

Введение в Matplotlib

Установка

Чтобы установить библиотеку matplotlib достаточно прописать в консоли:

```
pip install matplotlib
```

Простой пример использования

Чтобы начать пользоваться библиотекой в своём коде достаточно импортировать pyplot:

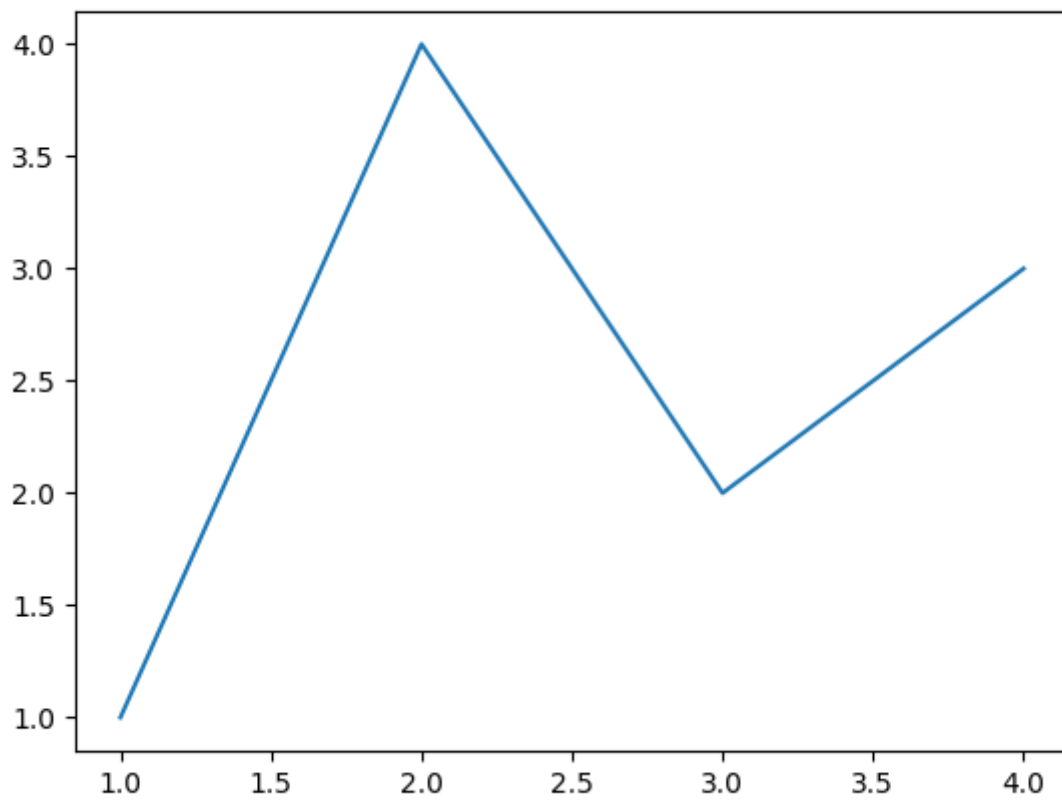
```
import matplotlib.pyplot as plt
```

Если вы используете Jupyter, то также следует добавить строчку кода, которая включает отображения графиков внутри jupyter notebook:

```
%matplotlib inline
```

Matplotlib отображает ваши данные на рисунке `Figures` (например, окнах, виджеты Jupyter и т.д.), каждый из которых может содержать одну или несколько осей `Axes`, область, где точки могут быть указаны в терминах координат x-y (или полярных координатах, x-y-z в 3D и т.д.). Самый простой способ создания `Figure` с `Axes` - это использование `plt.subplots`. Затем мы можем использовать `Axes.plot`, чтобы нарисовать данные по осям:

```
fig, ax = plt.subplots() # Создание области, которая содержит единственный график.  
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Вывод данных на график.  
plt.show() # Рендер области с графиками.
```



Результат выполнения кода выше.

Части графика

Класс `Axes` и его функции-члены являются основной точкой входа для работы с интерфейсом ООП, и в них определено большинство методов построения графиков (например, `ax.plot()`, показанный выше, использует метод построения графиков).

Axis

Эти объекты задают масштаб и пределы и генерируют `ticks` (метки на оси) и `ticklabels` (строки, обозначающие отметки). Расположение `ticks` определяется объектом `Locator`, а строки `ticklabel` форматируются с помощью `Formatter`. Сочетание правильного `Locator` и `Formatter` обеспечивает точный контроль над расположением `tick` и `labels`.

Artist

По сути, все, что видно на рисунке, принадлежит к классу `Artist` (фигура, оси и осевые объекты). Сюда входят `Text` объекты, объекты `Line2D`, объекты `collections`, объекты `Patch` и т.д. Когда `Figure` отрисована, все `Artist` рисуются на холсте. Большинство объектов класса `Artist` привязаны к осям; такие `Artist` объекты не могут быть перемещены или разделяться несколькими `Axes`.

Типы входа для функций отрисовки

В качестве входа функции отрисовки (`plot`) ожидают увидеть [np.array](#) или объект, который может быть конвертирован `np.array` с помощью `np.asarray`.

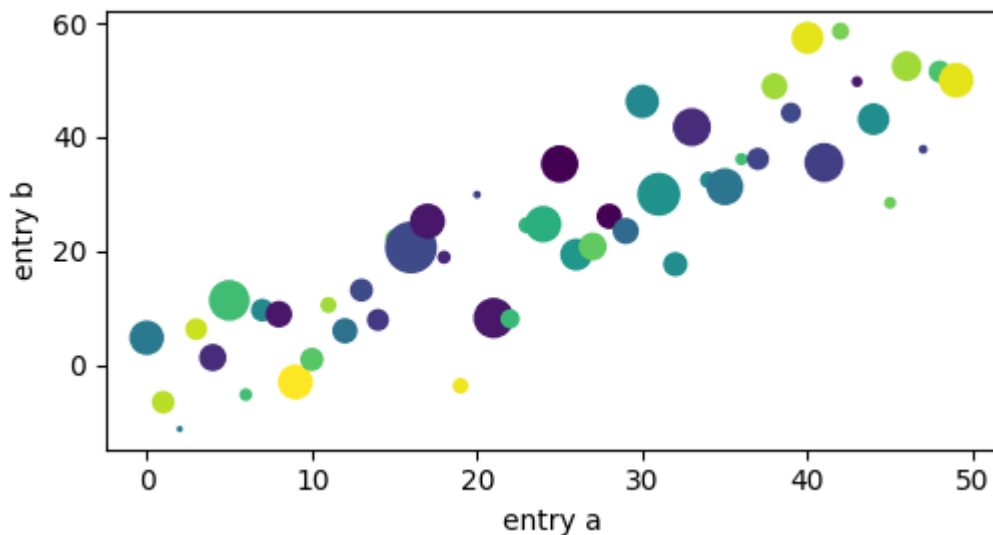
```
b = np.matrix([[1, 2], [3, 4]])
b_asarray = np.asarray(b)
```

Pandas

`pd.DataFrame` или `dict-like` объекты можно передавать в качестве аргумента `data` и генерировать графики указывая `str` соответствующие оси `x`, `y`.

```
np.random.seed(19680801) # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
```

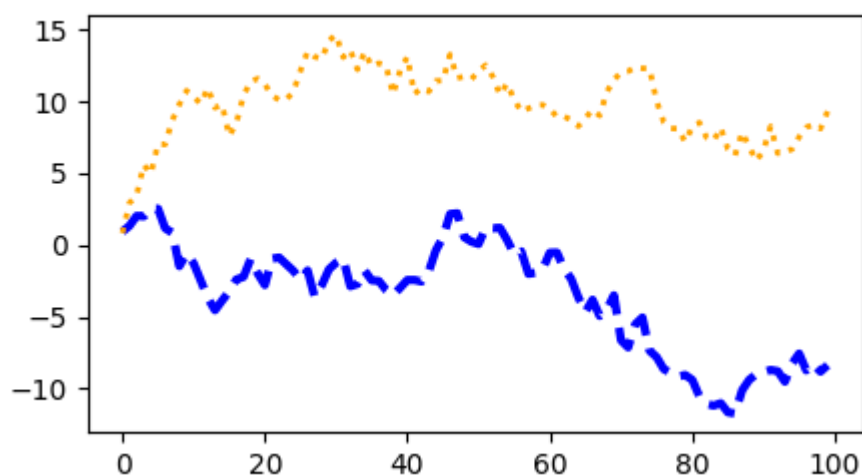


Результат выполнения кода выше

Styling Artists

Большинство методов построения графиков имеют параметры оформления для `Artist`, доступные либо при вызове метода построения графиков, либо из "setter" на художнике. На графике ниже мы вручную задаем цвет, ширину линии и стиль линий `Artist`, созданных с помощью `plot`, и мы задаем стиль линии второй строки постфактум с помощью `set_linestyle`.

```
fig, ax = plt.subplots(figsize=(5, 2.7))
x = np.arange(len(data1))
ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')
l = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)
l.set_linestyle(':')
```



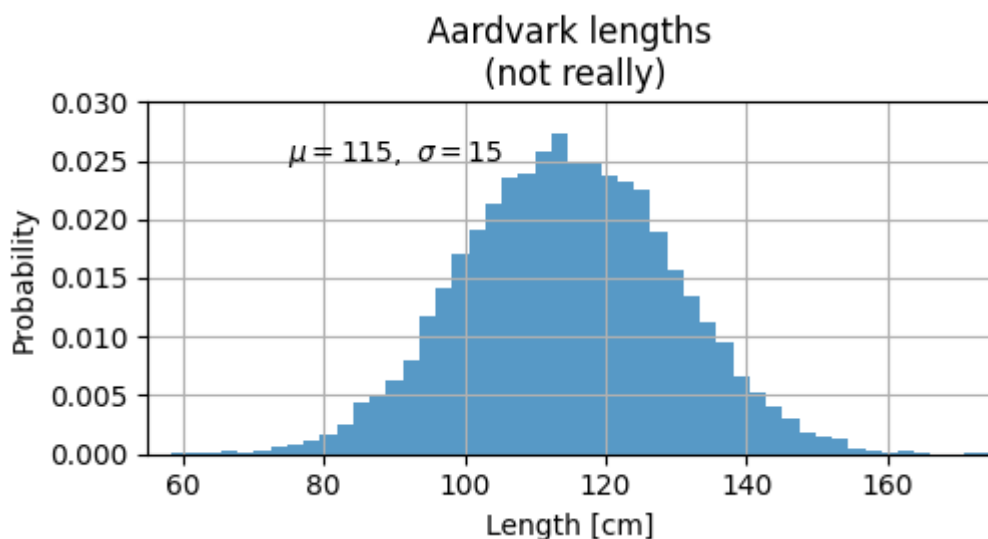
Добавление меток на график

`set_xlabel`, `set_ylabel`, и `set_title` используются, чтобы добавить текст в соответствующие места на графике. Текст также может быть напрямую добавлен на


график с использованием метода `text`:

```
mu, sigma = 115, 15
x = mu + sigma * np.random.randn(10000)
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
# the histogram of the data
n, bins, patches = ax.hist(x, 50, density=True, facecolor='C0',
alpha=0.75)

ax.set_xlabel('Length [cm]')
ax.set_ylabel('Probability')
ax.set_title('Aardvark lengths\n (not really)')
# Using Latex in text
ax.text(75, .025, r'$\mu=115,\ \sigma=15$')
ax.axis([55, 175, 0, 0.03])
ax.grid(True)
```



Результат выполнения кода выше

 Стоит заметить, что `matplotlib` поддерживает TeX выражение в любом `text` выражение.

annotation

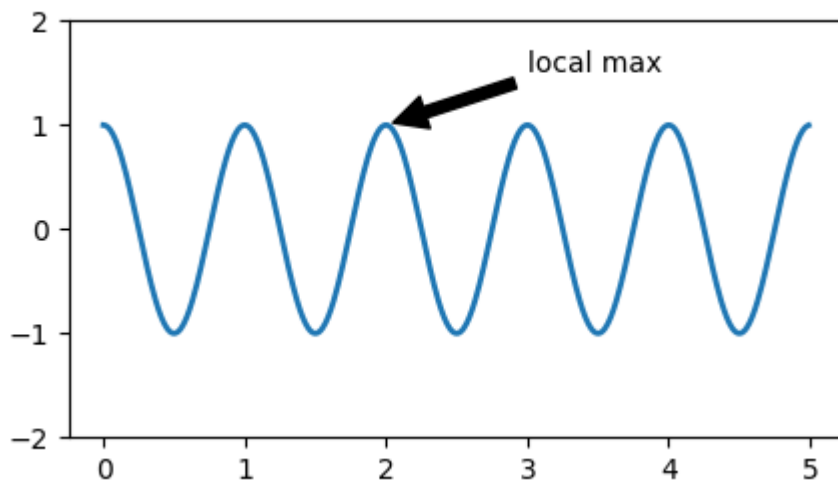
Мы также можем комментировать точки на графике, часто соединяя стрелку, указывающую на `xy`, с фрагментом текста в `xytext`:

```
fig, ax = plt.subplots(figsize=(5, 2.7))

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2 * np.pi * t)
line, = ax.plot(t, s, lw=2)

ax.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
```

```
arrowprops=dict(facecolor='black', shrink=0.05))  
  
ax.set_ylim(-2, 2)
```



Результат выполнения кода выше

✎ Стоит обратить внимания, что для добавления аннотации или текста на график указываются координаты этой точки в системе координат данных.

Заключение

Мы познакомились с основными методами и классами `matplotlib`, знание которых необходимо для дальнейшего изучения курса. В случае, если представленной информации недостаточно, следует воспользоваться [официальной документацией matplotlib](#).