

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

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

**Кафедра инфокоммуникаций
«Основы работы с пакетом matplotlib»**

**Отчет по лабораторной работе № 3.5
по дисциплине «Визуализация данных с помощью matplotlib»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности библиотеки matplotlib языка программирования Python

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

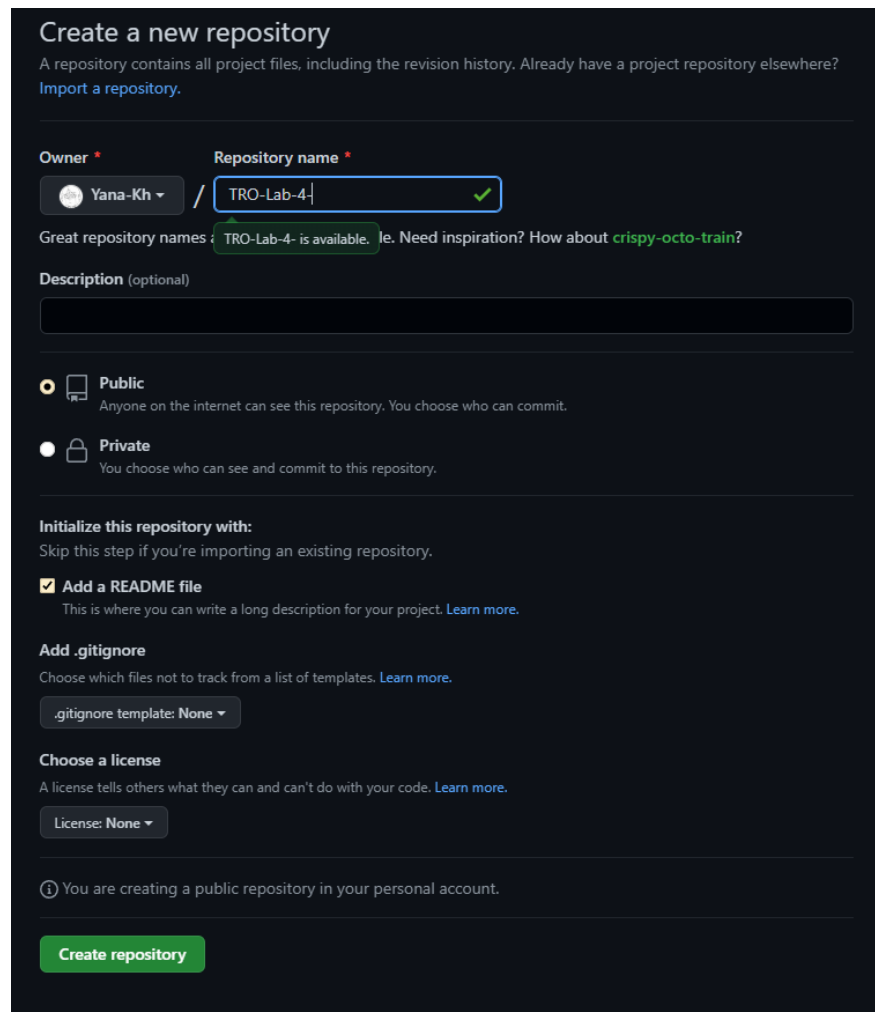


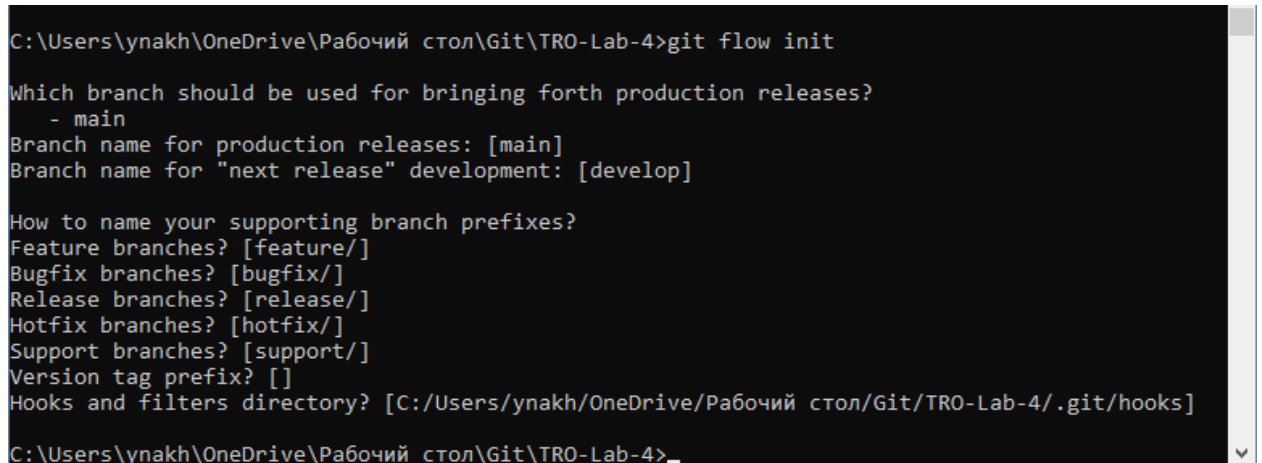
Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/TRO-Lab-4.git
Cloning into 'TRO-Lab-4'...
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

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

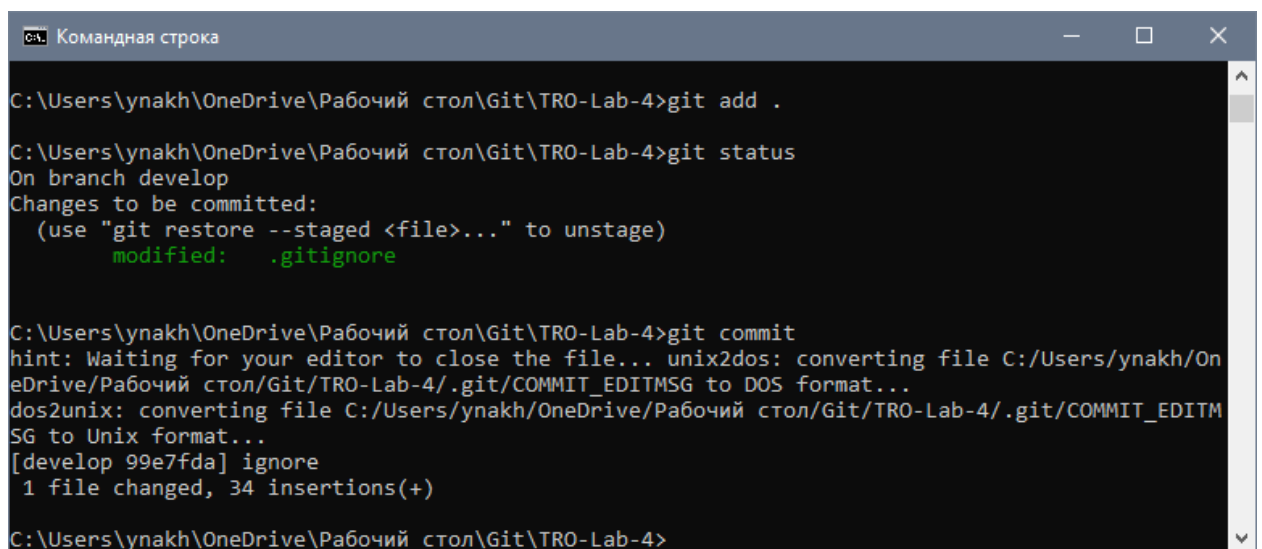


```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

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/ynakh/OneDrive/Рабочий стол/Git/TR0-Lab-4/.git/hooks]
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.



```
Командная строка
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>git add .
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>git commit
hint: Waiting for your editor to close the file... unix2dos: converting file C:/Users/ynakh/OneDrive/Рабочий стол/Git/TR0-Lab-4/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/ynakh/OneDrive/Рабочий стол/Git/TR0-Lab-4/.git/COMMIT_EDITMSG to Unix format...
[develop 99e7fda] ignore
1 file changed, 34 insertions(+)
C:\Users\ynakh\OneDrive\Рабочий стол\Git\TR0-Lab-4>
```

Рисунок 4 – Дополнение файла .gitignore

6. Проработать примеры лабораторной работы.

Лабораторная работа №5 Визуализация данных с помощью matplotlib

Построение линейного графика

```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: 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)
```

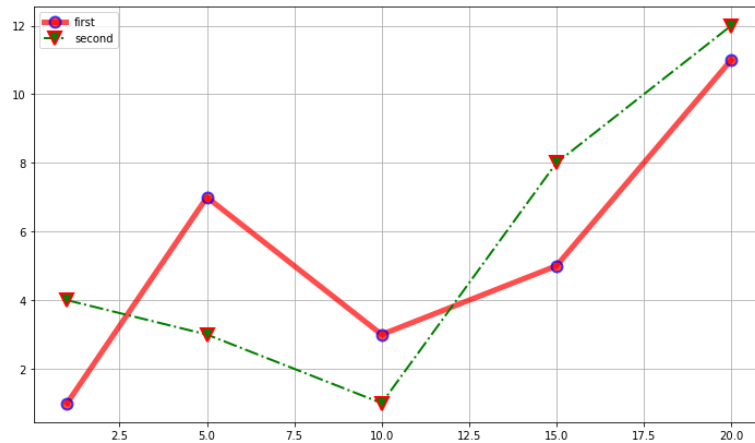


Рисунок 5 – Проработка примеров лабораторной работы

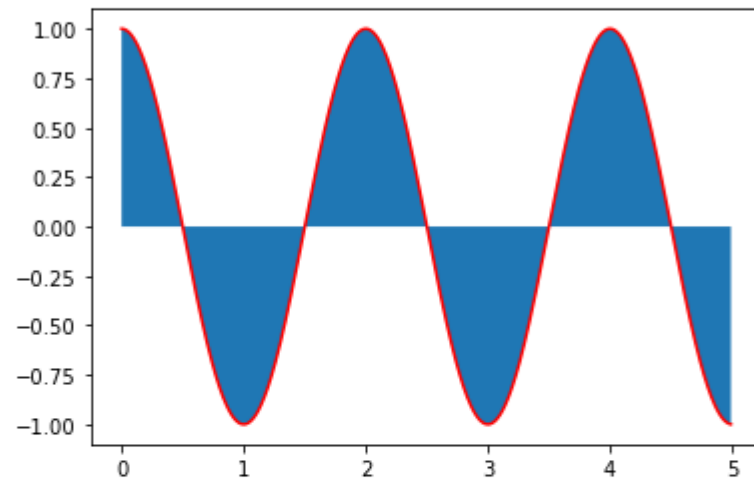
Заливка области между графиком и осью

```
In [6]: import numpy as np

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)
```

Out[6]: <matplotlib.collections.PolyCollection at 0x2e70e6e31f0>



```
In [7]: plt.plot(x, y, c="r")
plt.fill_between(x, y, where=(y > 0.75) | (y < -0.75))
```

Out[7]: <matplotlib.collections.PolyCollection at 0x2e70e7494c0>

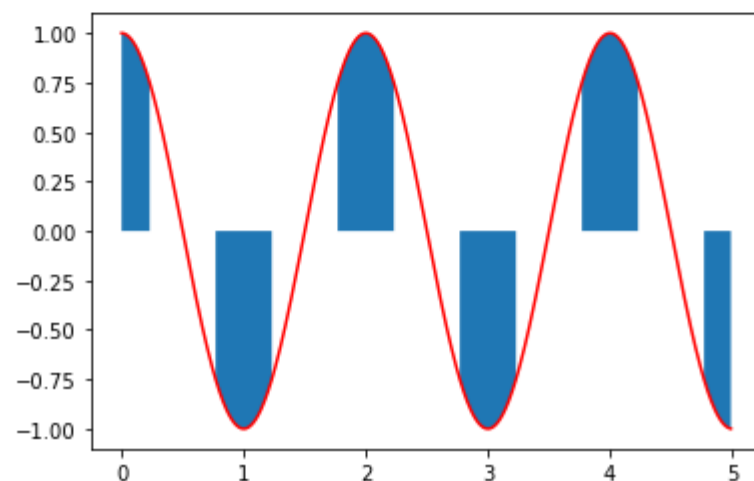
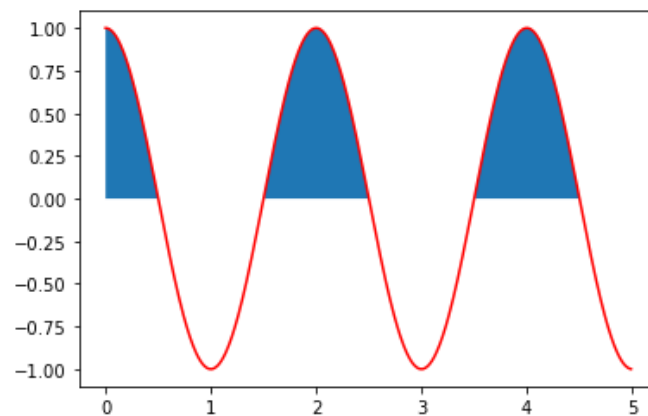


Рисунок 6 – Проработка примеров лабораторной работы

```
In [8]: plt.plot(x, y, c="r")  
plt.fill_between(x, y, where=(y > 0))
```

Out[8]: <matplotlib.collections.PolyCollection at 0x2e70e7b1a30>



```
In [9]: plt.plot(x, y, c="r")  
plt.grid()  
plt.fill_between(x, 0.5, y, where=(y >= 0.5))
```

Out[9]: <matplotlib.collections.PolyCollection at 0x2e70e82c3d0>

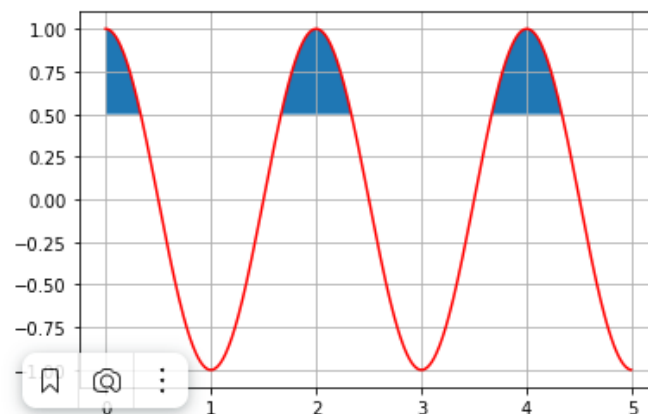
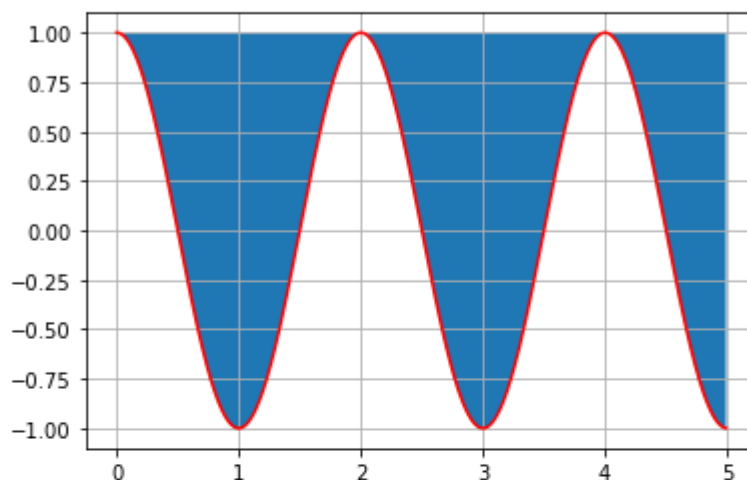


Рисунок 7 – Проработка примеров лабораторной работы

```
In [10]: plt.plot(x, y, c="r")  
plt.grid()  
plt.fill_between(x, y, 1)
```

Out[10]: <matplotlib.collections.PolyCollection at 0x2e70e891f70>



```
In [11]: 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)
```

Out[11]: <matplotlib.collections.PolyCollection at 0x2e70e9546d0>

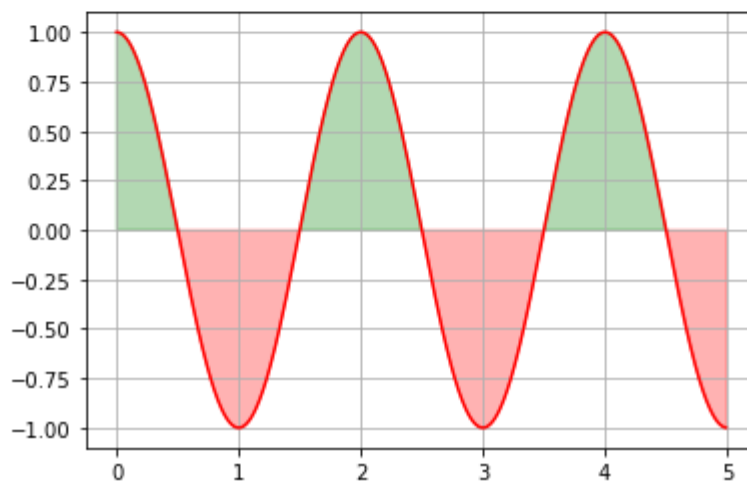
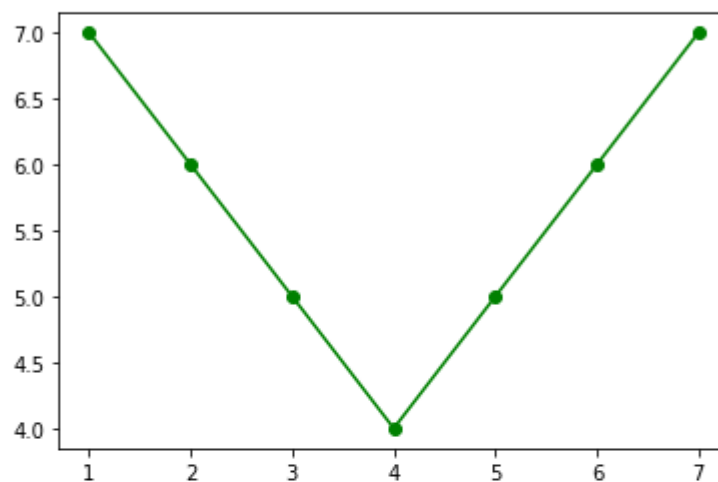


Рисунок 8 – Проработка примеров лабораторной работы

Настройка маркировки графиков

```
In [12]: x = [1, 2, 3, 4, 5, 6, 7]
y = [7, 6, 5, 4, 5, 6, 7]
plt.plot(x, y, marker="o", c="g")
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x2e70e9cc370>]
```



```
In [14]: x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)

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

```
Out[14]: [<matplotlib.lines.Line2D at 0x2e70ea95970>]
```

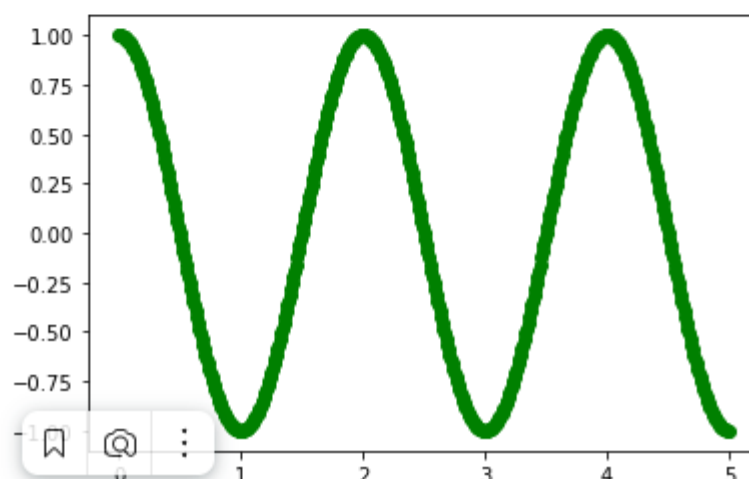


Рисунок 9 – Проработка примеров лабораторной работы


```
In [16]: x = np.arange(0.0, 5, 0.01)
y = np.cos(x * np.pi)
m_ev_case = [None, 10, (100, 30), slice(100,400,15), [0, 100, 200, 300], [10, 50, 100]]
fig, ax = plt.subplots(2, 3, figsize=(10, 7))
axs = [ax[i, j] for i in range(2) for j in range(3)]
for i, case in enumerate(m_ev_case):
    axs[i].set_title(str(case))
    axs[i].plot(x, y, "o", ls='-', ms=7, markevery=case)
```

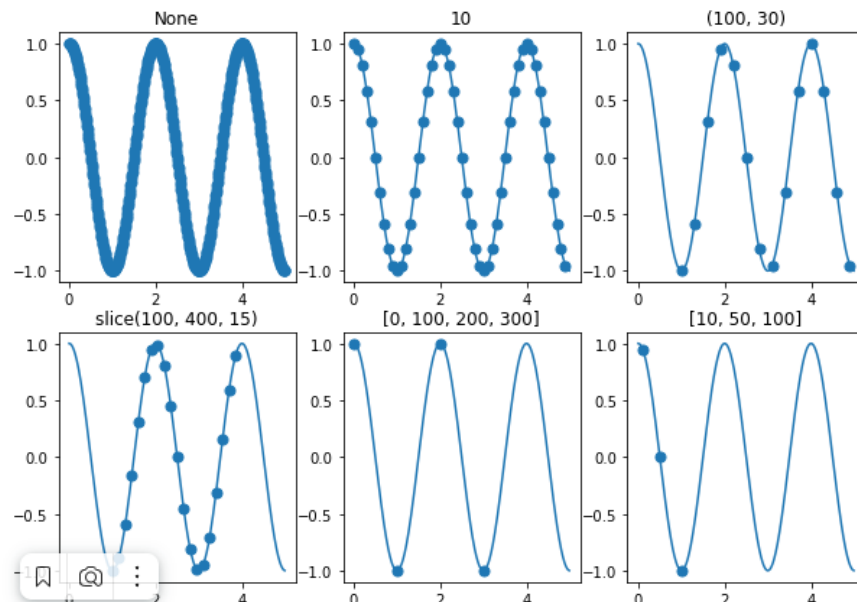


Рисунок 10 – Проработка примеров лабораторной работы

Обрезка графика

```
In [17]: 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)
```

Out[17]: [

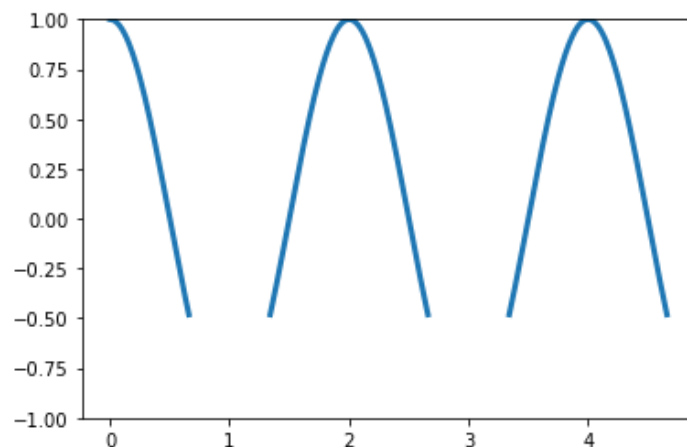


Рисунок 11 – Проработка примеров лабораторной работы

Ступенчатый график

```
In [18]: 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()
```

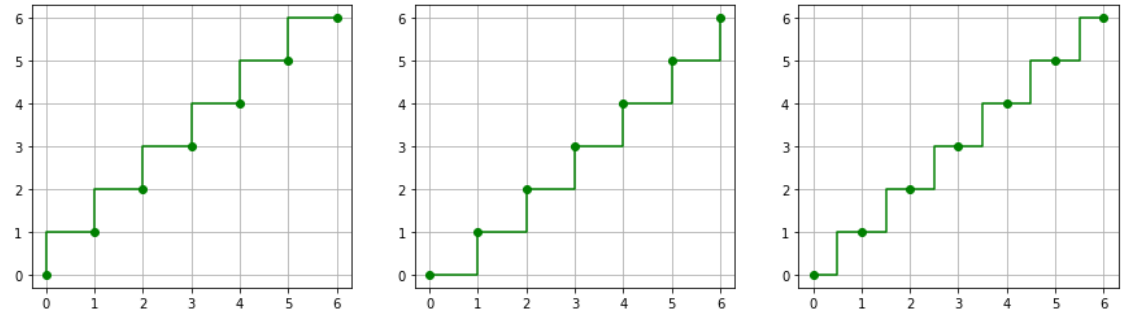


Рисунок 12 – Проработка примеров лабораторной работы

Стековый график

```
In [19]: 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')
```

Out[19]: <matplotlib.legend.Legend at 0x2e70ffd0c40>

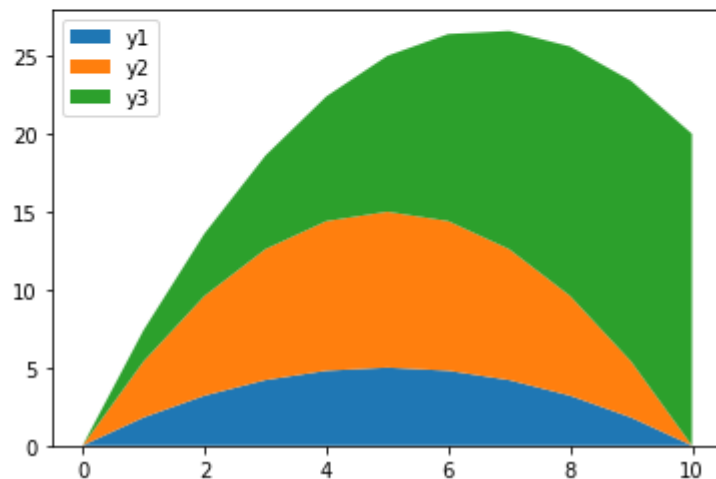
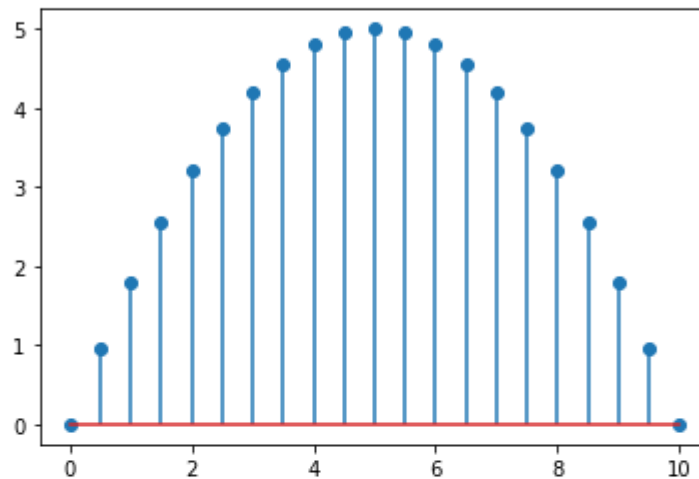


Рисунок 13 – Проработка примеров лабораторной работы

Stem-график

```
In [20]: 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)
```

Out[20]: <StemContainer object of 3 artists>



```
In [21]: plt.stem(x, y, linefmt="r--", markerfmt="^", bottom=1)
```

Out[21]: <StemContainer object of 3 artists>

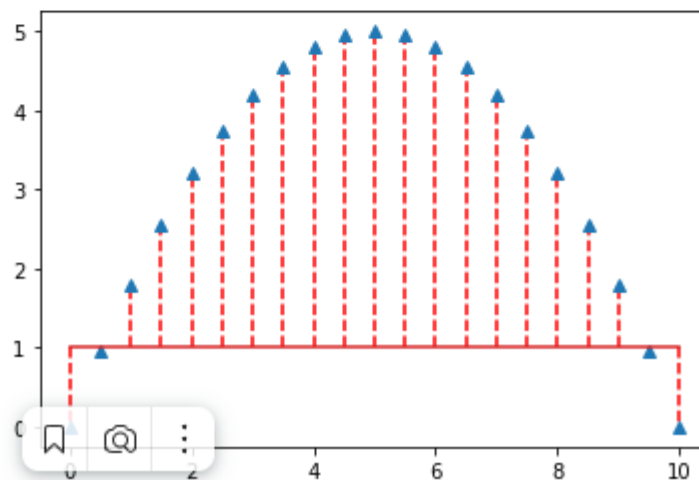
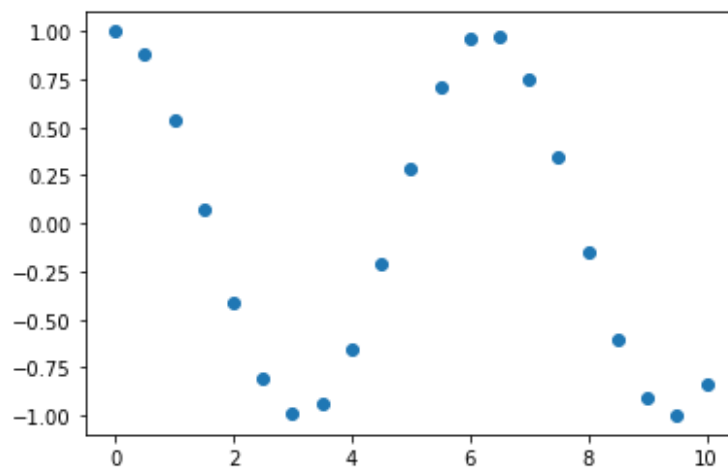


Рисунок 14 – Проработка примеров лабораторной работы

Точечный график

```
In [22]: x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y)
```

Out[22]: <matplotlib.collections.PathCollection at 0x2e71001ac40>



```
In [23]: x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y, s=80, c="r", marker="D", linewidths=2, edgecolors="g")
```

Out[23]: <matplotlib.collections.PathCollection at 0x2e71008df40>

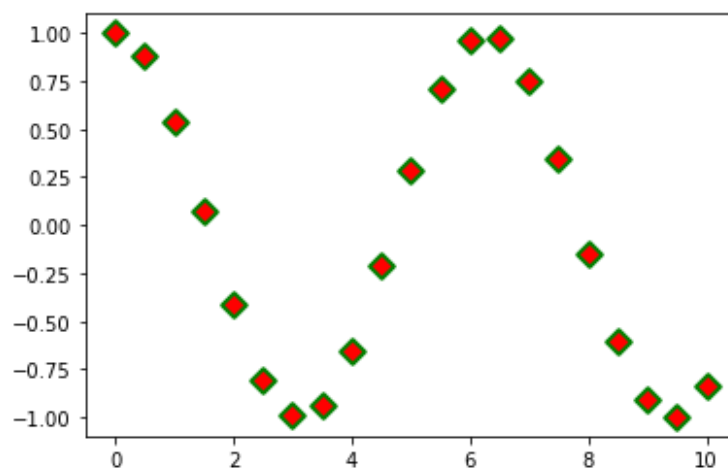


Рисунок 15 – Проработка примеров лабораторной работы

```
In [24]: import matplotlib.colors as mcolors
bc = mcolors.BASE_COLORS
x = np.arange(0, 10.5, 0.25)
y = np.cos(x)
num_set = np.random.randint(1, len(mcolors.BASE_COLORS), len(x))
sizes = num_set * 35
colors = [list(bc.keys())[i] for i in num_set]
plt.scatter(x, y, s=sizes, alpha=0.4, c=colors, linewidths=2, edgecolors="face")
plt.plot(x, y, "g--", alpha=0.4)
```

Out[24]: [matplotlib.lines.Line2D at 0x2e710100ee0]

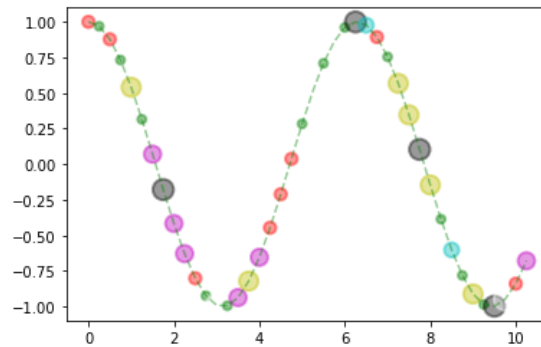
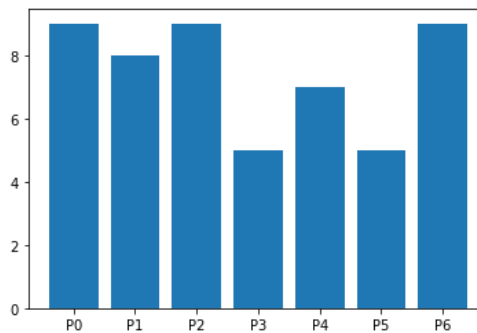


Рисунок 16 – Проработка примеров лабораторной работы

Столбчатые диаграммы

```
In [25]: 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)
```

Out[25]: <BarContainer object of 7 artists>



```
In [26]: plt.barh(groups, counts)
```

Out[26]: <BarContainer object of 7 artists>

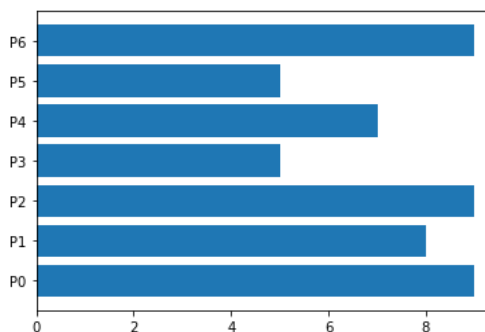


Рисунок 17 – Проработка примеров лабораторной работы

```
In [27]: import matplotlib.colors as mcolors

bc = mcolors.BASE_COLORS

np.random.seed(123)

groups = [f"P{i}" for i in range(7)]
counts = np.random.randint(0, len(bc), len(groups))

width = counts*0.1

colors = [r, b, g][int(np.random.randint(0, 3, 1))] for _ in counts]
plt.bar(groups, counts, width=width, alpha=0.6, bottom=2, color=colors,
edgecolor="k", linewidth=2)
```

Out[27]: <BarContainer object of 7 artists>

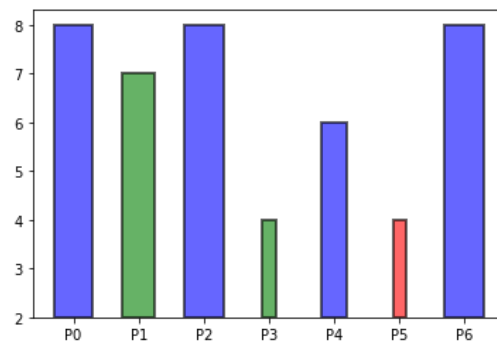


Рисунок 18 – Проработка примеров лабораторной работы

Групповые столбчатые диаграммы

```
In [28]: 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()
```

Out[28]: <matplotlib.legend.Legend at 0x2e71028cd90>

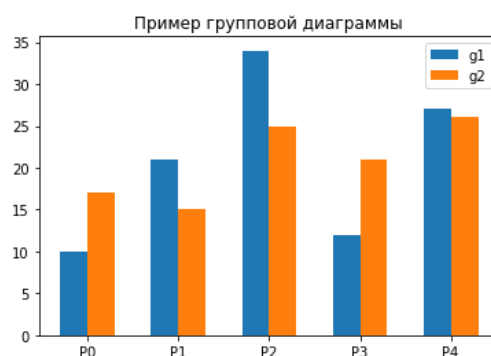


Рисунок 19 – Проработка примеров лабораторной работы

Диаграмма с errorbar элементом

```
In [29]: 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)
```

Out[29]: <BarContainer object of 5 artists>

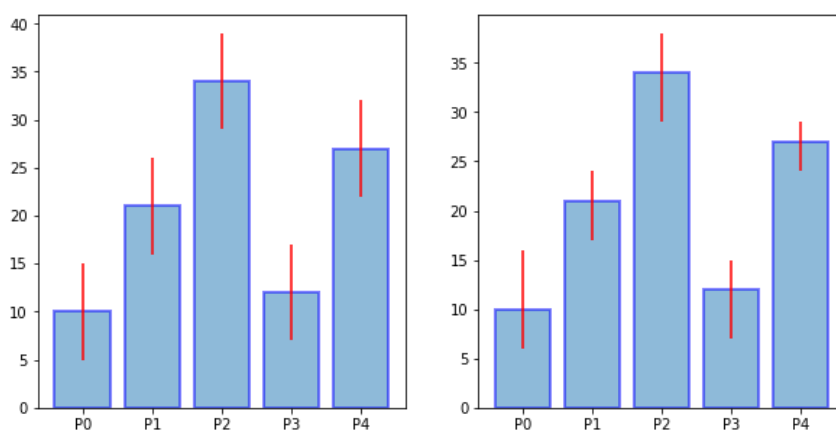


Рисунок 20 – Проработка примеров лабораторной работы

Круговые диаграммы

```
In [30]: vals = [24, 17, 53, 21, 35]

labels = ["Ford", "Toyota", "BMW", "AUDI", "Jaguar"]

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

Out[30]: (-1.1163226287452406,
1.1007772680354877,
-1.1107362350259515,
1.1074836529113834)

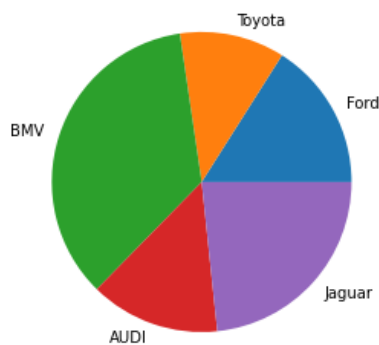


Рисунок 21 – Проработка примеров лабораторной работы

```

In [31]: vals = [24, 17, 53, 21, 35]

labels = ["Ford", "Toyota", "BMW", "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")

```

```

Out[31]: (-1.2704955621219602,
1.1999223938155328,
-1.1121847055183558,
1.1379015332518725)

```

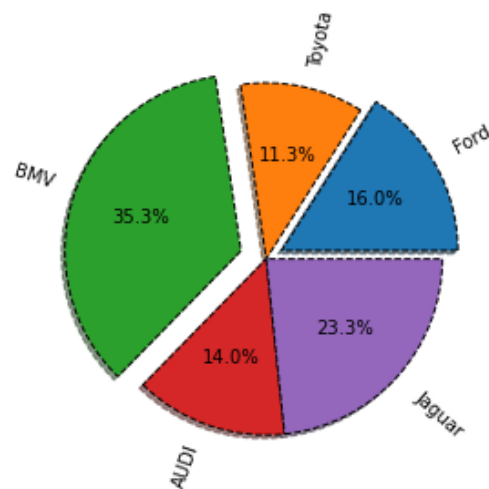


Рисунок 22 – Проработка примеров лабораторной работы


```
In [32]: 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'))
```

```
Out[32]: ([<matplotlib.patches.Wedge at 0x2e70fccd1c0>,
<matplotlib.patches.Wedge at 0x2e70fccdee0>,
<matplotlib.patches.Wedge at 0x2e70fccdc40>,
<matplotlib.patches.Wedge at 0x2e70e9fa670>,
<matplotlib.patches.Wedge at 0x2e70eaf6820>,
<matplotlib.patches.Wedge at 0x2e70eaf69d0>,
<matplotlib.patches.Wedge at 0x2e70eaf6dc0>,
<matplotlib.patches.Wedge at 0x2e70eaf6160>,
<matplotlib.patches.Wedge at 0x2e70fc462b0>],
[Text(0.646314344414094, 0.13370777166859046, ''),
Text(0.4521935266177387, 0.48075047008298655, ''),
Text(0.040366679721656945, 0.6587643973138266, ''),
Text(-0.34542288787409087, 0.5623904591409097, ''),
Text(-0.6578039053946477, 0.05379611554331286, ''),
Text(-0.48987451889717687, -0.44229283934431896, ''),
Text(-0.12049606360635531, -0.6489073112975174, ''),
Text(0.39011356818311405, -0.532363976917521, ''),
Text(0.6332653697075483, -0.1859434632601054, '')])
```



Рисунок 23 – Проработка примеров лабораторной работы

```

In [33]: vals = [24, 17, 53, 21, 35]

labels = ["Ford", "Toyota", "BMV", "AUDI", "Jaguar"]

fig, ax = plt.subplots()
ax.pie(vals, labels=labels, wedgeprops=dict(width=0.5))

Out[33]: ([<matplotlib.patches.Wedge at 0x2e7113c9af0>,
<matplotlib.patches.Wedge at 0x2e7113c9f40>,
<matplotlib.patches.Wedge at 0x2e7113d3460>,
<matplotlib.patches.Wedge at 0x2e7113d3940>,
<matplotlib.patches.Wedge at 0x2e7113d3e20>],
[Text(0.9639373540021144, 0.5299290306818474, 'Ford'),
Text(0.22870287165240302, 1.075962358309037, 'Toyota'),
Text(-1.046162158377023, 0.3399187231970734, 'BMV'),
Text(-0.3617533684721028, -1.0388139873909512, 'AUDI'),
Text(0.8174592712713289, -0.7360437078139777, 'Jaguar')])

```

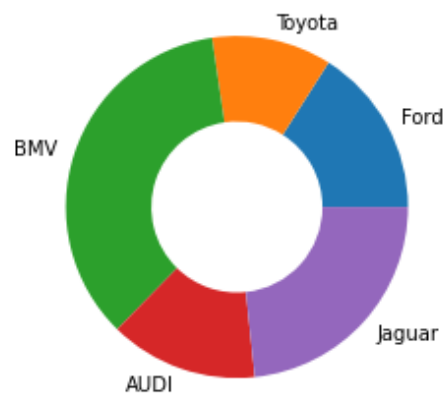


Рисунок 24 – Проработка примеров лабораторной работы

Цветовые карты (colormaps). Построение цветовой сетки

Отображение изображений

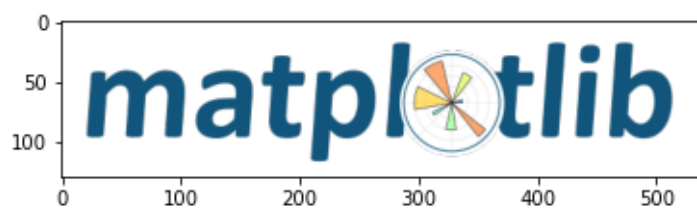
```
In [34]: from PIL import Image
import requests

from io import BytesIO

response = requests.get('https://matplotlib.org/_static/logo2.png')
img = Image.open(BytesIO(response.content))

plt.imshow(img)
```

Out[34]: <matplotlib.image.AxesImage at 0x2e7114dff10>



```
In [35]: np.random.seed(19680801)

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

Out[35]: <matplotlib.image.AxesImage at 0x2e711550730>

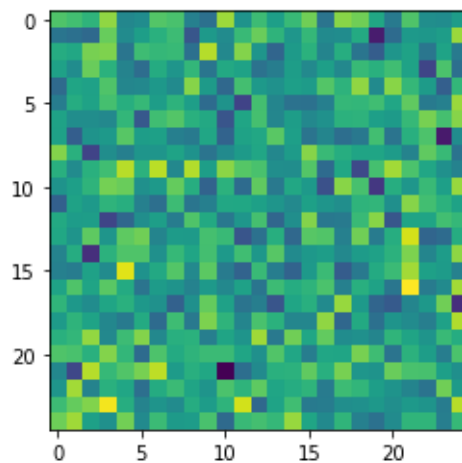


Рисунок 25 – Проработка примеров лабораторной работы

```
In [36]: fig, axs = plt.subplots(1, 2, figsize=(10,3), constrained_layout=True)

p1 = axs[0].imshow(data, cmap='winter', aspect='equal', vmin=-1, vmax=1,
origin="lower")
fig.colorbar(p1, ax=axs[0])

p2 = axs[1].imshow(data, cmap='plasma', aspect='equal',
interpolation='gaussian', origin="lower", extent=(0, 30, 0, 30))
fig.colorbar(p2, ax=axs[1])
```

Out[36]: <matplotlib.colorbar.Colorbar at 0x2e711596f70>

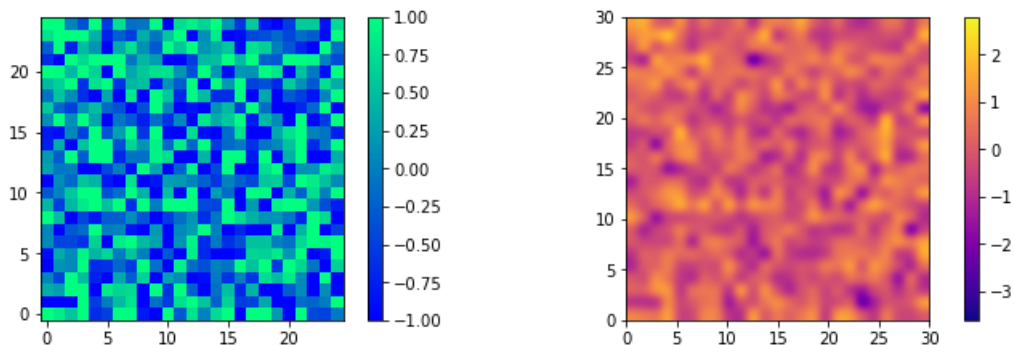


Рисунок 26 – Проработка примеров лабораторной работы

Отображение тепловой карты

```
In [37]: np.random.seed(123)

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

Out[37]: <matplotlib.collections.QuadMesh at 0x2e71173afd0>

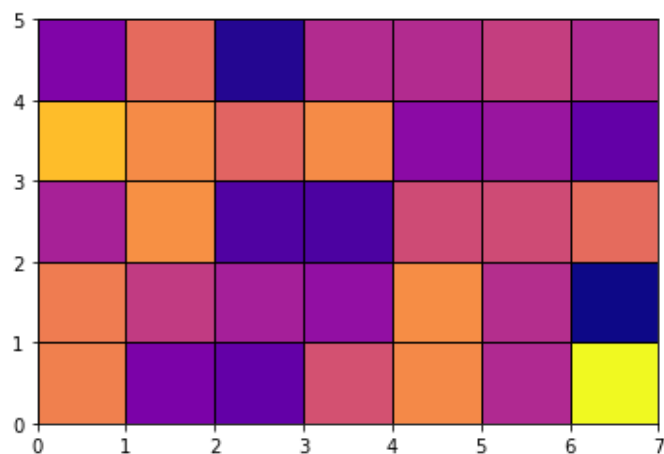


Рисунок 27 – Проработка примеров лабораторной работы

7. 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.

Задача:

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.

Условие:

Определить падение напряжения на полностью включенном реостате, изготовленном из никелинового провода, длиной 7,5 м. Плотность тока равна 0,5; 1,0; 1,5; 2,0; 2,5; 3,0 А/мм²

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

Дано: $L = 7,5 \text{ м}$; $j = 1,0 - 3,0 \text{ А/мм}^2$, $U = ?$

Решение задачи:

Согласно закону Ома для участка цепи сила тока I прямо пропорциональна напряжению на участке U и обратно пропорциональна сопротивлению этого участка R , поэтому:

$$I = \frac{U}{R}$$

Откуда имеем:

$$U = IR$$

Плотность тока j по определению равна отношению силы тока I на площадь поперечного сечения проводника S , то есть:

$$j = \frac{I}{S}$$

Тогда:

$$I = jS$$

Сопротивление проводника R , изготовленного из никелинового провода, легко определить по такой известной формуле (здесь ρ – удельное электрическое сопротивление никелина, равное 420 нОм·м):

$$R = \rho \frac{L}{S}$$

Подставим формулы и тогда:

$$U = jS\rho \frac{L}{S}$$
$$U = j\rho L$$

Задача решена в общем виде, теперь необходимо лишь подставить данные задачи в полученную формулу и посчитать ответ

Рисунок 28 – Результат работы

```
L = 7.5
j = 10**(6)
p = 420 * 10**(-9)
current_density = []
voltage = []
i = 0.5
```

Создадим цикл, в котором будем подсчитывать напряжение в зависимости от разной плотности тока

```
while i <= 3:
    U = i*j*p*L
    current_density.append(i)
    voltage.append(U)
    i += 0.5
```

Рисунок 29 – Результат работы

```
plt.title("Зависимость падение напряжения на полностью включенном реостате от плотности тока")
plt.xlabel("Плотность тока")
plt.ylabel("Напряжение")
plt.plot(current_density, voltage, '*-b', alpha=0.4, lw=2, mec='b', mew=2, ms=10)
```

[<matplotlib.lines.Line2D at 0x7f0908427d00>]

Зависимость падение напряжения на полностью включенном реостате от плотности тока

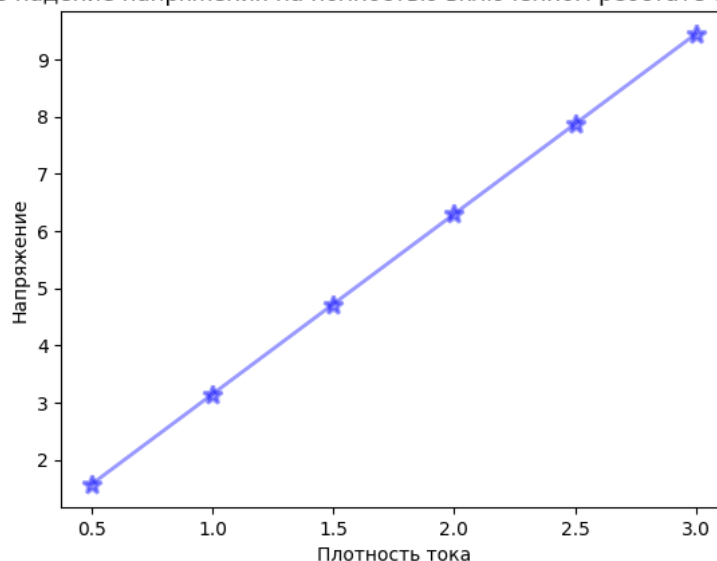


Рисунок 30 – Результат работы

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

Задача:

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

Условие:

Написать программу, которая вычисляет количество платежей по аннуитетному кредиту при вводе пользователем суммы кредита, процентной ставки и максимального платежа

Решение задачи:

```
import matplotlib.pyplot as plt
```

Для решения задачи введем переменные: S - сумма кредита, r - процент, b - множитель для нахождения новой суммы кредита после начисления процента; groups и counts переменные для хранения названий и значений столбцов для построения диаграммы

```
sum_cred = int(input("Введите сумму кредита: "))
r = float(input("Введите процентную ставку: "))
b = r/100 + 1.0
payment = int(input("Введите максимально возможный платеж: "))
groups = []
counts = []
i = 1
```

Введите сумму кредита: 100000
Введите процентную ставку: 12
Введите максимально возможный платеж: 15000

Рисунок 31 – Результат работы

Создадим цикл, в котором будем преобразовывать сумму кредита в соответствии с процентной ставкой и вычитая максимально возможный платеж

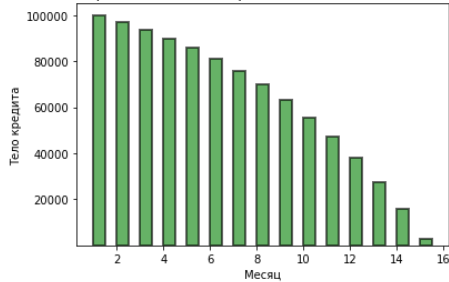
```
: S = sum_cred
while S > 0:
    groups.append(i)
    counts.append(S)
    S = S * b - payment
    i += 1
```

Составим столбчатую диаграмму, отражающую изменение суммы кредита после каждого платежа

```
: plt.title(f"Отношение тела кредита к месяцам при максимально возможном платеже: {payment}")
plt.xlabel("Месяц")
plt.ylabel("Тело кредита")
plt.bar(groups, counts, color = "green", width = 0.5, edgecolor="black", alpha=0.6, align = 'edge', bottom = 20, linewidth=2.0)
```

: <BarContainer object of 15 artists>

Отношение тела кредита к месяцам при максимально возможном платеже: 15000



```
: print(f"При взятии кредита на сумму {sum_cred}, под {r}% с выплатой до {payment} рублей минимальное количество платежей = {i - 1}
```

При взятии кредита на сумму 100000, под 12.0% с выплатой до 15000 рублей минимальное количество платежей = 15

Рисунок 32 – Результат работы

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

Задача:

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

Условие:

Необходимо проанализировать данные о типах личности пользователей твиттера, сделать выводы относительно преобладающих черт.

Решение задачи:

Теория:

Индикаторы типа Майерса-Бриггса (MBTI) - одна из самых популярных моделей личности, которая создает бинарную классификацию на основе четырех различных измерений и выдает 16 возможных типов личности в зависимости от комбинации этих четырех значений.

- Интроверсия / экстраверсия: первое измерение связано с энергией человека. Экстраверты предпочитают вкладывать свою энергию в общение с людьми, обстоятельствами или внешним миром. Интроверты предпочитают концентрировать свою энергию на работе с идеями, фактами, объяснениями, убеждениями или внутренним миром.

- Интуиция / восприятие: второе измерение касается того, как обрабатывается информация. Восприимчивый - это тип человека, который хочет смотреть на факты, опираясь на то, что известно. Человек с интуицией склонен экспериментировать с идеями, исследовать неизвестное.

- Чувства / мышление: третье измерение связано с принятием решений. Думающие люди принимают решения, основываясь на объективных рассуждениях и независимой точке зрения. Тип людей, которые предпочитают использовать ценности, - это чувства.

- Восприятие / суждение: Последнее измерение связано с выбранным образом жизни. Люди, склонные к суждениям, предпочитают, чтобы их жизнь была спланирована и структурирована. Восприятие - это тип человека, который предпочитает плыть по течению, быть гибким и реагировать на события по мере их возникновения.

Рисунок 34 – Результат работы

```
]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

По условию, существует 8 различных черт, в зависимости от сочетания которых, можно разделить людей на 16 групп.
Для начала проанализируем, какую долю от выборки составляют люди той или иной группы

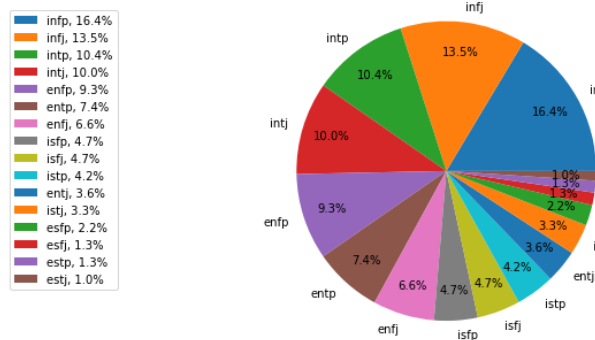
```
]: data = pd.read_csv('data.csv')
```

```
]: #Выберем значения
label_counts = data["label"].value_counts()
vals = label_counts.values
labels = label_counts.index

#Создадим круговую диаграмму
fig, ax = plt.subplots()
fig.subplots_adjust(0.3,0,1,1)
total = sum(vals)
ax.pie(vals, labels=labels, autopct='%1.1f%%', pctdistance=0.8, labeldistance= 1.1)
ax.legend(labels=['%s, %1.1f%%' % (l, (float(s) / total) * 100) for l, s in zip(labels, vals)],
          bbox_to_anchor=(0.0, 1),
          bbox_transform=fig.transFigure)
ax.axis("equal")
```

7811

```
]: (-1.1020387279441433,
1.1000970822830545,
-1.101632263181789,
1.1102406582440016)
```



Как мы видим, больше всего представителей группы infp, что трактуется как интровертный, интуитивный, чувствительный и склонный к суждениям тип, второе место занял infj, который отличается только последней характеристикой, которая трактуется как человек, предпочитающий плыть по течению, быть гибким и реагировать на события по мере их возникновения

Рисунок 35 – Результат работы

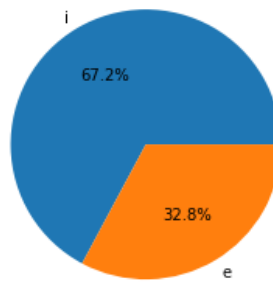
Проанализируем теперь процентное отношении по каждому из измерений

```
# Переменная для перехода
group = 0
#Создадим два списка с 4-мя различными элементами для разбиения каждого типов
first= ['i', 'n', 'f', 'p']
second = ['e', 's', 't', 'j']

while group < 4:
    #Создадим две переменные для подсчета по подгруппам
    fitst_n = 0
    second_n = 0
    # Проанализируем каждую из 16 групп
    for ind, i in enumerate(labels):
        if i[group] in first:
            fitst_n += vals[ind]
        else:
            second_n += vals[ind]
    # Построим диаграмму
    plt.pie([fitst_n, second_n],
            labels=[first[group], second[group]],
            autopct='%1.1f%%',)
    plt.title(f"Процентное соотношение по чертам {first[group]} и {second[group]}")
    plt.show()
    group += 1
```

Рисунок 36 – Результат работы

Процентное соотношение по чертам i и e



Процентное соотношение по чертам n и s

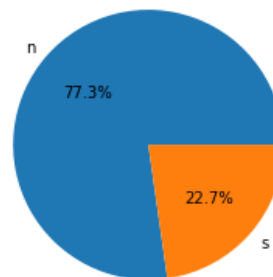
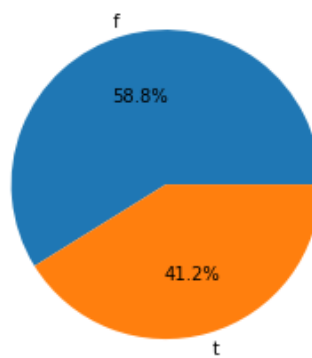


Рисунок 37 – Результат работы

Процентное соотношение по чертам f и t



Процентное соотношение по чертам p и j

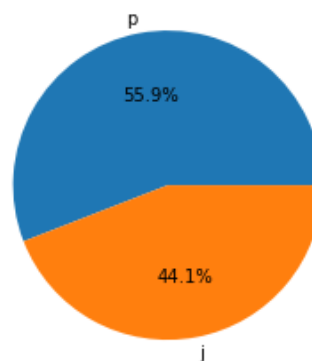


Рисунок 38 – Результат работы

Анализируя полученные данные можно сделать вывод, что больше всего среди людей, попавших в выборку, интровертов и интуитов, а соотношение чувственный/мыслящий и восприимчивый/рассуждающий близко к равному

Выводы

Исходя из проведенного анализа данных, можно сделать следующие выводы:

1. Наиболее распространённой среди пользователей Twitter, участвующих в опросе, оказалась группа *inpf*, в четверку также вошли *infj*, *intp*, *intj*
2. Вышеперечисленные 4 группы составляют примерно половину от всех опрошенных
3. Меньше всего оказалось *estj*, *estp*, *esfj* - обладателями этих типов оказались менее 1,5% людей по каждому типу
4. Больше 75% участников опроса в своей жизни чаще опираются на интуицию, а не на факты
5. Большинство людей из выборки являются интровертами
6. Наиболее близкая к равенству по соотношению количества человек является группа восприятие/суждение (*p/j*)

Рисунок 39 – Результат работы

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

Задание:

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

Решение задачи:

```
import matplotlib.pyplot as plt
%matplotlib inline

from PIL import Image
from io import BytesIO
import requests

plt.figure(figsize=(20, 10))

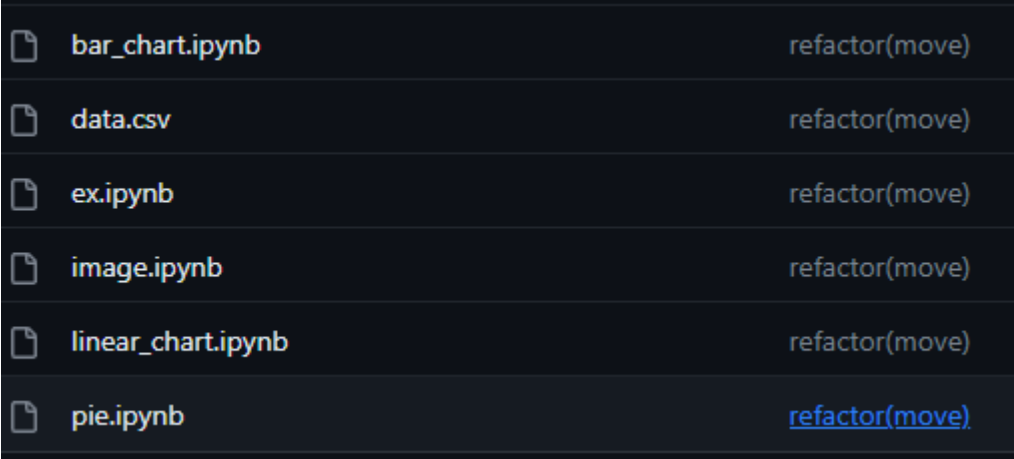
response = requests.get('https://sun2-14.userapi.com/img/Urz9jcVAR4N20wCLZN8LLYp0GmS4-HpCQ2DzdW/uAPiFyp0zds.jpg?size=1280x960&quality=96')
img = Image.open(BytesIO(response.content))
plt.imshow(img)
```

<matplotlib.image.AxesImage at 0x11abfc0cd0>



Рисунок 40– Результат работы

11. Зафиксируйте сделанные изменения в репозитории.









 bar_chart.ipynb	refactor(move)
 data.csv	refactor(move)
 ex.ipynb	refactor(move)
 image.ipynb	refactor(move)
 linear_chart.ipynb	refactor(move)
 pie.ipynb	refactor(move)

Рисунок 19 – Фиксирование изменений в репозитории

Вопросы для защиты работы

1. Как выполнить построение линейного графика с помощью matplotlib?

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

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

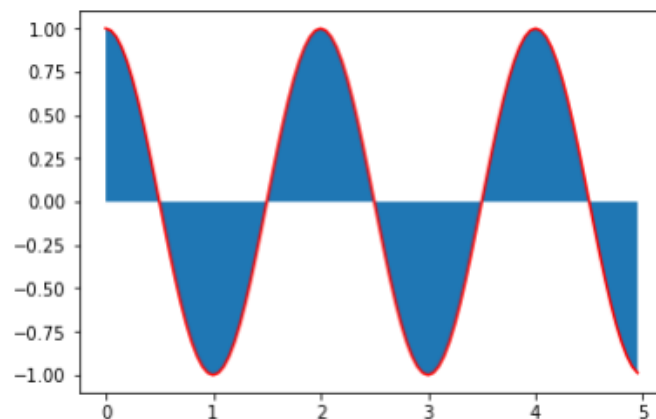
2. Как выполнить заливку области между графиком и осью? Между двумя графиками?

```
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)
```



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

Заливка области между 0.5 и y, при условии, что $y \geq 0.5$:

```
plt.plot(x, y, c="r")
```

```
plt.grid()
```

```
plt.fill_between(x, 0.5, y, where=(y>=0.5))
```

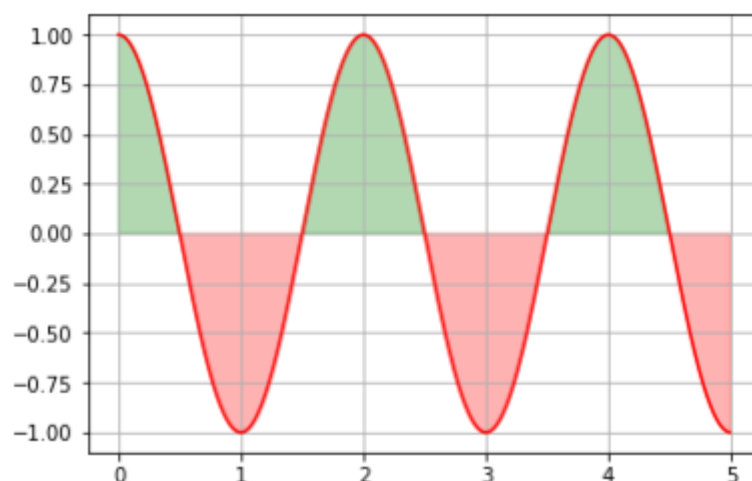
4. Как выполнить двухцветную заливку?

```
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?

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

Типы маркеров:

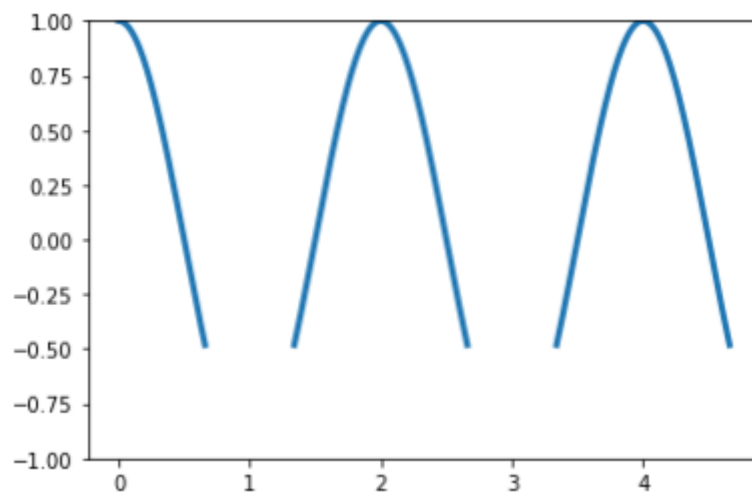
Символ	Описание
'.'	Точка (<i>point marker</i>)
','	Пиксель (<i>pixel marker</i>)
'o'	Окружность (<i>circle marker</i>)
'v'	Треугольник, направленный вниз (<i>triangle_down marker</i>)
'^'	Треугольник, направленный вверх (<i>triangle_up marker</i>)
'<'	Треугольник, направленный влево (<i>triangle_left marker</i>)
'>'	Треугольник, направленный вправо (<i>triangle_right marker</i>)
'1'	Треугольник, направленный вниз (<i>tri_down marker</i>)
'2'	Треугольник, направленный вверх (<i>tri_up marker</i>)
'3'	Треугольник, направленный влево (<i>tri_left marker</i>)
'4'	Треугольник, направленный вправо (<i>tri_right marker</i>)
's'	Квадрат (<i>square marker</i>)
'p'	Пятиугольник (<i>pentagon marker</i>)
'*'	Звезда (<i>star marker</i>)
'h'	Шестиугольник (<i>hexagon1 marker</i>)
'H'	Шестиугольник (<i>hexagon2 marker</i>)
'+'	Плюс (<i>plus marker</i>)
'x'	Х-образный маркер (<i>x marker</i>)
'D'	Ромб (<i>diamond marker</i>)
'd'	Ромб (<i>thin_diamond marker</i>)
' '	Вертикальная линия (<i>vline marker</i>)
'_'	Горизонтальная линия (<i>hline marker</i>)

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 для интервала $(x[i]; x[i+1])$.

‘mid’: значение y ставится в середине интервала.

Код:

```
x = np.arange(0, 7)
```

```
y = x
```

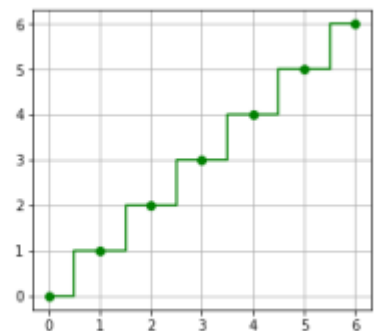
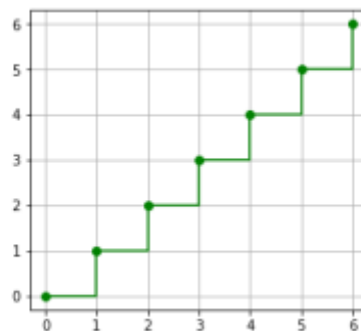
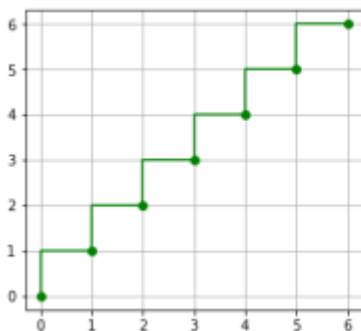
```
where_set = ['pre', 'post', 'mid']
```

```
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])
```

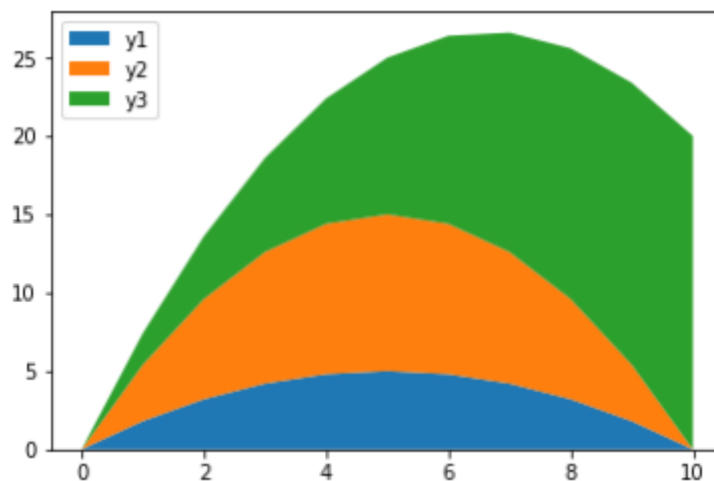
```
y3 = np.array([2*i for i in x])
```

```
labels = ["y1", "y2", "y3"]
```

```
fig, ax = plt.subplots()
```

```
ax.stackplot(x, y1, y2, y3, labels=labels)
```

```
ax.legend(loc='upper left')
```



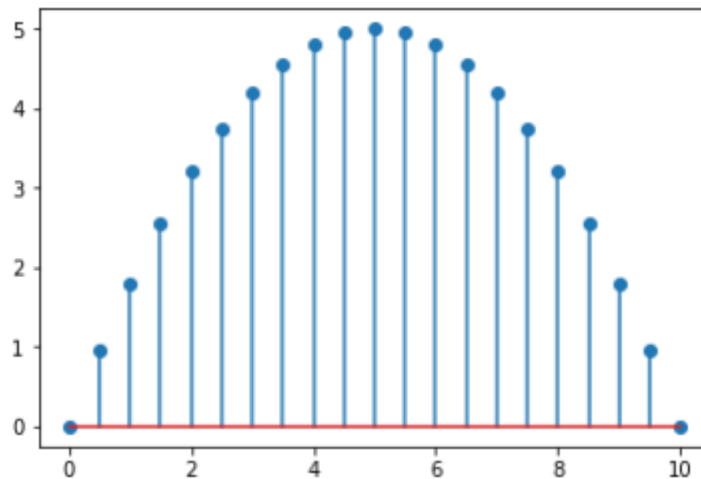
9. Как построить stem-график? В чем особенность stem-графика?

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)
```

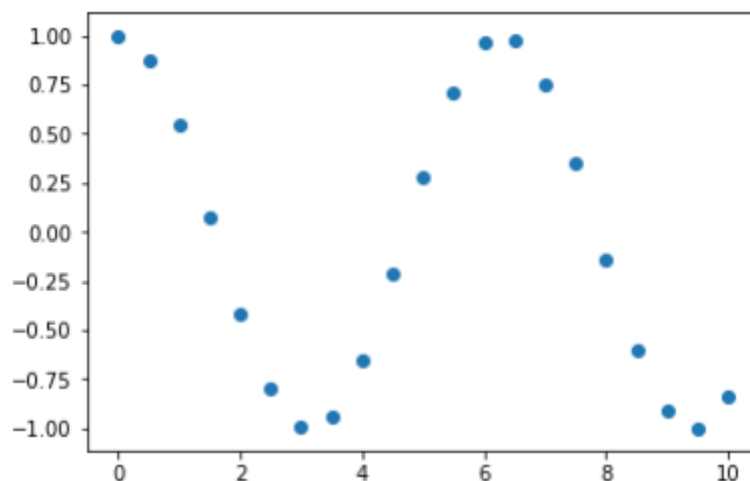
10. Как построить точечный график? В чем особенность точечного графика?

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?

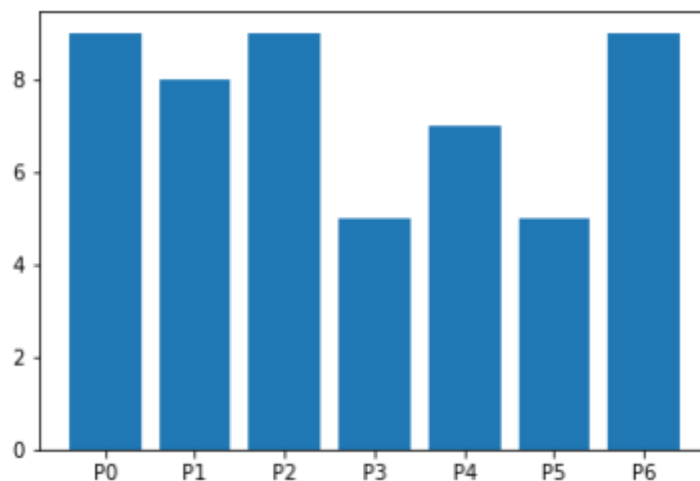
Для их построения используются функции: `bar()` – для построения вертикальной диаграммы `barh()` – для построения горизонтальной диаграммы.

```
np.random.seed(123)
```

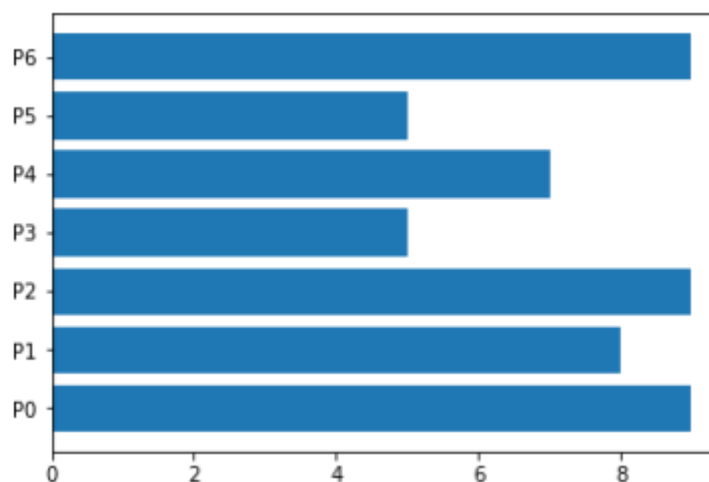
```
groups = [f"P{i}" for i in range(7)]
```

```
counts = np.random.randint(3, 10, len(groups))
```

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



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



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

Это столбчатая диаграмма, которая используется для одновременного отображения больших или нескольких наборов данных.

```
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')
```

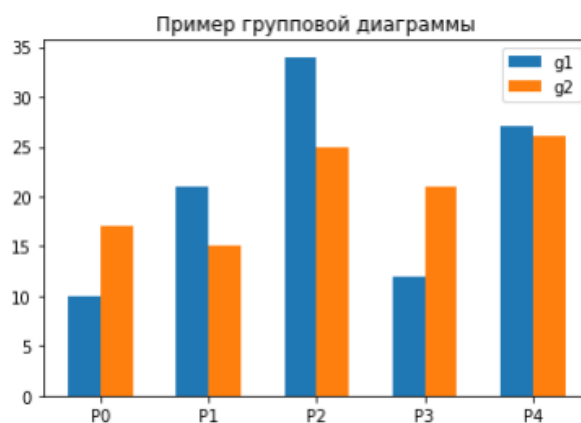
```
rects2 = ax.bar(x + width/2, g2, width, label='g2')
```

```
ax.set_title('Пример групповой диаграммы')
```

```
ax.set_xticks(x)
```

```
ax.set_xticklabels(cat_par)
```

```
ax.legend()
```



13. Как выполнить построение круговой диаграммы средствами matplotlib?

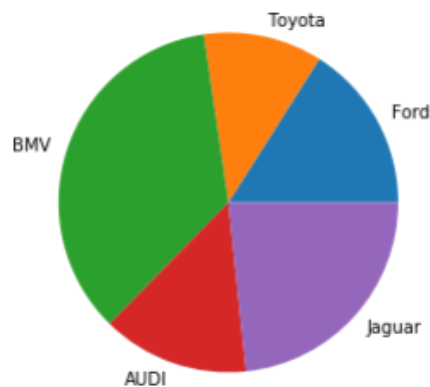
```
vals = [24, 17, 53, 21, 35]
```

```
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.

```
fig, axs = plt.subplots(1, 2, figsize=(10,3), constrained_layout=True)
```

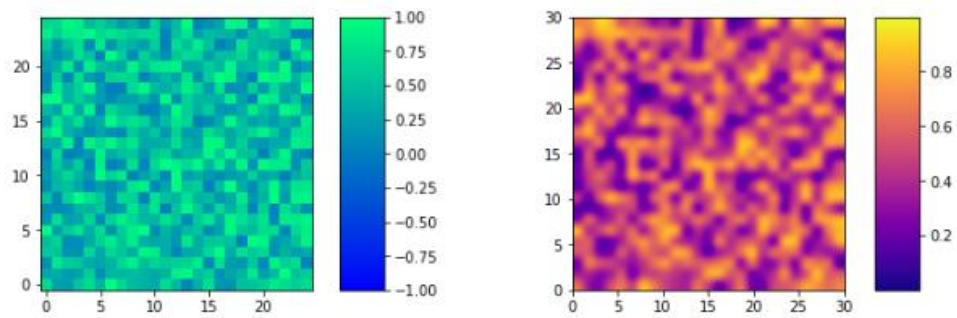
```
p1 = axs[0].imshow(data, cmap='winter', aspect='equal', vmin=-1, vmax=1, origin="lower")
```

```
fig.colorbar(p1, ax=axs[0])
```

```
p2 = axs[1].imshow(data, cmap='plasma', aspect='equal',
```

```
interpolation='gaussian', origin="lower", extent=(0, 30, 0, 30))
```

```
fig.colorbar(p2, ax=axs[1])
```



15. Как отобразить изображение средствами matplotlib?

```
from PIL import Image
```

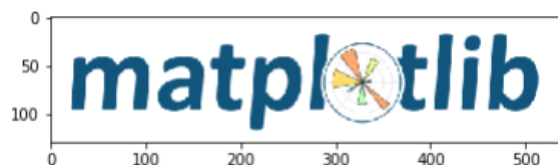
```
import requests
```

```
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?

```
np.random.seed(123)
```

```
data = np.random.rand(5, 7)
```

```
plt.pcolormesh(data, cmap='plasma', edgecolors="k", shading='flat')
```

