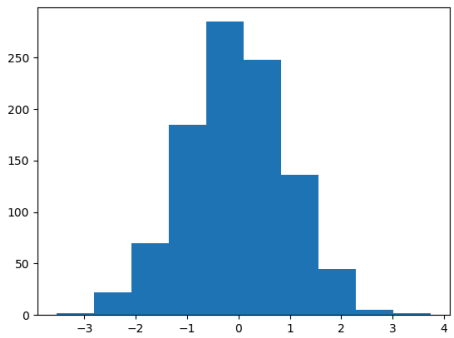
2.6 Визуализация данных в Matplotlib

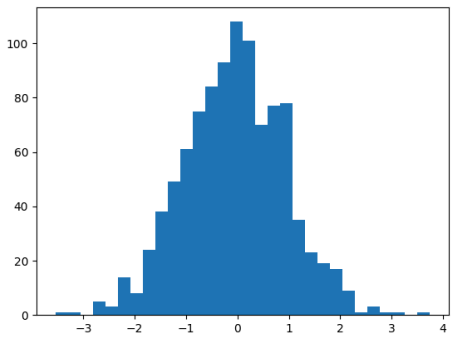
Одномерная гистограмма

Параметры

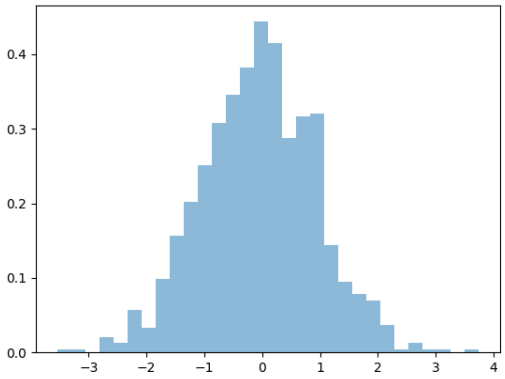
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

rng = np.random.default_rng(1)
data = rng.normal(size=1000)

plt.hist(data,
        bins=30,
        density=True,
        alpha=.5,
        histtype="step")

plt.savefig('img.png')
plt.show()
```



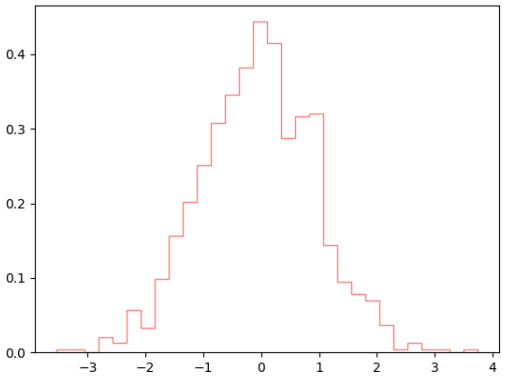
bins = 10 (default)



bins = 30



bins = 30, density=True, alpha=0.5



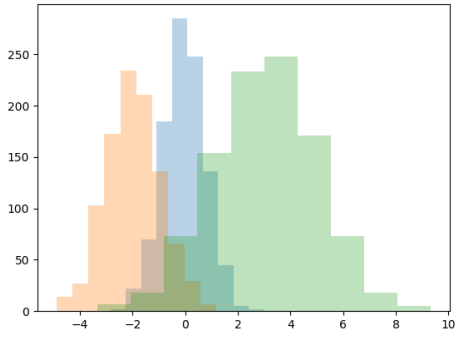
bins = 30, density=True, alpha=0.5, histtype="step"

- bins - (бинирование) используется для определения количества интервалов (или "бинов"), на которые будет разбито множество данных. Каждый "бин" представляет собой диапазон значений, и гистограмма показывает, сколько данных попадает в каждый из этих интервалов.
- density - нормализации гистограммы. Когда вы устанавливаете density=True, значения на оси Y представляют собой вероятностную плотность, а не абсолютные частоты. Это означает, что площадь под гистограммой будет равна 1.
- alpha - используется для управления прозрачностью (или непрозрачностью) графических элементов, таких как гистограммы. Значение alpha варьируется от 0 до 1.
- histtype - определяет стиль (тип) гистограммы, который будет использован для отображения данных.
    1. 'bar' (по умолчанию): Стандартная гистограмма с прямоугольниками, представляющими частоты значений.
    2. 'barstacked': Cтacked bar histogram. Гистограммы накладываются друг на друга, что позволяет видеть общий объем значений.
    3. 'step': Гистограмма, представленная линиями, которые соединяют верхние точки столбиков. Это полезно для более четкого отображения распределения.
    4. 'stepfilled': Похож на 'step', но заполнен цветом под линией.
- edgecolor - позволяет задать цвет границ (контуров) столбиков гистограммы.

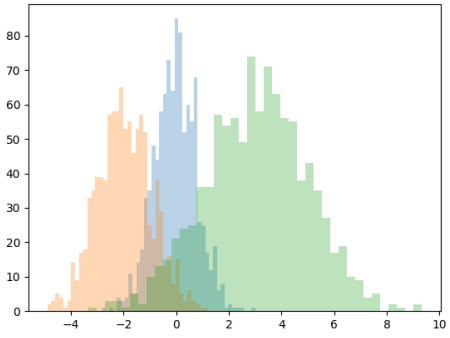
Отображение нескольких гистограмм на одной области

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

rng = np.random.default_rng(1)
```

```
x1 = rng.normal(0, 0.8, 1000)
x2 = rng.normal(-2, 1, 1000)
x3 = rng.normal(3, 2, 1000)

args = dict(
    alpha=0.3,
    bins=40
)

plt.hist(x1, **args)
plt.hist(x2, **args)
plt.hist(x3, **args)

plt.savefig('img.png')
plt.show()
```



alpha=0.3



alpha=0.3, bins=40

1. 0 — это среднее значение (mean) нормального распределения. В данном случае, это значение, вокруг которого будут сосредоточены сгенерированные случайные числа. То есть, в среднем, значения будут близки к 0.
2. 0.8 — это стандартное отклонение (standard deviation) нормального распределения. Стандартное отклонение определяет, насколько сильно значения будут разбросаны относительно среднего. Чем больше значение стандартного отклонения, тем шире будет распределение. В данном случае стандартное отклонение равно 0.8.
3. 1000 — это количество случайных чисел, которые вы хотите сгенерировать. В этом случае будет создан массив из 1000 значений, распределенных по нормальному закону с заданными средним и стандартным отклонением.

Обзор массива данных гистограммы

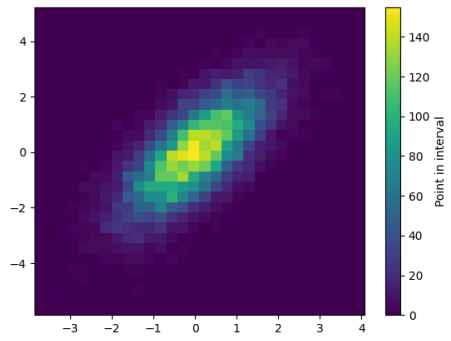
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

rng = np.random.default_rng(1)

x1 = rng.normal(0, 0.8, 1000)
x2 = rng.normal(-2, 1, 1000)
x3 = rng.normal(3, 2, 1000)

print(np.histogram(x1, bins=1))
# ⇒ (array([1000], dtype=int64), array([-2.83904398,  3.00130797]))

print(np.histogram(x1, bins=2))
# ⇒ (array([564, 436], dtype=int64), array([-2.83904398,  0.081132  ,  3.00130797]))

print(np.histogram(x1, bins=40))
# ⇒(array([ 1,  0,  1,  0,  4,  3,  4, 11,  5, 14, 18, 33, 35, 48, 44, 58, 63,
#    73, 64, 85, 81, 52, 60, 55, 68, 26, 25, 17, 12, 19,  6,  8,  1,  2,
#     1,  1,  1,  0,  0,  1], dtype=int64),
#          array([-2.83904398, -2.69303518, -2.54702638, -2.40101758, -2.25500878,
#    -2.10899998, -1.96299118, -1.81698239, -1.67097359, -1.52496479,
#    -1.37895599, -1.23294719, -1.08693839, -0.94092959, -0.79492079,
#    -0.648912  , -0.5029032 , -0.3568944 , -0.2108856 , -0.0648768 ,
#     0.081132  ,  0.2271408 ,  0.3731496 ,  0.51915839,  0.66516719,
#     0.81117599,  0.95718479,  1.10319359,  1.24920239,  1.39521119,
#     1.54121999,  1.68722878,  1.83323758,  1.97924638,  2.12525518,
#     2.27126398,  2.41727278,  2.56328158,  2.70929038,  2.85529918,
#     3.00130797]))
```

данный фрагмент кода позволяет увидеть, как распределены значения в массиве x1 по n (1, 2, 40)интервалам.

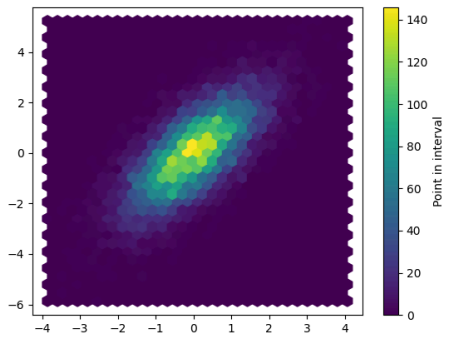
Двумерная гистограмма

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

rng = np.random.default_rng(1)
```

```
mean = [0, 0]
cov = [[1, 1], [1, 2]]

x, y = rng.multivariate_normal(mean, cov, 10000).T

plt.hist2d(x, y, bins=30)
#plt.hexbin(x, y, gridsize=30)

cb = plt.colorbar()
cb.set_label("Point in interval")

plt.savefig("img.png")
plt.show()
```



plt.hist2d(x, y, bins=30)



plt.hexbin(x, y, gridsize=30)

Обзор массива данных двумерной гистограммы

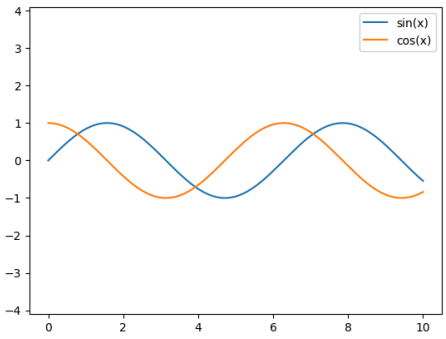
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

rng = np.random.default_rng(1)

mean = [0, 0]
cov = [[1, 1], [1, 2]]

x, y = rng.multivariate_normal(mean, cov, 10000).T
plt.hist2d(x, y, bins=30)

print(np.histogram2d(x, y, bins=1))
```

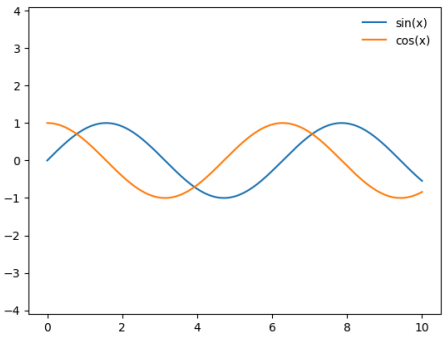
```
# ⇒ (array([[10000.]]), array([-3.85933046,  4.06952191]), array([-5.86398754,  5.23174817]))

print(np.histogram2d(x, y, bins=10))
# ⇒ (array([[3.000e+00, 2.000e+00, 2.000e+00, 3.000e+00, 0.000e+00, 0.000e+00,
#     0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00],
#    [2.000e+00, 1.000e+01, 2.700e+01, 3.100e+01, 1.600e+01, 1.000e+00,
#     0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00],
#    [1.000e+00, 1.400e+01, 1.160e+02, 2.040e+02, 1.530e+02, 4.000e+01,
#     4.000e+00, 0.000e+00, 0.000e+00, 0.000e+00],
#    [2.000e+00, 1.200e+01, 1.180e+02, 4.850e+02, 7.390e+02, 3.510e+02,
#     6.100e+01, 7.000e+00, 0.000e+00, 0.000e+00],
#    [0.000e+00, 4.000e+00, 3.300e+01, 3.710e+02, 9.940e+02, 1.120e+03,
#     4.120e+02, 5.800e+01, 0.000e+00, 0.000e+00],
#    [0.000e+00, 0.000e+00, 3.000e+00, 8.400e+01, 5.240e+02, 1.084e+03,
#     8.170e+02, 2.300e+02, 1.000e+01, 1.000e+00],
#    [0.000e+00, 0.000e+00, 0.000e+00, 4.000e+00, 9.200e+01, 3.820e+02,
#     5.830e+02, 2.850e+02, 4.700e+01, 5.000e+00],
#    [0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 2.000e+00, 3.700e+01,
#     1.480e+02, 1.360e+02, 5.700e+01, 2.000e+00],
#    [0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 3.000e+00,
#     1.200e+01, 2.100e+01, 2.000e+01, 7.000e+00],
#    [0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
#     0.000e+00, 4.000e+00, 1.000e+00, 3.000e+00]]), array([-3.85933046, -3.06644522, -2.27355
#     0.10509573,  0.89798096,  1.6908662 ,  2.48375144,  3.27663667,
#     4.06952191]), array([-5.86398754, -4.75441397, -3.6448404 , -2.53526683, -1.42569326,
#    -0.31611969,  0.79345388,  1.90302746,  3.01260103,  4.1221746 ,
#     5.23174817]))
```

Легенда для графика

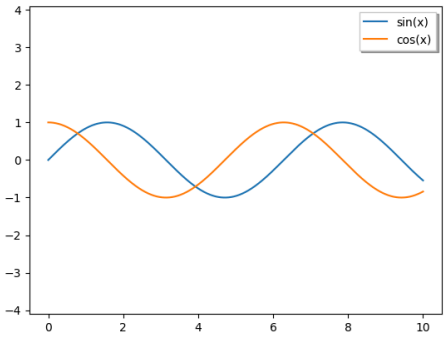
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)

fig, ax = plt.subplots()

ax.plot(x, np.sin(x), label="sin(x)")
ax.plot(x, np.cos(x), label="cos(x)")
ax.axis("equal")
```

```
plt.legend(frameon=True,
           fancybox=False,
           shadow=True)

plt.savefig("img.png")
plt.show()
```





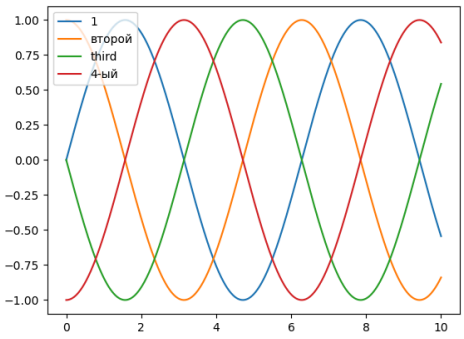
frameon=False



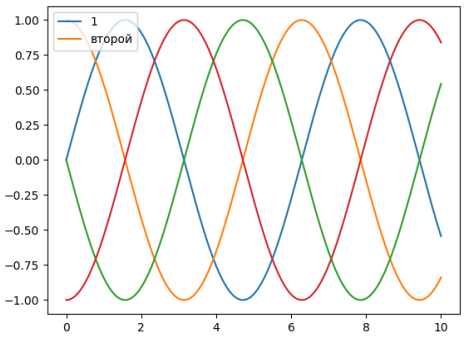
frameon=True, fancybox=True, shadow=True

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

x = np.linspace(0, 10, 1000)
y = np.sin(x[:, np.newaxis] + np.pi * np.arange(0, 2, 0.5))

lines = plt.plot(x, y) # plt.Line2d
```

```
plt.legend(lines, ['1', 'второй', 'third', '4-ый'], loc="upper left")
#plt.legend(lines[:2], ['1', 'второй'], loc="upper left")

plt.savefig("img.png")
plt.show()
```



(lines, ['1', 'второй', 'third', '4-ый'])

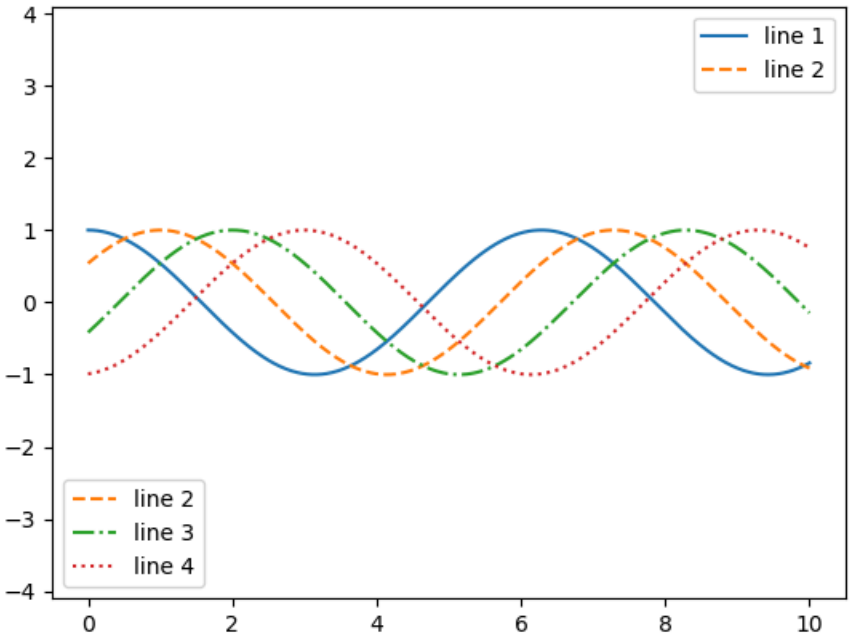


(lines[:2], ['1', 'второй'])

Размещение нескольких легенд

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

lines = []
styles = ["-", "--", "-.", ":"]

x = np.linspace(0, 10, 1000)

for i in range(4):
    lines += ax.plot(
        x,
        np.sin(x - i + np.pi/2),
        styles[i]
    )

ax.axis("equal")  # Выравнивание графиков

ax.legend(lines[:2], ["line 1", "line 2"], loc="upper right")
```

```
leg = mpl.legend.Legend(ax, lines[1:], ["line 2", "line 3", "line 4"], loc="lower left")

ax.add_artist(leg)

plt.savefig("img.png")
plt.show()
```



Шкалы

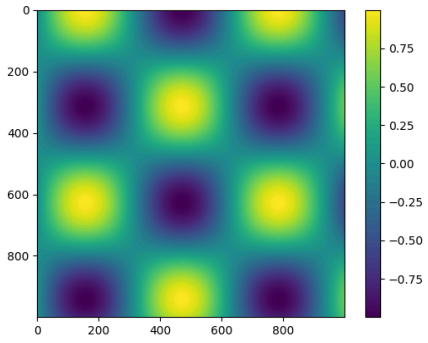
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x) * np.cos(x[:, np.newaxis])

plt.imshow(y, cmap="Greys")
plt.colorbar()
```

```
plt.savefig("img.png")
plt.show()
```

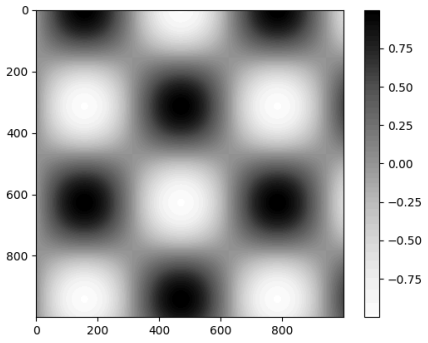
Карты цветов cmap

- Последовательные Sequential Colormaps

  Последовательные карты цветов предназначены для представления непрерывных данных, где значения варьируются от низкого к высокому. Они обычно имеют один цвет, который постепенно меняется от светлого к темному (или наоборот).



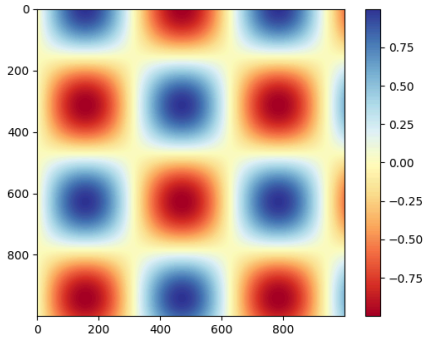
cmap="viridis"



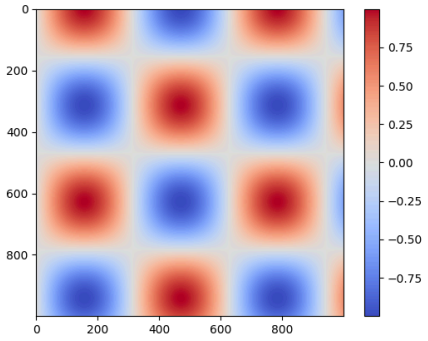
cmap="Greys"

- Дивергентные Diverging Colormaps (два цвета)

  Дивергентные карты цветов используются для данных, которые имеют два крайних значения и важен центральный ноль или среднее значение. Обычно они состоят из двух разных цветов, которые расходятся от центра.



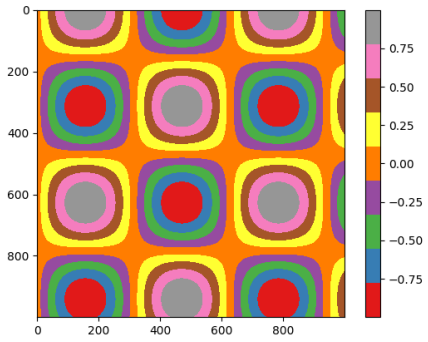
cmap="RdYlBu"



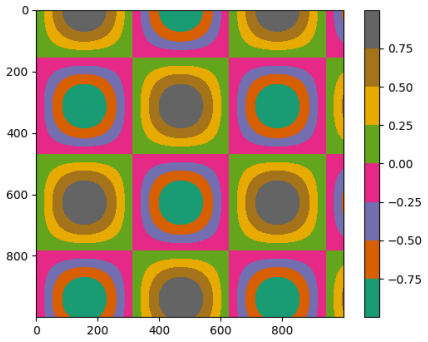
cmap="coolwarm"

- Качественные Qualitative Colormaps (смешиваются без четкого порядка)

Качественные карты цветов предназначены для представления категориальных данных, где значения не имеют четкого порядка. Эти карты используют разные цвета для различения категорий.

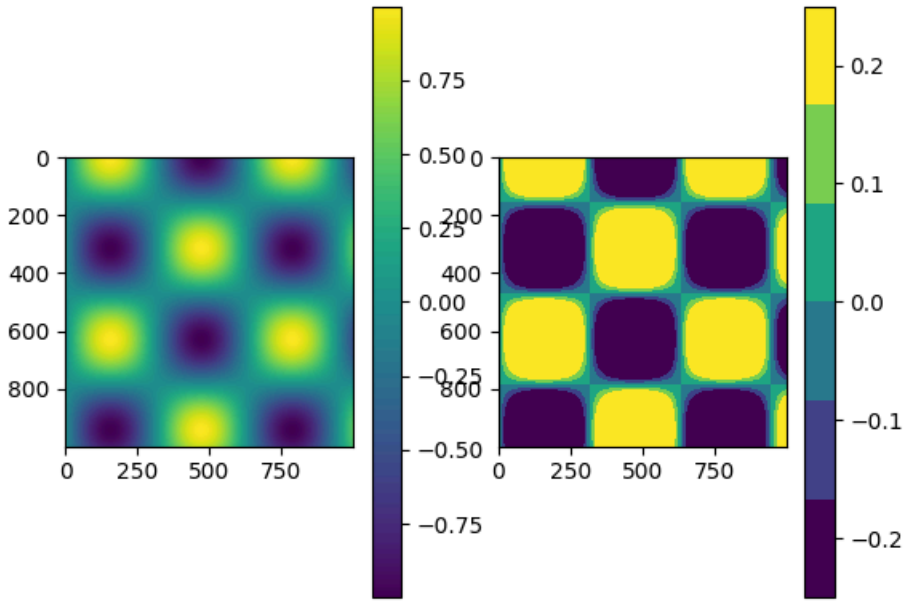


cmap="Set1"



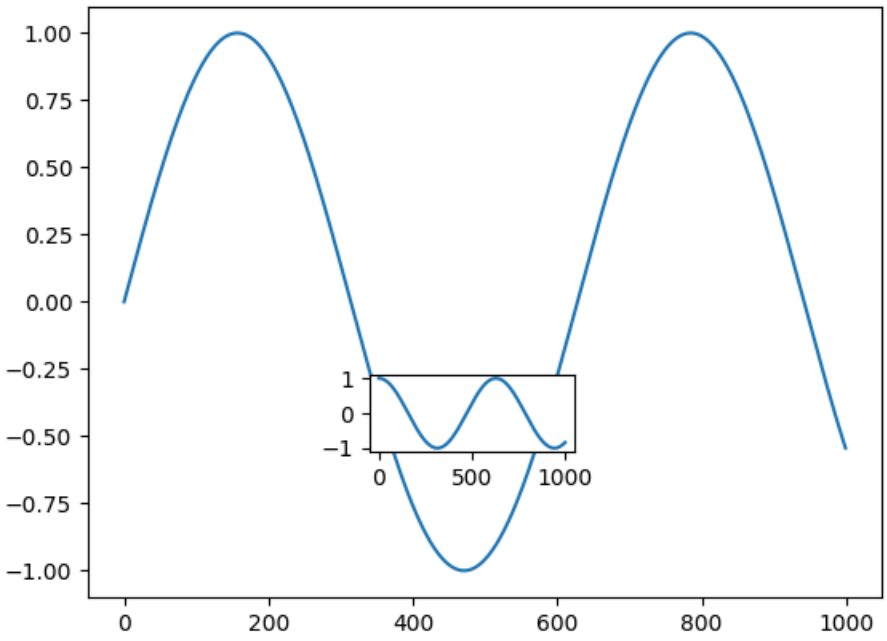
cmap="Dark2"

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x) * np.cos(x[:, np.newaxis])

plt.figure()

plt.subplot(1, 2, 1)
plt.imshow(y, cmap="viridis")
plt.colorbar()

plt.subplot(1, 2, 2)
plt.imshow(y, cmap=plt.cm.get_cmap("viridis", 6))
plt.colorbar()
plt.clim(-0.25, 0.25)


plt.savefig("img.png")
plt.show()
```

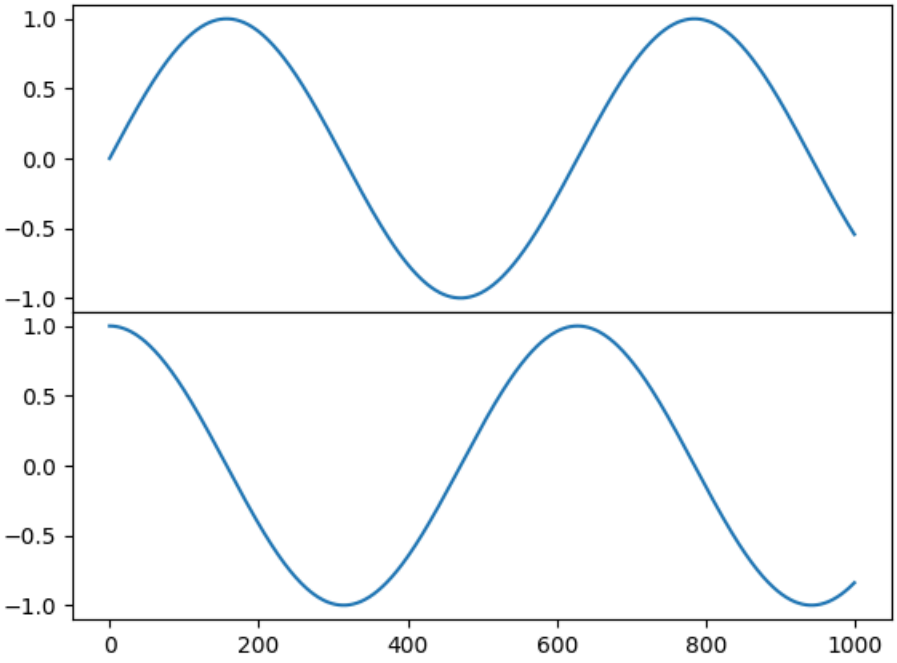


Управление Layout (Несколько областей)

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x) * np.cos(x[:, np.newaxis])

ax1 = plt.axes()

# [нижний угол, левый угол, ширина, высота]
ax2 = plt.axes([0.4, 0.3, 0.2, 0.1])

ax1.plot(np.sin(x))
ax2.plot(np.cos(x))
plt.savefig("img.png")
plt.show()
```

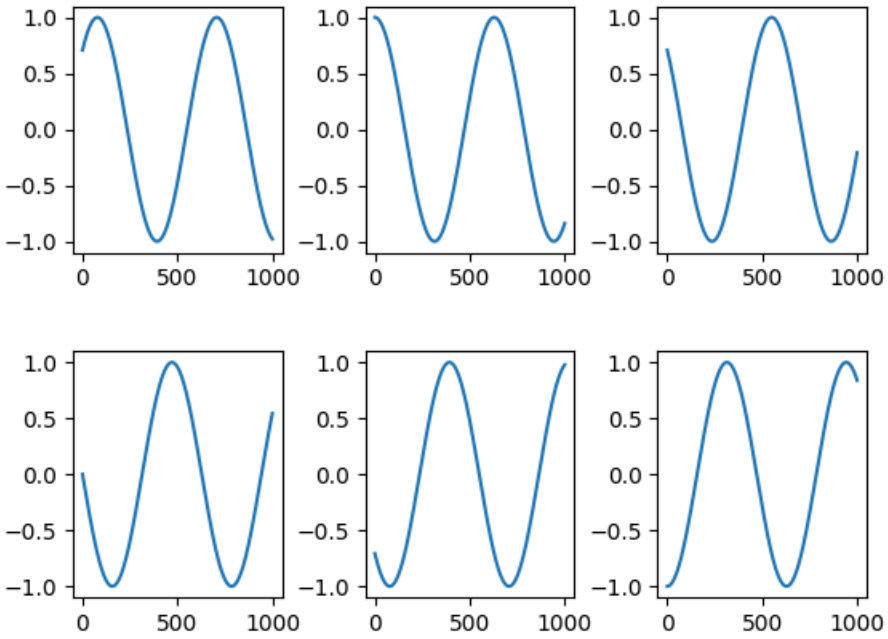



```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x) * np.cos(x[:, np.newaxis])

ax1 = plt.axes()

# [нижний угол, левый угол, ширина, высота]
ax2 = plt.axes([0.4, 0.3, 0.2, 0.1])

ax1.plot(np.sin(x))
ax2.plot(np.cos(x))
plt.savefig("img.png")
plt.show()
```



Простые сетки

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x) * np.cos(x[:, np.newaxis])


plt.savefig("img.png")
plt.show()
```



Использование DataSet

data.tar.gz

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

cities = pd.read_csv("./data/california_cities.csv")

lat, lon, pop, area = cities["latd"], cities["longd"], \
    cities["population_total"], cities["area_total_km2"]

plt.scatter(lon, lat, c=np.log10(pop), s=area) # c - color, s - size

plt.xlabel("Широта")
plt.ylabel("Долгота")

plt.colorbar()
```

```
plt.clim(3, 7)

plt.scatter([], [], c="k", alpha=0.5, s=100, label="$100 km^2$")
plt.legend()

plt.savefig("img.png")
plt.show()
```