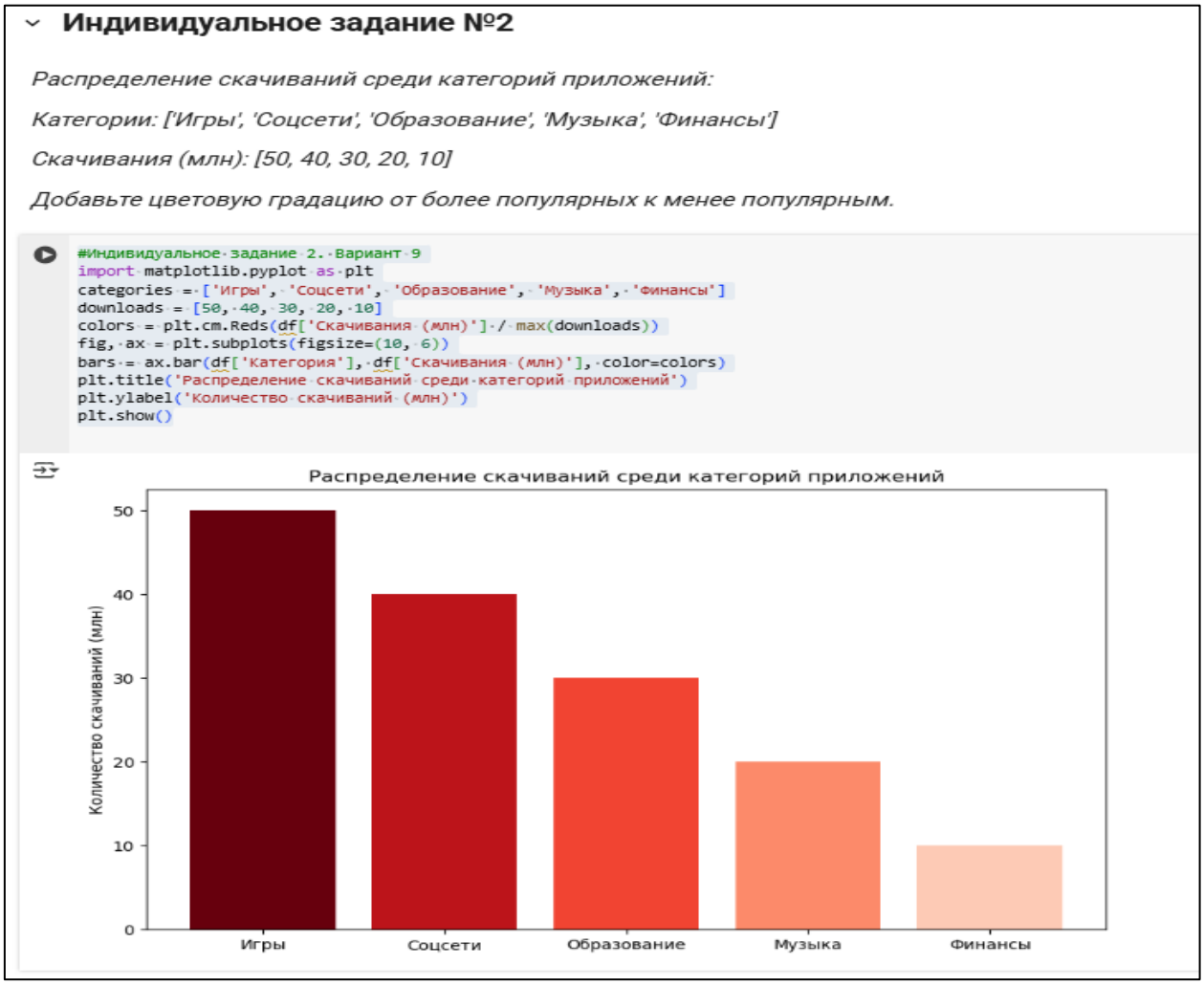


Рисунок 29. Индивидуальное задание №2

**Индивидуальное задание №3**

**Задачи на вычисление определенного интеграла с помощью Matplotlib.**

В каждой задаче требуется:

1. Построить график подинтегральной функции.
2. Вычислить площадь под кривой на заданном отрезке как значение определенного интеграла.
3. Закрасить область под графиком, чтобы визуализировать интеграл. Определите площадь под графиком: f(x) = x−−√ на интервале [0, 4].

Закрасьте соответствующую область.

**Листинг кода:**

```
#Индивидуальное задание 3. Вариант 9
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return np.sqrt(x)
a, b = 0, 4
area, _ = quad(f, a, b)
x = np.linspace(a, b, 100)
y = f(x)
plt.figure(figsize=(8, 6))
plt.plot(x, y, color='blue', label='$f(x) = \\sqrt{x}$')
plt.fill_between(x, y, color='lightblue', alpha=0.5)
plt.title('Площадь под графиком функции $f(x) = \\sqrt{x}$ на интервале $[0,4]$')
plt.xlabel('$x$')
plt.ylabel('$f(x)$')
plt.grid()
plt.legend()
print(f"Площадь под графиком функции на интервале [{a},{b}] равна приблизительно
{area:.3f}")
plt.show()
```

```python
#Индивидуальное задание 3. Вариант 9
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return np.sqrt(x)
a, b = 0, 4
area, _ = quad(f, a, b)
x = np.linspace(a, b, 100)
y = f(x)
plt.figure(figsize=(8, 6))
plt.plot(x, y, color='blue', label='$f(x) = \\sqrt{x}$')
plt.fill_between(x, y, color='lightblue', alpha=0.5)
plt.title('Площадь под графиком функции $f(x) = \\sqrt{x}$ на интервале $[0,4]$')
plt.xlabel('$x$')
plt.ylabel('$f(x)$')
plt.grid()
plt.legend()
print(f"Площадь под графиком функции на интервале [{a},{b}] равна приблизительно {area:.3f}")
plt.show()
```

```
Площадь под графиком функции на интервале [0,4] равна приблизительно 5.333
```

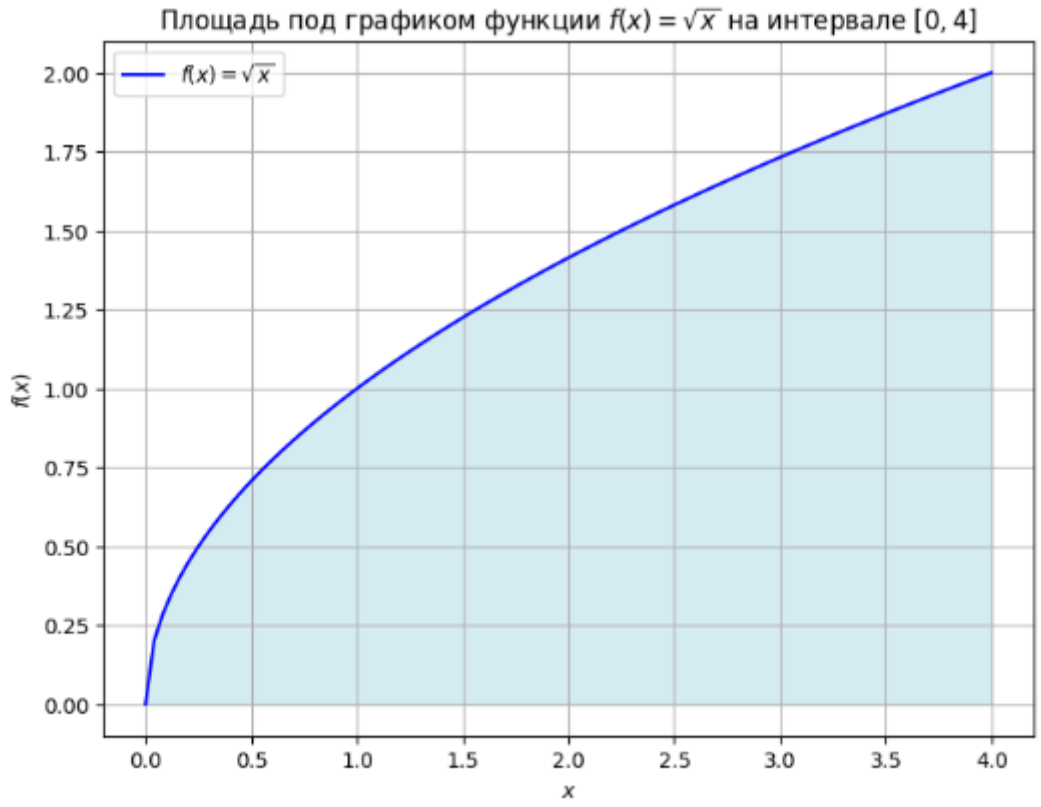


Рисунок 30. Индивидуальное задание №3

**Индивидуальное задание №4**

**Задачи на построение 3D-графиков с помощью Matplotlib**

**Во всех задачах требуется:**

1. Построить трехмерный график функции f(x, y) в заданных пределах.
2. Использовать библиотеку Matplotlib для визуализации.
3. Оформить график: добавить заголовок, подписи осей и цветовую карту (если уместно).
4. Постройте поверхность:f(x, y) = sin(x) + cos(y) на x, y ∈ [−2π, 2π]

**Листинг кода:**

```
#Индивидуальное задание 4. Вариант 9
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
x = np.linspace(-2 * np.pi, 2 * np.pi, 100)
y = np.linspace(-2 * np.pi, 2 * np.pi, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(X) + np.cos(Y)
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')
ax.set_title("Поверхность функции f(x, y) = sin(x) + cos(y)")
ax.set_xlabel("Ось X")
ax.set_ylabel("Ось Y")
ax.set_zlabel("Ось Z")
fig.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```

```python
#Индивидуальное задание 4. Вариант 9
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
x = np.linspace(-2 * np.pi, 2 * np.pi, 100)
y = np.linspace(-2 * np.pi, 2 * np.pi, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(X) + np.cos(Y)
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')
ax.set_title("Поверхность функции f(x, y) = sin(x) + cos(y)")
ax.set_xlabel("Ось X")
ax.set_ylabel("Ось Y")
ax.set_zlabel("Ось Z")
fig.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```

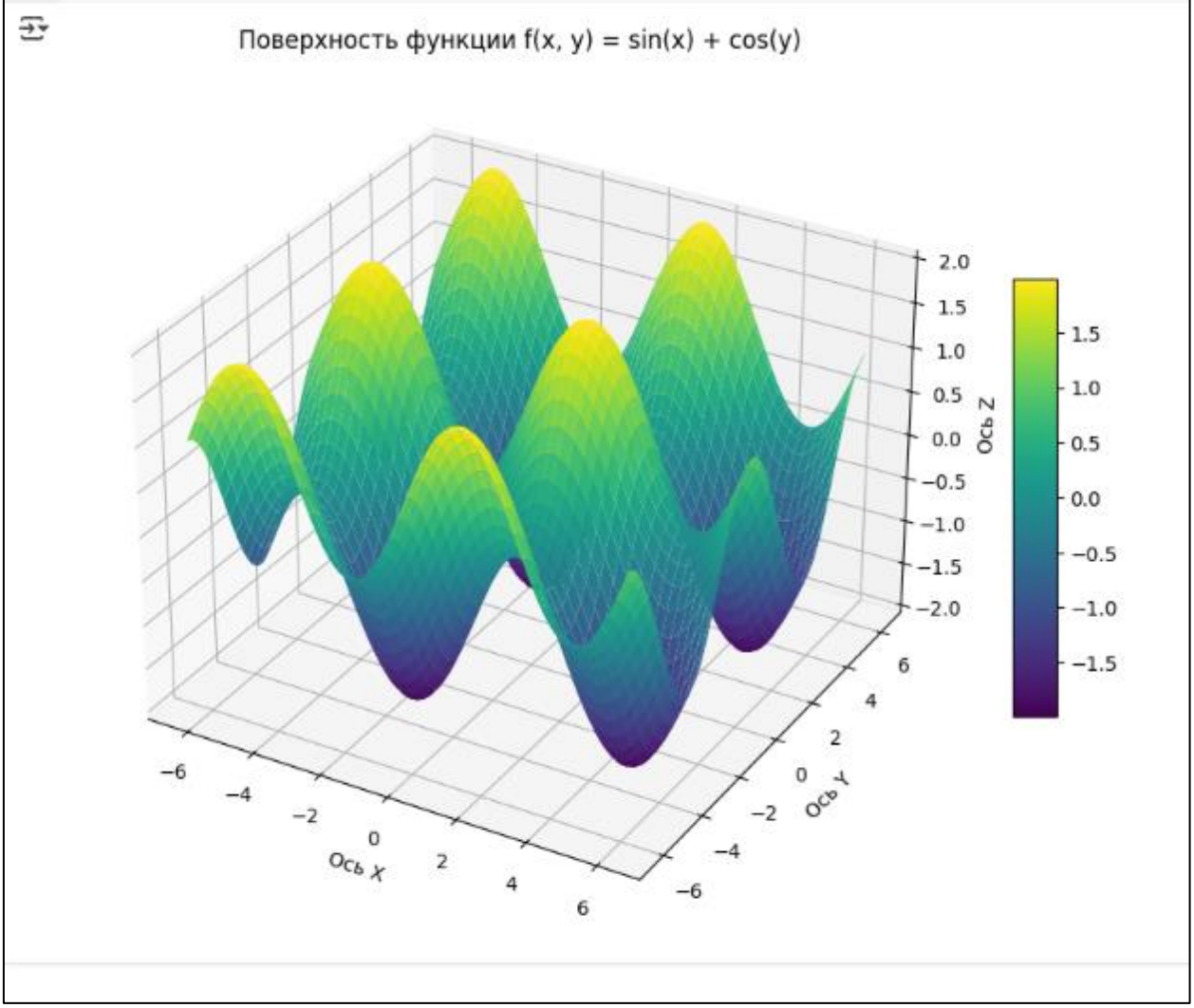


Рисунок 31. Индивидуальное задание №4

7.Зафиксированы изменения на репозитории и отправлены на сервер GitHub.

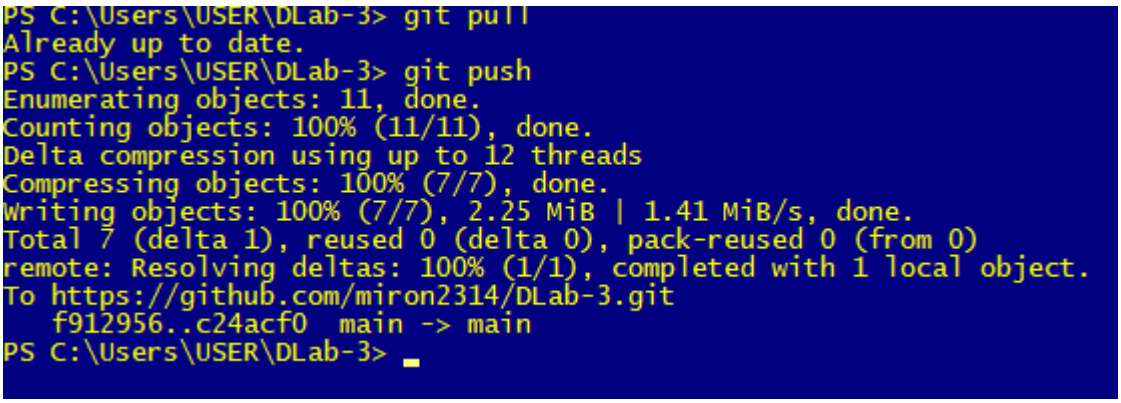


Рисунок 32. Отправка на сервер GitHub

**Вывод:** в ходе выполнения лабораторной работы были исследованы базовые возможности библиотеки matplotlib языка программирования Python.

**Ответы на контрольные вопросы:**

**1. Как осуществляется установка пакета matplotlib?**

Установка пакета matplotlib в Python обычно осуществляется с помощью менеджера пакетов pip (или pip3, если у вас установлено несколько версий Python).

**2. Какая "магическая" команда должна присутствовать в ноутбуках Jupyter для корректного отображения графиков matplotlib**

Для корректного отображения графиков matplotlib в ноутбуках Jupyter необходимо использовать команду: %matplotlib inline

**3. Как отобразить график с помощью функции plot?**

Чтобы отобразить график с помощью функции plot из библиотеки Matplotlib, выполните следующие шаги:

1. Импортируйте необходимые библиотеки:

   import matplotlib.pyplot as plt

2. Подготовьте данные для графика:

   x = [1, 2, 3, 4, 5] y = [2, 3, 5, 7, 11]

3. Используйте функцию plot для построения графика:

   plt.plot(x, y)

4. Добавьте заголовок и метки осей (по желанию):

   plt.title('Пример графика')

plt.xlabel('Ось X')

plt.ylabel('Ось Y')

5. Отобразите график:

plt.show()

В результате вы получите график, отображающий данные из списков x и y.

**4. Как отобразить несколько графиков на одном поле?**

Чтобы отобразить несколько графиков на одном поле с помощью Matplotlib, вы можете использовать функцию plot несколько раз перед вызовом plt.show().

**5.Какой метод Вам известен для построения диаграмм категориальных данных?**

1. Столбчатая диаграмма (Bar Chart):

Столбчатые диаграммы хорошо подходят для отображения категориальных данных. Вы можете использовать plt.bar() для создания столбчатой диаграммы.

2. Гистограмма (Histogram):

Гистограммы используются для отображения распределения числовых данных, но их также можно адаптировать для категориальных данных, если у вас есть количество наблюдений в каждой категории.

3. Круговая диаграмма (Pie Chart):

Круговые диаграммы полезны для отображения долей категорий в общем объеме.

4. Скаттер-плот (Scatter Plot):

Хотя он чаще используется для числовых данных, его можно применять и для категориальных данных в случае, если вы хотите показать взаимосвязь между двумя категориальными переменными.

**6. Какие основные элементы графика Вам известны?**

1. Оси (Axes):

• Ось X: Горизонтальная ось, обычно отображает независимую переменную или категориальные данные.

• Ось Y: Вертикальная ось, обычно отображает зависимую переменную или значения.

2. Метки осей (Axis Labels):

• Названия, которые описывают, что представляют собой оси. Например, "Время" для оси X и "Температура" для оси Y.

3. Заголовок (Title):

• Краткое описание графика, которое помогает понять его содержание и цель.

4. Легенда (Legend):

• Объясняет символы, цвета или линии на графике, особенно если представлено несколько наборов данных.

5. Сетка (Grid):

• Линии, которые помогают визуально ориентироваться на графике и облегчают чтение значений.

6. Точки данных (Data Points):

• Конкретные значения, представленные на графике, которые могут быть обозначены маркерами или линиями.

7. Линии (Lines):

• Используются для соединения точек данных в линейных графиках или для обозначения трендов.

8. Шкала (Scale):

• Определяет диапазон значений, отображаемых на осях. Может быть линейной или логарифмической.

9. Аннотации (Annotations):

• Дополнительные заметки или комментарии на графике, которые помогают объяснить определенные аспекты данных.

10. Фон (Background):

• Цвет или текстура фона графика, который может улучшать визуальное восприятие

**7. Как осуществляется управление текстовыми надписями на графике?**

1. Добавление заголовка и меток осей:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.title('Заголовок графика') # Заголовок графика
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.show()
```

2. Добавление легенды:

```
plt.plot([1, 2, 3], [4, 5, 6], label='Линия 1')
plt.plot([1, 2, 3], [6, 5, 4], label='Линия 2')
plt.legend() # Отображение легенды
plt.show()
```

3.Добавление аннотаций:

```
plt.plot([1, 2, 3], [4, 5, 6])
plt.text(2, 5, 'Аннотация', fontsize=12)
plt.show()
```

4. Настройка шрифта и цвета:

```
plt.title('Заголовок', fontsize=14, color='blue')
plt.xlabel('Ось X', fontsize=12, color='green')
plt.ylabel('Ось Y', fontsize=12, color='red')
```

**8. Как осуществляется управление легендой графика?**

1. Добавление легенды:

Используйте параметр label в функции plot() для каждой линии, а затем вызовите plt.legend() для отображения легенды.

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6], label='Линия 1')
```

plt.plot([1, 2, 3], [6, 5, 4], label='Линия 2')

plt.legend() # Отображение легенды

plt.show()

2. Настройка положения:

Вы можете указать положение легенды с помощью параметра loc: plt.legend(loc='upper left') # Положение в верхнем левом углу

3. Настройка внешнего вида: Можно изменять шрифт, цвет и другие параметры через аргументы функции legend():

plt.legend(fontsize=10, frameon=False) # Настройка шрифта и отключение рамки

1. Добавление легенды:

Используйте параметр 'DisplayName' в функции plot() и затем вызовите legend show.

plot([1, 2, 3], [4, 5, 6], 'DisplayName', 'Линия 1');

hold on; plot([1, 2, 3], [6, 5, 4], 'DisplayName', 'Линия 2');

legend show; % Отображение легенды hold off;

2. Настройка положения:

Вы можете установить положение легенды с помощью legend('Location', 'northeast'):

legend('Location', 'northeast'); % Положение в правом верхнем углу

3. Настройка внешнего вида:

Можно изменять шрифт и другие свойства через параметры legend:

legend('FontSize', 12, 'TextColor', 'b'); % Настройка шрифта и цвета текста

**9. Как задать цвет и стиль линий графика?**

1. Цвет линии:

Вы можете задать цвет линии, указывая его в формате RGB или используя предопределенные названия цветов:

plot([1, 2, 3], [4, 5, 6], 'Color', [1, 0, 0]); % Красный цвет в формате RGB

plot([1, 2, 3], [6, 5, 4], 'g'); % Зеленый цвет

2. Стиль линии:

Стиль линии можно указать в строке форматирования:

• '-': сплошная линия

• '--': пунктирная линия

• '-.': пунктирно-штриховая линия

• ':': точечная линия

Пример: plot([1, 2, 3], [4, 5, 6], 'r--'); % Пунктирная красная линия

**10. Как выполнить размещение графика в разных полях?**

Для размещения графиков в разных полях (подграфиках) используется функция subplot.

**11. Как выполнить построение линейного графика с помощью matplotlib?**

1. Импортируем библиотеку: import matplotlib.pyplot as plt.

2. Подготавливаем данные для осей.

3. Используем функци. Plt.plot() для создания графика.

**12. Как выполнить заливку области между графиком и осью? Между двумя графиками?**

1. Заливка области между графиком и осью:

Используется fill_between(x,y), где x - значения по оси X, а y – значения по оси Y.

2. Заливка области между двумя графиками:

Используется fill_between(x, y1, y2), где y1 и y2 – значения двух графиков. Можно применять условие where для выбора области заливки.

**13. Как выполнить выборочную заливку, которая удовлетворяет некоторому условию?**

Используется fill_between(): Передаются массивы x, y1, y2 и логический массив where в функцию fill_between(). Можно дополнительно настроить цвет и прозрачность заливки с помощью параметров color и alpha.

**14. Как выполнить двухцветную заливку?**

1. Определите условия:

Создайте два логических массива, каждый из которых будет указывать, какие области должны быть залиты разными цветами.

2. Первый вызов fill_between():

Залейте первую область с одним цветом, используя первое условие.

3. Второй вызов fill_between():

Залейте вторую область с другим цветом, используя второе условие.

4. Настройте визуализацию:

Вы можете настроить цвета и прозрачность для каждой заливки.

**15. Как выполнить маркировку графиков?**

1. Добавление заголовка:

Используйте plt.title() для задания заголовка графика.

2. Подписи осей:

Используйте plt.xlabel() и plt.ylabel() для добавления подписей к осям X и Y соответственно.

3. Легенда:

Для отображения легенды используйте plt.legend(), чтобы обозначить различные линии или данные на графике.

4. Подписи точек данных:

Для индивидуальной маркировки точек данных используйте plt.text() или plt.annotate() для добавления текста рядом с конкретными точками на графике.

5. Сетка:

Включите сетку с помощью plt.grid(), чтобы улучшить читаемость графика.

**16. Как выполнить обрезку графиков?**

Для обрезки графиков в Matplotlib используются методы set_xlim() и set_ylim() для задания пределов осей X и Y соответственно. Также можно использовать plt.axis() для одновременной настройки всех пределов.

**17.Как построить ступенчатый график? В чем особенность ступенчатого графика?**

Ступенчатый график в Matplotlib строится с помощью функции plt.step(). Особенность ступенчатого графика заключается в том, что он отображает изменения значений на горизонтальных отрезках, что позволяет лучше визуализировать дискретные изменения во времени или других переменных. Это особенно полезно для представления кумулятивных данных или для отображения функций с резкими переходами.

**18. Как построить стековый график? В чем особенность стекового графика?**

Стековый график в Matplotlib строится с помощью функции plt.stackplot(). Особенность стекового графика заключается в том, что он отображает несколько наборов данных, складывая их друг на друга, что позволяет визуализировать общую величину и относительное значение каждого набора данных по сравнению с другими. Это удобно для анализа изменений во времени и распределения значений между категориями.

**19. Как построить stem-график? В чем особенность stem-графика?**

Stem-график (стебельчатый график) в Matplotlib строится с помощью функции plt.stem(). Особенность stem-графика заключается в том, что он отображает дискретные данные, представляя каждую точку как вертикальную линию (стебель), исходящую от оси X, с пометкой на верхней части, что позволяет легко видеть значения и их распределение. Это полезно для визуализации последовательностей данных и их изменений.

**20. Как построить точечный график?**

В чем особенность точечного графика? Точечный график (scatter plot) строится с помощью функции plt.scatter() в Matplotlib. Особенность точечного графика заключается в том, что он отображает индивидуальные точки данных на плоскости, позволяя визуализировать взаимосвязи между двумя переменными. Это помогает выявлять паттерны, тренды и корреляции в данных.

**21. Как осуществляется построение столбчатых диаграмм с помощью matplotlib?**

Построение столбчатых диаграмм (bar charts) в Matplotlib осуществляется с помощью функции plt.bar() или plt.barh() (для горизонтальных столбцов) из модуля matplotlib.pyplot.

**22. Что такое групповая столбчатая диаграмма? Что такое столбчатая диаграмма с errorbar элементом?**

Групповая столбчатая диаграмма — это способ визуализации данных, который позволяет сравнивать несколько групп категорий одновременно Столбчатая диаграмма с полосами погрешностей— это тип столбчатой диаграммы, который отображает неопределенность или изменчивость данных с помощью "полос" (обычно линий) над каждым столбцом. Эти полосы показывают диапазон возможных значений вокруг среднего значения, представленного высотой столбца.

**23. Как выполнить построение круговой диаграммы средствами matplotlib?**

Для построения круговой диаграммы в Matplotlib используется функция plt.pie() Основные шаги для построения: Подготовить данные Вызов функции plt.pie() Настройка графика Отображение графика

**24. Что такое цветовая карта? Как осуществляется работа с цветовыми картами в matplotlib?**

Цветовая карта — это отображение, которое связывает числа из заданного диапазона с цветами. Как работает цветовая карта:

1. Данные:

У вас есть набор данных, где каждый элемент представляет собой некоторое значение.

2. Нормализация:

Значения данных обычно нормализуются в диапазон от 0 до 1 (или другой подходящий диапазон).

3. Отображение:

Каждый нормализованный элемент данных отображается в цвет из цветовой карты. Цветовая карта определяет, какой цвет соответствует каждому значению от 0 до 1.

4.Визуализация: Полученные цвета используются для отображения данных

**25. Как отобразить изображение средствами matplotlib?**

Отображение изображения из массива NumPy:

import matplotlib.pyplot as plt import numpy as np # Создадим случайный массив для примера (замените на ваш массив изображения)

image_array = np.random.rand(100, 100, 3) # 100x100 пикселей, 3 канала (RGB)

plt.imshow(image_array) # Отображает массив как изображение

plt.axis('off') # Убирает оси координат (по желанию)

plt.title("Изображение из массива NumPy")

plt.show()

**26. Как отобразить тепловую карту средствами matplotlib?**

Отображение тепловой карты с помощью imshow():

import matplotlib.pyplot as plt import numpy as np # Создадим случайные данные для тепловой карты (замените на ваши данные)

data = np.random.rand(10, 10)

plt. imshow(data, cmap='viridis', interpolation='nearest') # Отображаем данные как тепловую карту

plt.colorbar(label='Значение') # Добавляем цветовую шкалу

plt.title('Тепловая карта с imshow()')

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

**27. Как выполнить построение линейного 3D-графика с помощью matplotlib?**

Для построения линейного 3D-графика в Matplotlib необходимо использовать модуль mplot3d

```
import matplotlib.pyplot as plt
from mpl_toolkits. mplot3d
import Axes3D
import numpy as np
```

1. Создание данных

Генерируем случайные данные для x, y и z координат

```
num_points = 100
z = np.linspace(0, 10, num_points)
x = np.sin(z)
y = n.cos(z)
```

2. Создание фигуры и 3D-подграфика

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(projection='3d')
```

3. Построение графика

```
ax.plot(x, y, z, label='Линейный 3D-график')
```

4. Настройка графика

```
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Пример линейного 3D-графика')
ax.legend()
ax.view_init(elev=20, azim=45)
```

5. Отображение графика

```
plt.show()
```

**28. Как выполнить построение точечного 3D-графика с помощью matplotlib?**

Для построения точечного 3D-графика в matplotlib используется функция ax.scatter()

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D import numpy as np
1. Создание данных
Генерируем случайные данные для x, y и z координат
num_points = 100
x = np.random.rand(num_points)
y = np.random.rand(num_points)
z = np.random.rand(num_points)
sizes = np.random.rand(num_points) * 100
colors = np.random.rand(num_points)
Или задаем массив с конкретными цветами, если это необходимо
colors = ['red', 'blue', 'green'] * (num_points // 3)
Повторяем список цветов
2. Создание фигуры и 3D-подграфика
 fig = plt.figure(figsize=(10, 8))
 ax = fig.add_subplot(projection='3d')
 3. Построение точечного графика
 s - размер точки (scale)
 c - цвет точки (color)
 marker - форма маркера (например, 'o', '^', 's')
 ax.scatter(x, y, z, s=sizes, c=colors, marker='o')
 4. Настройка графика (необязательно)
 ax.set_xlabel('X')
 ax.set_ylabel('Y')
 ax.set_zlabel('Z')
 ax.set_title('Пример точечного 3D-графика')
 5. Отображение графика
 plt.show()
```

**29. Как выполнить построение каркасной поверхности с помощью matplotlib?**

Для построения каркасной поверхности в matplotlib используется функция ax.plot_wireframe()

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d
import Axes3D import numpy as np
1. Создание данных
Создадим сетку координат
num_points = 50
x = np.linspace(-5, 5, num_points)
y = np.linspace(-5, 5, num_points)
X, Y = np.meshgrid(x, y)
Определим функцию поверхности
Z = np.sin(np.sqrt(X**2 + Y**2))
2. Создание фигуры и 3D-подграфика
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(projection='3d')
3. Построение каркасной поверхности
ax.plot_wireframe(X, Y, Z, color='black')
4. Настройка графика
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Каркасная поверхность')
5. Отображение графика
plt.show()
```

**30. Для построения трехмерной поверхности с помощью matplotlib используется функция ax.plot_surface()**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

```
from matplotlib import cm # Для цветовых карт
```

1. Создание данных

Создадим сетку координат

```
num_points = 100
x = np.linspace(-5, 5, num_points)
y = np.linspace(-5, 5, num_points)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))
```

2. Создание фигуры и 3D-подграфика

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(projection='3d')
```

3. Построение поверхности

cmap - цветовая карта

rstride - шаг по строкам

cstride - шаг по столбцам

```
ax.plot_surface(X, Y, Z, cmap=cm.viridis, rstride=1, cstride=1)
```

4. Настройка графика

```
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D Поверхность')
```

5. Добавление цветовой шкалы (colorbar)

```
fig.colorbar(ax.plot_surface(X, Y, Z, cmap=cm.viridis))
```

6. Отображение графика

```
plt.show()
```