

**РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное**  
**образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**  
**«Визуализация данных с помощью matplotlib»**

**Отчет по лабораторной работе № 3.5**  
**по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Уланбекова Айканыш.

« » \_\_\_\_\_ 2023г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

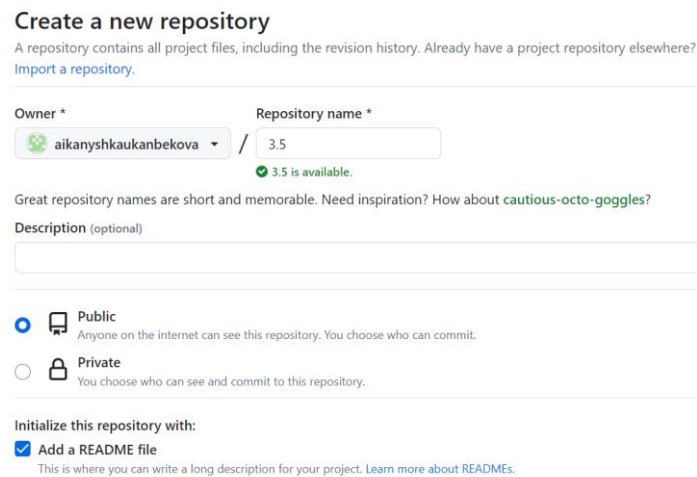
Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

**Цель работы:** исследовать базовые возможности визуализации данных на плоскости средствами библиотеки matplotlib языка программирования Python.

**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* aikanyshkaukanbekova / Repository name \* 3.5

Great repository names are short and memorable. Need inspiration? How about [cautious-octo-goggles?](#)

Description (optional)

☐ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
C:\Users\User>cd C:\Users\User\Desktop\2.1\3.5
C:\Users\User\Desktop\2.1\3.5>git clone https://github.com/aikanyshkaukanbekova/3.5.git
Cloning into '3.5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\User\Desktop\2.1\3.5>_
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

#### 4. Проработать примеры лабораторной работы.

##### Пример 1.

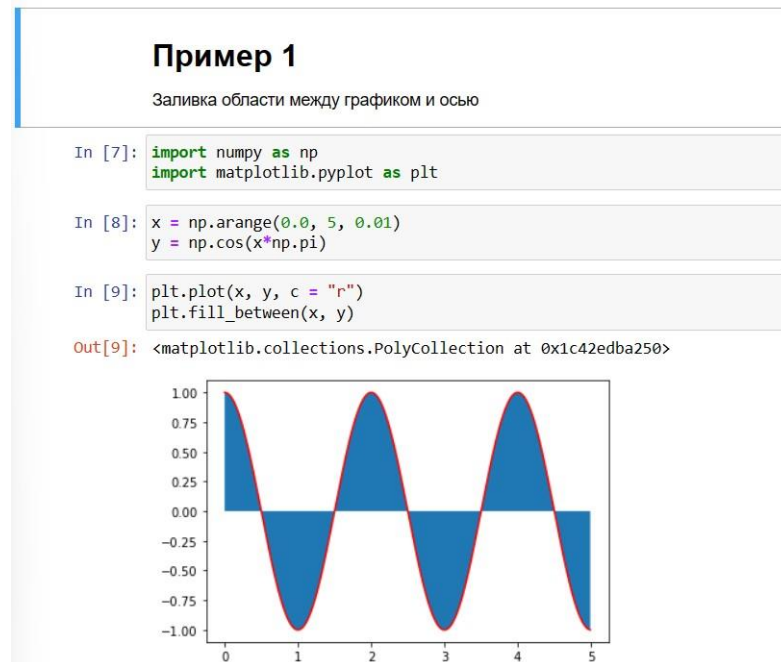


Рисунок 4 - Результат выполнения примера 1

##### Пример 2.

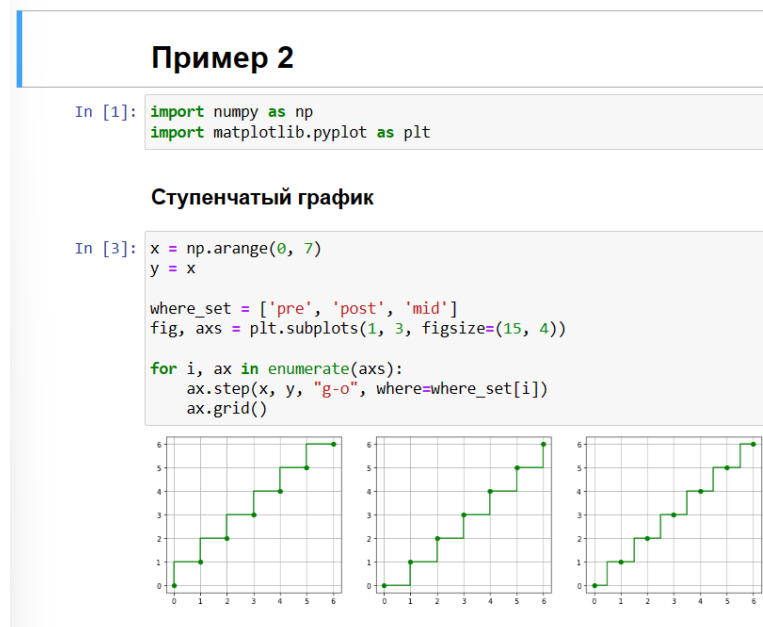


Рисунок 5 - Результат выполнения примера 2

### Пример 3.

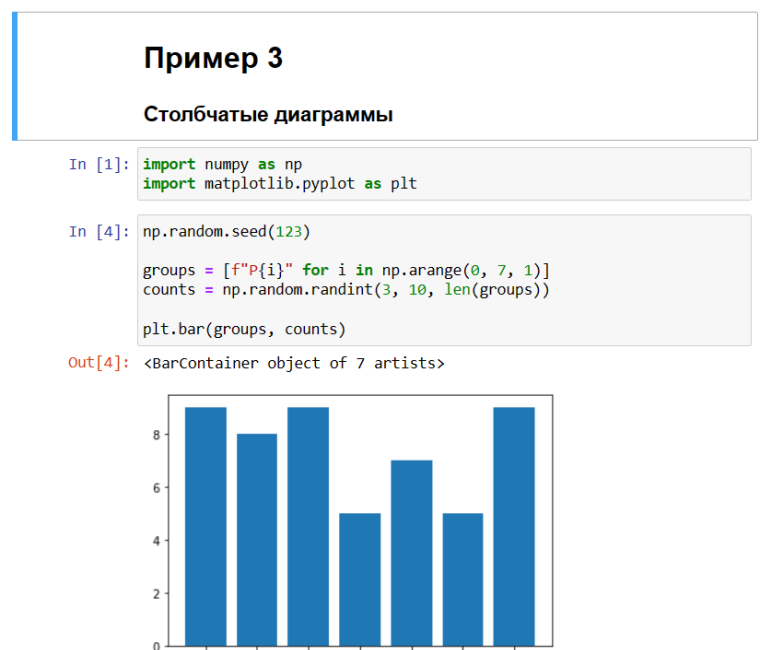


Рисунок 6 - Результат выполнения примера 3

### Пример 4.

#### Пример 4 ¶

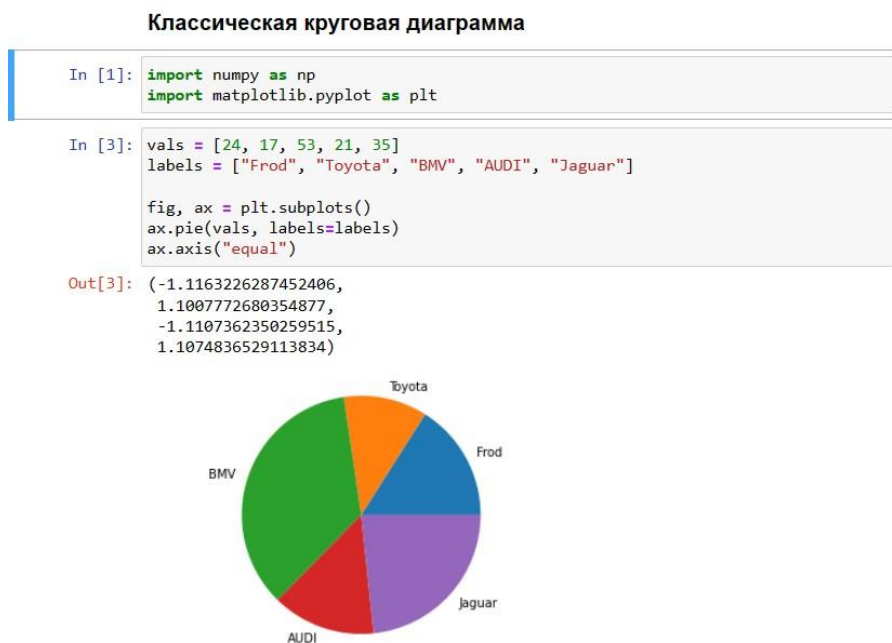


Рисунок 7 - Результат выполнения примера 4

### Пример 5.

## Пример 5

### Построение цветовой сетки

```
In [3]: from PIL import Image
import requests
from io import BytesIO
import numpy as np
import matplotlib.pyplot as plt

In [4]: response = requests.get('https://matplotlib.org/_static/logo2.png')
img = Image.open(BytesIO(response.content))
plt.imshow(img)

Out[4]: <matplotlib.image.AxesImage at 0x1d965b842e0>
```

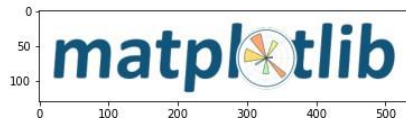


Рисунок 8 - Результат выполнения примера 5

5. Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения линейного графика.

### Контрольные вопросы:

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

Для построения линейного графика используется функция `plot()`, со следующей сигнатурой:

```
plot([x], y, [fmt], *, data=None, **kwargs)
```

```
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

**2. Как выполнить заливку области между графиком и осью?**

**Между двумя графиками?**

Для заливки областей используется функция `fill_between()`. Сигнатура функции:

```
fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, *,
data=None, **kwargs)
```

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

where: массив bool элементов (длины N), optional, значение по умолчанию: None – задает заливаемый цветом регион, который определяется координатами  $x[where]$ : интервал будет залит между  $x[i]$  и  $x[i+1]$ , если  $where[i]$  и  $where[i+1]$  равны True.

Заливка области между 0 и  $y$ , при условии, что  $y \geq 0$ :

```
plt.plot(x, y, c="r")  
plt.fill_between(x, y, where=(y > 0))
```

#### **4. Как выполнить двухцветную заливку?**

Вариант двухцветной заливки:

```
plt.plot(x, y, c="r")  
plt.grid()  
plt.fill_between(x, y, where=y>=0, color="g", alpha=0.3)  
plt.fill_between(x, y, where=y<=0, color="r", alpha=0.3)
```

#### **5. Как выполнить маркировку графиков?**

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

#### **6. Как выполнить обрезку графиков?**

Для того, чтобы отобразить только часть графика, которая отвечает определенному условию используйте предварительное маскирование данных с помощью функции `masked_where` из пакета `numpy`.

```
x = np.arange(0.0, 5, 0.01)  
y = np.cos(x * np.pi)  
y_masked = np.ma.masked_where(y < -0.5, y)  
plt.ylim(-1, 1)  
plt.plot(x, y_masked, linewidth=3)
```

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

Такой график строится с помощью функции `step()`, которая принимает следующий набор параметров:

x: array\_like - набор данных для оси абсцисс  
y: array\_like - набор данных для оси ординат  
fmt: str, optional - задает отображение линии (см. функцию plot()).  
data: indexable object, optional - метки.  
where : {'pre', 'post', 'mid'}, optional , по умолчанию 'pre' - определяет место, где будет установлен шаг.

```
x = np.arange(0, 7)
y = x
where_set = ['pre', 'post', 'mid']
fig, axs = plt.subplots(1, 3, figsize=(15, 4))
for i, ax in enumerate(axs):
    ax.step(x, y, "g-o", where=where_set[i])
    ax.grid()
```

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

Для построения стекового графика используется функция `stackplot()`. Суть его в том, что графики отображаются друг над другом, и каждый следующий является суммой предыдущего и заданного набора данных:

```
x = np.arange(0, 11, 1)
y1 = np.array([(-0.2)*i**2+2*i for i in x])
y2 = np.array([(-0.4)*i**2+4*i for i in x])
y3 = np.array([2*i for i in x])
labels = ["y1", "y2", "y3"]
fig, ax = plt.subplots()
ax.stackplot(x, y1, y2, y3, labels=labels)
ax.legend(loc='upper left')
```

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

Визуально этот график выглядит как набор линий от точки с координатами (x, y) до базовой линии, в верхней точке ставится маркер:

```
x = np.arange(0, 10.5, 0.5)
y = np.array([(-0.2)*i**2+2*i for i in x])
plt.stem(x, y)
```

Дополнительные параметры функции stem():

linefmt: str, optional - стиль вертикальной линии

markerfmt: str, optional - формат маркера

basefmt: str, optional - формат базовой линии

bottom: float, optional , по умолчанию: 0 - y-координата базовой линии

## **10. Как построить точечный график? В чем особенность точечного графика?**

Для отображения точечного графика предназначена функция scatter(). В простейшем виде точечный график можно получить передав функции scatter() наборы точек для x, y координат:

```
x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y)
```

Для более детальной настройки отображения необходимо воспользоваться дополнительными параметрами функции scatter(), сигнатура ее вызова имеет следующий вид:

```
scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None,
vmin=None, vmax=None, alpha=None, linewidths=None, verts=None,
edgecolors=None, *, plotnonfinite=False, data=None, **kwargs)
```

## **11. Как осуществляется построение столбчатых диаграмм с помощью matplotlib?**

Для визуализации категориальных данных хорошо подходят столбчатые диаграммы. Для их построения используются функции:

bar() – для построения вертикальной диаграммы



`barh()` – для построения горизонтальной диаграммы.

Построим простую диаграмму:

```
np.random.seed(123)
groups = [f"P{i}" for i in range(7)]
counts = np.random.randint(3, 10, len(groups))
plt.bar(groups, counts)
```

Если заменим `bar()` на `barh()` получим горизонтальную диаграмму:

```
plt.barh(groups, counts)
```

## **12. Что такое групповая столбчатая диаграмма? Что такое столбчатая диаграмма с `errorbar` элементом?**

Используя определенным образом подготовленные данные можно строить групповые диаграммы:

```
cat_par = [f"P{i}" for i in range(5)]
g1 = [10, 21, 34, 12, 27]
g2 = [17, 15, 25, 21, 26]
width = 0.3
x = np.arange(len(cat_par))
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, g1, width, label='g1')
rects2 = ax.bar(x + width/2, g2, width, label='g2')
ax.set_title('Пример групповой диаграммы')
ax.set_xticks(x)
ax.set_xticklabels(cat_par)
ax.legend()
```

`Errorbar` элемент позволяет задать величину ошибки для каждого элемента графика. Для этого используются параметры `xerr`, `yerr` и `ecolor` (для задания цвета):

```
np.random.seed(123)
rnd = np.random.randint
```

```

cat_par = [f"P{i}" for i in range(5)]
g1 = [10, 21, 34, 12, 27]
error = np.array([[rnd(2,7),rnd(2,7)] for _ in range(len(cat_par))]).T
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].bar(cat_par, g1, yerr=5, ecolor="r", alpha=0.5, edgecolor="b",
linewidth=2)
axs[1].bar(cat_par, g1, yerr=error, ecolor="r", alpha=0.5, edgecolor="b",
linewidth=2)

```

### **13. Как выполнить построение круговой диаграммы средствами matplotlib?**

Круговые диаграммы – это наглядный способ показать доли компонент в наборе. Они идеально подходят для отчетов, презентаций и т.п. Для построения круговых диаграмм в Matplotlib используется функция `pie()`.

Пример построения 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")

```

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

Цветовая карта представляет собой подготовленный набор цветов, который хорошо подходит для визуализации того или иного набора данных. Подробное руководство по цветовым картам вы можете найти на официальном сайте Matplotlib (<https://matplotlib.org/tutorials/colors/colormaps.html#sphx-glr-tutorials-colors-colormaps-py>). Также отметим, что такие карты можно создавать самостоятельно, если среди существующих нет подходящего решения.

Рассмотрим две функции для построения цветовой сетки: `imshow()` и `pcolormesh()`.

## 15. Как отобразить изображение средствами matplotlib?

Основное назначение функции `imshow()` состоит в представлении 2d растров. Это могут быть картинки, двумерные массивы данных, матрицы и т.п. Напишем простую программу, которая загружает картинку из интернета по заданному URL и отображает ее с использованием библиотеки Matplotlib:

```
from PIL import Image
import requests
from io import BytesIO
response = requests.get('https://matplotlib.org/_static/logo2.png')
img = Image.open(BytesIO(response.content))
plt.imshow(img)
```

## 16. Как отобразить тепловую карту средствами matplotlib?

Рассмотрим ещё одну функцию для визуализации 2D наборов данных – `pcolormesh()`. В библиотеке Matplotlib есть ещё одна функция с аналогичным функционалом – `pcolor()`, в отличие от нее рассматриваемая нами `pcolormesh()` более быстрая и является лучшим вариантом в большинстве случаев. Функция `pcolormesh()` похожа по своим возможностям на `imshow()`, но есть и отличия.

Пример использования функции `pcolormesh()`:

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

**Вывод:** были исследованы базовые возможности визуализации данных на плоскости средствами библиотеки matplotlib языка программирования Python.