МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.5

«Визуализация данных с помощью matplotlib»

по дисциплине «Основы программной инженерии»

Выполнил студент группы
ПИЖ-б-о-21-1
Зиберов Александр
« » апреля 2023 г.
Подпись студента_____
Работа защищена
« »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

Цель работы:

Исследовать базовые возможности визуализации данных на плоскости средствами библиотеки matplotlib языка программирования Python.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: https://github.com/afk552/trolab_5

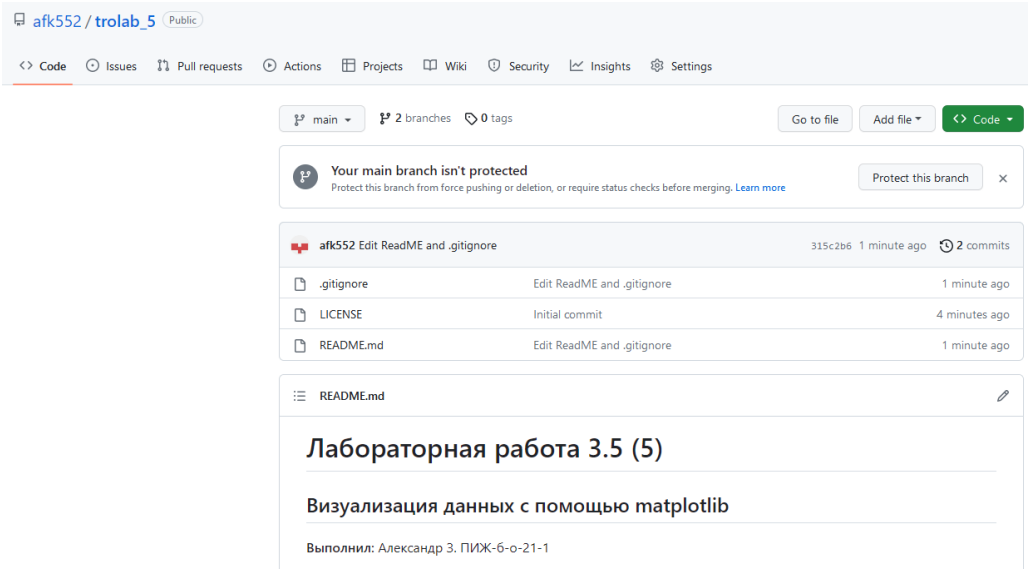


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

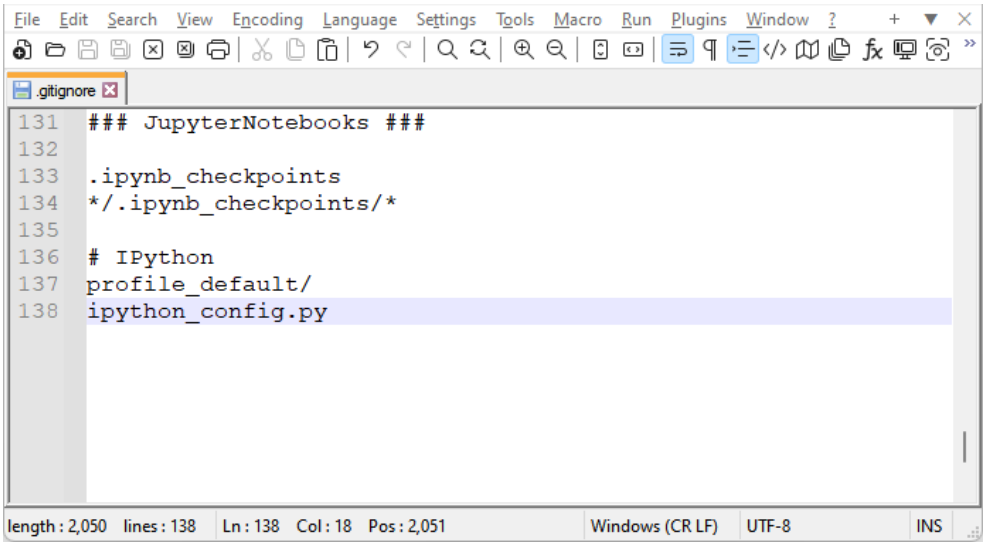


Рисунок 2 – Окно блокнота

Организуйте свой репозиторий в соответствие с моделью ветвления git-flow, рисунок 3.

```
* develop
  main
```

Рисунок 3 – Окно командной строки

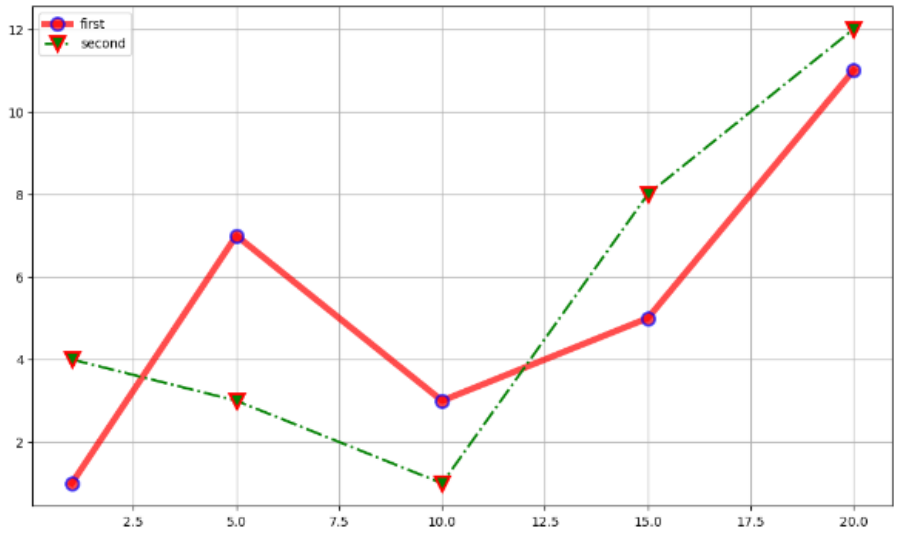
Проработать примеры лабораторной работы в отдельном ноутбуке.

Визуализация данных с помощью matplotlib

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

Построение графика при помощи plot()

In [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)
```



Тестирование заливки графика

In [3]:
```
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 0x20542b37760>
```

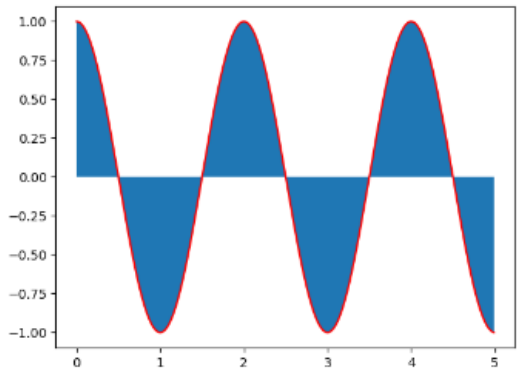
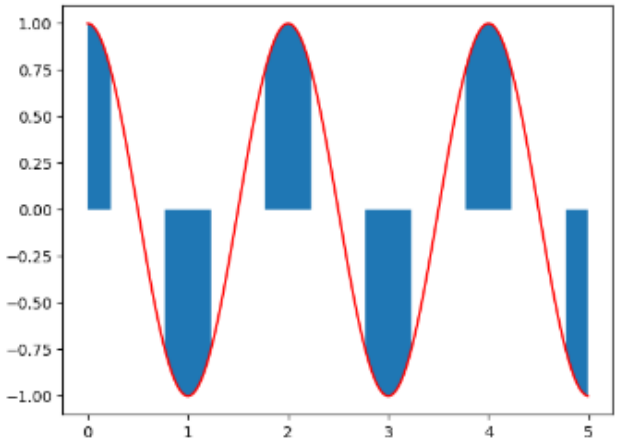


Рисунок 4 – Пример 1

Заливка при y > 0.75 и y < 0.75:

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

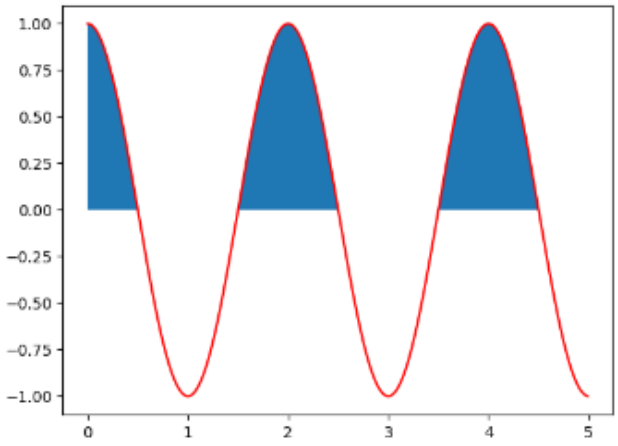
Out[4]: <matplotlib.collections.PolyCollection at 0x20542436640>



Заливка области между 0 и y, при условии, что y >= 0:

In [5]:
```
plt.plot(x, y, c="r")
plt.fill_between(x, y, where=(y > 0))
```

Out[5]: <matplotlib.collections.PolyCollection at 0x205425dbf70>



Заливка области между 0.5 и y, при условии, что y >= 0.5:

In [6]:
```
plt.plot(x, y, c="r")
plt.grid()
plt.fill_between(x, 0.5, y, where=(y>=0.5))
```

Out[6]: <matplotlib.collections.PolyCollection at 0x2054231f400>

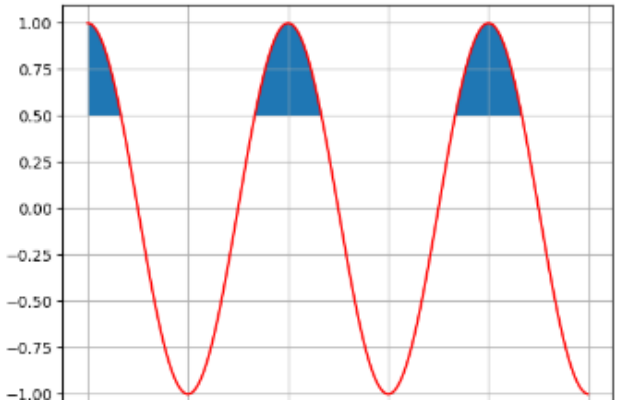
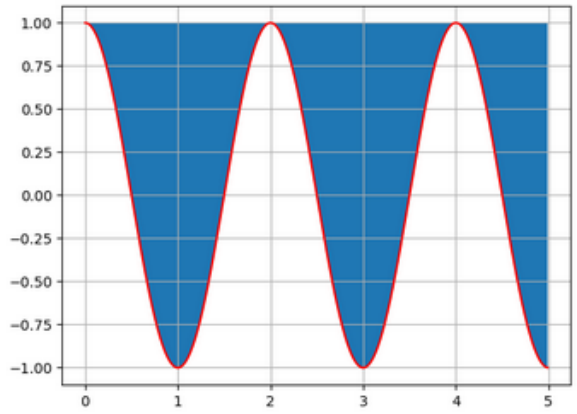


Рисунок 5 – Пример 2

Заливка область между y и 1:

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

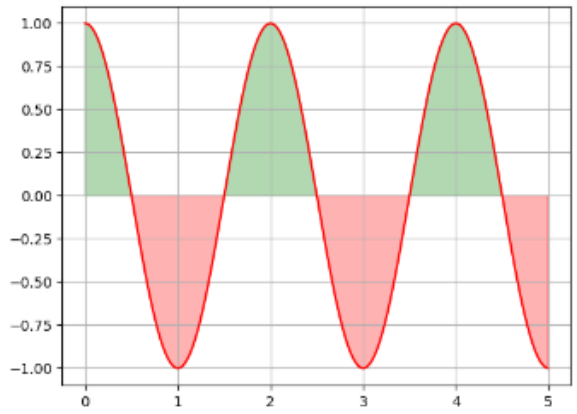
Out[7]:
```
<matplotlib.collections.PolyCollection at 0x20542392250>
```



Вариант двухцветной заливки:

In [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 0x20542443dc0>
```



Настройка маркировки графиков

In [9]:
```
x = np.arange(0.0, 5, 0.01)
y = np.cos(x*np.pi)

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

Out[9]:
```
[<matplotlib.lines.Line2D at 0x2054267b970>]
```

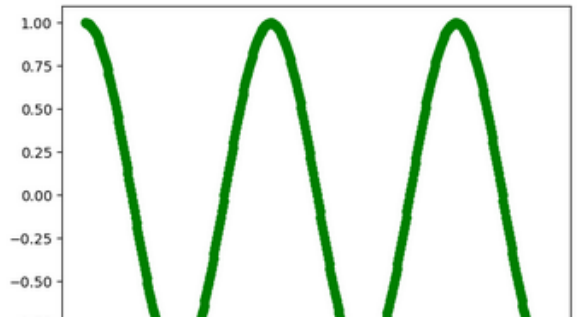
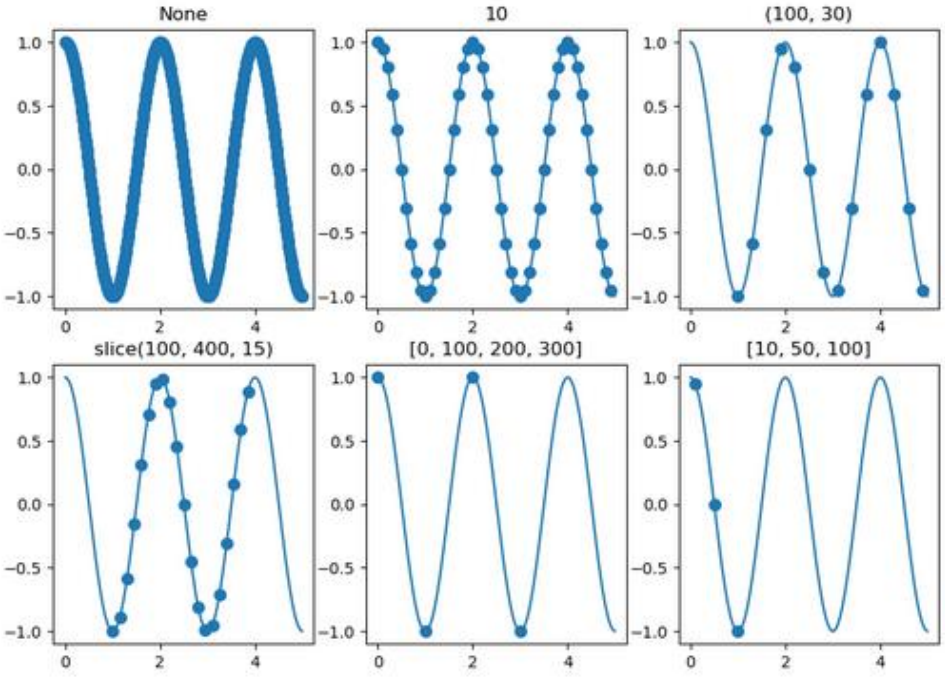


Рисунок 6 – Пример 3

Пример, демонстрирующий работу с markevery

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



Обрезка графиков

In [11]:
```
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[11]:
```
[<matplotlib.lines.Line2D at 0x2054272fd90>]
```

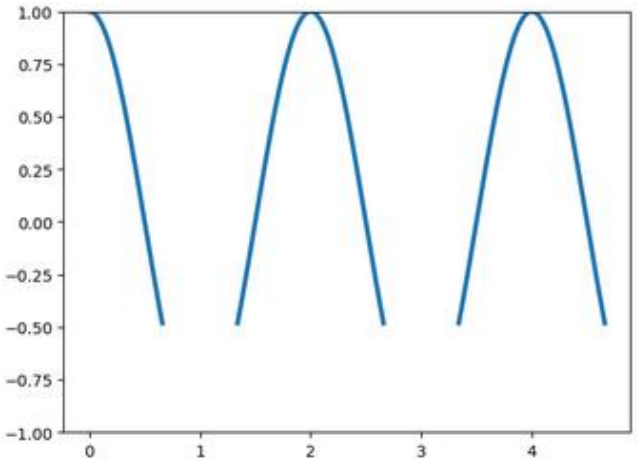
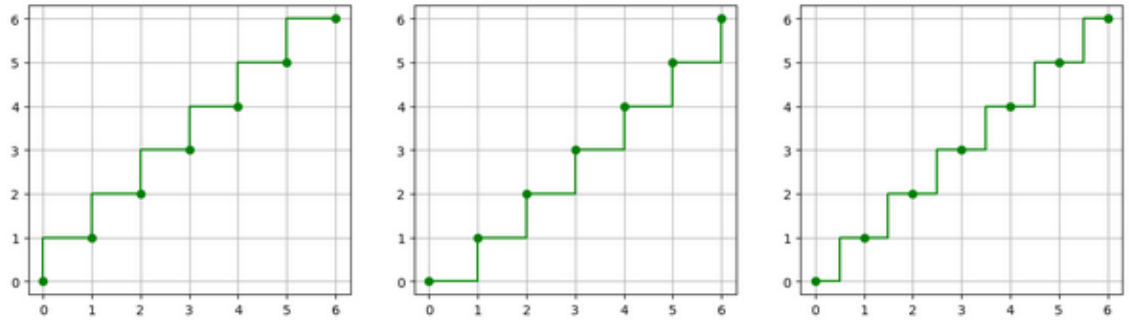


Рисунок 7 – Пример 4

Ступенчатый график

In [12]:
```
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

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

In [13]:
```
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[13]:
```
<matplotlib.legend.Legend at 0x20542b07f70>
```

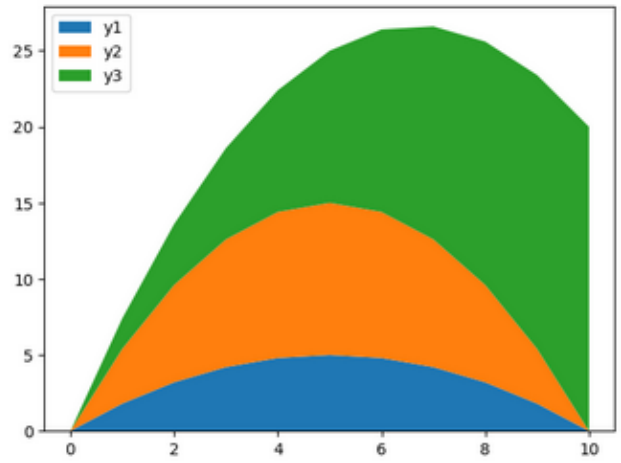


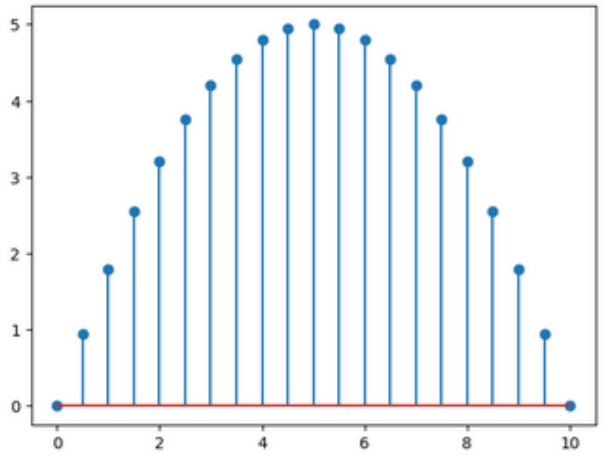
Рисунок 8 – Пример 5

Stem-график

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

In [14]:
```
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[14]:
```
<StemContainer object of 3 artists>
```



Stem-график с добавлением доп. параметрами:

- linefmt - стиль вертикальной линии
- markerfmt - формат маркера
- bottom - y-координата базовой линии

In [15]:
```
plt.stem(x, y, linefmt="r--", markerfmt="^", bottom=1)
```

Out[15]:
```
<StemContainer object of 3 artists>
```

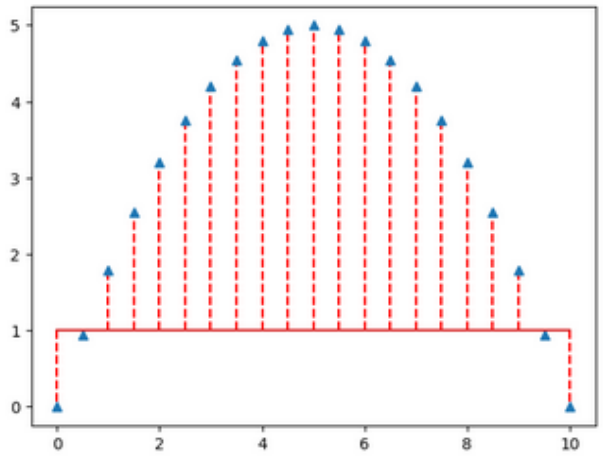


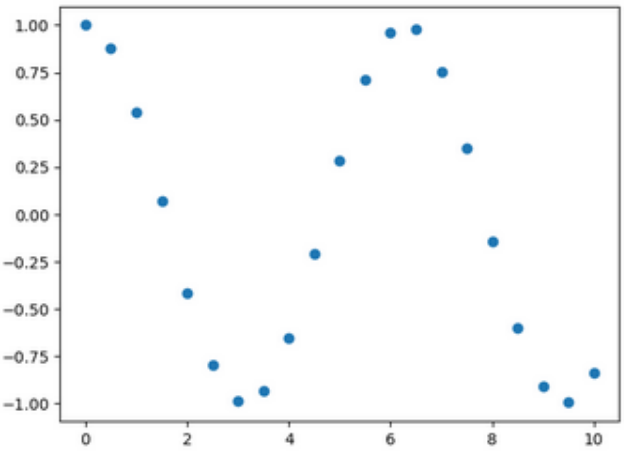
Рисунок 9 – Пример 6

Точечный график

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

In [16]:
```
x = np.arange(0, 10.5, 0.5)
y = np.cos(x)

plt.scatter(x, y)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x205426a9d90>
```



Решение, использующее расширенные параметры для настройки отображения графика:

In [17]:
```
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[17]:
```
<matplotlib.collections.PathCollection at 0x2054253c130>
```

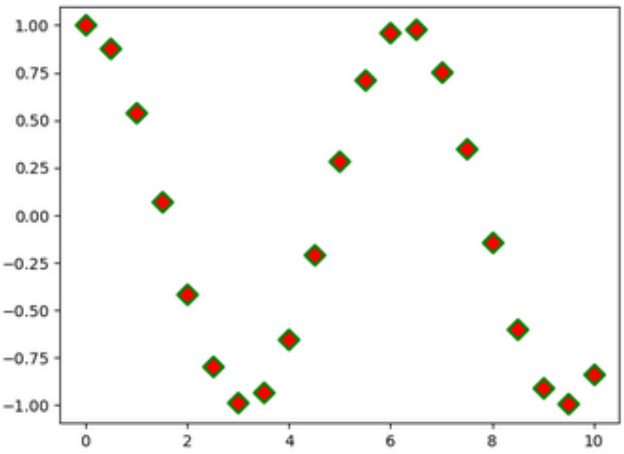
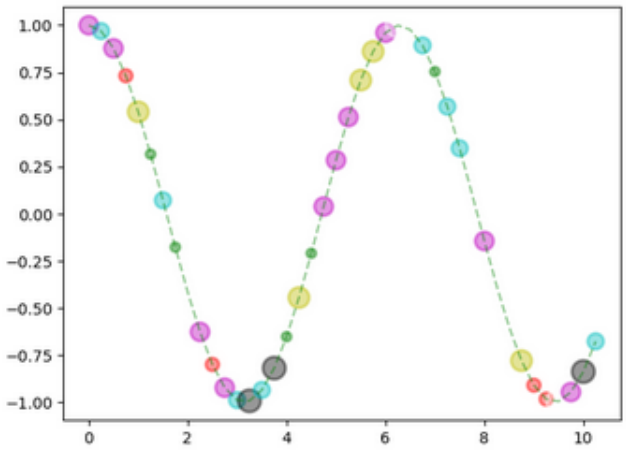


Рисунок 10 – Пример 7

Пример, демонстрирующий работу с цветом и размером:

In [18]:
```
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[18]:
```
[<matplotlib.lines.Line2D at 0x205429504c0>]
```

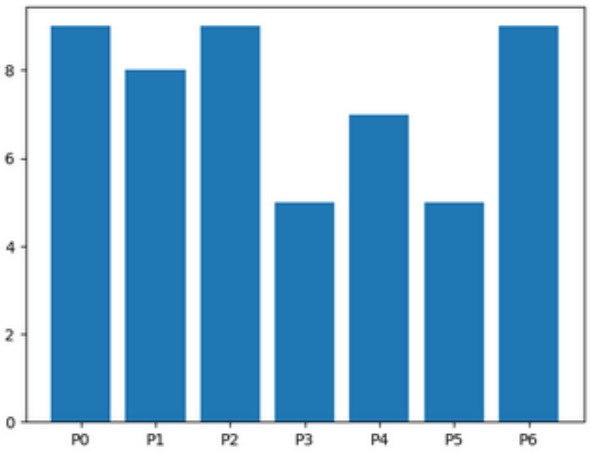


Столбчатые диаграммы

Для визуализации категориальных данных хорошо подходят столбчатые диаграммы. Для их построения используются функции:
bar() – для построения вертикальной диаграммы
barh() – для построения горизонтальной диаграммы

In [19]:
```
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[19]:
```
<BarContainer object of 7 artists>
```

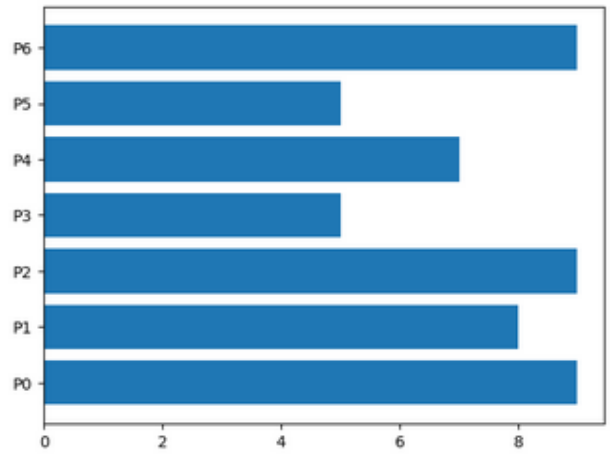


Если заменим bar() на barh() получим горизонтальную диаграмму

In [20]:
```
plt.barh(groups, counts)
```

Out[20]:
```
<BarContainer object of 7 artists>
```

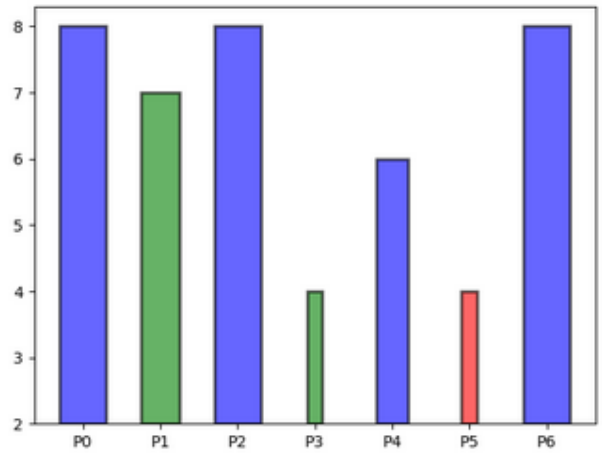
Рисунок 11 – Пример 8



Более сложный пример, демонстрирующий работу с параметрами:

In [21]:
```
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[21]:
```
<BarContainer object of 7 artists>
```



Type *Markdown* and LaTeX: α^2

Групповые столбчатые диаграммы

In [22]:
```
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[22]:
```
<matplotlib.legend.Legend at 0x20543bb38e0>
```

Рисунок 12 – Пример 9

Групповые столбчатые диаграммы

In [22]:
```
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[22]:
```
<matplotlib.legend.Legend at 0x20543bb38e0>
```

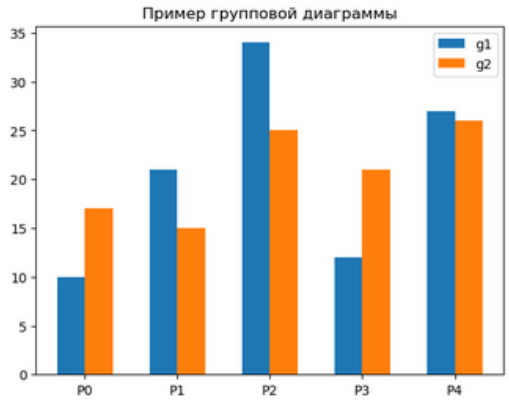


Диаграмма с errorbar элементом

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

In [23]:
```
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="n", alpha=0.5, edgecolor="b",
linewidth=2)
axs[1].bar(cat_par, g1, yerr=error, ecolor="n", alpha=0.5, edgecolor="b",
linewidth=2)
```

Out[23]:
```
<BarContainer object of 5 artists>
```

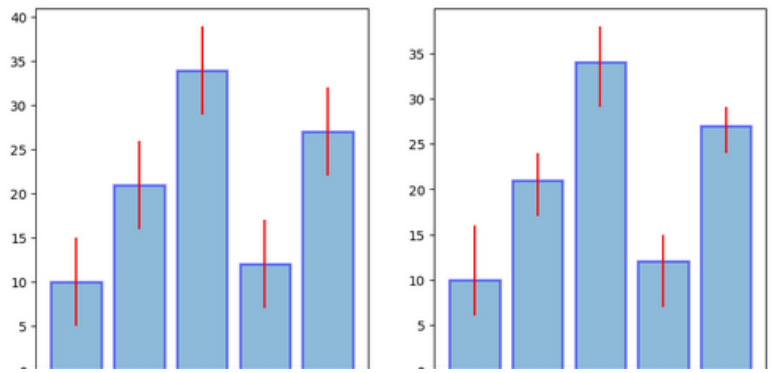
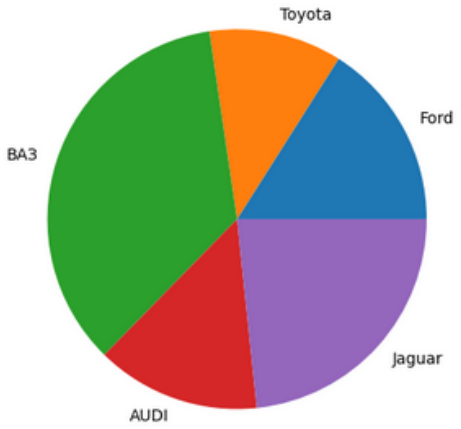


Рисунок 13 – Пример 10

Классическая круговая диаграмма

In [24]:
```
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BA3", "AUDI", "Jaguar"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis("equal")
```

Out[24]:
```
(-1.1163226287452406,
 1.1007772680354877,
 -1.1107362350259515,
 1.1074836529113834)
```



In [25]:
```
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BA3", "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[25]:
```
(-1.2704955621219602,
 1.1999223938155328,
 -1.1121847055183558,
 1.1379015332518725)
```

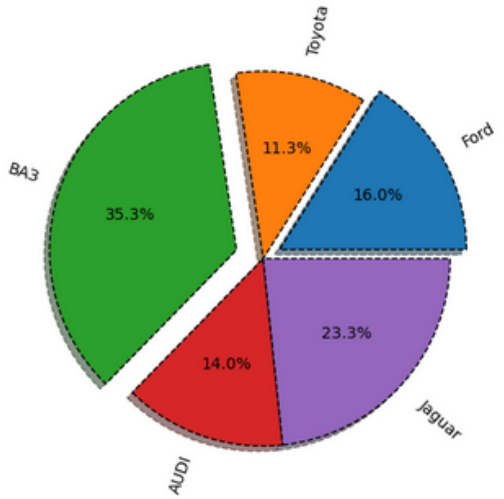


Рисунок 14 – Пример 11

Вложенные круговые диаграммы

In [26]:
```
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[26]:
```
([<matplotlib.patches.Wedge at 0x205427bc850>,
  <matplotlib.patches.Wedge at 0x205427bcd30>,
  <matplotlib.patches.Wedge at 0x20542799250>,
  <matplotlib.patches.Wedge at 0x20542799730>,
  <matplotlib.patches.Wedge at 0x20542799c10>,
  <matplotlib.patches.Wedge at 0x20542790130>,
  <matplotlib.patches.Wedge at 0x20542790610>,
  <matplotlib.patches.Wedge at 0x20542790af0>,
  <matplotlib.patches.Wedge at 0x20542790fd0>],
 [Text(0.6463143444414094, 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(-0.12048606360635531, -0.6489073112975174, ''),
  Text(0.39011356818311405, -0.5323639769175 21, ''),
  Text(0.6332653697075483, -0.1859434632601054, '')])
```

Круговая диаграмма в виде бублика

In [27]:
```
vals = [24, 17, 53, 21, 35]
labels = ["Ford", "Toyota", "BA3", "AUDI", "Jaguar"]
fig, ax = plt.subplots()
ax.pie(vals, labels=labels, wedgeprops=dict(width=0.5))
```

Out[27]:
```
([<matplotlib.patches.Wedge at 0x20542741e80>,
  <matplotlib.patches.Wedge at 0x2054277a3e0>,
  <matplotlib.patches.Wedge at 0x2054277a880>,
  <matplotlib.patches.Wedge at 0x2054277ad60>,
  <matplotlib.patches.Wedge at 0x20542763280>],
 [Text(0.9639375400211 44, 0.5299290306818474, 'Ford'),
  Text(0.2287028716524 0302, 1.0759623583090 37, 'Toyota'),
  Text(-1.046162158377023, 0.3399187231970734, 'BA3'),
  Text(-0.3617533684721028, -1.0388139873909512, 'AUDI'),
  Text(0.8174592712713289, -0.7360437078139777, 'Jaguar')])
```

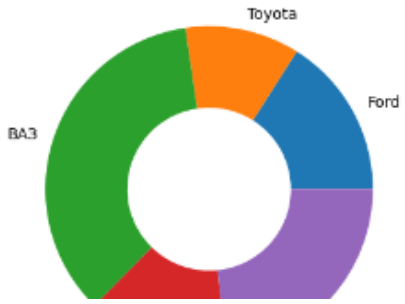


Рисунок 15 – Пример 12

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.

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.

Материальная точка движется вдоль оси OX по закону: $x(t) = 4 \cdot t$

Чему равна скорость материальной точки?
Какой путь она пройдёт за 7 с движения?

Постройте графики зависимости (линейные диаграммы):

- скорости от времени
- координаты от времени

Импорт модулей

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

Материальная точка будет двигаться прямолинейно, следовательно, справедливы следующие формулы:

Скорость материальной точки: $v = \frac{S}{t}$

Путь, пройденный материальной точкой: $\Delta S = v * t$

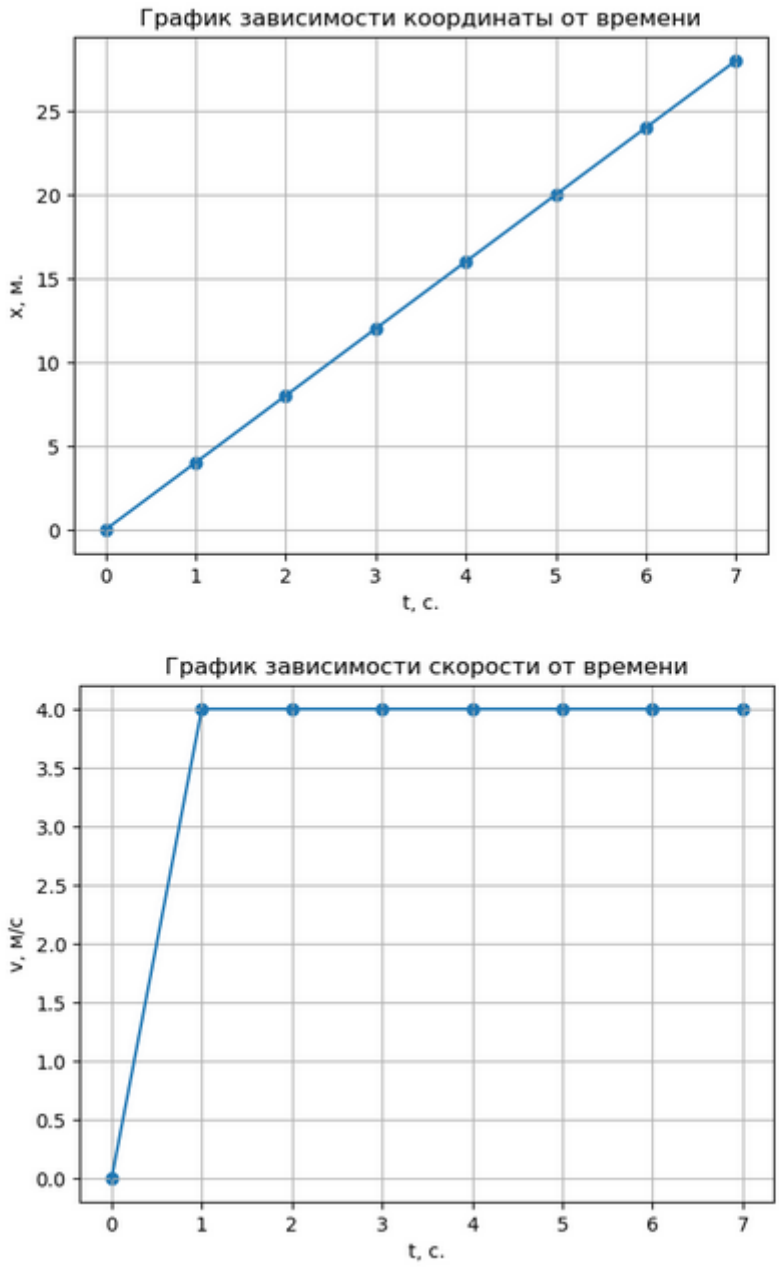
In [2]:
```
v, x, t = [0], [0], [0]
time = 7
for i in range(1, 8):
    xt = 4*i
    v2 = xt/i
    v.append(v2)
    t.append(i)
    x.append(xt)
print('Скорость мат. точки равна: ', v2, 'м/с')
print('Путь, пройденный за 7 с движения: ', v2*time,'м')
```

```
Скорость мат. точки равна:  4.0 м/с
Путь, пройденный за 7 с движения:  28.0 м
```

Построим линейные диаграммы по полученным данным:

In [3]:
```
plt.grid()
plt.title('График зависимости координаты от времени')
plt.xlabel('t, с.')
plt.ylabel('x, м.')
plt.plot(t, x)
plt.scatter(t, x)
plt.show()

plt.grid()
plt.title('График зависимости скорости от времени')
plt.xlabel('t, с.')
plt.ylabel('v, м/с')
plt.plot(t, v)
plt.scatter(t, v)
plt.show()
```

Рисунок 16 – Задание 1 (1)



In [ ]:

Рисунок 17 – Задание 1 (2)

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.

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.

Дан набор данных по продажам на разных платформах видеоигр, продажи которых превысили 100 000 копий с 1992 по 2020 год. Построить график мировых продаж игр на всех платформах по годам и найти год, в который было продано наибольшее количество копий игр.

Импорт модулей

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import csv
%matplotlib inline
```

Импорт набора данных

In [2]:
```
data = []
x = []
y = []

df = pd.read_csv('vgsales.csv', encoding="cp1252")
df
```

Out[2]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
...	...	...	...	...	...	...	...	...	...	...	...
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.00	0.00	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.00	0.00	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.00	0.00	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.00	0.00	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.00	0.01

16598 rows × 11 columns

В этом наборе нас интересуют столбцы Year - год и Global_Sales - показатель проданных копий игр в мире в млн. копиях. Собираем данные по годам и считаем сумму проданных копий.

In [3]:
```
data = []
x = []
y = []

res = df.groupby(['Year'], sort=True)['Global_Sales'].sum()

years = df["Year"].values.tolist()
years = list(set(years))
years = [x for x in years if str(x) != 'nan']
years.sort()
amount = res.tolist()
```

Рисунок 18 – Задание 2 (1)

Полученные данные отобразим в виде столбчатой диаграммы.

In [4]:
```
plt.figure(figsize=(10,6))
plt.bar(years, amount)
plt.title('Мировые продажи игр (в  млн. копиях) по годам', fontsize=12)
plt.grid(True)
```

Из полученных данных можно отметить следующее:

- Самое большее число проданных копий игр приходится на 2008 год.
- С появлением более доступных платформ для игр, количество проданных копий значительно увеличилось (с 1995 года видим резкий взлет)

In [ ]:

Рисунок 19 – Задание 2 (2)

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.

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.

Дан набор данных обзора показателей самоубийств с 1985 по 2016 год. Сравнивает социально-экономическую информацию с уровнем самоубийств по годам и странам.

Задача: проанализировать соотношение полов по совершенным самоубийствам трех стран в возрасте от 15 до 24 лет.

Импорт модулей

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

Импорт данных

In [2]:
```
x = []
y = []
df = pd.read_csv('master.csv', encoding="UTF-8")
df
```

Out[2]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	generation
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796	Generation X
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796	Silent
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796	Generation X
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796	G.I. Generation
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796	Boomers
...	...	...	...	...	...	...	...	...	...	...	...	...
27815	Uzbekistan	2014	female	35-54 years	107	3620833	2.96	Uzbekistan2014	0.675	63,067,077,179	2309	Generation X
27816	Uzbekistan	2014	female	75+ years	9	348465	2.58	Uzbekistan2014	0.675	63,067,077,179	2309	Silent
27817	Uzbekistan	2014	male	5-14 years	60	2762158	2.17	Uzbekistan2014	0.675	63,067,077,179	2309	Generation Z
27818	Uzbekistan	2014	female	5-14 years	44	2631600	1.67	Uzbekistan2014	0.675	63,067,077,179	2309	Generation Z
27819	Uzbekistan	2014	female	55-74 years	21	1438935	1.46	Uzbekistan2014	0.675	63,067,077,179	2309	Boomers

27820 rows × 12 columns

Рисунок 20 – Задание 3 (1)

Эти данные можно представить круговой диаграммой следующим образом:

In [3]:
```
import random
import matplotlib.colors as mcolors

countries = df.loc[:,"country"]
countries = list(set(countries))
countries.sort()

su_sum = df.groupby(['country'])['suicides_no'].sum()
values = su_sum.tolist()

cs = random.choices(list(mcolors.CSS4_COLORS.values()),k = len(countries))
plt.figure(figsize=(1280, 720))
explode = (0.1, 0.15)
labels = ["Женщины", "Мужчины"]
fig, ax = plt.subplots()
ax.pie(su_sum, colors=cs)
ax.legend(countries, loc="upper right", bbox_to_anchor=(2, 1.05), fontsize=6 ,ncol=3, fancybox=True, shadow=True)

plt.title("Число самоубийств по странам")
plt.show()
```

```
<Figure size 128000x72000 with 0 Axes>
```

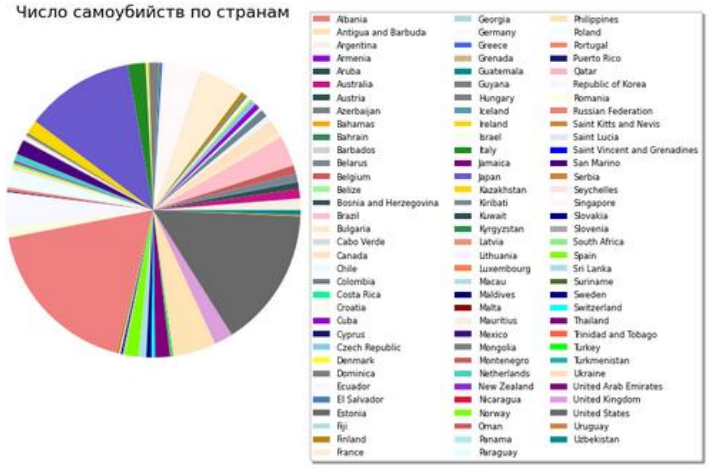


Рисунок 21 – Задание 1 (2)

Из исходного набора данных нас интересуют столбцы: Страна и год, поколение (Generation X) и пол. Составим из них датафрейм.

In [4]:
```
res = df.groupby(['country-year','sex', 'generation'], as_index=False)['suicides_no'].sum()
res
```

Out[4]:

	country-year	sex	generation	suicides_no
0	Albania1987	female	Boomers	4
1	Albania1987	female	G.I. Generation	1
2	Albania1987	female	Generation X	14
3	Albania1987	female	Silent	6
4	Albania1987	male	Boomers	9
...	...	...	...	...
21341	Uzbekistan2014	male	Boomers	144
21342	Uzbekistan2014	male	Generation X	519
21343	Uzbekistan2014	male	Generation Z	60
21344	Uzbekistan2014	male	Millenials	665
21345	Uzbekistan2014	male	Silent	17

21346 rows × 4 columns

Отфильтруем значения датафрейма. В качестве стран данных для анализа возьмем статистические данные за 2015 год по странам: Россия, США и Германия.

In [5]:
```
res1 = res.loc[res['country-year'].isin(['Russian Federation2015']) & res['generation'].isin(['Generation X'])]
res2 = res.loc[res['country-year'].isin(['United States2015']) & res['generation'].isin(['Generation X'])]
res3 = res.loc[res['country-year'].isin(['Germany2015']) & res['generation'].isin(['Generation X'])]
print(res1, '\n', res2, '\n', res3)
x = res1['suicides_no'].values.tolist()
y = res2['suicides_no'].values.tolist()
z = res3['suicides_no'].values.tolist()
print(x, y, z)
```

```
               country-year     sex    generation  suicides_no
16313  Russian Federation2015  female  Generation X         1391
16318  Russian Federation2015    male  Generation X         7898
            country-year     sex    generation  suicides_no
20877  United States2015  female  Generation X         4053
20882  United States2015    male  Generation X        11634
      country-year     sex    generation  suicides_no
7695  Germany2015  female  Generation X          800
7700  Germany2015    male  Generation X         2267
[1391, 7898] [4053, 11634] [800, 2267]
```

Рисунок 22 – Задание 3 (3)

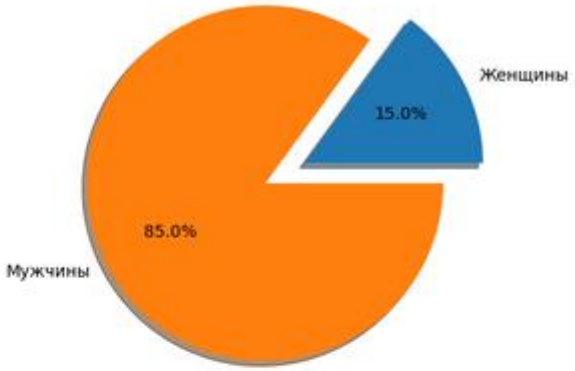
Для наглядности построим круговую диаграмму по полученным данным.

In [6]:
```
plt.figure(figsize=(30, 30))
explode = (0.1, 0.15)
labels = ["Женщины", "Мужчины"]
fig, ax = plt.subplots()
ax.pie(x, labels=labels, shadow=True, autopct='%1.1f%%',explode=explode)
plt.title("Количество самоубийств, совершенных в России в 2015 году")
plt.show()

plt.figure(figsize=(30, 30))
explode = (0.1, 0.15)
labels = ["Женщины", "Мужчины"]
fig, ax = plt.subplots()
ax.pie(y, labels=labels, shadow=True, autopct='%1.1f%%',explode=explode)
plt.title("Количество самоубийств, совершенных в США в 2015 году")
plt.show()

plt.figure(figsize=(30, 30))
explode = (0.1, 0.15)
labels = ["Женщины", "Мужчины"]
fig, ax = plt.subplots()
ax.pie(z, labels=labels, shadow=True, autopct='%1.1f%%',explode=explode)
plt.title("Количество самоубийств, совершенных в Германии в 2015 году")
plt.show()
```

```
<Figure size 3000x3000 with 0 Axes>
```

Рисунок 23 – Задание 3 (4)

Количество самоубийств, совершенных в России в 2015 году

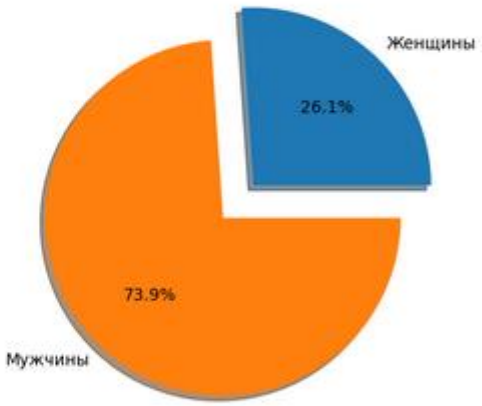


<Figure size 3000x3000 with 0 Axes>

Количество самоубийств, совершенных в США в 2015 году

<Figure size 3000x3000 with 0 Axes>

Количество самоубийств, совершенных в Германии в 2015 году



Из полученных результатов можно отметить следующее:

- В каждой из анализируемых стран большее количество самоубийств совершено мужчинами.
- В России, показатели соотношения значительно отличаются.

Рисунок 24 – Задание 3 (5)

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

Найти какое-либо изображение в сети Интернет.

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

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

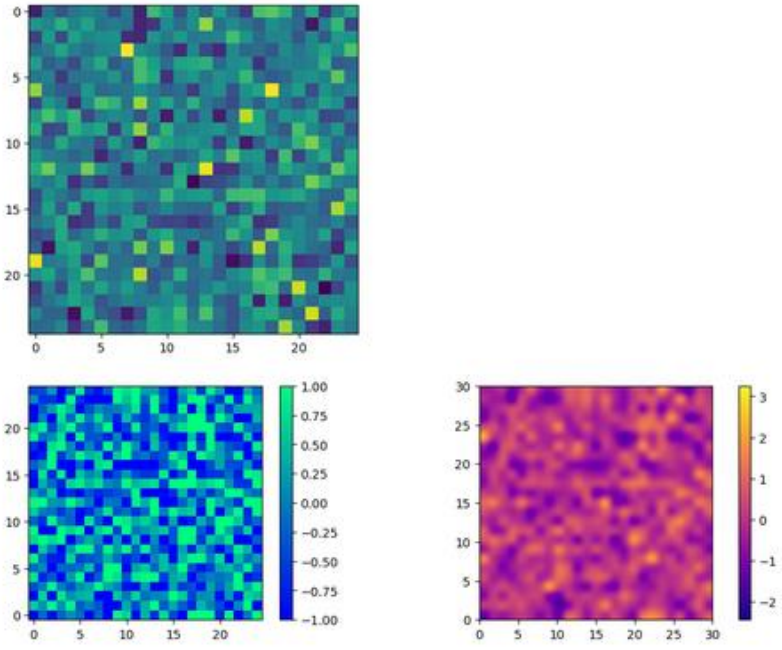
In [2]:
```
from PIL import Image
import requests
from io import BytesIO

response = requests.get('https://archive.org/download/windows-xp-bliss-wallpaper/windows-xp-bliss-4k-lu-1920x1080.jpg')
img = Image.open(BytesIO(response.content))
plt.imshow(img)
```

Out[2]: <matplotlib.image.AxesImage at 0x1f7834e5b80>

Тепловая карта изображения

In [3]:
```
np.random.seed(19680)
data = np.random.randn(25, 25)
plt.imshow(data)

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[3]: <matplotlib.colorbar.Colorbar at 0x1f784d23100>



In [ ]:

Рисунок 4 – Задание 4

Вывод: В результате выполнения работы были исследованы возможности визуализации данных при помощи пакета matplotlib языка Python.

Контрольные вопросы:

1. Как выполнить построение линейного графика с помощью matplotlib?

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

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

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

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

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

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

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

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')
```

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)
```

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")

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. Также, такие карты можно создавать самостоятельно, если среди существующих нет подходящего решения. imshow() и pcolormesh().

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

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

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

16. Как отобразить тепловую карту средствами matplotlib?

Рассмотрим ещё одну функцию для визуализации 2D наборов данных – pcolormesh(). В библиотеке Matplotlib есть ещё одна функция с аналогичным функционалом – pcolor(), в 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.