МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Кафедра инфокоммуникаций

Отчет

по лабораторной работе №12

«Визуализация данных с помощью matplotlib»

по дисциплине:

«Введение в системы искусственного интеллекта»

Вариант 3

Выполнил: студент группы ИВТ-б-о-18-1
Данченко Максим Игоревич
_____(подпись)
Проверил:
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

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

Ход работы:

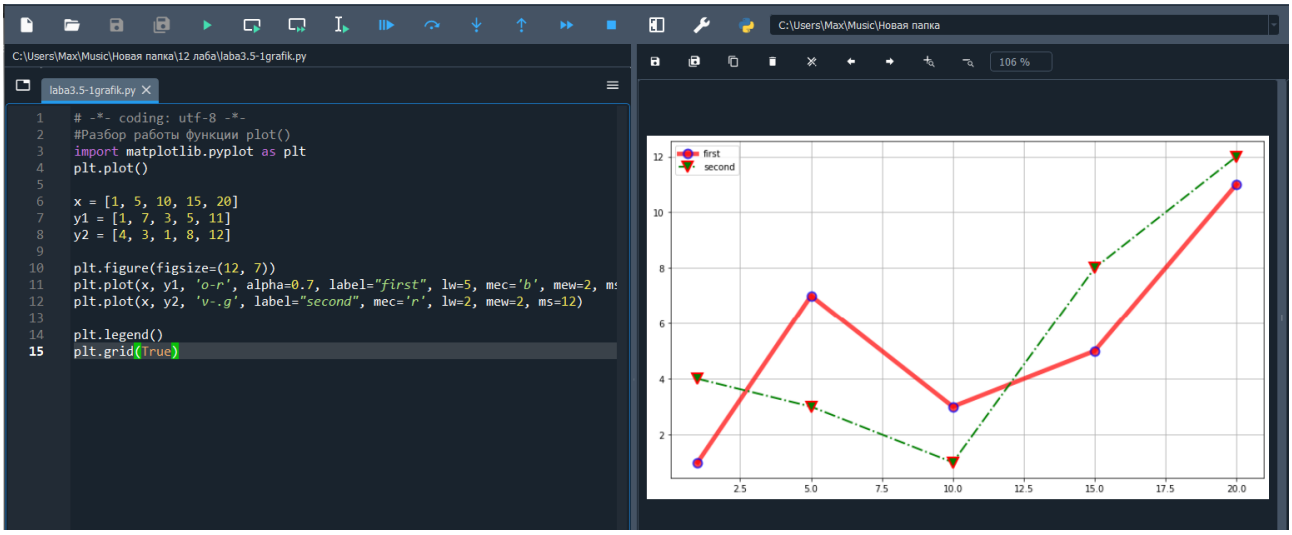


Рисунок 1 –график с параметрами аргумента fmt

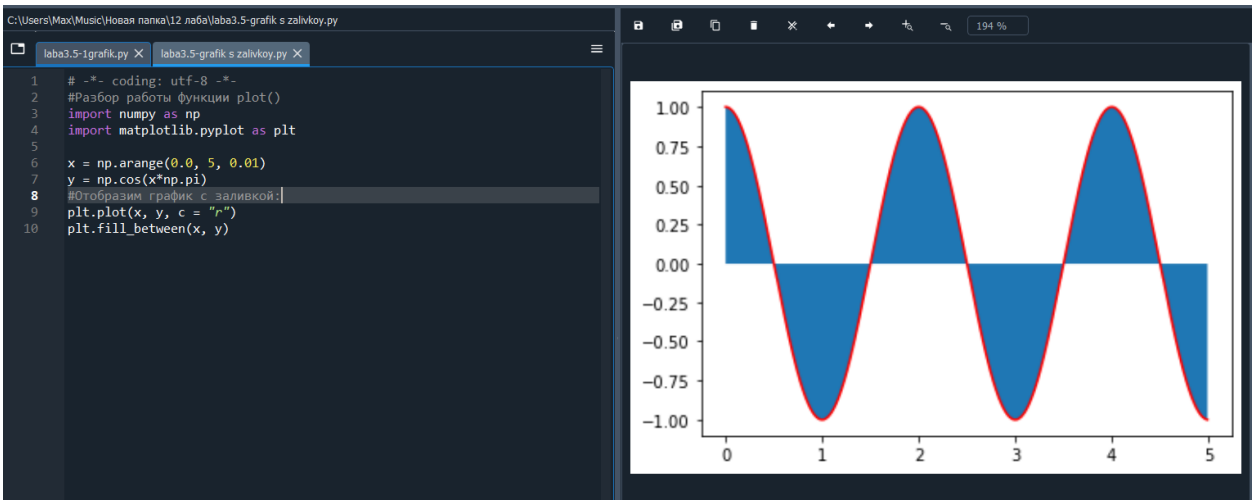


Рисунок 2 – график с заливкой области между графиком и осью

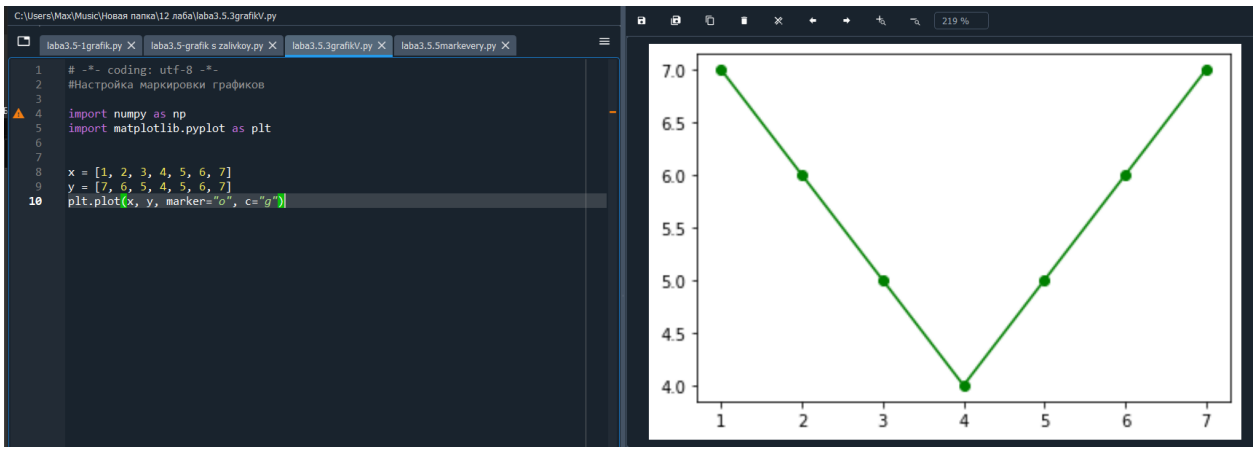


Рисунок 3 – график с настройкой маркировки графиков

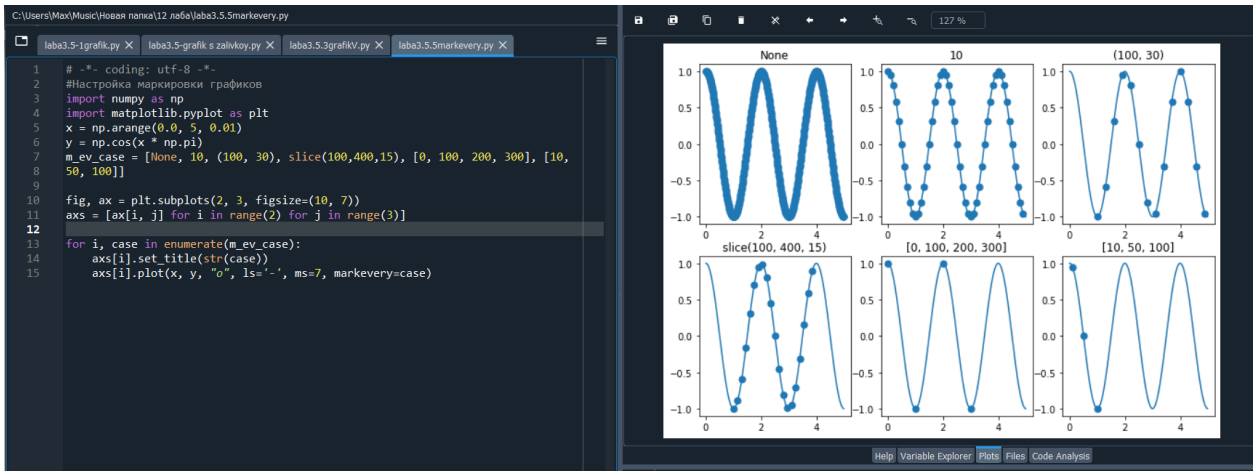


Рисунок 4 – график, демонстрирующий работу с markevery

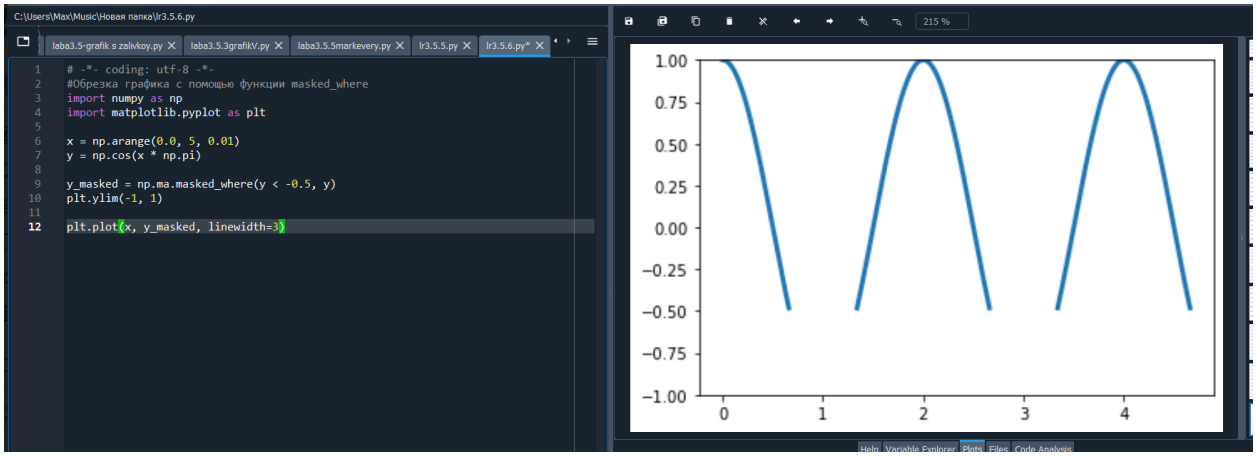


Рисунок 5 – обрезка графика

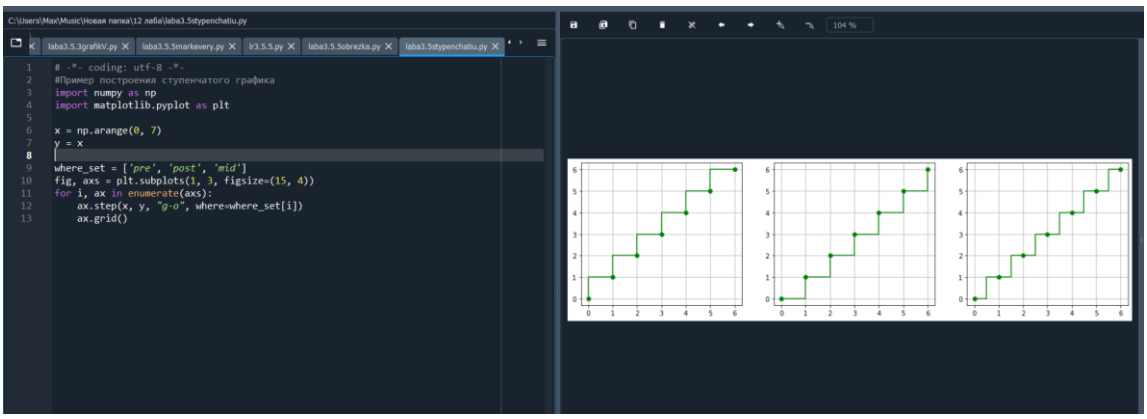


Рисунок 6 – ступенчатый график

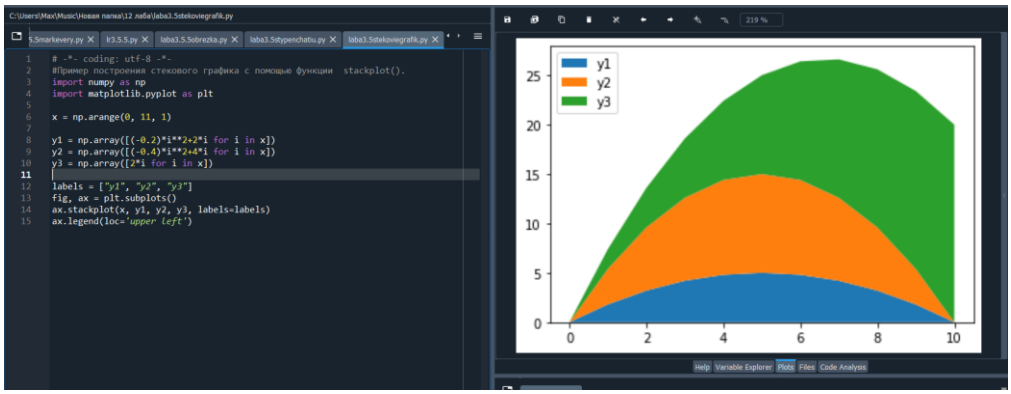


Рисунок 7 – стековый график

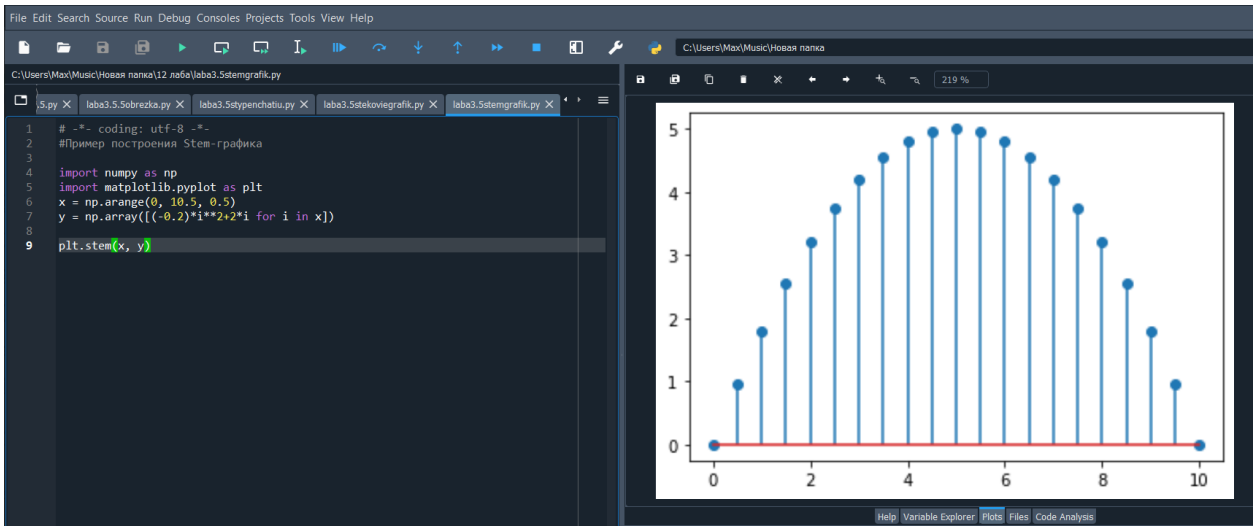


Рисунок 8 – Stem-график

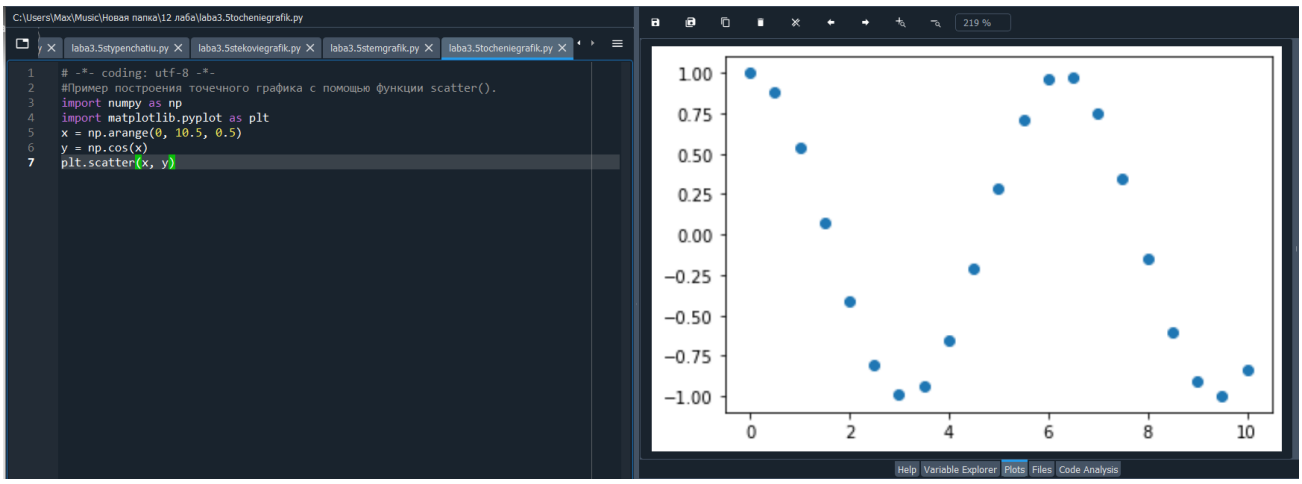


Рисунок 9 – точечный график

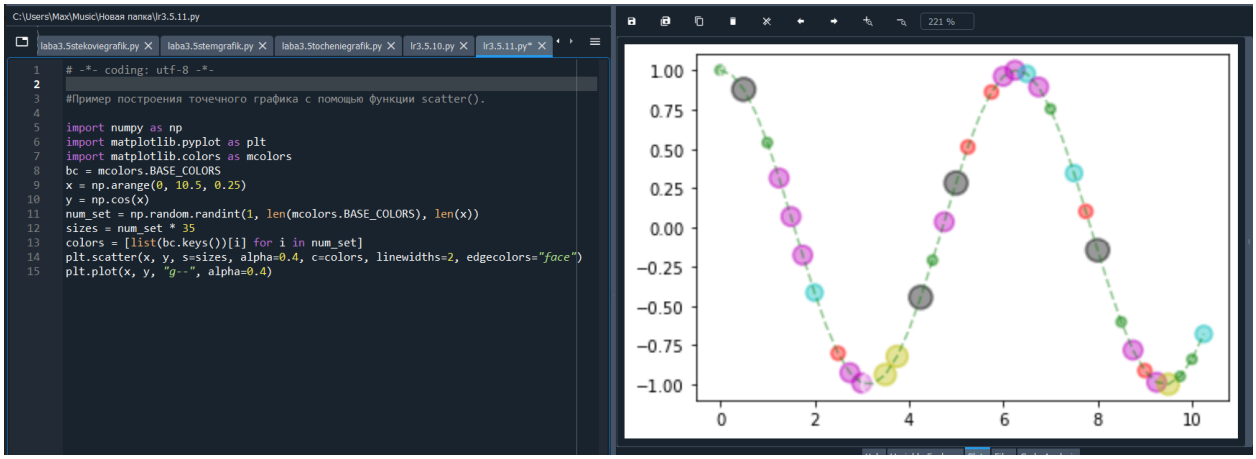


Рисунок 10 – точечный график, демонстрирующий работу с цветом и размером:

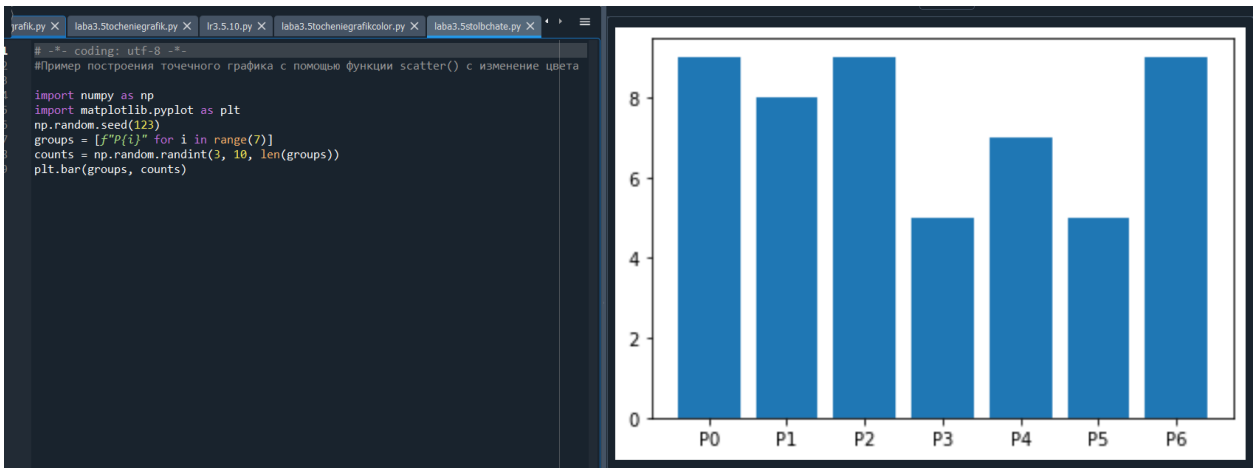


Рисунок 11 – столбчатые диаграммы

Рисунок 12 – столбчатые диаграммы с параметрами

Рисунок 13 – Диаграмма с errorbar элементом

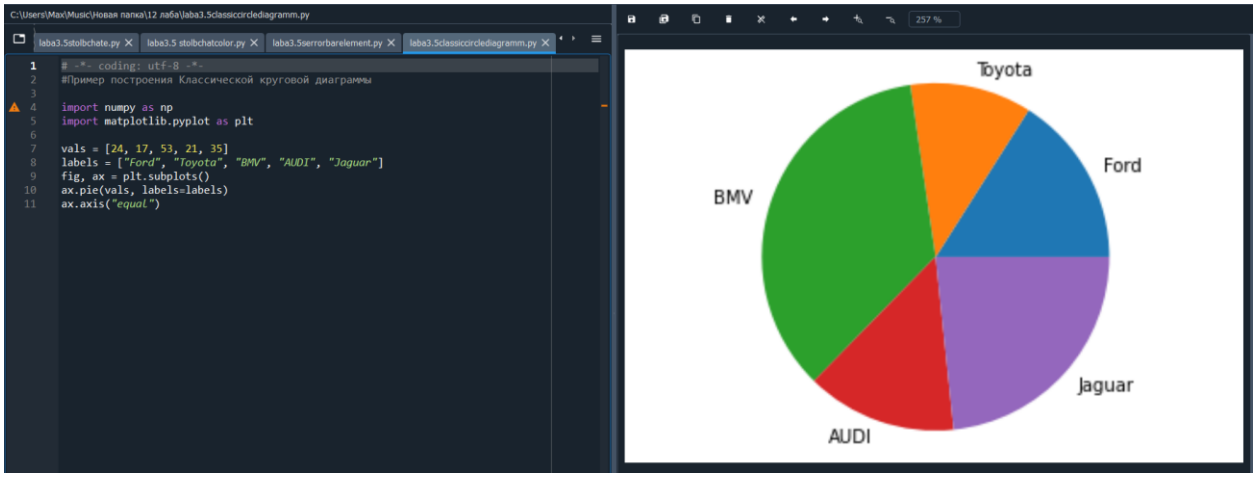


Рисунок 14 – Классическая круговая диаграмма

Рисунок 15 – Изображение логотипа Matplotlib.

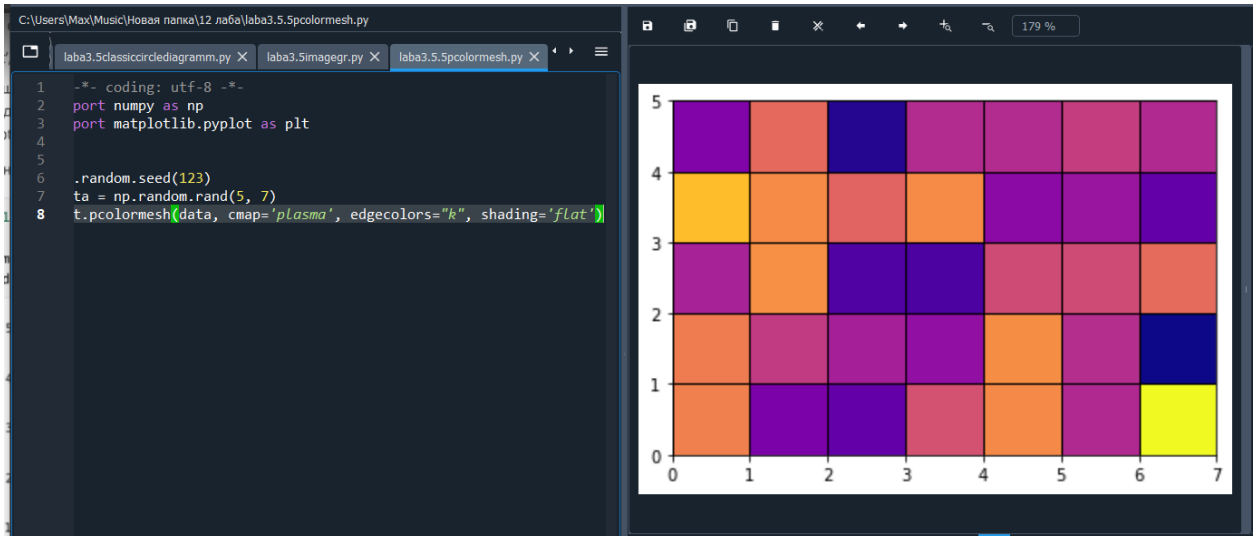


Рисунок 16 – Пример использования функции pcolormesh():.

Вывод: В ходе лабораторной работы были исследованы базовые возможности визуализации данных на плоскости средствами библиотеки matplotlib языка программирования Python..

Ответы на вопросы:

1. Как выполнить построение линейного графика с помощью 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)

Если вызвать функцию plot() с одним аргументом – вот так: plot(y), то мы получим график, у которого по оси ординат (ось y) будут отложены значения из переданного списка, по по оси абсцисс (ось x) – индексы элементов массива.

Рассмотрим аргументы функции plot():

1)x, x2, ...: array - набор данных для оси абсцисс первого, второго и т.д. графика.

2)y, y2, ...: array - набор данных для оси ординат первого, второго и т.д. графика.

3)fmt: str - формат графика, задается в виде строки: [marker][line][color]

4)**kwargs – свойства класса Line2D (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D), которые предоставляют доступ к большому количеству настроек внешнего вида графика, отметим наиболее полезные:

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)

Основные параметры функции:

x : массив длины N`- набор данных для оси абсцисс.

y1 : массив длины N или скалярное значение - набор данных для оси ординат – первая кривая.

y2 : массив длины N или скалярное значение - набор данных для оси ординат – вторая кривая.

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.

step : {‘pre’, ‘post’, ‘mid’}, optional - определяет шаг, если используется step- функция для отображения графика (будет рассмотрена далее в данной лабораторной работе).

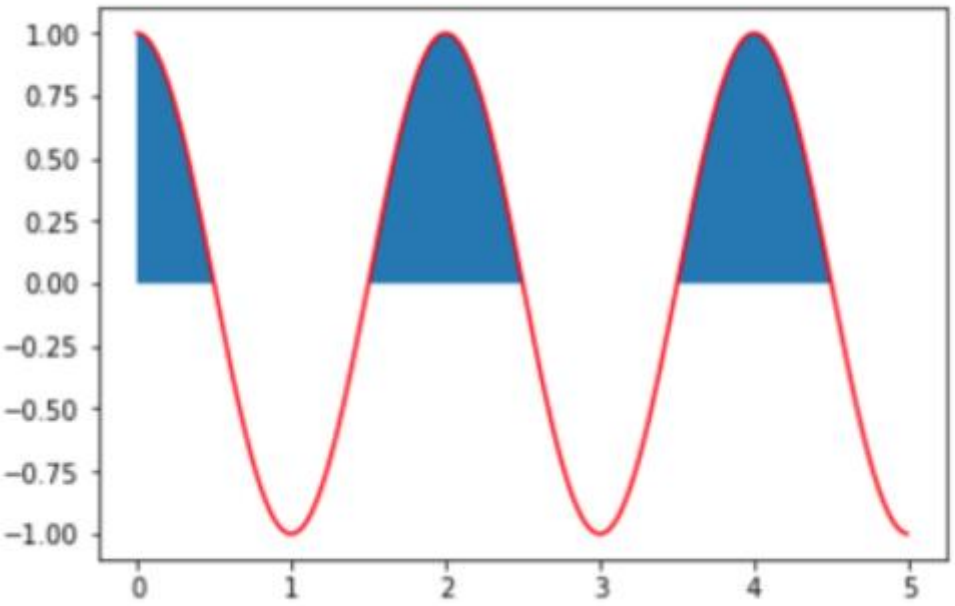
**kwargs - свойства класса Polygon (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.patches.Polygon.html#matplotlib.patches.Polygon)

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

Используя параметры y1 и y2 можно формировать более сложные решения.

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")
```

В наборе данных, который создает код:

```
import numpy as np
x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)
```

количество точек составляет 500, поэтому подход, представленный выше уже будет не применим:

```
plt.plot(x, y, marker="o", c="g")
```

В этой случае нужно задать интервал отображения маркеров, для этого используется параметр markevery, который может принимать одно из следующих значений:

None – отображаться будет каждая точка;

N – отображаться будет каждая N-я точка;

(start, N) – отображается каждая N-я точка начиная с точки start;

slice(start, end, N) – отображается каждая N-я точка в интервале от start до end;

[i, j, m, n] – будут отображены только точки i, j, m, n.

6. Как выполнить обрезку графиков?

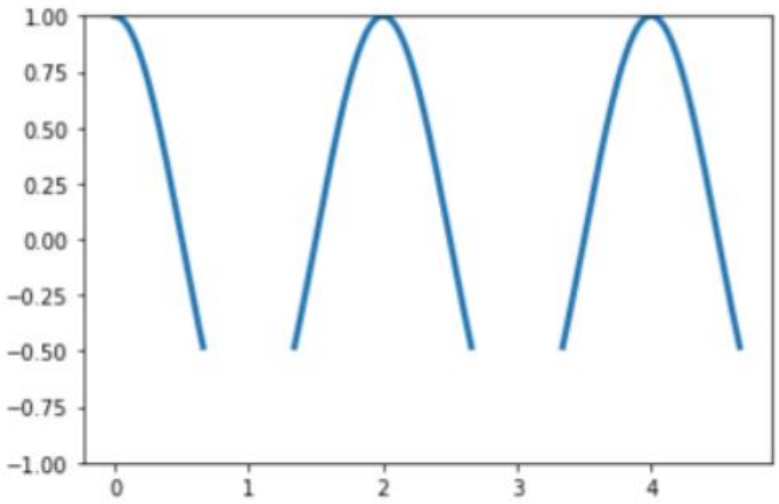
Для того, чтобы отобразить только часть графика, которая отвечает определенному условию используйте предварительное маскирование данных

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.

‘pre’: значение y ставится слева от значения x, т.е. значение y[i] определяется для интервала (x[i-1]; x[i]).

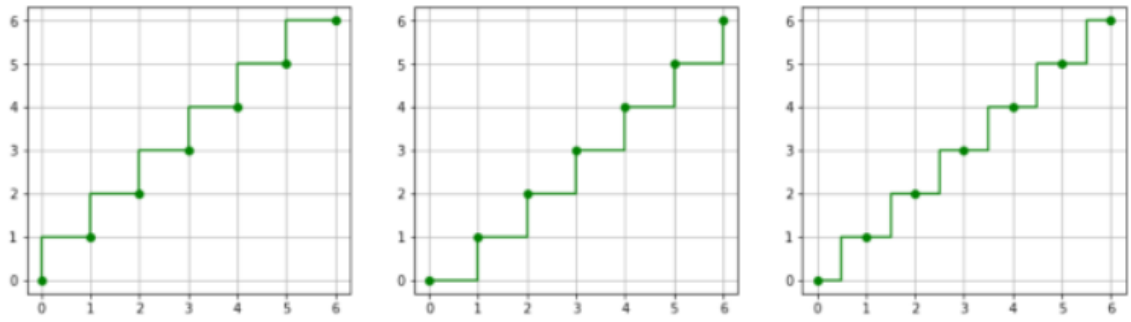
‘post’: значение y ставится справа от значения x, т.е. значение y[i] определяется дляинтервала (x[i]; x[i+1]).

‘mid’: значение y ставится в середине интервала.

```
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:

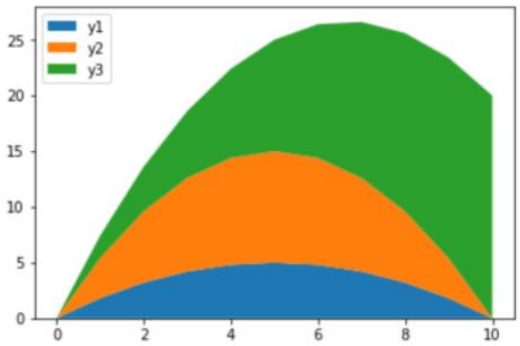
```
x = np.arange(0, 11, 1)

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')
```

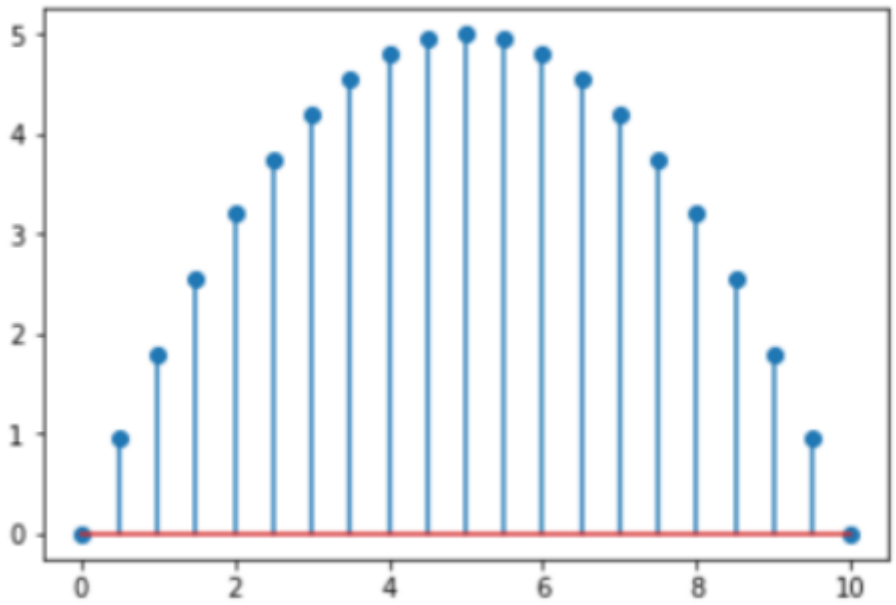


Верхний край области y2 определяется как сумма значений из наборов y1 и y2, y3 – соответственно сумма y1, y2 и y3.

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:

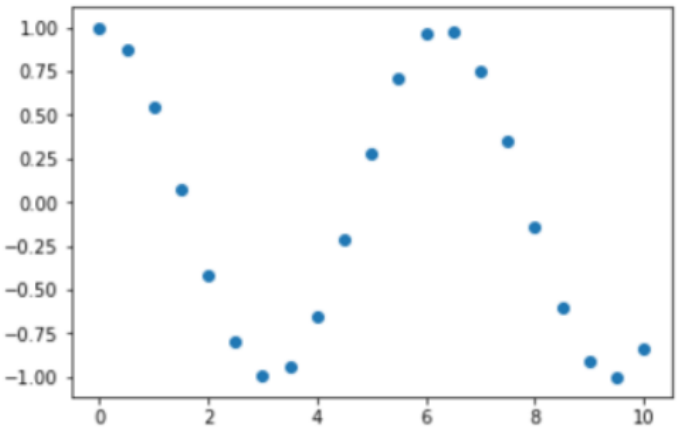
```
x = np.arange(0, 10.5, 0.5)
y = np.array([(-0.2)*i**2+2*i for i in x])
plt.stem(x, y)
```



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)
```



11. Как осуществляется построение столбчатых диаграмм с помощью matplotlib?

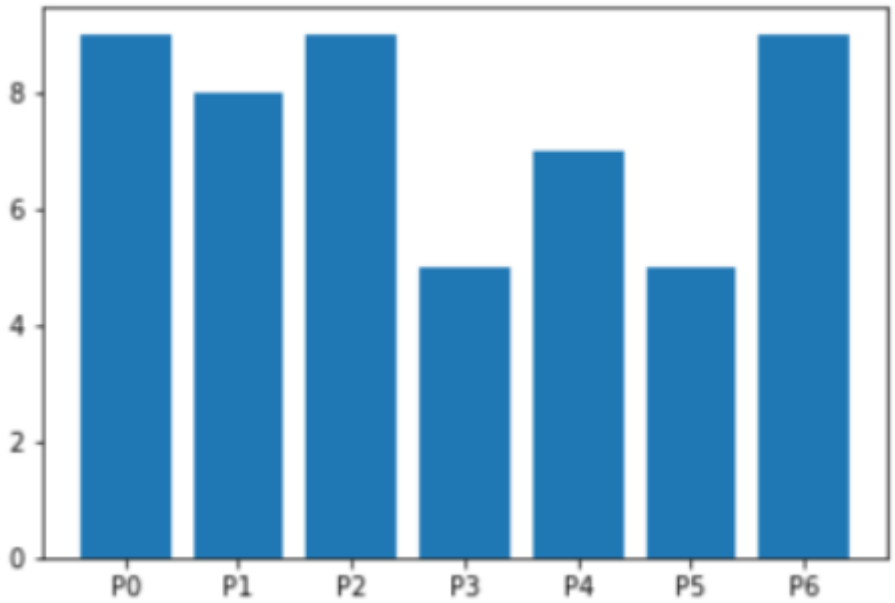
Для визуализации категориальных данных хорошо подходят столбчатые диаграммы. Для их построения используются функции:

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)
```



12. Что такое групповая столбчатая диаграмма? Что такое столбчатая диаграмма с errorbar элементом?

Используя определенным образом подготовленные данные можно строить групповые диаграммы:

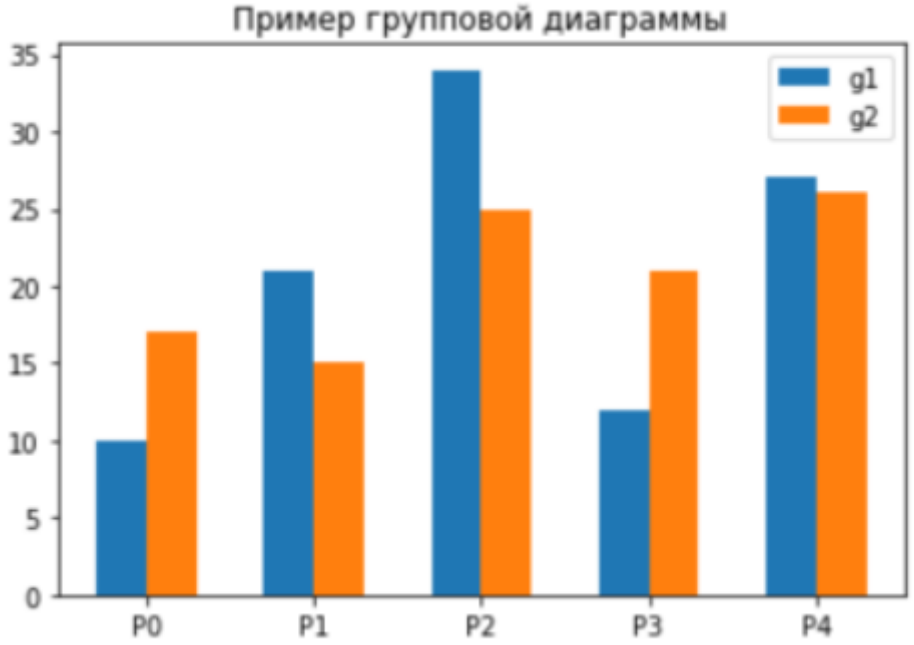
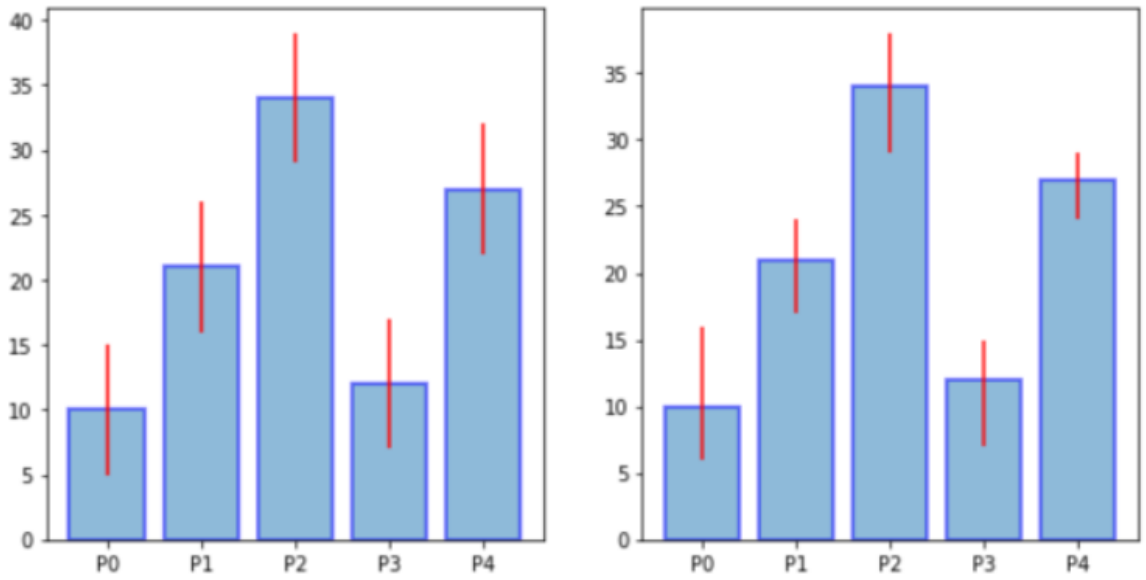


Диаграмма с errorbar элементом

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)
```

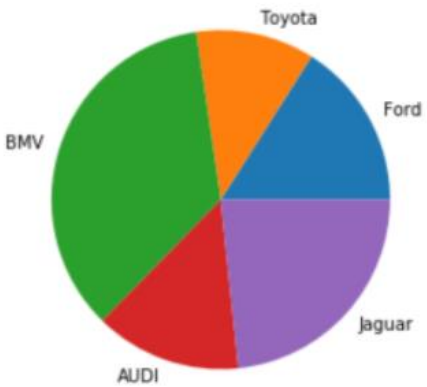


13. Как выполнить построение круговой диаграммы средствами matplotlib?

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?

Цветовые карты (colormaps)

Цветовая карта представляет собой подготовленный набор цветов, который хорошо подходит для визуализации того или иного набора данных.

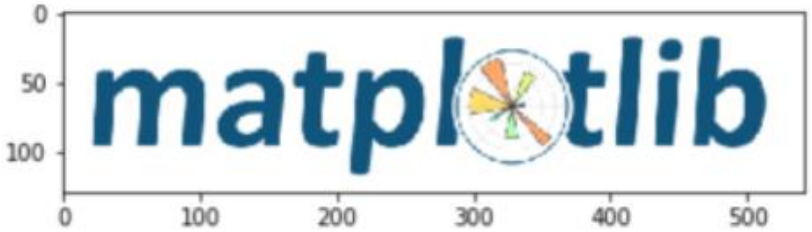
Также отметим, что такие карты можно создавать самостоятельно, если среди существующих нет подходящего решения.

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)
```

В результате получим изображение логотипа *Matplotlib*.



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 отличии от нее рассматриваемая нами pcolormesh() более быстрая и является лучшим вариантом в большинстве лучаев. Функция pcolormesh() похожа по своим возможностям на imshow(), но есть и отличия.

Рассмотрим параметры функции pcolormesh():

C : массив - 2D массив скалярных значений

cmap : str или Colormap, optional - см. cmap в imshow()

norm : Normalize, optional - см. norm в imshow()

vmin , vmax : scalar, optional, значение по умолчанию: None - см. vmin, vmax в imshow()

edgecolors : {‘none’, None, ‘face’, color, color sequence}, optional - цвет границы, по

умолчанию: ‘none’, возможны следующие варианты:

‘none’ or ”: без отображения границы.

None: черный цвет.

‘face’: используется цвет ячейки.

Можно выбрать цвет из доступных наборов.

alpha : scalar, optional, значение по умолчанию: None - см. alpha в imshow().

shading : {‘flat’, ‘gouraud’}, optional - стиль заливки, доступные значения:

‘flat’: сплошной цвет заливки для каждого квадрата.

‘gouraud‘: для каждого квадрата будет использован метод затенения Gouraud.

snap : bool, optional, значение по умолчанию: False - привязка сетки к границам пикселей.

Пример использования функции pcolormesh():

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