

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Институт цифрового развития**

ОТЧЁТ

по лабораторной работе №3.5

Дисциплина: «Анализ данных»

Тема: «Визуализация данных с помощью matplotlib»

Выполнил: студент 2 курса
группы ИВТ-б-о-21-1
Кочкаров Умар Ахматович

Ставрополь 2023


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

Ход работы:

1. Создать общедоступный репозиторий с лицензией MIT и языком 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 *  umarkochkarov ▾ / Repository name *

✓ lb3.5 is available.

Great repository names are short and memorable. Need inspiration? How about [studious-sniffle?](#)

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.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1. Создание репозитория

2. Клонировать репозиторий на ПК:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python/лаб3.5
$ git clone https://github.com/umarkochkarov/lb3.5.git
Cloning into 'lb3.5'...
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.
```

Рисунок 2. Клонирование репозитория на ПК

3. Организовать репозиторий в соответствии с моделью ветвления git-flow:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python/лаб3.5/lb3.5 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/erken/Desktop/python/лаб3.5/.git/hooks]
```

Рисунок 3. Организация репозитория в соответствии с git-flow

4. Проработка примеров лабораторной работы:

1)

Заливка области между графиком и осью

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

2: x = np.arange(0.0, 5, 0.01)
   y = np.cos(x*np.pi)

3: plt.plot(x, y, c = "r")
   plt.fill_between(x, y)

4: <matplotlib.collections.PolyCollection at 0x18139440130>
```

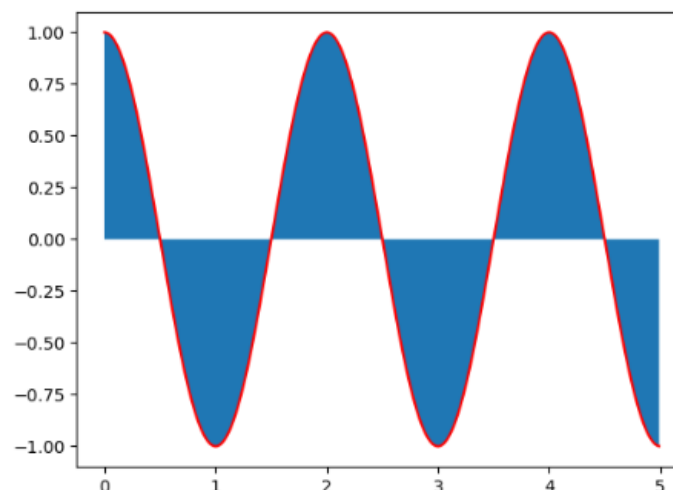


Рисунок 4. Пример 1

2)

Ступенчатый график

```
In [10]: 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()
```

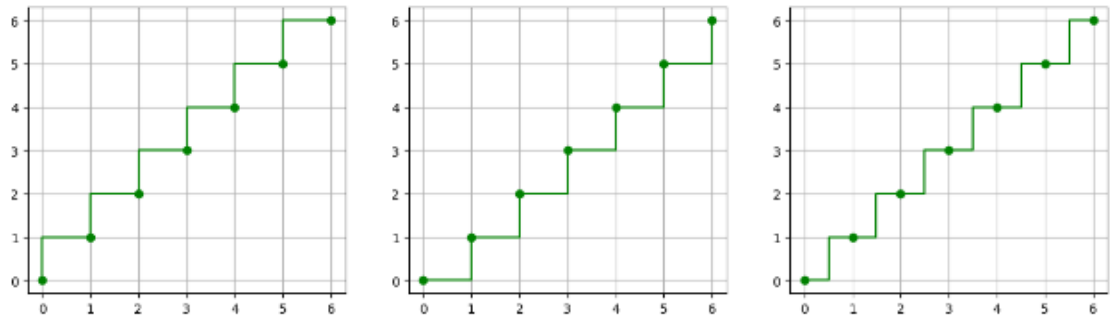


Рисунок 5. Пример 2

3)

Столбчатые диаграммы

```
In [17]: np.random.seed(123)

groups = [f"P{i}" for i in np.arange(0, 7, 1)]
counts = np.random.randint(3, 10, len(groups))

plt.bar(groups, counts)
```

```
Out[17]: <BarContainer object of 7 artists>
```

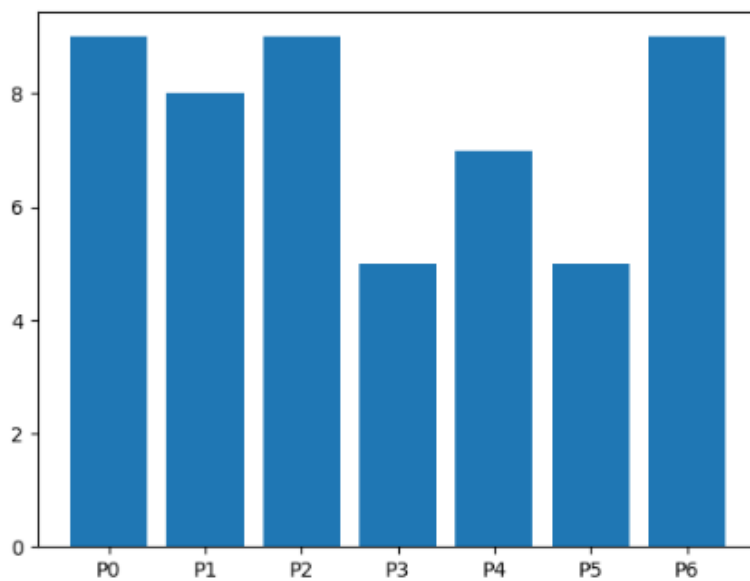


Рисунок 6. Пример 3

4)

Классическая круговая диаграмма

```
[22]: vals = [24, 17, 53, 21, 35]
      labels = ["Ford", "Toyota", "BMW", "AUDI", "Jaguar"]

      fig, ax = plt.subplots()
      ax.pie(vals, labels=labels)
      ax.axis("equal")
```

```
t[22]: (-1.1163226287452406,
        1.1007772680354877,
        -1.1107362350259515,
        1.1074836529113834)
```

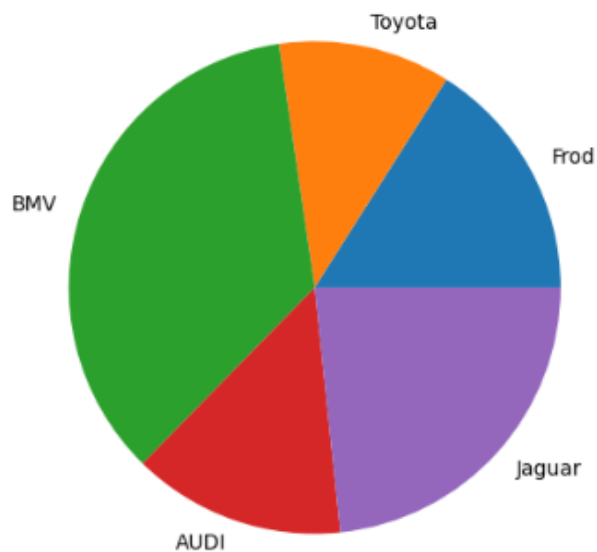


Рисунок 7. Пример 4

5)

Построение цветовой сетки

```
[31]: from PIL import Image
      import requests

      from io import BytesIO
```

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

```
[32]: <matplotlib.image.AxesImage at 0x1813a943760>
```

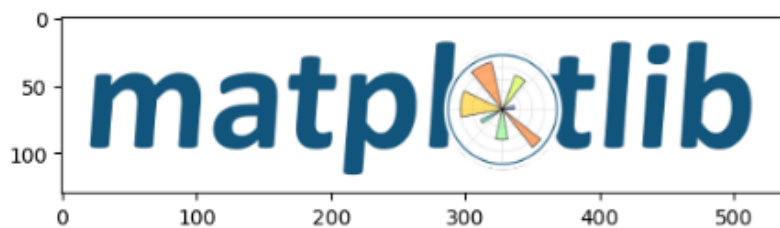


Рисунок 8. Пример 5

5. Индивидуальные задания:

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения линейного графика, условие которой предварительно необходимо согласовать с преподавателем.

Себестоимость 1 тонны кукурузы, выращенной на первом участке составляет 5000 руб., на втором участке – 8 тыс. руб., на третьем – 12 тыс. руб. Оптовая цена 1 тонны кукурузы составляет 12 тыс. руб. Чему равна дифференциальная рента, получаемая на первом и втором участке при производстве от 1 до 10 тонн кукурузы? Чему равна дифференциальная рента на каждом участке при производстве 10 тонн кукурузы?

```
In [1]: import matplotlib.pyplot as plt
```

```
In [20]: costs = [5000, 8000, 12000]
wholesale_price = 12000
productions = range(1, 11)
```

Линейный график

```
In [23]: cost_1 = [costs[0]/ p for p in productions]
rent_1 = [wholesale_price - c for c in cost_1]
```

```
In [26]: plt.plot(productions, cost_1, label='Стоимость производства')
plt.axhline(y=wholesale_price, color='r', linestyle='-', label='Оптовая цена')
plt.plot(productions, rent_1, label='Дифференциальная рента')
plt.xlabel('Объем производства, тонн')
plt.ylabel('Стоимость, руб.')
plt.legend()
plt.show()
```

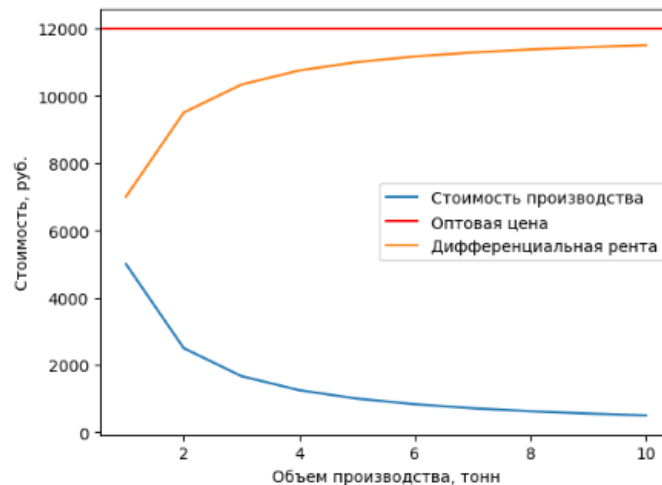


Рисунок 9. Индивидуальное задание 1

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения линейного графика, условие которой предварительно необходимо согласовать с преподавателем.

Столбчатая диаграмма

```
cost_2 = [costs[1] / p for p in productions]
rent_2 = [wholesale_price - c for c in cost_2]
```

```
plt.bar(productions, rent_2, label='Дифференциальная рента')
plt.xlabel('Объем производства, тонн')
plt.ylabel('Дифференциальная рента, руб.')
plt.legend()
plt.show()
```

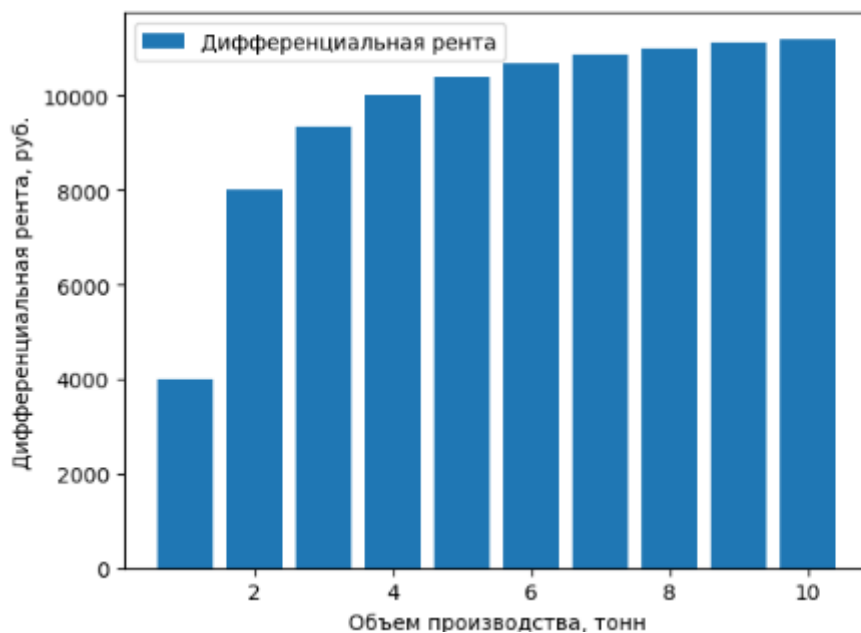


Рисунок 10. Индивидуальное задание 2

Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) требующей построения линейного графика, условие которой предварительно необходимо согласовать с преподавателем.

Круговая диаграмма

```
] : rent1 = wholesale_price - costs[0]/10  
    rent2 = wholesale_price - costs[1]/10  
    rent3 = wholesale_price - costs[2]/10  
  
    rents = [rent1, rent2, rent3]  
  
    labels = ['участок 1', 'участок 2', 'участок 3']  
  
] : plt.pie(rents, labels=labels, autopct='%1.1f%%')  
    plt.title('Дифференциальная рента на каждом участке')  
    plt.show()
```

Дифференциальная рента на каждом участке

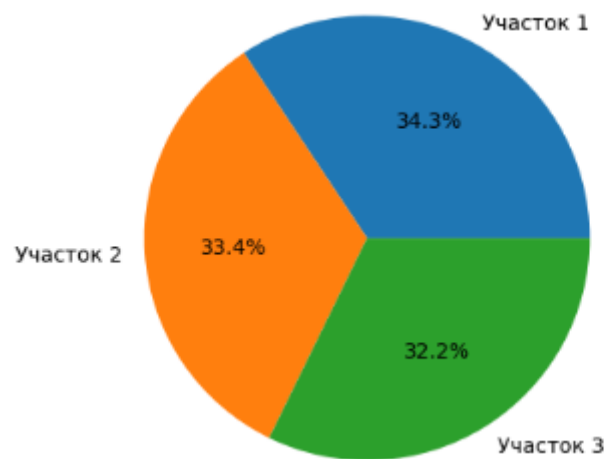


Рисунок 11. Индивидуальное задание 3

Найти какое-либо изображение в сети Интернет. Создать ноутбук, в котором будет отображено выбранное изображение средствами библиотеки matplotlib по URL из сети Интернет.

```
import requests
import io
from PIL import Image
import matplotlib.pyplot as plt

image_url = "https://klassicheskij-recept.ru/wp-content/uploads/2022/10/nastojashhiy-uzbekskij-plov.jpg"

response = requests.get(image_url)
image_data = response.content

image = Image.open(io.BytesIO(image_data))

plt.imshow(image)
plt.axis('off')
plt.show()
```



Рисунок 12. Индивидуальное задание 4

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 >= 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-colorscolormaps-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: исследованы базовые возможности визуализации данных на плоскости средствами библиотеки matplotlib языка программирования Python