2.5 Введение в Matplotlib

Python программы можно запускать в разных оболочках:

- Сценарий
- Командная оболочка IPython
- Jupyter

Запуск через IDLE

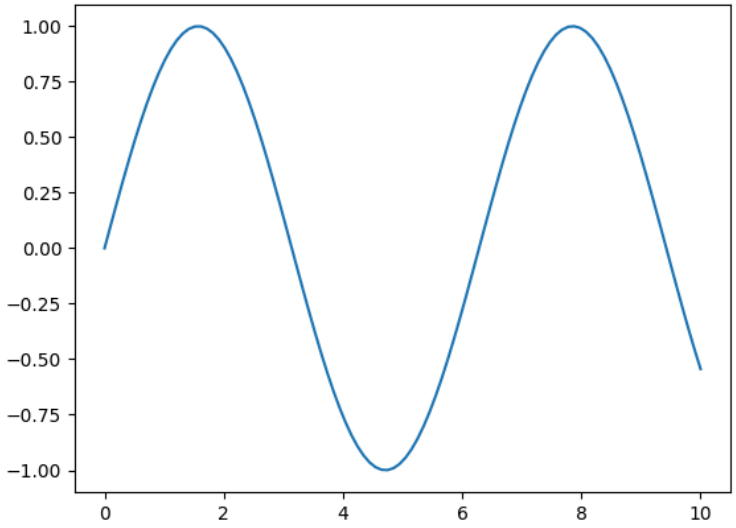
Для демонстрации результата используется команда plt.show() - запускается только один раз

В процессе работы код создает объекты класса Figure.

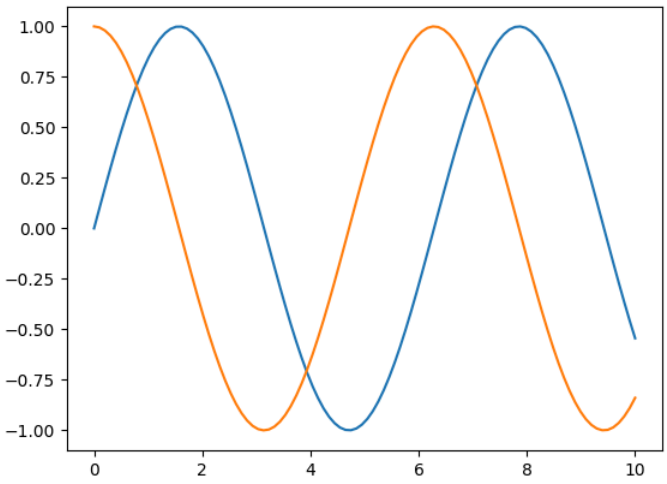
Для сохранения изображений в выбранной директории используется команда plt.savefig('file_name.png')

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))

plt.savefig('img.png')

plt.show()
plt.plot(x, np.cos(x))
```

Код отобразить только то, что было вызвало до команды plt.show()




```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.savefig('img.png')

plt.show()
```



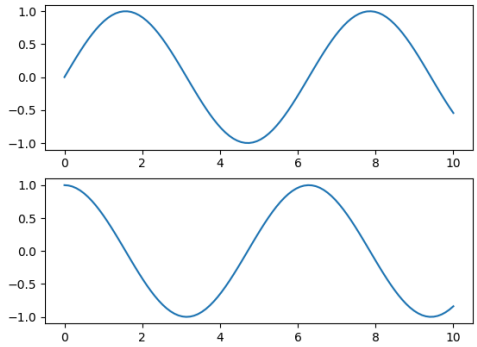
Для того, чтобы узнать с какими расширениями файл может сохраняться можно использовать plt.figure().canvas.get_supported_filetypes()

```
import matplotlib.pyplot as plt
import numpy as np

print(plt.figure().canvas.get_supported_filetypes())
# ⇒ {'eps': 'Encapsulated Postscript', 'jpg': 'Joint Photographic Experts Group',
#    'jpeg': 'Joint Photographic Experts Group', 'pdf': 'Portable Document Format',
#    'pgf': 'PGF code for LaTeX', 'png': 'Portable Network Graphics', 'ps': 'Postscript',
#    'raw': 'Raw RGBA bitmap', 'rgba': 'Raw RGBA bitmap',
#    'svg': 'Scalable Vector Graphics', 'svgz': 'Scalable Vector Graphics',
#    'tif': 'Tagged Image File Format', 'tiff': 'Tagged Image File Format',
#    'webp': 'WebP Image Format'}
```

Два способа выведения графиков

1. MATLAB-подобный стиль

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)

plt.figure()
                #строка, колонка, обращение к какому графику
plt.subplot(2, 1, 1)
plt.plot(x, np.sin(x))

plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x))

plt.savefig('img.png')

plt.show()
```



Недостатки:

- Когда мы находимся в n-ой строчке, вернуться к начальному графику затруднительно (необходимо пролистать весь код вверх).
- Нет контроля что и куда рисуется.

2. Объектно-ориентированный (OO) стиль

В OO стиле сначала создается сетка графиков

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)

#fig:Figure, ax:Axes
fig, ax = plt.subplots(2)
ax[0].plot(x, np.sin(x))
ax[1].plot(x, np.cos(x))

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

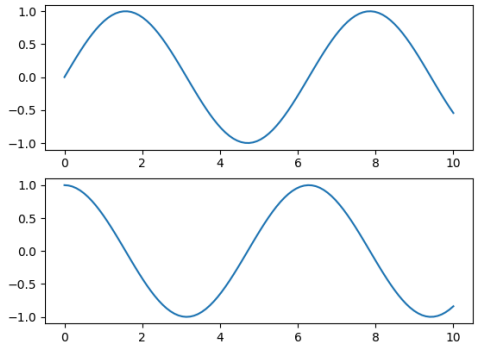
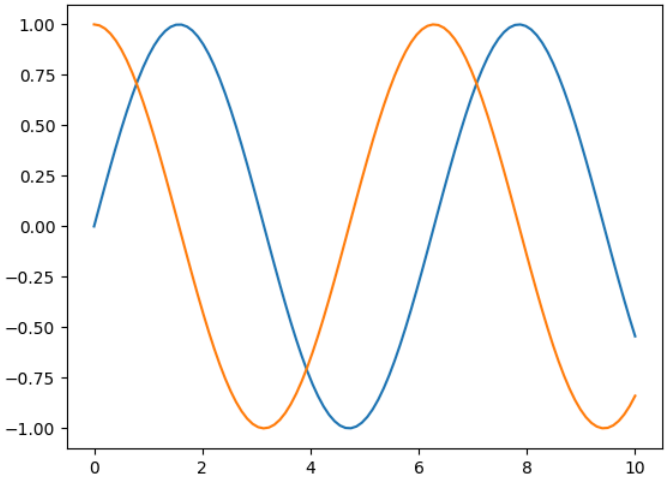


fig:plt.Figure - контейнер, содержит объекты (СК, тексты, метки),

ax:Axes - система координат (СК) - прямоугольник, деления, метки

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)

fig = plt.figure()
ax = plt.axes()

ax.plot(x, np.sin(x))
ax.plot(x, np.cos(x))

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



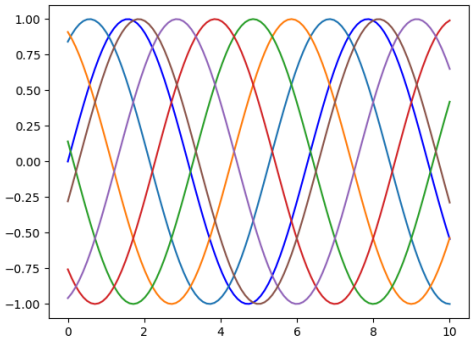
Стилизация графиков

Цвета линий (color)

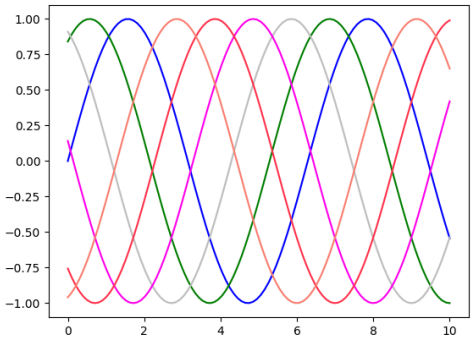
Способы задания:

- ‘blue’, ‘red’, ‘orange’
- ‘rgbcmyk’ → ‘rg’
- ‘0.14’ - градация серого от 0 до 1
- RRGGBB - ‘FF00EE’
- RGB - (1.0, 0.2, 0.3)
- HTML - ‘salmon’

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

x = np.linspace(0, 10, 100)

fig = plt.figure()
ax = plt.axes()

ax.plot(x, np.sin(x))
ax.plot(x, np.sin(x + 1))
ax.plot(x, np.sin(x + 2))
ax.plot(x, np.sin(x + 3))
ax.plot(x, np.sin(x + 4))
ax.plot(x, np.sin(x + 5))
ax.plot(x, np.sin(x + 6))

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




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

x = np.linspace(0, 10, 100)

fig = plt.figure()
ax = plt.axes()

ax.plot(x, np.sin(x), color='blue')
ax.plot(x, np.sin(x + 1), color='g')
ax.plot(x, np.sin(x + 2), color='0.75')
ax.plot(x, np.sin(x + 3), color='#FF00EE')
ax.plot(x, np.sin(x + 4), color=(1.0, 0.2, 0.3))
ax.plot(x, np.sin(x + 5), color='salmon')

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

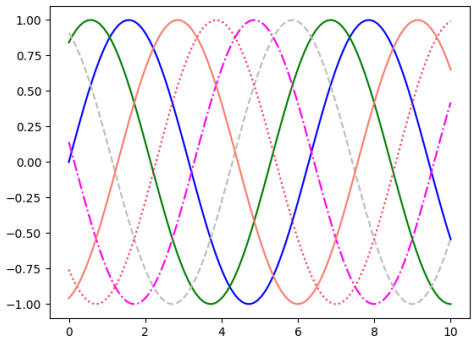


Стили линий (вид кривой linestyle)

Виды линий :

- сплошная ‘-’, ‘solid’
- штриховая ‘--’, ‘dashed’
- штрих-пунктирная ‘-.’, ‘dashdot’
- пунктирная ‘:’, ‘dotted’

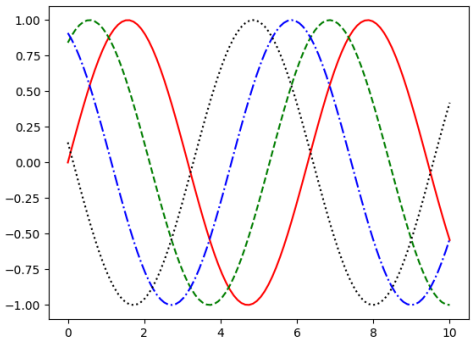
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)

fig = plt.figure()
ax = plt.axes()

ax.plot(x, np.sin(x), color='blue',)
ax.plot(x, np.sin(x + 1), color='g', linestyle='solid')
ax.plot(x, np.sin(x + 2), color='0.75', linestyle='dashed')
ax.plot(x, np.sin(x + 3), color='#FF00EE', linestyle='dashdot')
ax.plot(x, np.sin(x + 4), color=(1.0, 0.2, 0.3), linestyle='dotted')
ax.plot(x, np.sin(x + 5), color='salmon')

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



Короткая запись цветов и вида кривой

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

```
x = np.linspace(0, 10, 100)

fig = plt.figure()
ax = plt.axes()

ax.plot(x, np.sin(x), '-r',)
ax.plot(x, np.sin(x + 1), '--g')
ax.plot(x, np.sin(x + 2), '-.b')
ax.plot(x, np.sin(x + 3), ':k')

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

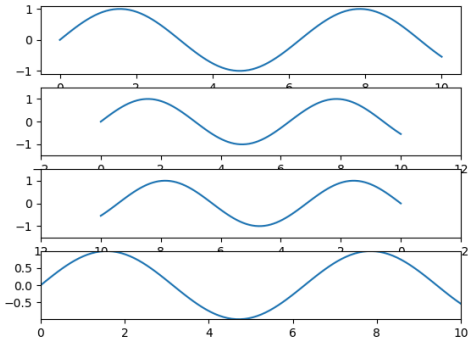


Редактирование осей координат

Ограничение ширины и высоты осей

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

x = np.linspace(0, 10, 100)

fig, ax = plt.subplots(4)

ax[0].plot(x, np.sin(x))
ax[1].plot(x, np.sin(x))
ax[2].plot(x, np.sin(x))
ax[3].plot(x, np.sin(x))

# Ограничение по X
```

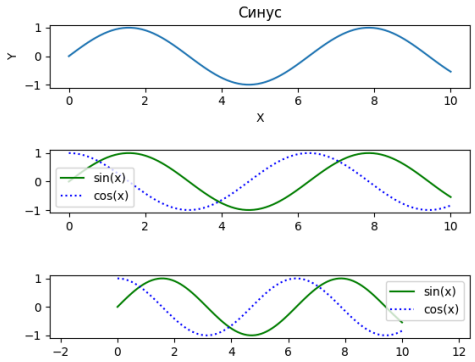
```
ax[1].set_xlim(-2, 12)

# Ограничение по Y
ax[1].set_ylim(-1.5, 1.5)

# Отзеркаливание функции
ax[2].set_xlim(12, -2)
ax[2].set_ylim(-1.5, 1.5)

# Автоматическое масштабирование
ax[3].autoscale(tight=True)

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



Название осей, title легенда (label)

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

x = np.linspace(0, 10, 100)

plt.subplot(3, 1, 1)
plt.plot(x, np.sin(x))

# Имя графика
plt.title('Синус')

# Имя оси OX
plt.xlabel('X')
```

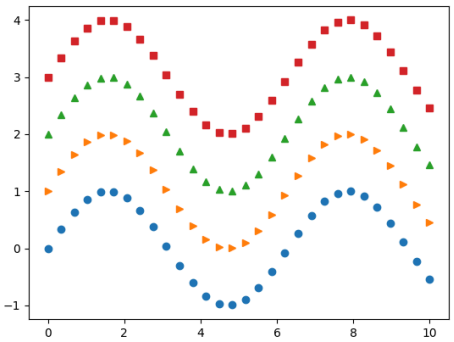
```
# Имя оси OY
plt.ylabel('Y')


# Задание легенды графика
plt.subplot(3, 1, 2)
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')
plt.legend()

# Выравнивание осей координат
plt.subplot(3, 1, 3)
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')

plt.axis('equal')
plt.legend()

# Изменение расстояния между областями с графиками
plt.subplots_adjust(hspace=1)

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



Отметки на кривой графика (маркеры)

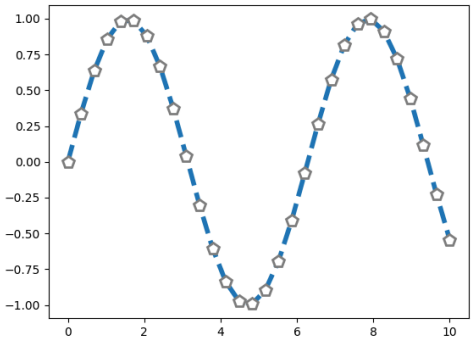
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 30)
```

```
plt.plot(x, np.sin(x), 'o')
plt.plot(x, np.sin(x) + 1, '>')
plt.plot(x, np.sin(x) + 2, '^')
plt.plot(x, np.sin(x) + 3, 's')


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



Параметры маркера

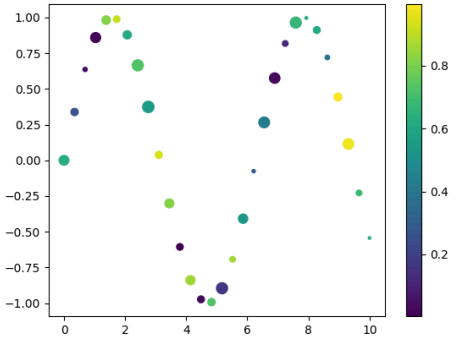
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 30)

# markersize - размер маркера
# linewidth - толщина линии графика
# markerfacecolor - цвет маркера
# markeredgecolor - цвет контура маркера
# markeredgewidth - толщина контура маркера

plt.plot(x, np.sin(x), '--p', markersize=10, linewidth=4,
        markerfacecolor='white', markeredgecolor='gray', markeredgewidth=2)

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



Scatter

Разница scatter и plot ? У scatter для каждой точки можно задать отдельные характеристики

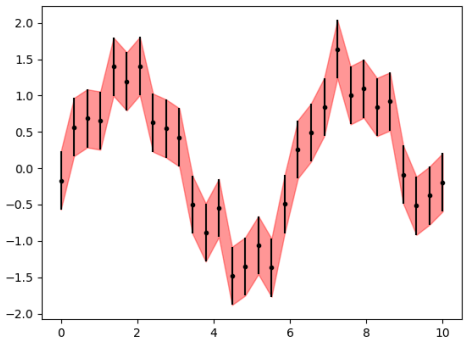
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 30)

rng = np.random.default_rng(0)
colors = rng.random(30)
sizes = 100 * rng.random(30)
plt.scatter(x, np.sin(x), marker='o', c=colors, s=sizes)

plt.colorbar()

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



Если точек > 1000, то plot предпочтительней из-за производительности

Визуализация погрешности (errorbar)

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

x = np.linspace(0, 10, 30)

# Внесем погрешность в y
dy = 0.4
y = np.sin(x) + dy * np.random.randn(30)

plt.errorbar(x, y, yerr=dy, fmt='.k')

# Заполнение промежутка
plt.fill_between(x, y - dy, y + dy, color='red', alpha=0.4)
plt.savefig('img.png')
plt.show()
```

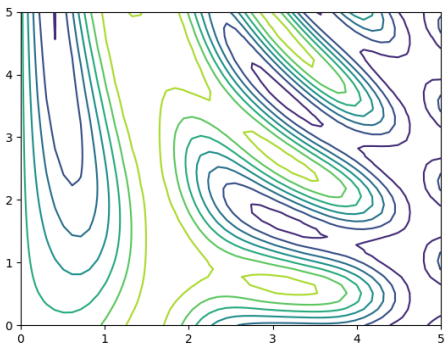


Трехмерный график

Контур

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


def f(x, y):
    return np.sin(x) ** 5 + np.cos(20 + x * y) * np.cos(x)
```

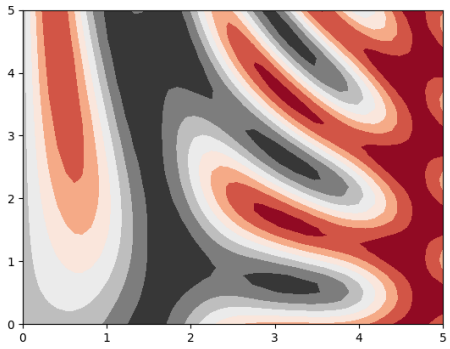
```
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)

Z = f(X, Y)

plt.contour(X, Y, Z, color='green')

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



Заливка

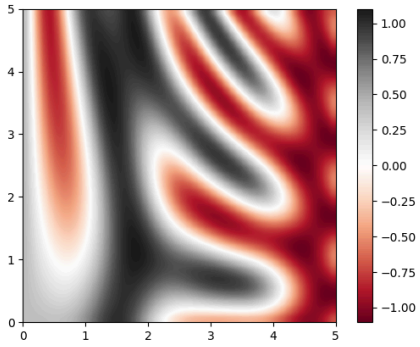
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


def f(x, y):
    return np.sin(x) ** 5 + np.cos(20 + x * y) * np.cos(x)


x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)

Z = f(X, Y)

plt.contourf(X, Y, Z, cmap='RdGy')
```

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



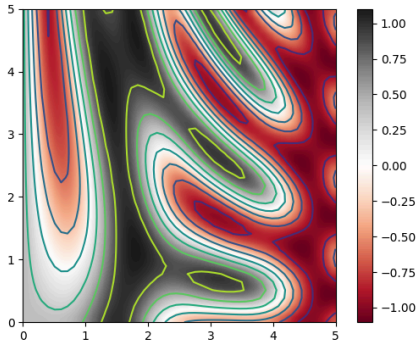
Сглаживание переходов

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


def f(x, y):
    return np.sin(x) ** 5 + np.cos(20 + x * y) * np.cos(x)


x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)

Z = f(X, Y)

plt.imshow(Z, extent=[0, 5, 0, 5], cmap='RdGy',
           interpolation='gaussian', origin='lower')
plt.colorbar()

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



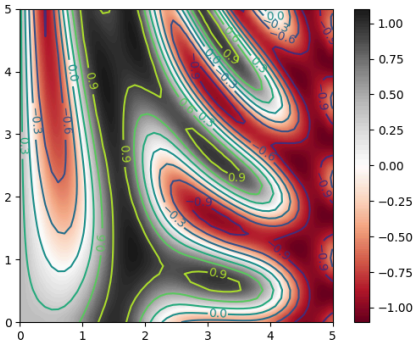
Дополнительно отображение контуров

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


def f(x, y):
    return np.sin(x) ** 5 + np.cos(20 + x * y) * np.cos(x)


x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)

Z = f(X, Y)

plt.contour(X, Y, Z, color='green')
plt.imshow(Z, extent=[0, 5, 0, 5], cmap='RdGy',
           interpolation='gaussian', origin='lower')
plt.colorbar()

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



Отображение значений на контуре

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


def f(x, y):
    return np.sin(x) ** 5 + np.cos(20 + x * y) * np.cos(x)


x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)

Z = f(X, Y)

c = plt.contour(X, Y, Z, color='green')
plt.clabel(c)

plt.imshow(Z, extent=[0, 5, 0, 5], cmap='RdGy',
           interpolation='gaussian', origin='lower')
plt.colorbar()

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


0
1
2
3
4
5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00