МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №5 по дисциплине технологии распознавания образов

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

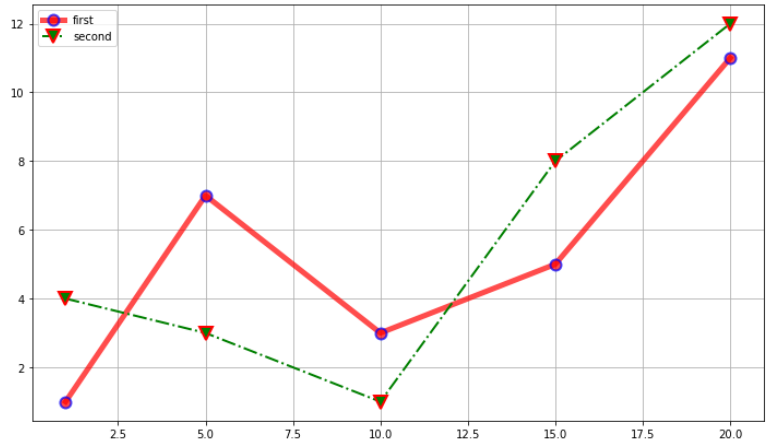
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний

Ввод [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

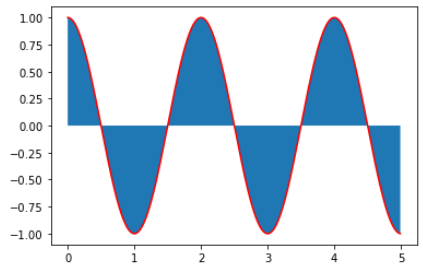
Ввод [2]:
```
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [4, 3, 1, 8, 12]
plt.figure(figsize=(12, 7))
plt.plot(x, y1, 'o-r', alpha=0.7, label="first", lw=5, mec='b', mew=2, ms=10)
plt.plot(x, y2, 'v-.g', label="second", mec='r', lw=2, mew=2, ms=12)
plt.legend()
plt.grid(True)
```



Заливка области между графиком и осью

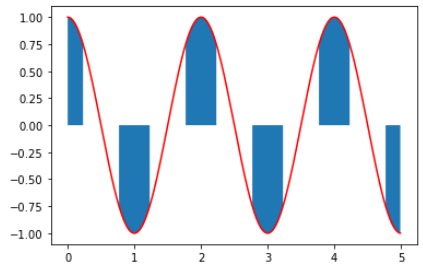
Ввод [3]:
```
import numpy as np

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[3]:
```
<matplotlib.collections.PolyCollection at 0x25a2aaae4f0>
```



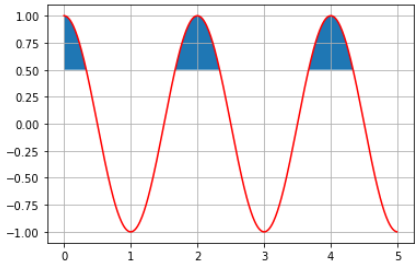
Ввод [4]:
```
plt.plot(x, y, c="r")
plt.fill_between(x, y, where=(y > 0.75) | (y < -0.75))
```

Out[4]:
```
<matplotlib.collections.PolyCollection at 0x25a2ab1d0a0>
```



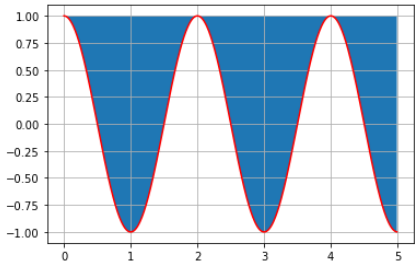
```
plt.grid()
plt.fill_between(x, 0.5, y, where=(y>=0.5))
```

Out[6]: <matplotlib.collections.PolyCollection at 0x25a2abdd790>



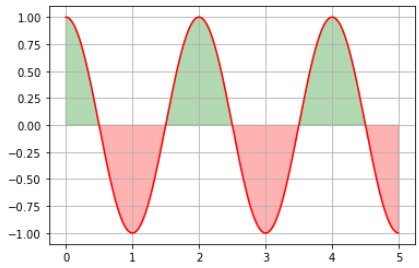
Ввод [7]:
```
plt.plot(x, y, c="r")
plt.grid()
plt.fill_between(x, y, 1)
```

Out[7]: <matplotlib.collections.PolyCollection at 0x25a2ac475e0>



Ввод [8]:
```
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)
```

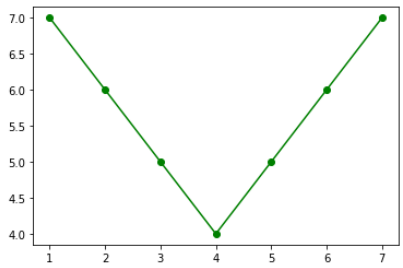
Out[8]: <matplotlib.collections.PolyCollection at 0x25a2ae0ed30>



Настройка маркировки графиков

Ввод [9]:
```
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[9]: [<matplotlib.lines.Line2D at 0x25a2ae7a670>]

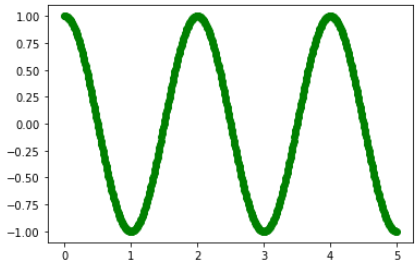


Ввод [10]:
```
x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)
plt.plot(x, y, marker="o", c="g")
```

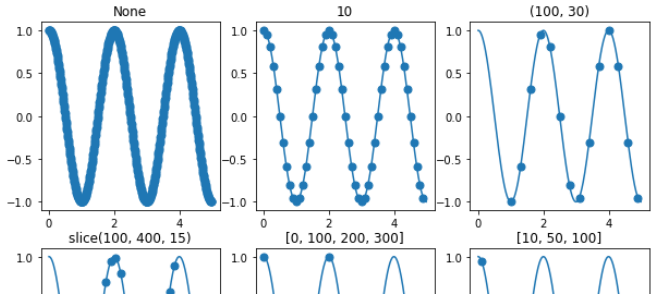
Out[10]: [<matplotlib.lines.Line2D at 0x25a2af448b0>]

Ввод [10]:
```
x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)
plt.plot(x, y, marker="o", c="g")
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x25a2af448b0>]
```



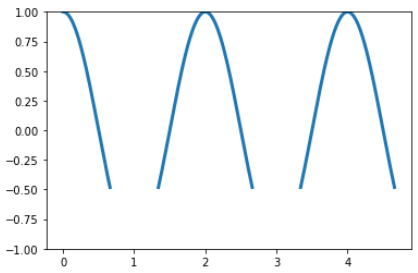
Ввод [14]:
```
x = np.arange(0.0, 5, 0.01)
y = np.cos(x * np.pi)
m_ev_case = [None, 10, (100, 30), slice(100,400,15), [0, 100, 200, 300], [10,
50, 100]]
fig, ax = plt.subplots(2, 3, figsize=(10, 7))
ax = [ax[i, j] for i in range(2) for j in range(3)]
for i, case in enumerate(m_ev_case):
    ax[i].set_title(str(case))
    ax[i].plot(x, y, "o", ls='-', ms=7, markevery=case)
```



Обрезка графика

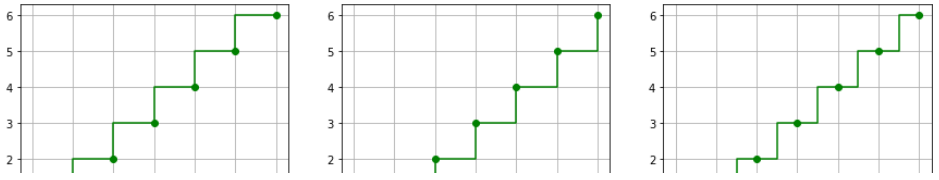
Ввод [15]:
```
x = np.arange(0.0, 5, 0.01)
y = np.cos(x * np.pi)
y_masked = np.ma.masked_where(y < -0.5, y)
plt.ylim(-1, 1)
plt.plot(x, y_masked, linewidth=3)
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x25a2aae32e0>]
```



Ступенчатый график

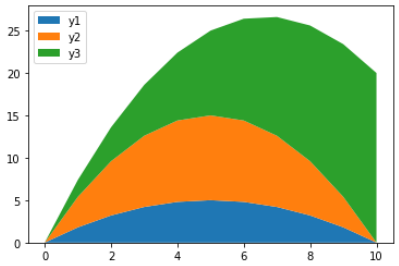
Ввод [17]:
```
x = np.arange(0, 7)
y = x
where_set = ['pre', 'post', 'mid']
fig, axs = plt.subplots(1, 3, figsize=(15, 4))
for i, ax in enumerate(axs):
    ax.step(x, y, "g-o", where=where_set[i])
    ax.grid()
```



Стековый график

Ввод [18]:
```
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')
```

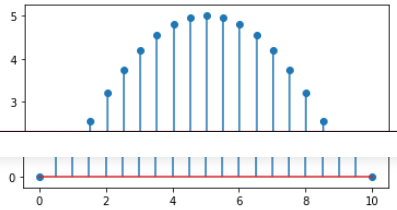
Out[18]:
```
<matplotlib.legend.Legend at 0x25a2b238fd0>
```



Stem-график

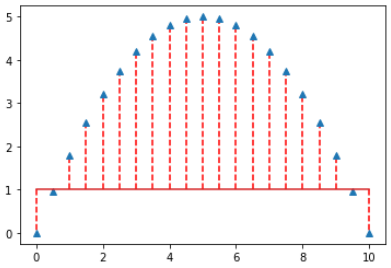
Ввод [19]:
```
x = np.arange(0, 10.5, 0.5)
y = np.array([(-0.2)*i**2+2*i for i in x])
plt.stem(x, y)
```

Out[19]:
```
<StemContainer object of 3 artists>
```



Ввод [20]:
```
plt.stem(x, y, linefmt="r--", markerfmt="^", bottom=1)
```

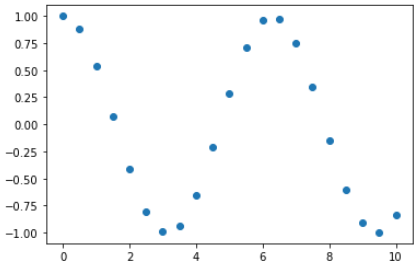
Out[20]:
```
<StemContainer object of 3 artists>
```



Точечный график

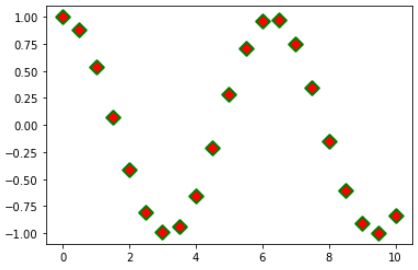
Ввод [21]:
```
x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x25a2b42cc10>
```



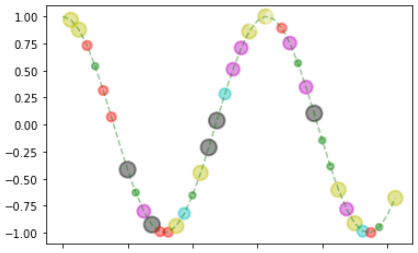
Ввод [22]:
```
x = np.arange(0, 10.5, 0.5)
y = np.cos(x)
plt.scatter(x, y, s=80, c="r", marker="D", linewidths=2, edgecolors="g")
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x25a2c4695e0>
```



Ввод [23]:
```
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)
```

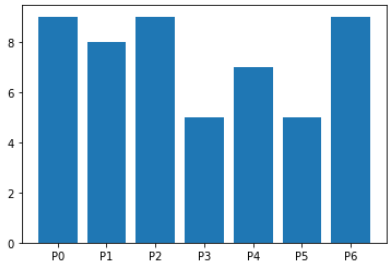
Out[23]:
```
[<matplotlib.lines.Line2D at 0x25a2c4ccbe0>]
```



Столбчатые диаграммы

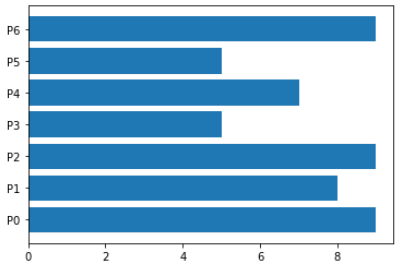
Ввод [24]:
```
np.random.seed(123)
groups = [f"P{i}" for i in range(7)]
counts = np.random.randint(3, 10, len(groups))
plt.bar(groups, counts)
```

Out[24]:
```
<BarContainer object of 7 artists>
```



Ввод [25]:
```
plt.barh(groups, counts)
```

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

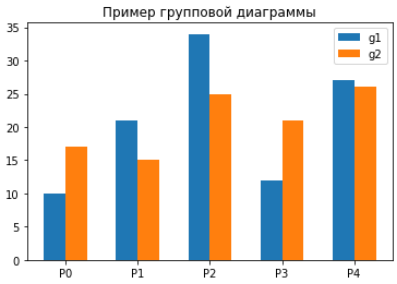


Ввод [26]:
```
import matplotlib.colors as mcolors
bc = mcolors.BASE_COLORS
np.random.seed(123)
groups = [f"P{i}" for i in range(7)]
```

Групповые столбчатые диаграммы

Ввод [27]:
```
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[27]:
```
<matplotlib.legend.Legend at 0x25a2b2b2460>
```



Ввод [28]:
```
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)
```

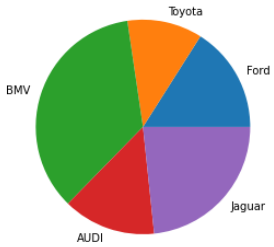
Out[28]:
```
<BarContainer object of 5 artists>
```

Круговые диаграммы

Классическая круговая диаграмма

Ввод [29]:
```
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BMV", "AUDI", "Jaguar"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis("equal")
```

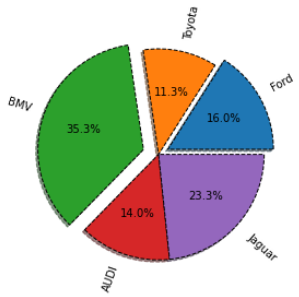
Out[29]:
```
(-1.1163226287452406,
 1.1007772680354877,
 -1.1107362350259515,
 1.1074836529113834)
```



Ввод [30]:
```
vals = [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")
```

Out[30]:
```
(-1.2704955621219602,
 1.1999223938155328,
 -1.1121847055183558,
 1.1379015332518725)
```

1.1379015332518725)



Вложенные круговые диаграммы

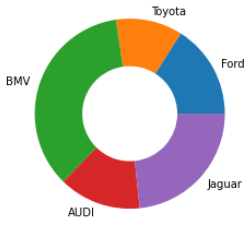
Ввод [31]:
```
fig, ax = plt.subplots()
offset=0.4
data = np.array([[5, 10, 7], [8, 15, 5], [11, 9, 7]])
cmap = plt.get_cmap("tab20b")
b_colors = cmap(np.array([0, 8, 12]))
sm_colors = cmap(np.array([1, 2, 3, 9, 10, 11, 13, 14, 15]))
ax.pie(data.sum(axis=1), radius=1, colors=b_colors,
wedgeprops=dict(width=offset, edgecolor='w'))
ax.pie(data.flatten(), radius=1-offset, colors=sm_colors,
wedgeprops=dict(width=offset, edgecolor='w'))
```

Out[31]:
```
([<matplotlib.patches.Wedge at 0x25a2c5babe0>,
  <matplotlib.patches.Wedge at 0x25a2c5c6130>,
  <matplotlib.patches.Wedge at 0x25a2c5c65b0>,
  <matplotlib.patches.Wedge at 0x25a2c5c6a30>,
  <matplotlib.patches.Wedge at 0x25a2c5c6eb0>,
  <matplotlib.patches.Wedge at 0x25a2c5d3370>,
  <matplotlib.patches.Wedge at 0x25a2c5d37f0>,
  <matplotlib.patches.Wedge at 0x25a2c5d3c70>,
  <matplotlib.patches.Wedge at 0x25a2c5e2130>],
 [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, ''),
```

Круговая диаграмма в виде бублика

Ввод [32]:
```
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[32]:
```
([<matplotlib.patches.Wedge at 0x25a2c614e20>,
  <matplotlib.patches.Wedge at 0x25a2c620280>,
  <matplotlib.patches.Wedge at 0x25a2c620640>,
  <matplotlib.patches.Wedge at 0x25a2c620ac0>,
  <matplotlib.patches.Wedge at 0x25a2c620f40>],
 [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')])
```



Цветовые карты (colormaps)

Отображение изображений

Ввод [33]:
```
from PIL import Image
import requests
from io import BytesIO
```

Цветовые карты (colormaps)

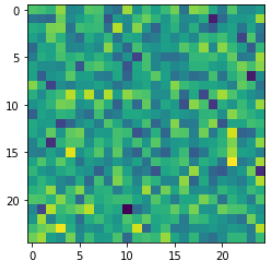
Отображение изображений

Ввод [33]:
```
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[33]: <matplotlib.image.AxesImage at 0x25a2c8d7f40>

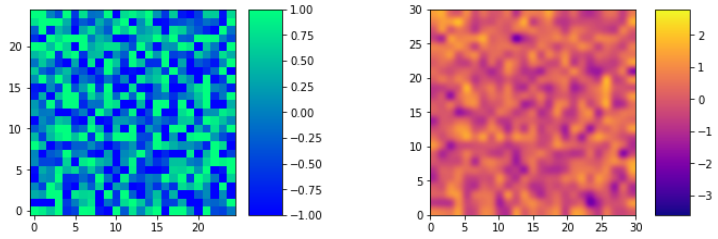
Ввод [34]:
```
np.random.seed(19680801)
data = np.random.randn(25, 25)
plt.imshow(data)
```

Out[34]: <matplotlib.image.AxesImage at 0x25a2887a610>



Ввод [35]:
```
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])
```

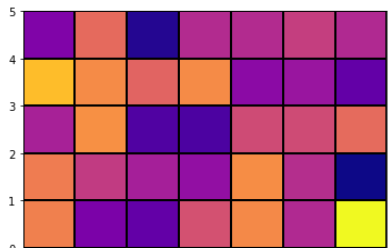
Out[35]: <matplotlib.colorbar.Colorbar at 0x25a2b15c280>



Отображение тепловой карты

Ввод [36]:
```
np.random.seed(123)
data = np.random.rand(5, 7)
plt.pcolormesh(data, cmap='plasma', edgecolors="k", shading='flat')
```

Out[36]: <matplotlib.collections.QuadMesh at 0x25a2b1575e0>

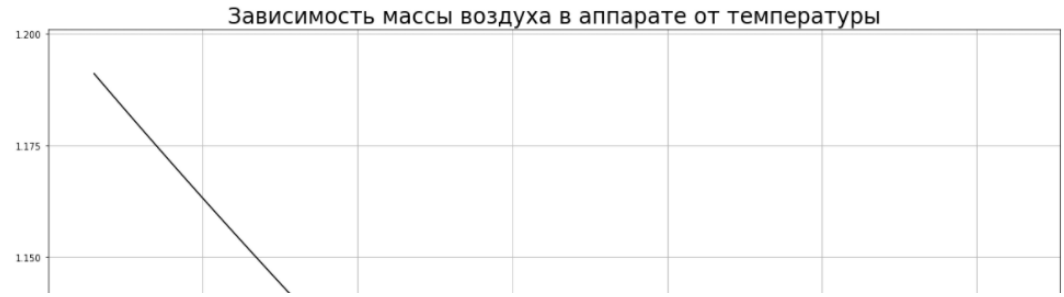


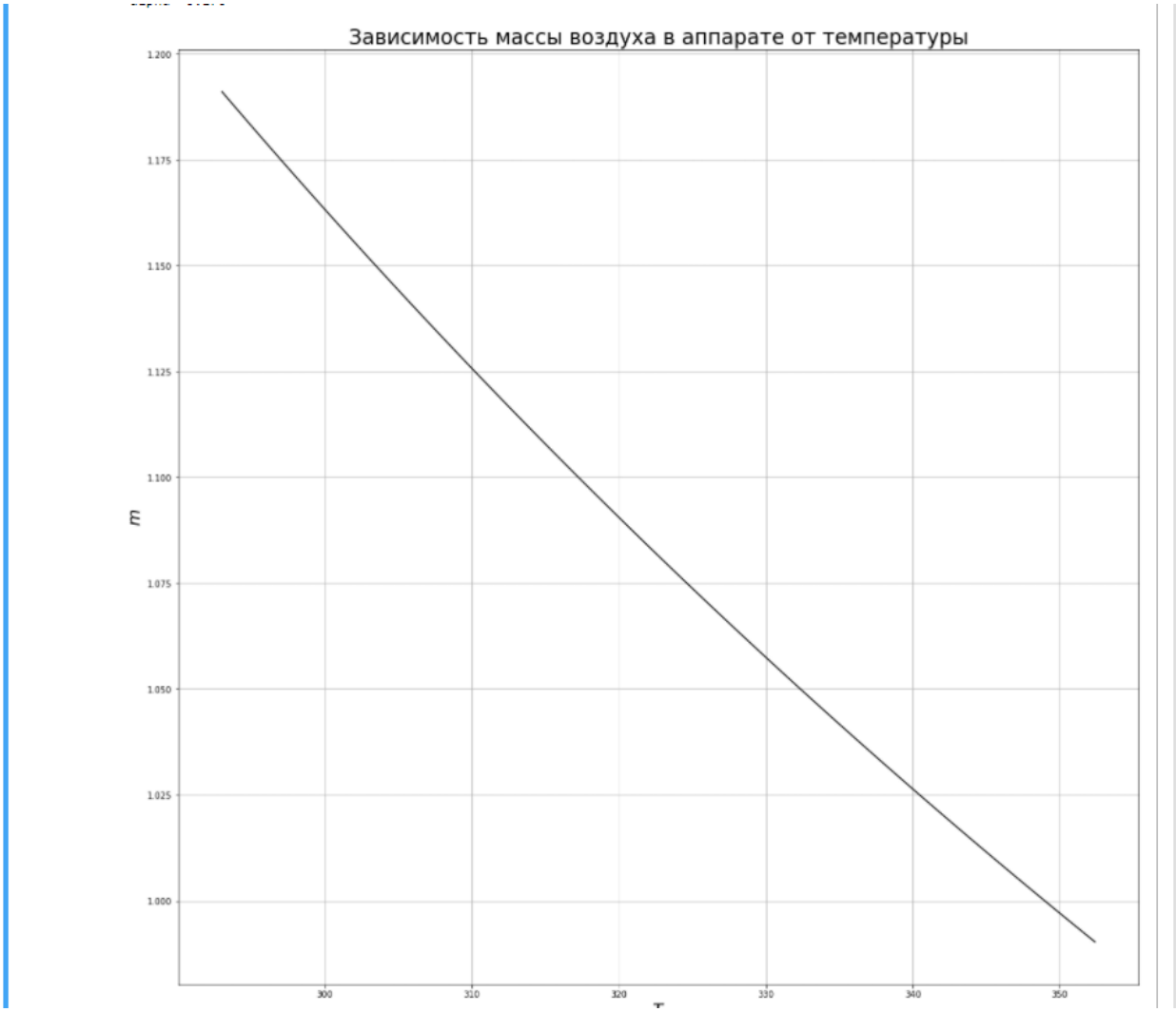
2. Задание с линейным графиком

Условие: При аэродинамическом торможении в атмосфере планеты температура внутри автоматического спускаемого аппарата повысилась с T = 293 K до T = 353 K. Какую часть воздуха необходимо выпустить, чтобы давление внутри аппарата не изменилось? Построить график процесса.

Ввод [18]:
```
import matplotlib.pyplot as plt
R = 8.31; T1 = 293; T2 = 353; p = 1.0e5 ; V = 1; mu = 0.029;
N = 100
dT = (T2 - T1) / N
alpha = 1 - T1 / T2
print(" alpha=%6.3f"%alpha)
x=[]; m=[];
x.append(T1)
m.append((p * V * mu / R) / T1)
for i in range(1, N):
    t = T1 + i * dT;
    x.append(t)
    m.append((p * V * mu / R) / t)

plt.figure(figsize=(20, 20))
plt.plot(x, m, 'k-')
plt.xlabel('$T$', fontsize=20)
plt.ylabel('$m$', fontsize=20)
plt.title('Зависимость массы воздуха в аппарате от температуры', fontsize=24)
plt.grid(True)
```

```
alpha= 0.170
```





3. Задание со столбчатой диаграммой

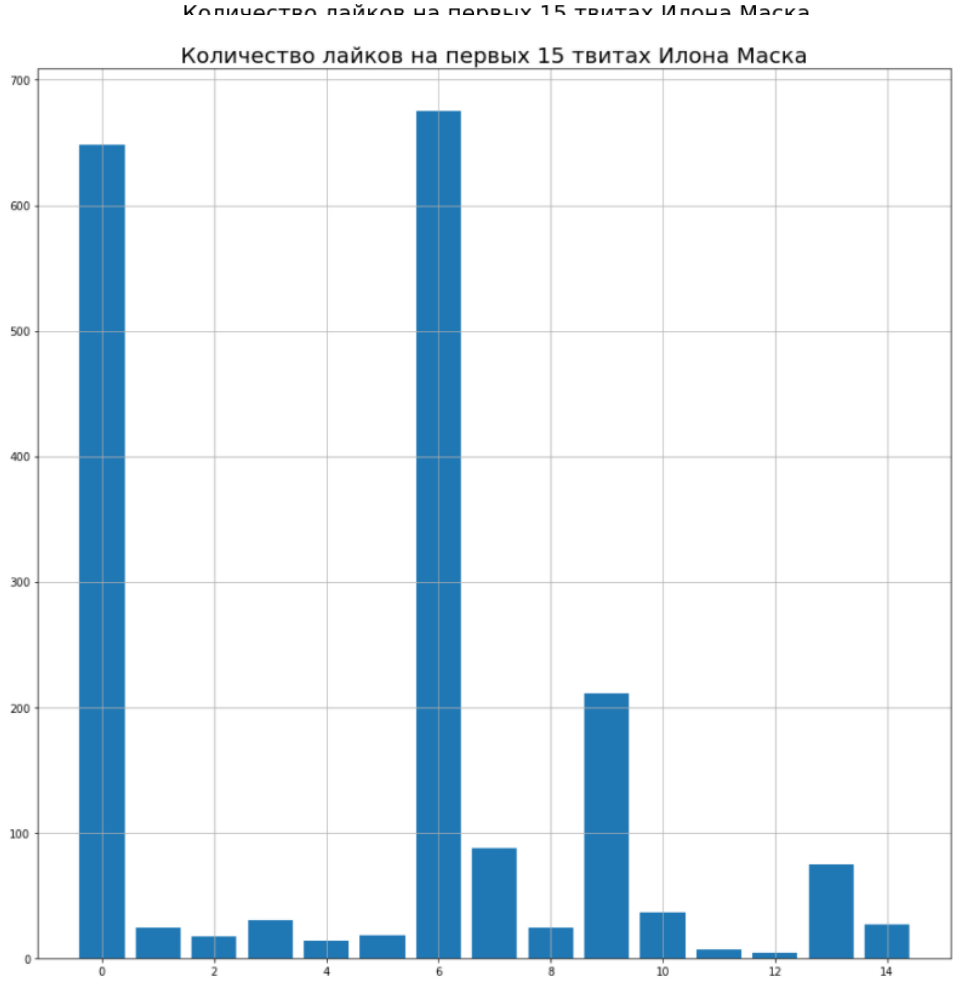
Ввод [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Количество лайков на первых твитах Илона Маска

Ввод [8]:
```
import csv


with open('elonmusk.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    tweet_no = []
    likes = []
    k = 0
    for row in data:
        print(f"{row[0]} {row[7]}")
        if k > 0:
            tweet_no.append(int(row[0]))
            likes.append(int(row[7]))
        k += 1
        if k > 15:
            break
```

```
Tweet Likes
0 648
1 24
2 17
3 30
4 14
5 18
6 675
7 88
8 24
9 211
10 36
11 7
12 4
13 75
14 27
```

Ввод [12]:
```
plt.figure(figsize=(15, 15))
plt.bar(tweet_no, likes)
plt.title('Количество лайков на первых 15 твитах Илона Маска', fontsize=20)
plt.grid(True)
```



Ввод [ ]:

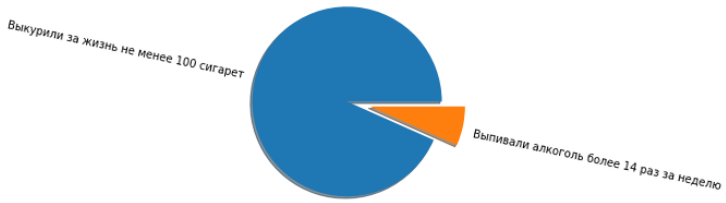
4. Задание с круговой диаграммой

Датасет, содержащий результаты опроса людей по поводу сердечных заболеваний. На графике показано для людей, которые жаловались на проблемы с сердцем: соотношение курящих из них к пьющим.

Ввод [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Ввод [10]:
```
import csv


with open('heart_2020_cleaned.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    smoking = 0
    alcohol = 0
    values = []
    k = 0
    for row in data:
        if row[0] == 'Yes':
            if row[2] == 'Yes':
                smoking += 1;
            if row[3] == 'Yes':
                alcohol += 1
values.append(smoking)
values.append(alcohol)
plt.figure(figsize=(30, 30))
explode = (0.1, 0.15)
labels = ["Выкурили за жизнь не менее 100 сигарет", "Выпивали алкоголь более 14 раз за неделю"]
fig, ax = plt.subplots()
ax.pie(values, labels=labels, shadow=True, explode=explode, rotatelabels=True)
plt.show()
```

```
<Figure size 2160x2160 with 0 Axes>
```



5. Картинка из сети интернет

Ввод [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
import requests
from io import BytesIO
```

Ввод [8]:
```
plt.figure(figsize=(20, 20))
response_stone = requests.get('https://cs7.pikabu.ru/post_img/big/2018/04/10/7/1523355917172135432.png')
img_stone = Image.open(BytesIO(response_stone.content))
plt.imshow(img_stone)
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x21a8043dd90>
```

6. Ответы на 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)

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.

**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?

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

plt.fill_between(x, y, where=(y > 0.75) | (y < -0.75))

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")

используется параметр

markevery, который может принимать одно из следующих значений:

None – отображаться будет каждая точка;

N – отображаться будет каждая N-я точка;

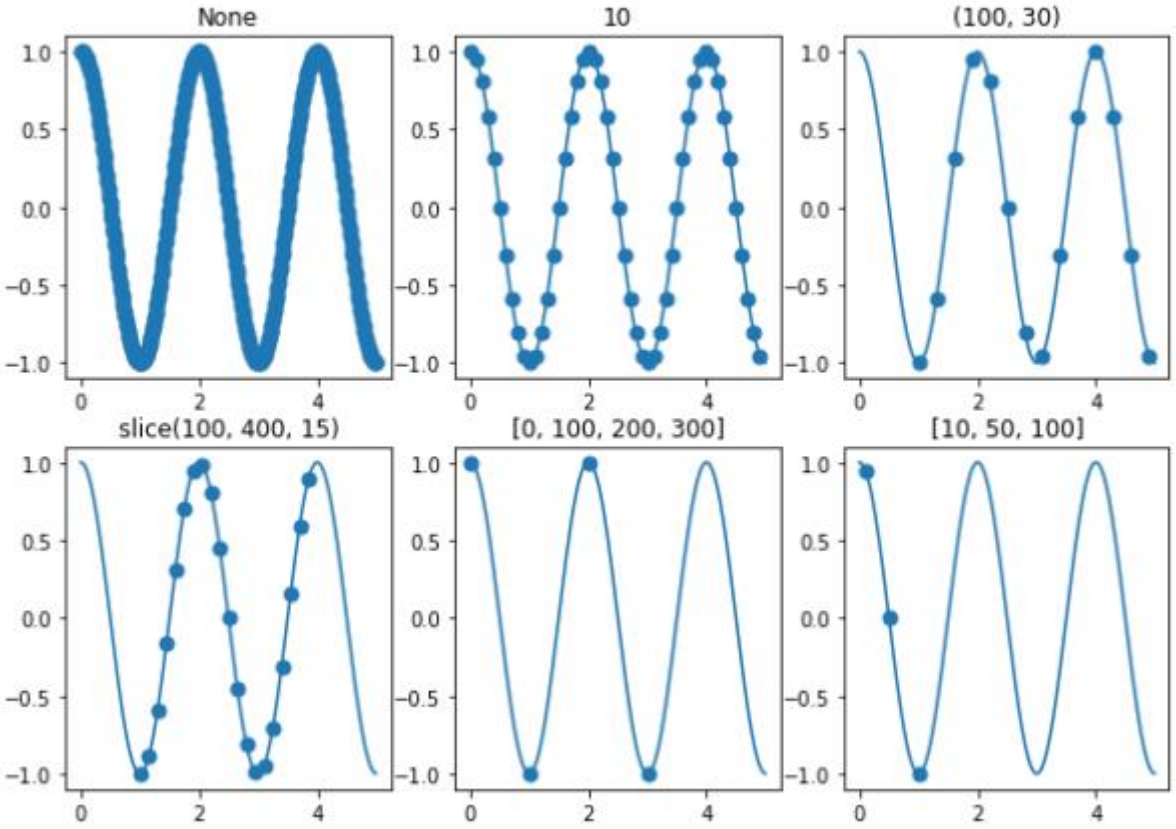
(start, N) – отображается каждая N-я точка начиная с точки start;

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.

Ниже представлен пример, демонстрирующий работу с markevery:

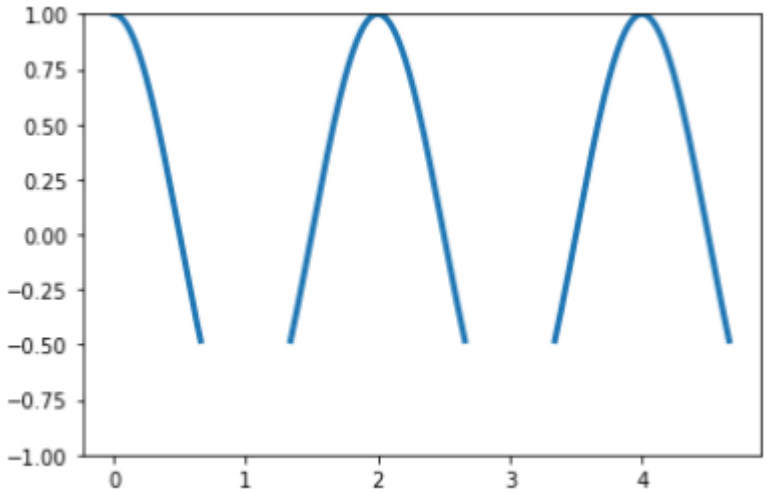
```
x = np.arange(0.0, 5, 0.01)

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)
```



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?

Рассмотрим еще одни график – ступенчатый. 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()).

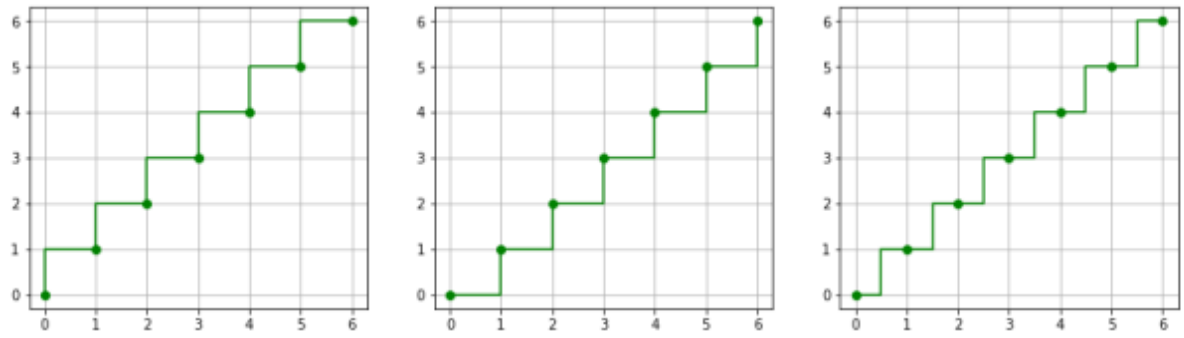
data: indexable object, optional - метки.

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]).

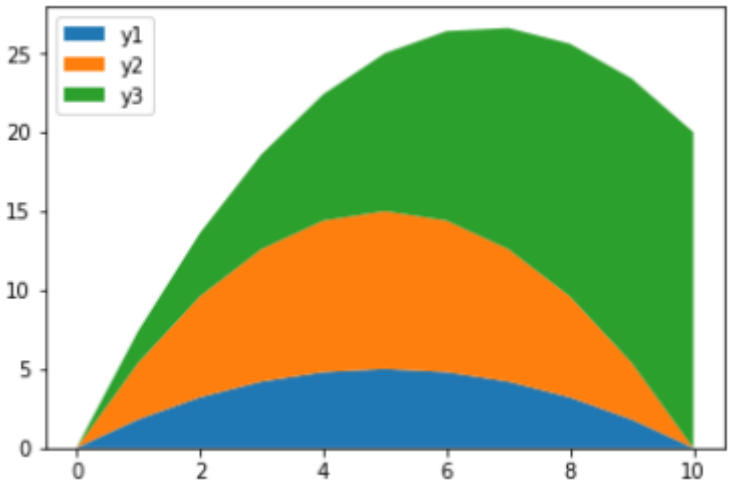
‘mid’: значение y ставится в середине интервала.



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')
```

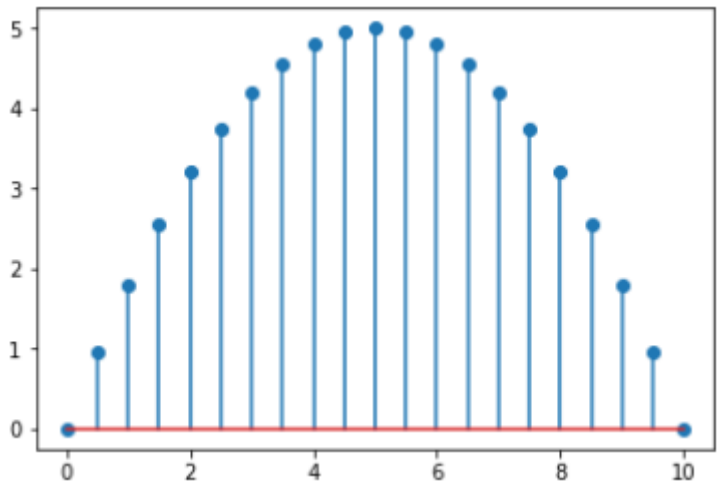


Верхний край области y2 определяется как сумма значений из наборов y1 и y2, y3 – соответственно сумма y1, y2 и y3.

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)
```

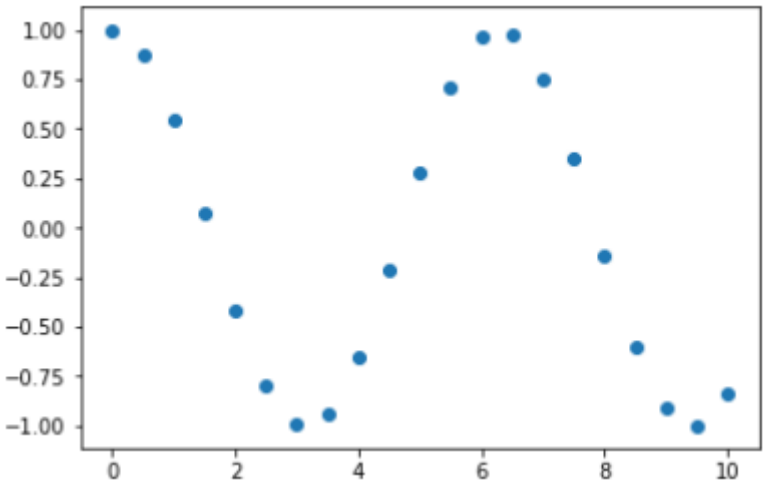


10. Как построить точечный график? В чем особенность точечного графика?

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

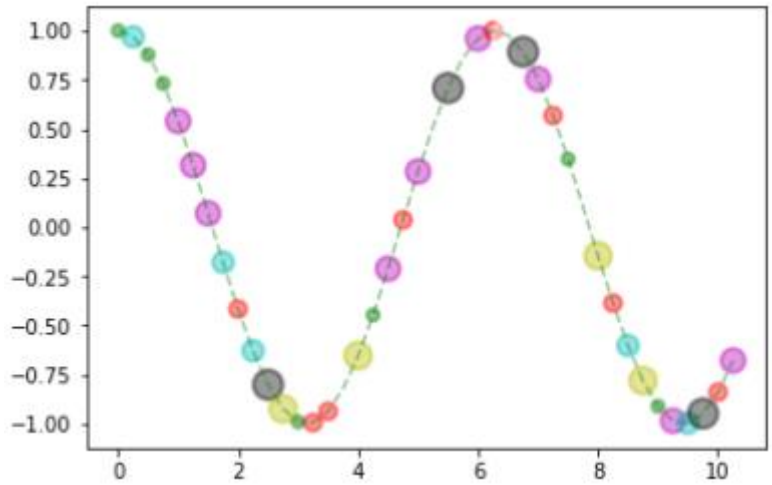
```
x = np.arange(0, 10.5, 0.5)
```

```
y = np.cos(x)
plt.scatter(x, y)
```



Пример, демонстрирующий работу с цветом и размером:

```
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)
```



11. Как осуществляется построение столбчатых диаграмм с помощью matplotlib?

Для визуализации категориальных данных хорошо подходят столбчатые диаграммы. Для их

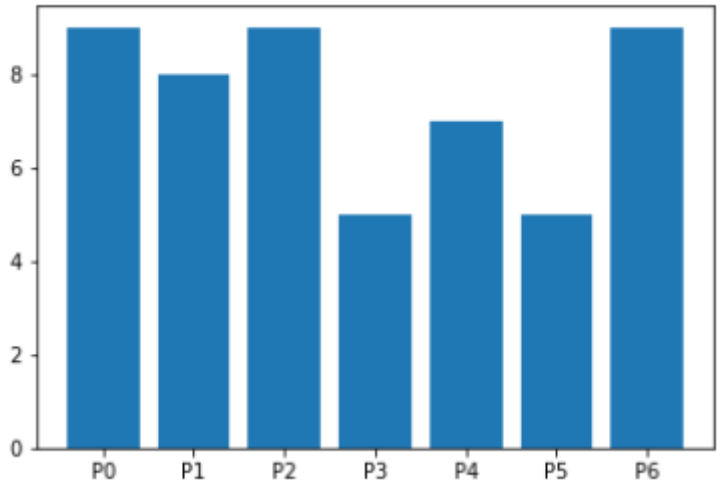
построения используются функции:

bar() – для построения вертикальной диаграммы

barh() – для построения горизонтальной диаграммы.

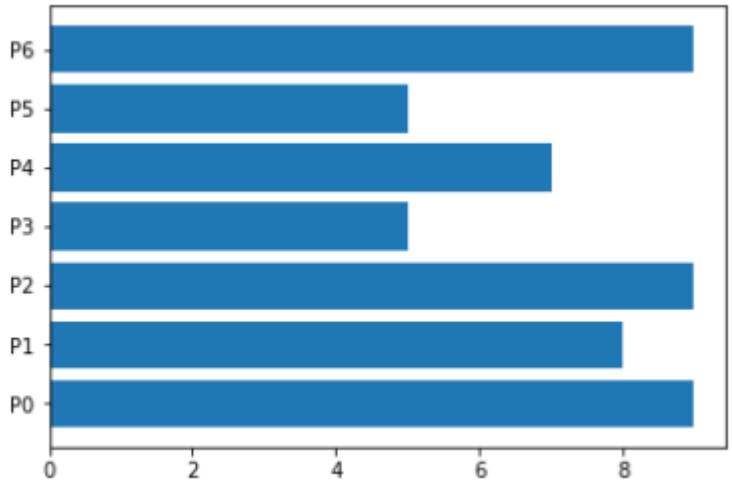
Построим простую диаграмму:

```
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?

Групповые столбчатые диаграммы

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()
```

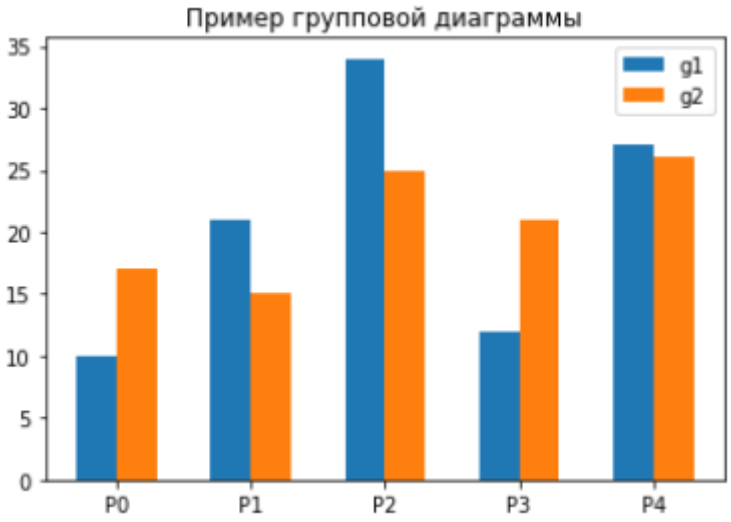
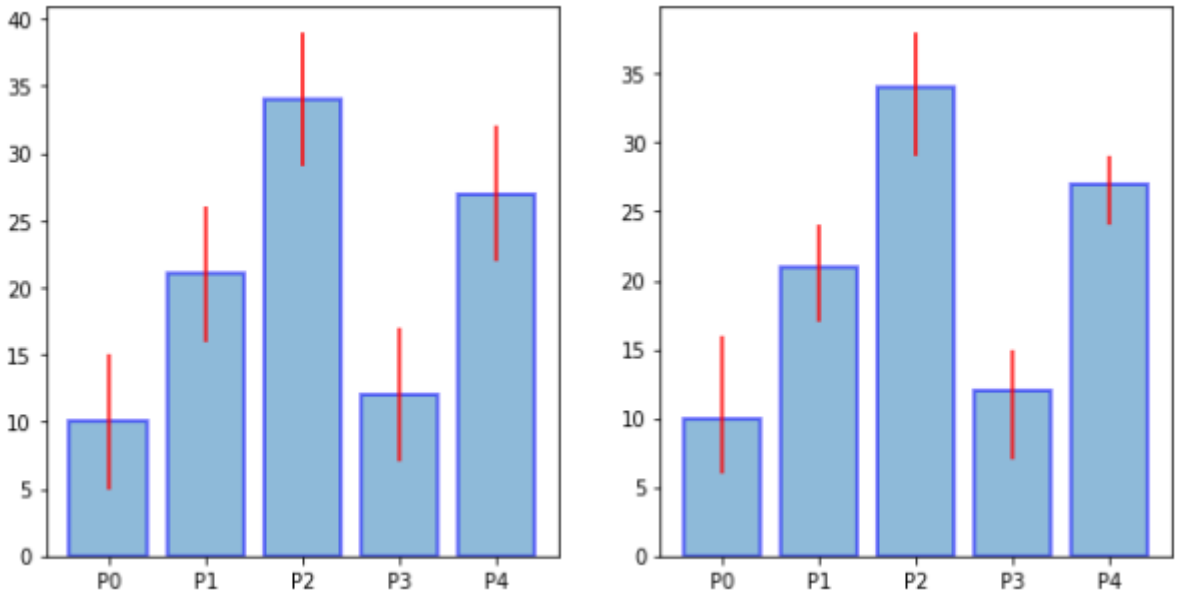


Диаграмма с errorbar элементом

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

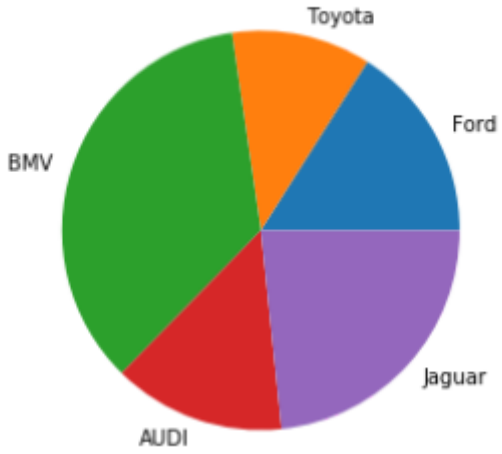
```
np.random.seed(123)

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().

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

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

fig, ax = plt.subplots()

ax.pie(vals, labels=labels)

ax.axis("equal")
```

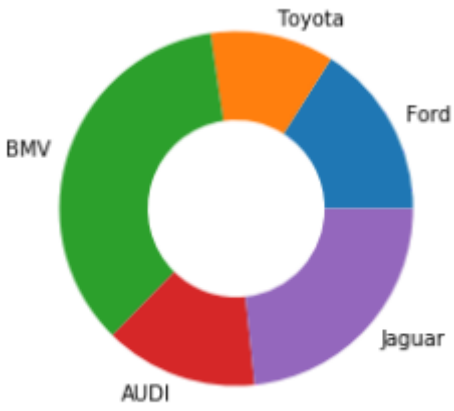


Вложенные круговые диаграммы

```
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'))
```

Круговая диаграмма в виде бублика

Построим круговую диаграмму в виде бублика (с отверстием посередине). Это можно сделать через параметр wedgeprops, который отвечает за внешний вид долей:



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

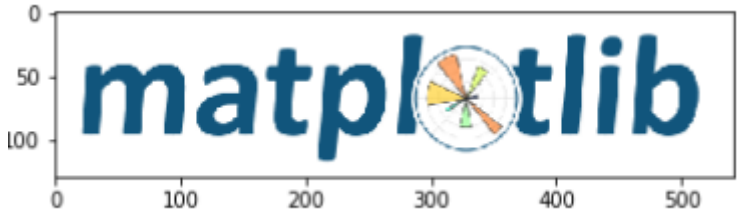
Цветовая карта представляет собой подготовленный набор цветов, который хорошо подходит для визуализации того или иного набора данных. Подробное руководство по цветовым картам вы можете найти на официальном сайте Matplotlib (https://matplotlib.org/tutorials/colors/colormaps.html#sphx-glr-tutorials-colors-colormaps-py).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

imshow() и pcolormesh().

15. Как отобразить изображение средствами matplotlib?

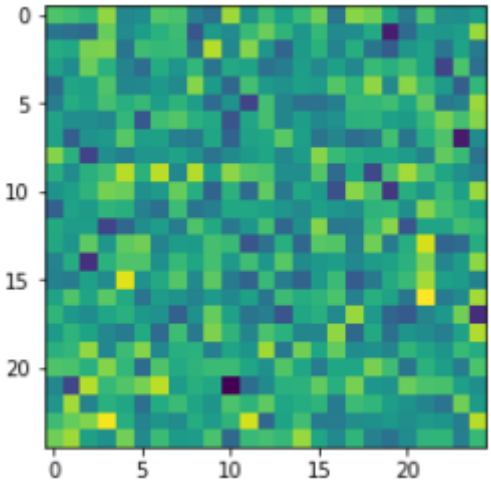
Основное назначение функции imshow() состоит в представлении 2d растров. Это могут быть картинки, двумерные массивы данных, матрицы и т.п. Напишем простую программу, которая загружает картинку из интернета по заданному URL и отображает ее с использованием библиотеки Matplotlib:

```
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)
```



Создадим двумерный набор данных и отобразим его с помощью imshow():

```
np.random.seed(19680801)

data = np.random.randn(25, 25)

plt.imshow(data)
```



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 случаев. 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()

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])

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')
```