Основы Matplotlib

Технологии и языки программирования

Юдинцев В. В.

Кафедра теоретической механики

13 апреля 2019 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Содержание

1. Простые графики
2. Стили
3. Экспорт
4. Надписи на графиках (аннотации)
5. Несколько графиков на рисунке
6. Графики по данным из файла

Библиотека matplotlib

Библиотека графических функций (пакет) для построения графиков в среде python.

Подключение библиотеки

Стандартный способ подключения модуля matplotlib.pyplot

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

matplotlib часто используется вместе с модулем numpy

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

Простые графики

Построение графика функции

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

Генерация списка 10 значений аргумента в интервале от 0 до 5 (включительно):

```
1 x = np.linspace(0.0,8.0,20)
```

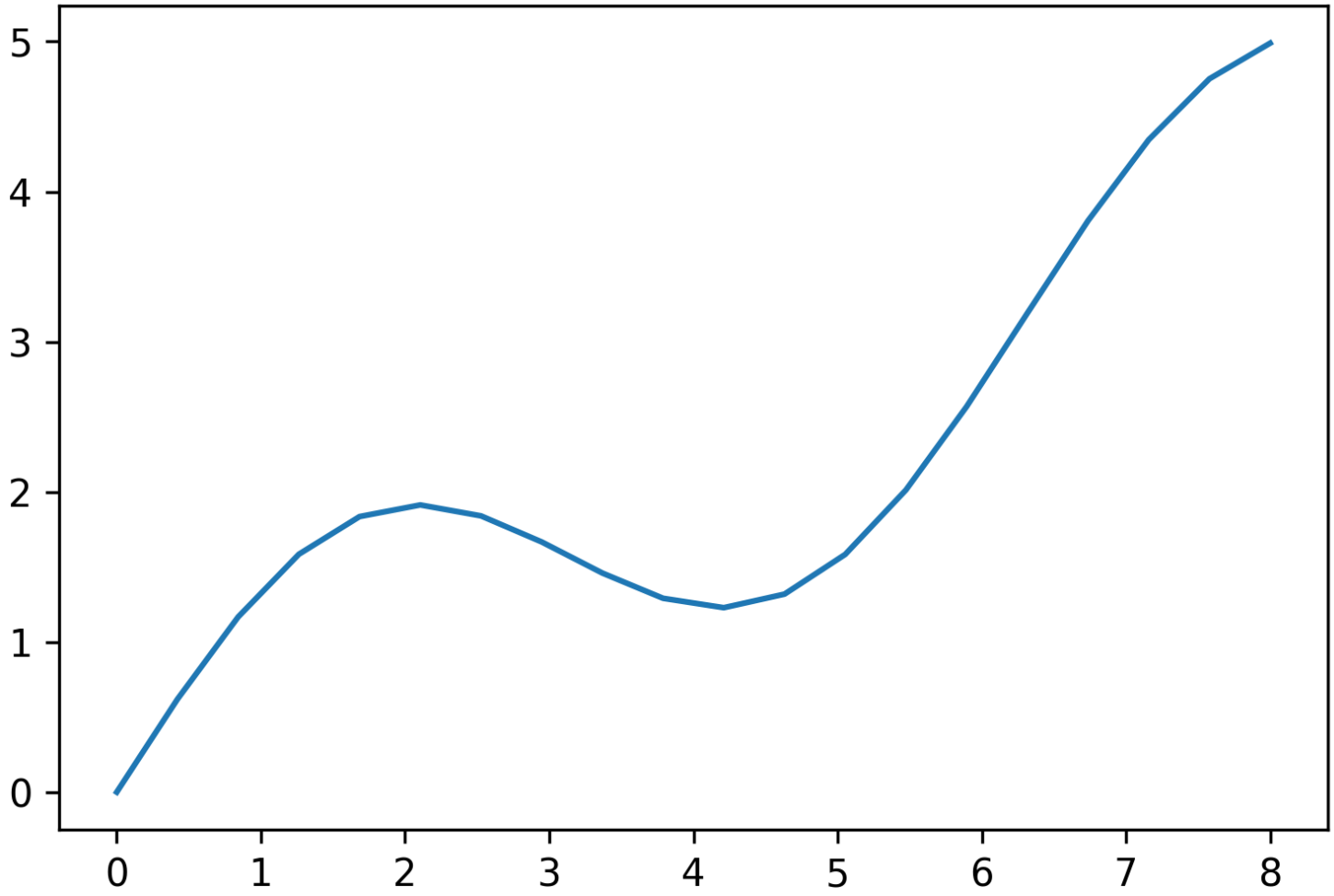
Построение списка значений функции:

```
1 y = np.sin(x) + x/2
```

Построение графика:

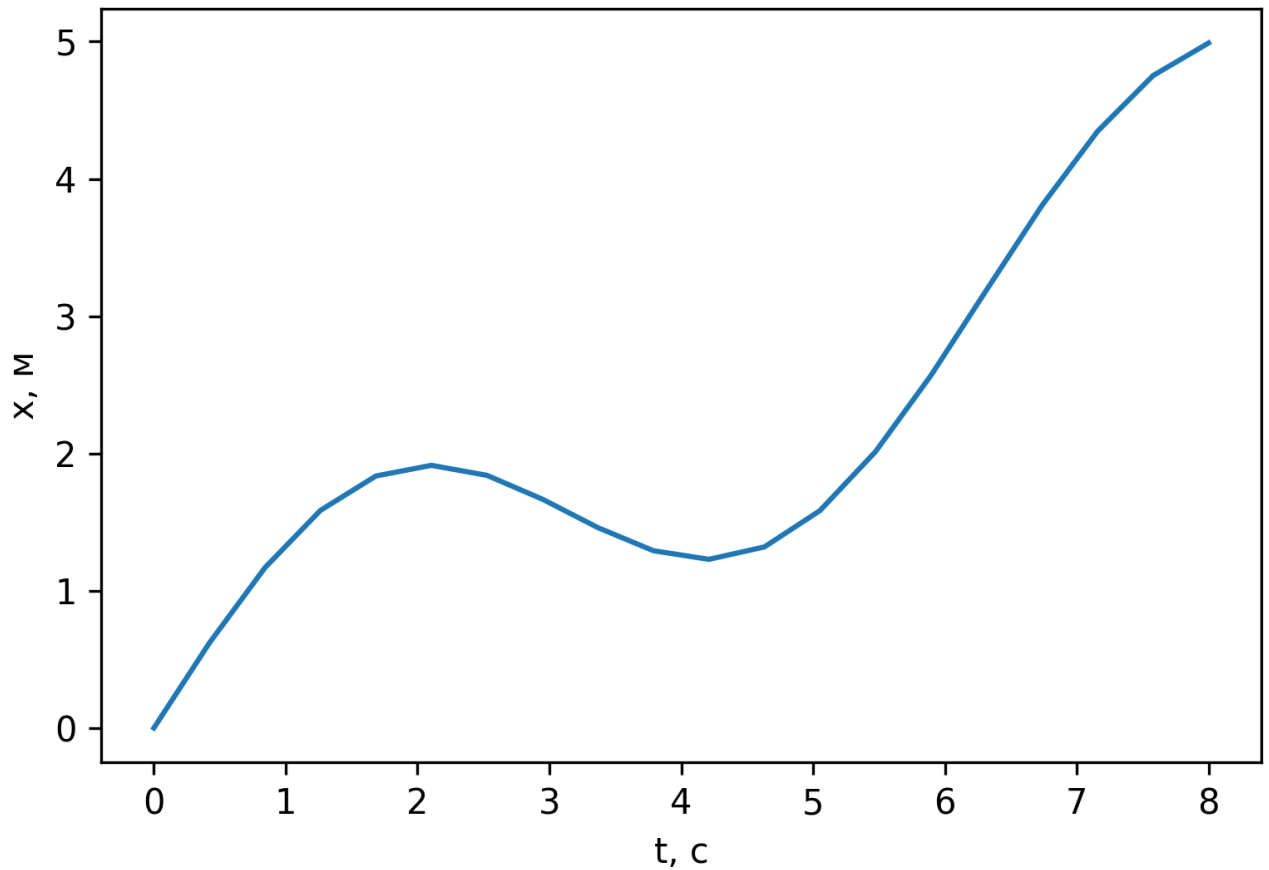
```
1 plt.plot(x, y)
```

Построение графика функции



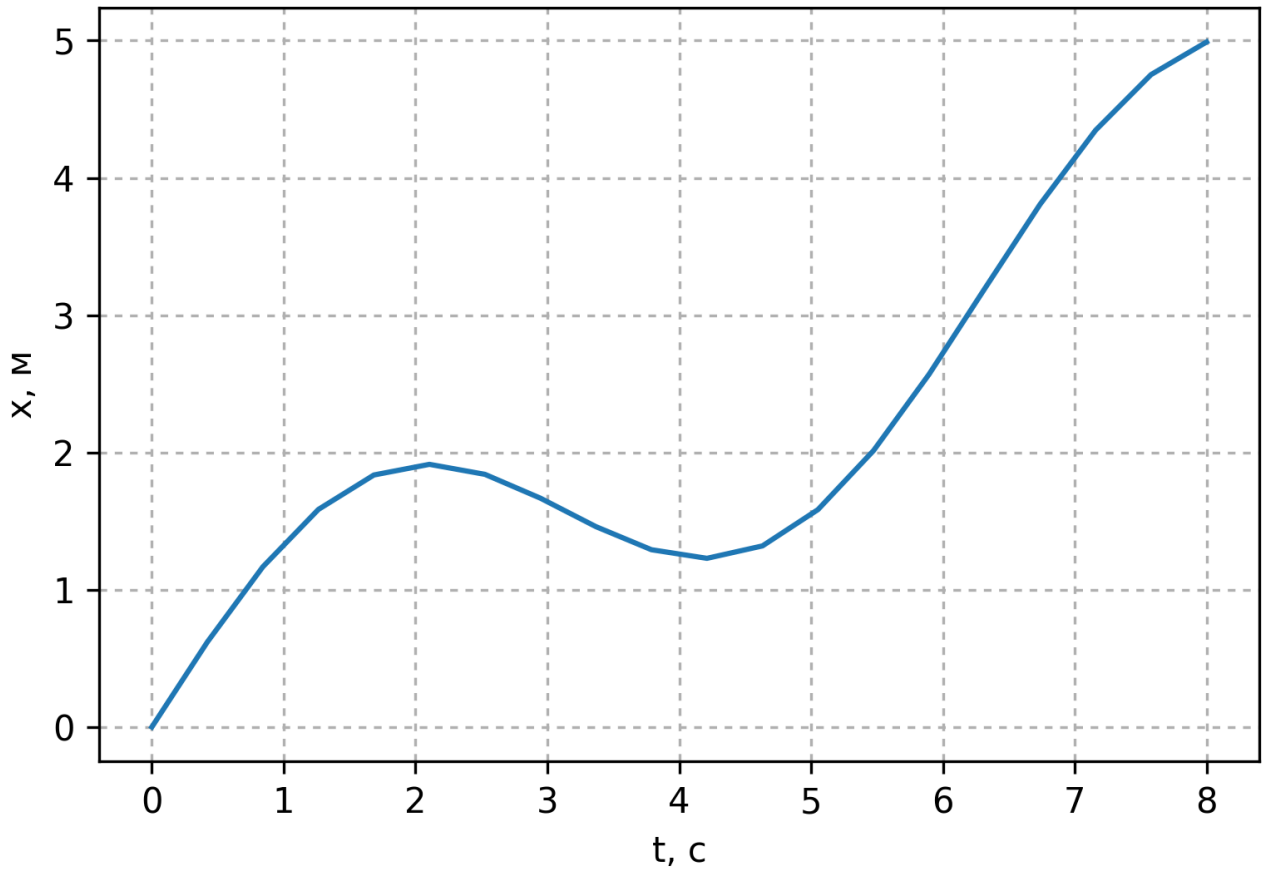
Подписи осей

```
1 plt.xlabel('t, с');
2 plt.ylabel('x, м');
```



Линии сетки

```
1 plt.grid(True, linestyle='dotted')
```



Шаблон

```
x = np.linspace(0.0,8.0,20)
y = np.sin(x) + x/2

plt.plot(x, y)

plt.xlabel('t, c');
plt.ylabel('x, м');
plt.grid(True,linestyle='dotted')
```

Функция scatter

```
1 x = np.linspace(0,10,20)
2 y = np.sin(x)
3 plt.scatter(x, y, c = 'r', s = np.abs(y)*20)
```

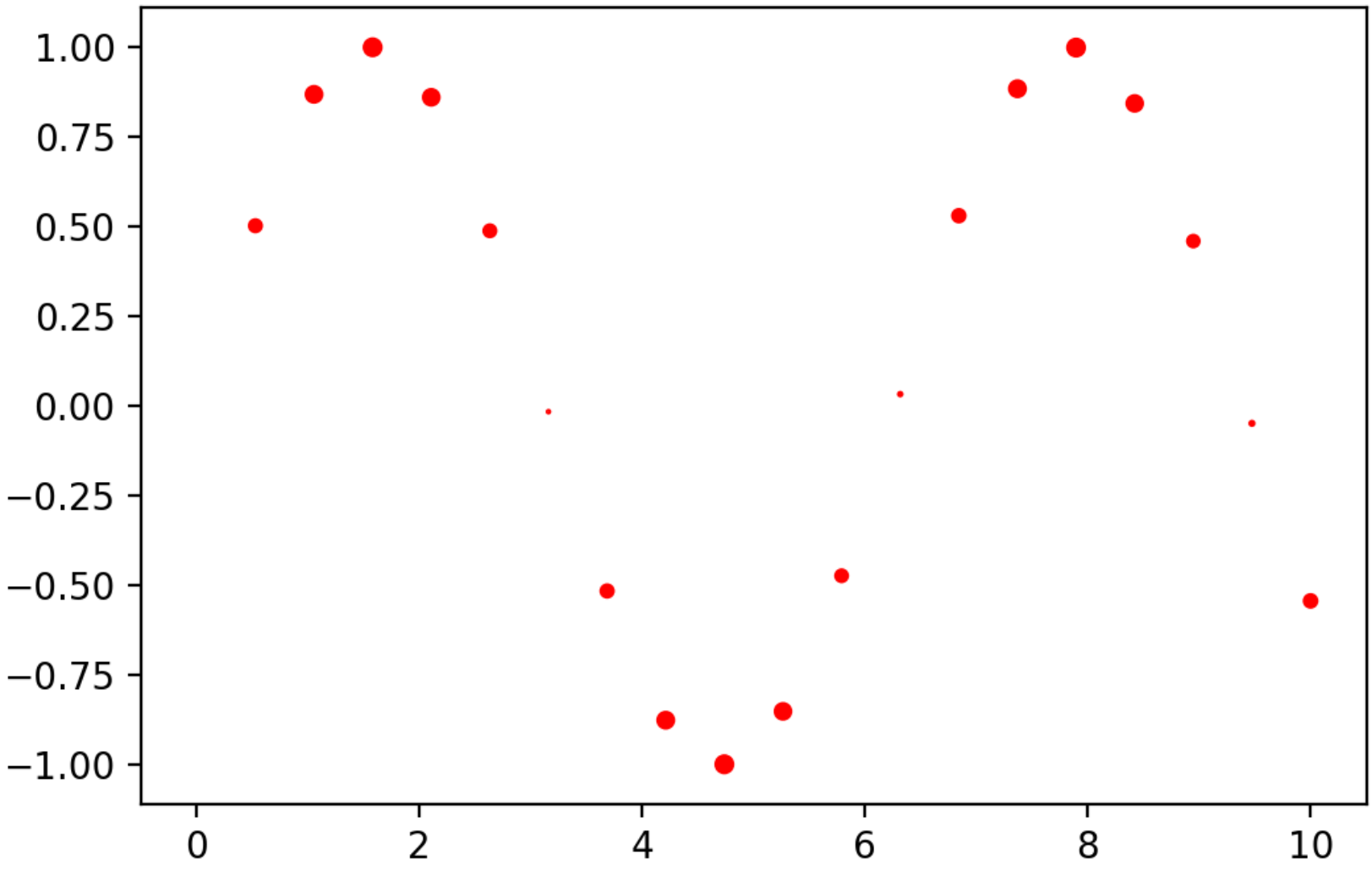
Позиционные аргументы:

- 1 - координаты x точек
- 2 - координаты y точек

Именованные аргументы:

- c - цвет маркера
- s - размер маркера

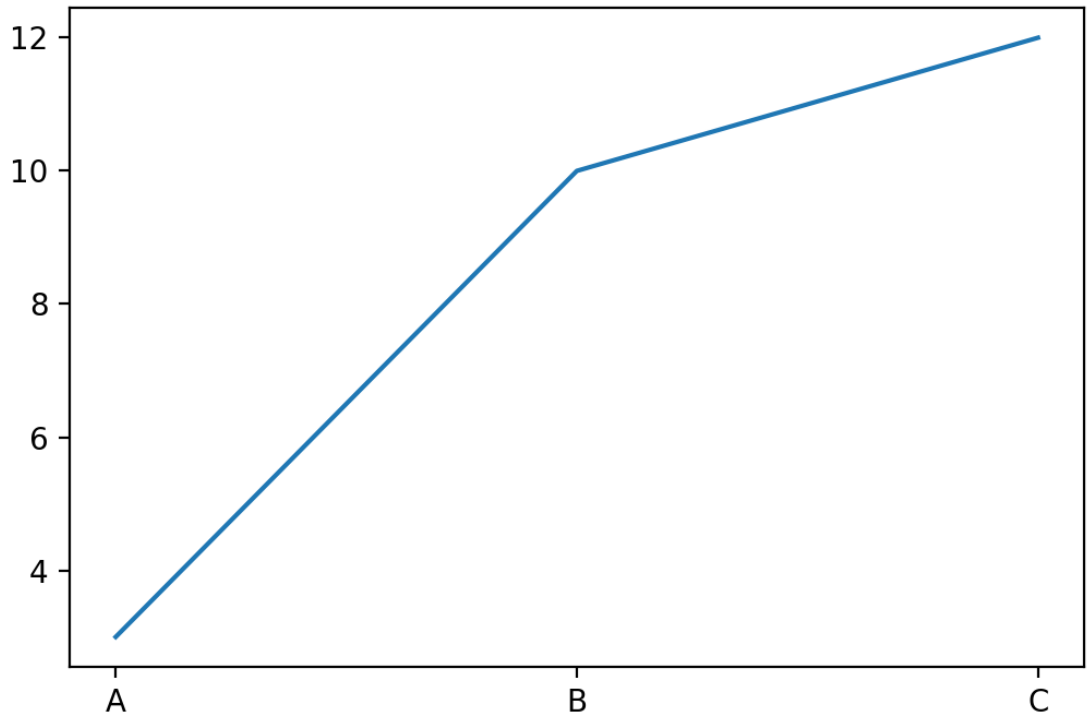
Функция scatter



Категории, как подписи оси x

Вместо координат точек по оси x могут быть передан список меток (категорий)

```
1 names = ['A', 'B', 'C']
2 values = [3, 10, 12]
3 plt.plot(names, values)
```



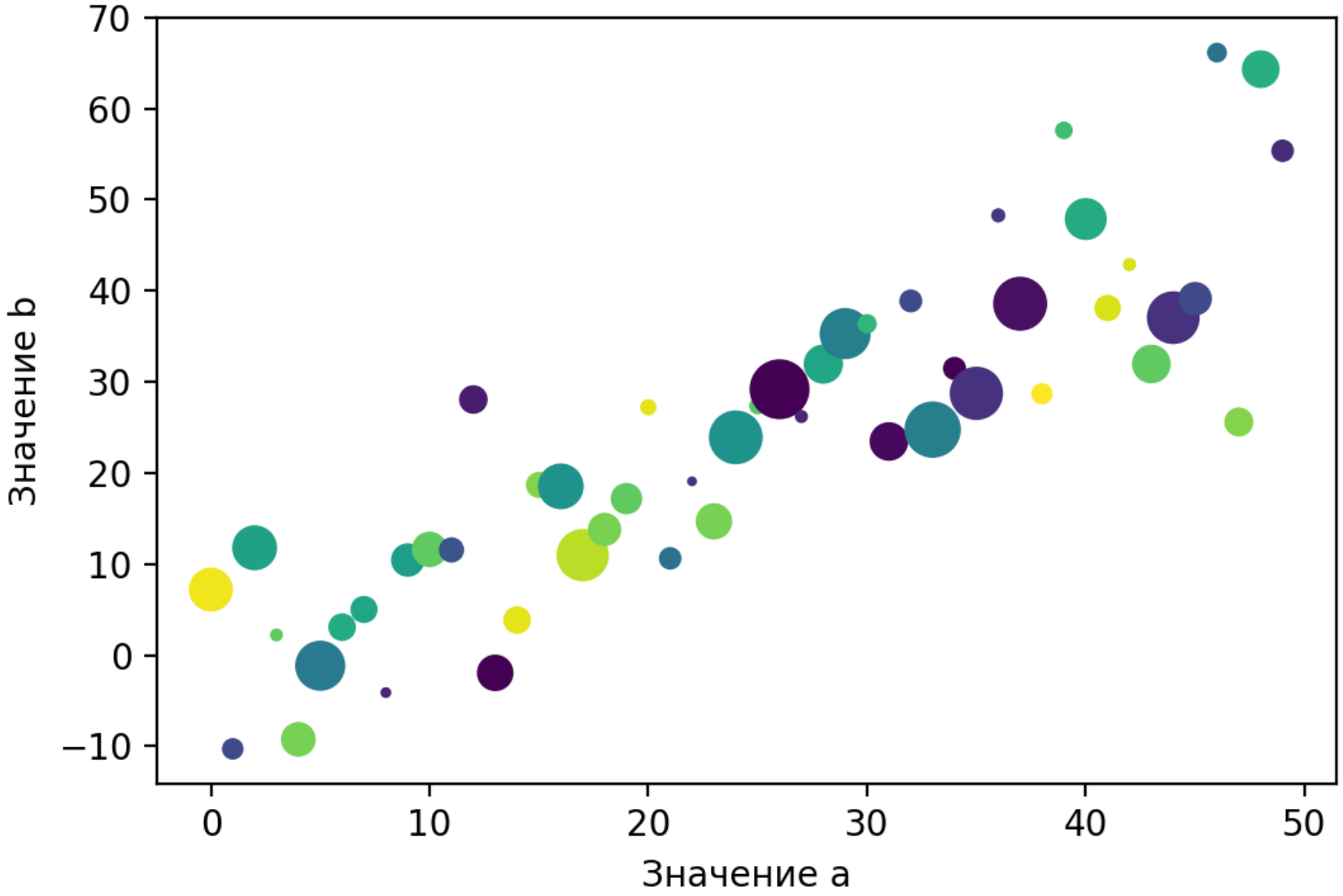
Использование словаря

Функции построения графика (scatter) можно передать ключи словаря, вместо явных значений массивов x и y:

```
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
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('Значение a')
plt.ylabel('Значение b')
```

- 'a' - значения x
- 'b' - значения y
- 'c' - цвет маркера
- 's' - размер маркера
- 'data' - ссылка на словарь с ключами 'a', 'b', 'c' и 'd'

Использование словаря

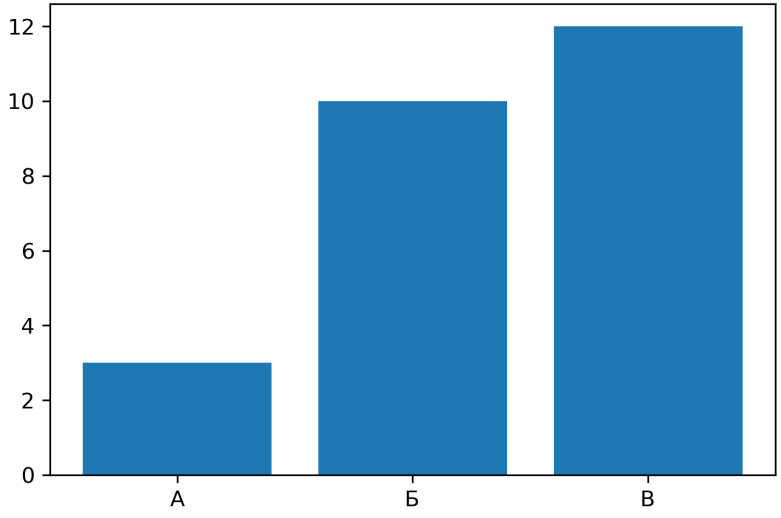


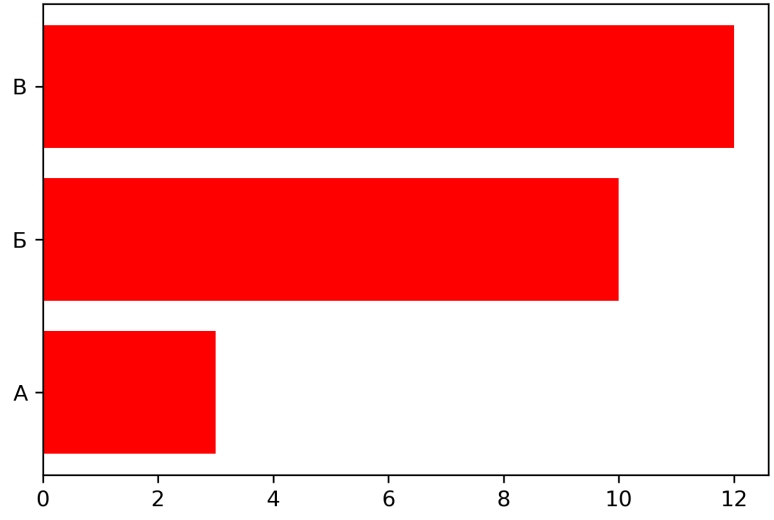
Столбчатые диаграммы

Функции bar и barh

```
names = ['А', 'Б', 'В']
values = [3, 10, 12]
plt.bar(names, values)
;
```

```
names = ['А', 'Б', 'В']
values = [3, 10, 12]
plt.barh(names, values,
         facecolor = 'red'
    )
```





Гистограммы

```
mu, sigma = 0, 1
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 16, density=1,
                            facecolor='g', alpha=0.75)
plt.xlabel('Величина')
plt.ylabel('Вероятность')
plt.text(-3, 0.35, r'$m=0,\ \sigma=1$')
```

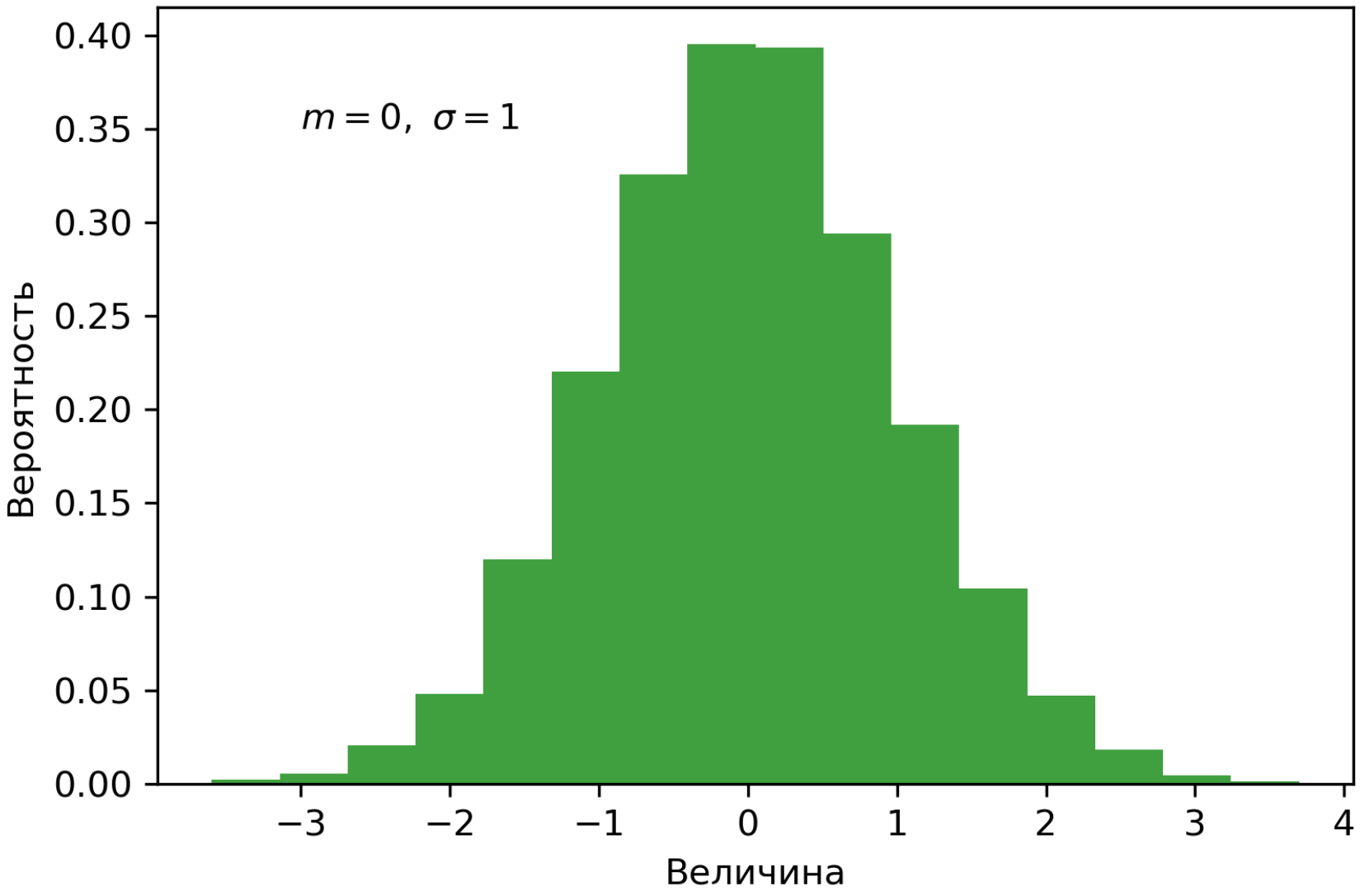
Позиционные аргументы:

- x - данные
- 16 - количество диапазонов (столбцов)

Именованные аргументы:

- density - 1 - плотность, 0 - количество
- facecolor - цвет заливки столбцов
- alpha - прозрачность

Гистограммы



Круговая диаграмма

```
labels = 'США', 'Россия', 'Китай', 'Евросоюз', 'Другие'
sizes = [29, 21, 18, 9, 13]
explode = (0, 0.1, 0, 0, 0)


plt.pie(sizes, explode=explode, labels=labels,
        autopct='%1.1f%%', shadow=True,
        startangle=90)
plt.axis('equal')
plt.title('Космические запуски в 2017 году')
plt.savefig('plot_pie.png',dpi=300)
```

Круговая диаграмма

Настройки по умолчанию

В начале файла программы после подлючения библиотеки matplotlib можно указать параметы по умолчанию для всех графиков.

Ширина и высота графика в дюймах:

```
plt.rcParams["figure.figsize"] = (17/2.5, 10/2.5)
```

Размер шрифта (оси, подписи осей, легенда):

```
plt.rcParams["font.size"] = 14
```

Тип шрифта:

```
plt.rcParams["font.family"] = 'Arial'
```

Стили

Несколько графиков, стили

Графики трёх функций $y = t$, $y = t^2$, $y = t^3$

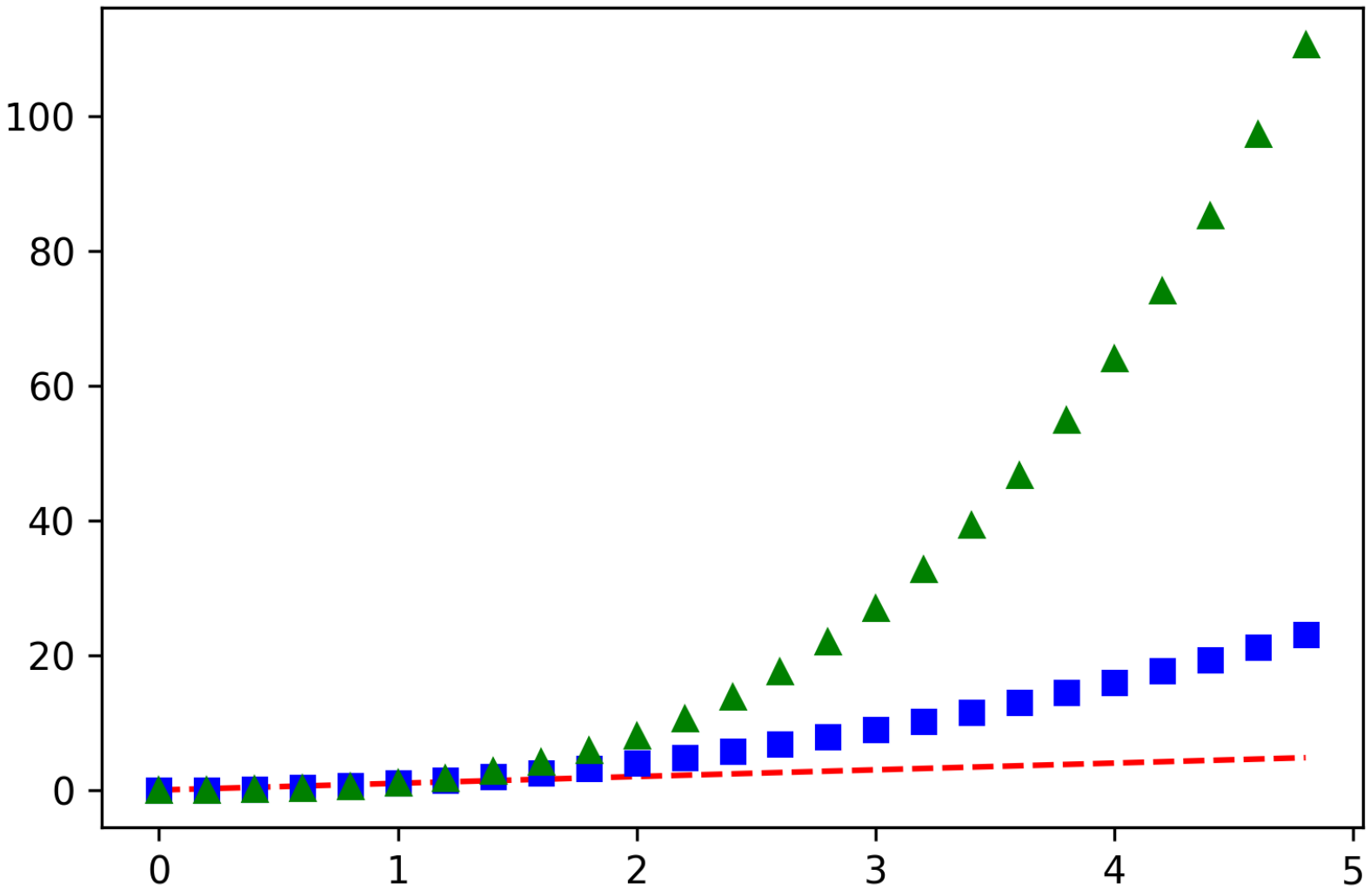
```
t = np.arange(0., 5., 0.2)

plt.plot(t, t, 'r--',
         t, t**2, 'bs',
         t, t**3, 'g^')
```

- $y = t$: красным цветом (red), пунктирной линией ('–'),
- $y = t^2$: синим цветом (blue), квадратными точками ('s'),
- $y = t^3$: зелёным цветом (green) треугольными точками ('').

аргумент, функция, стиль

Стили точек

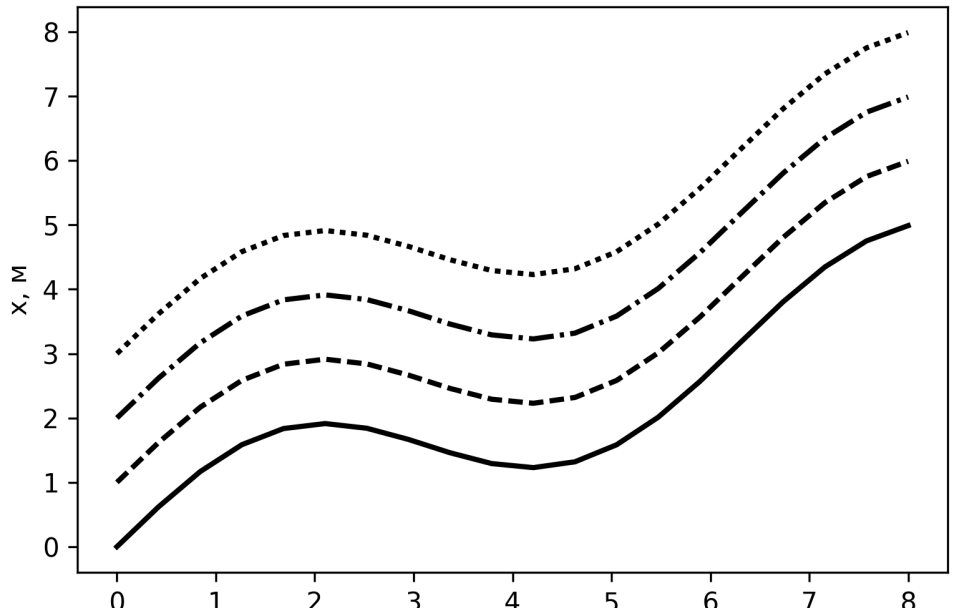


Стили линий

```
x = np.linspace(0.0,8.0,20)
y = np.sin(x) + x/2
plt.plot(x, y, lw = 2)
plt.xlabel('t, c'), plt.ylabel('x, м')
plt.grid(True,ls=':')
plt.plot(x, y+1, lw = 2)
plt.plot(x, y+2, lw = 2)
```

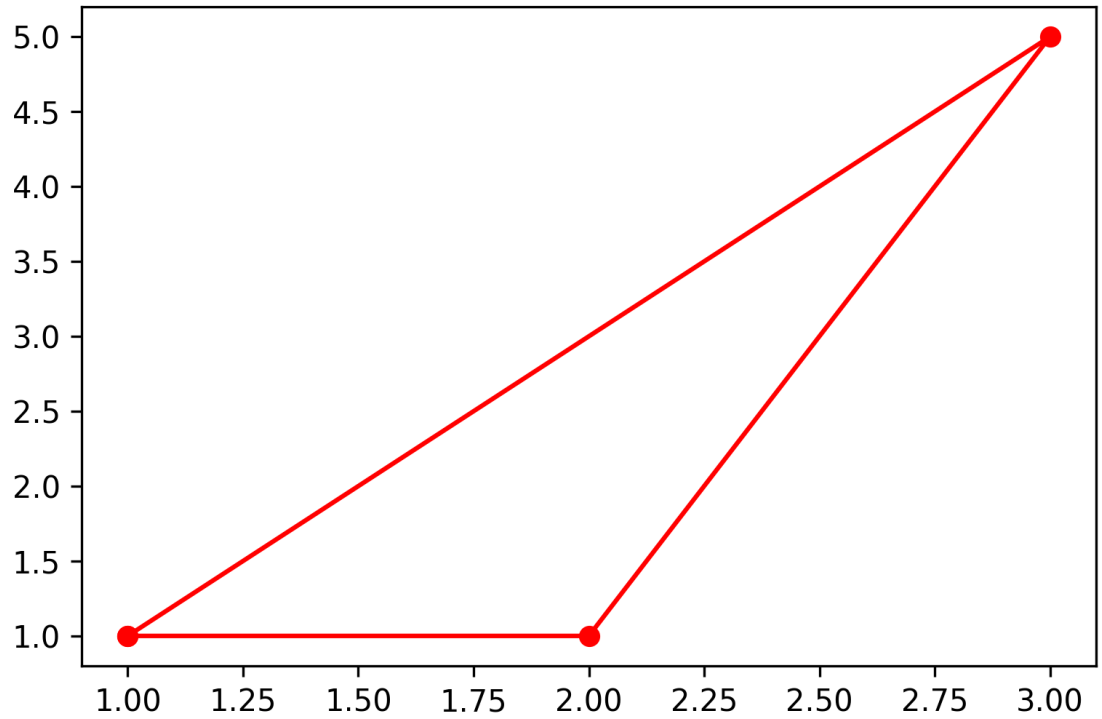
Стили линий

```
x = np.linspace(0.0,8.0,20)
y = np.sin(x) + x/2
plt.plot(x, y, 'k-', lw = 2)
plt.xlabel('t, c'), plt.ylabel('x, м')
plt.plot(x, y+1,'k--', lw = 2)
plt.plot(x, y+2,'k-.', lw = 2)
plt.plot(x, y+3,'k:', lw = 2)
```



Замкнутая линия

```
1 plt.plot([1, 2, 3, 1], [1, 1, 5, 1],'ro-')
```



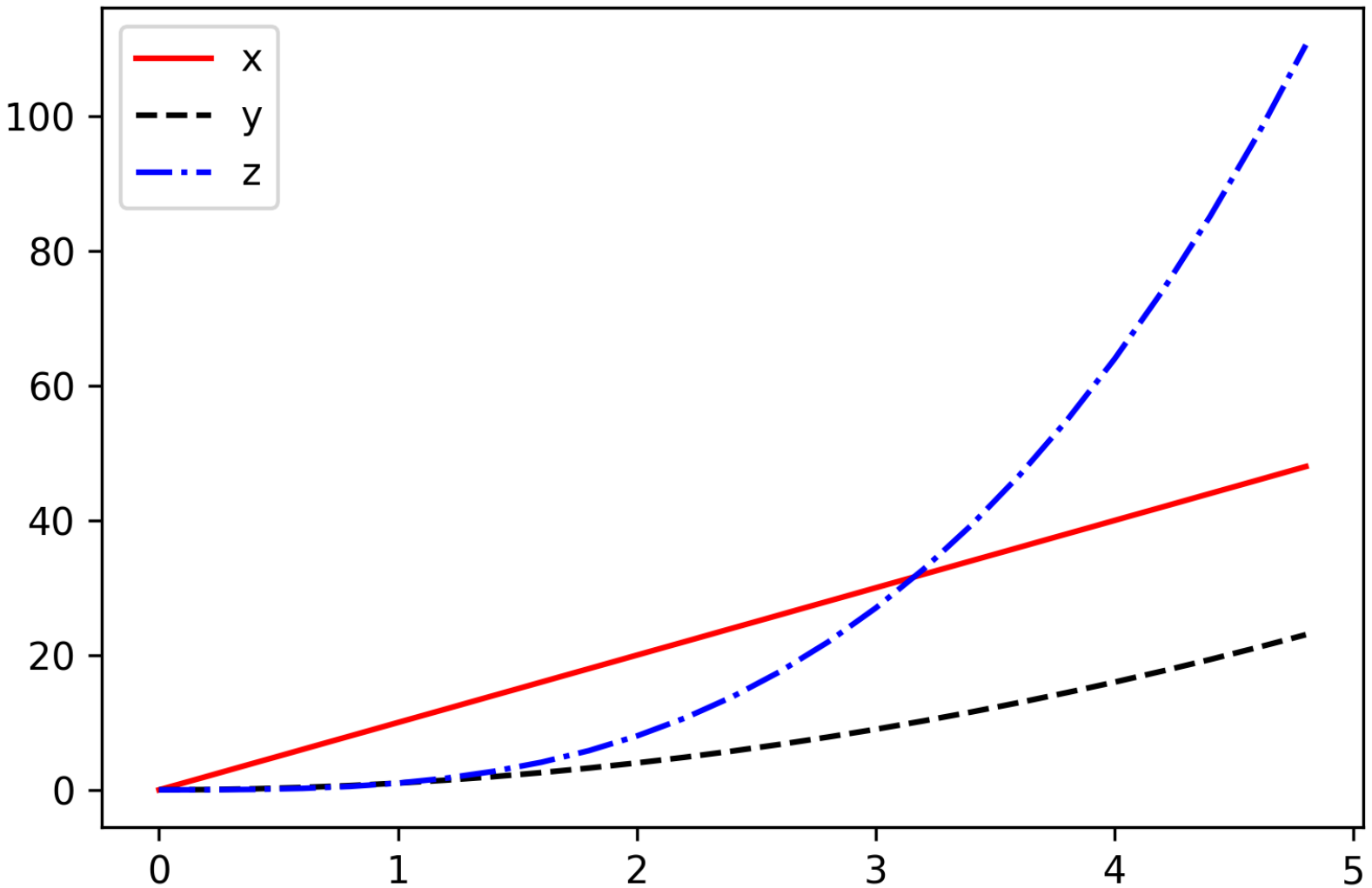
Несколько графиков, легенда

```
t = np.arange(0., 5., 0.2)

plt.plot(t, t, 'r--',
         t, t**2, 'bs',
         t, t**3, 'g^')

plt.legend(['x','y','z'])
```

Легенда



Область построения

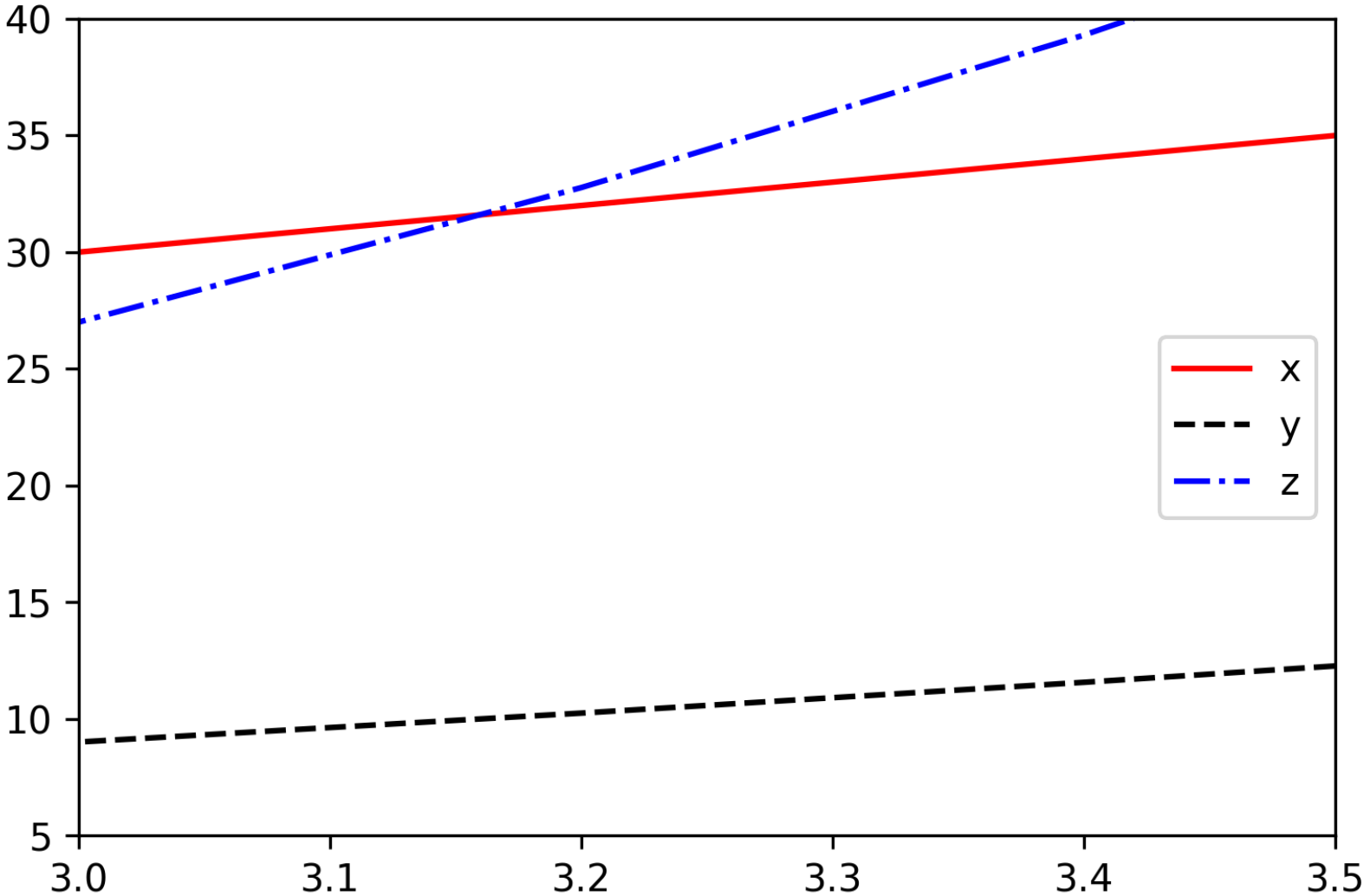
```
t = np.arange(0., 5., 0.2)

plt.plot(t, t, 'r--',
         t, t**2, 'bs',
         t, t**3, 'g^')

plt.xlim([3,3.5])
plt.ylim([5,40])

plt.legend(['x','y','z'])
```

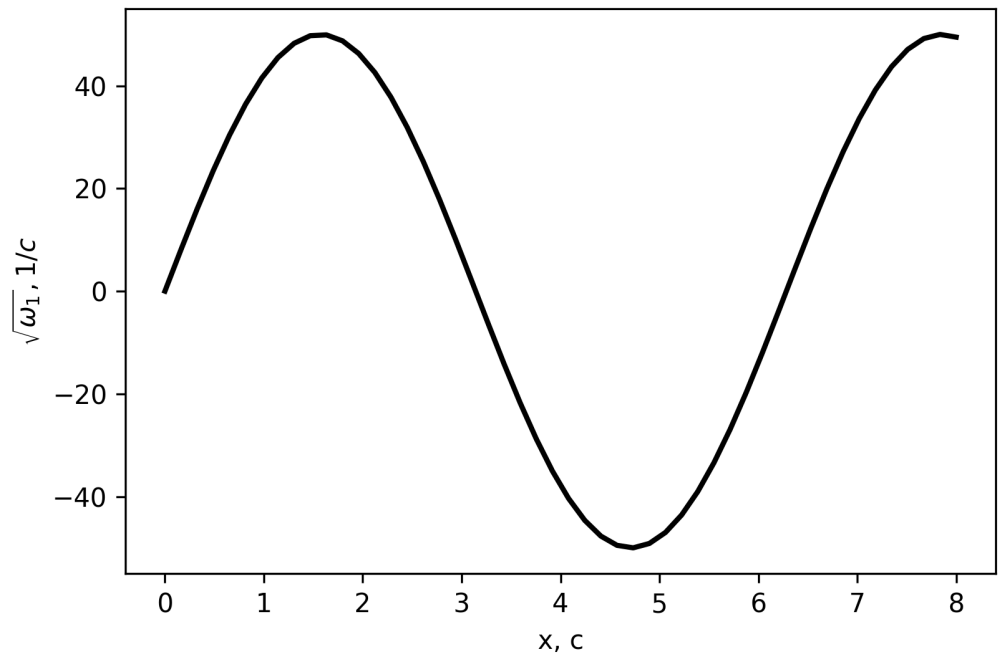
Область построения



Математические символы

В подписях могут использоваться математические символы и стили (надстрочные и подстрочные элементы).

```
1 t = np.linspace(0.0,8.0,50), v = 50*np.sin(t)
2 plt.plot(t, v, 'k—' , lw = 2)
3 plt.xlabel( 'x, c')
4 plt.ylabel(r'$\sqrt{\omega_1}, 1/c$')
```



Экспорт

Экпорт

- Для экспорта изображения в файл используется функция pyplot.savefig.
- Векторные графические форматы:

```
plt.savefig('plot_math.pdf')
plt.savefig('plot_math.svg')
```

- Растровые графические форматы:

```
plt.savefig('plot_math.png',dpi=300)
plt.savefig('plot_math.jpg',dpi=300)
plt.savefig('plot_math.tif',dpi=300)
```

- При экспорте в растровые форматы для рекомендуется указывать разрешение изображения dpi – количество точек на дюйм. Для дальнейшей печати изображения это значение должно быть не менее 300.

Экпорт

```
t = np.linspace(0.0,8.0,50)
v = 50*np.sin(t)

plt.plot(t, v, 'k-' , lw = 2)

plt.xlabel( 'x, c')
plt.ylabel(r'$\sqrt{\omega_1}, 1/c$')
plt.grid(True,ls=':')

plt.savefig('plot_math.png',dpi=300)
```

Надписи на графиках (аннотации)

Надписи на графиках

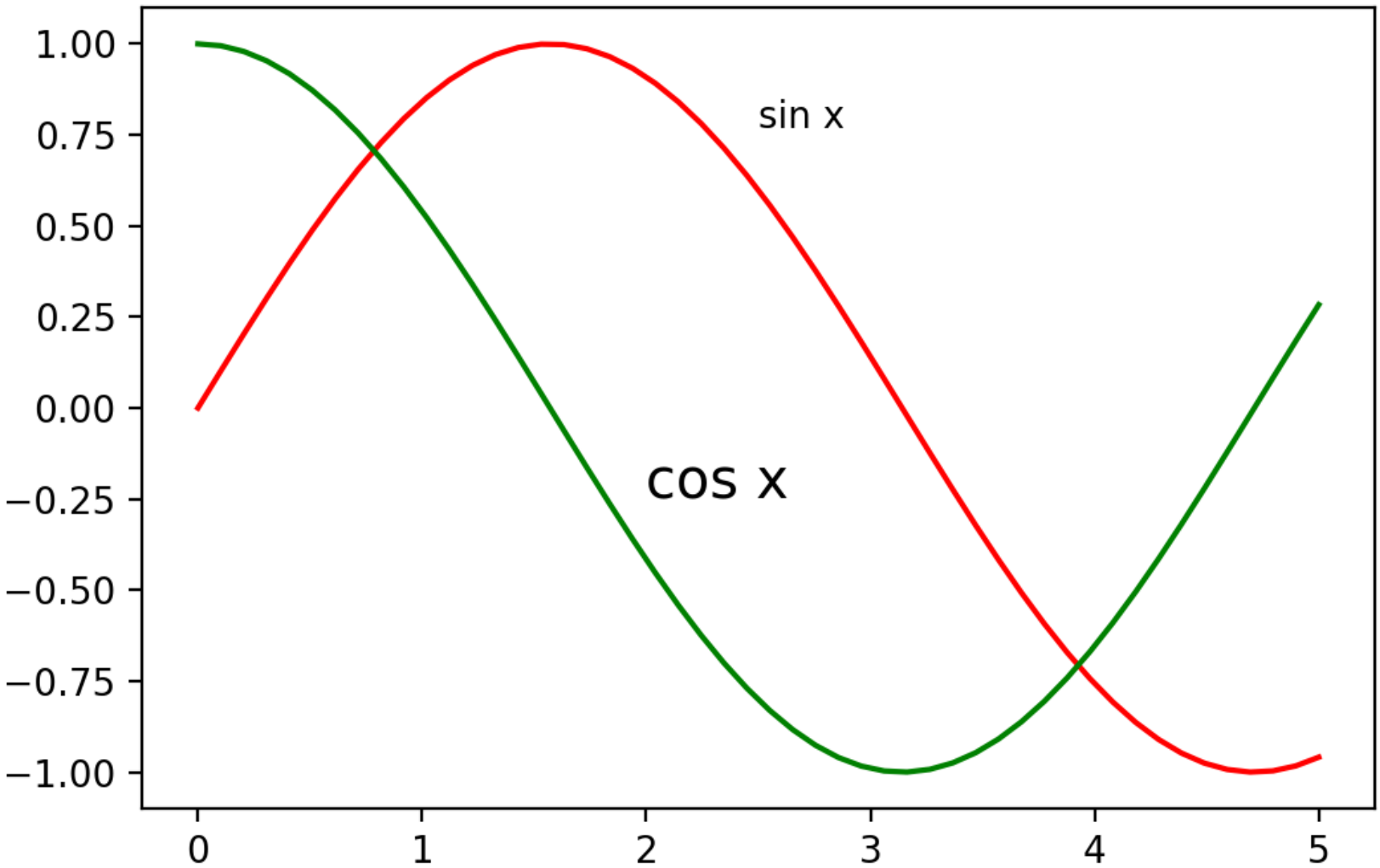
```
x = np.linspace(0,5,50)

plt.plot(x,np.sin(x),'r-')
plt.plot(x,np.cos(x),'g-')

plt.annotate('sin x', xy=(0.5, 0.85),
             xycoords = 'axes fraction',
             ha = 'left')

plt.annotate('cos x', xy=(2, -0.25),
             ha = 'left', size='15')
```

Надписи на графике



Аннотации

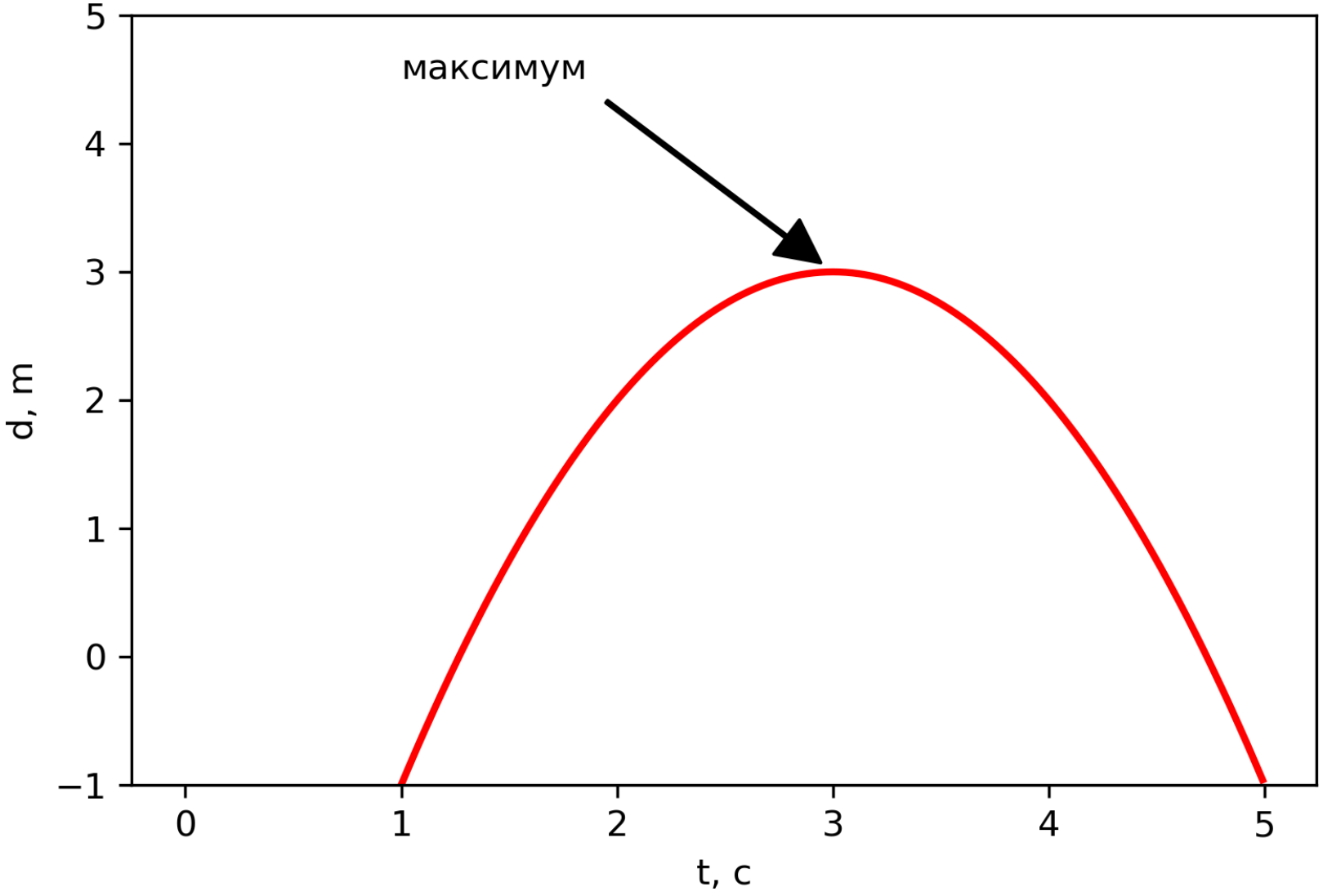
```
t = np.arange(0.0, 5.0, 0.01)

y = -(t-3)**2+3

plt.plot(t, y, lw=2, c='r')

plt.annotate('максимум', xy=(3, 3), xytext=(1, 4.5),
             arrowprops = dict(facecolor='black',
                               shrink=0.05,width=0.8))
plt.ylim(-1, 5)
plt.xlabel('t, c')
plt.ylabel('d, m')
```

Аннотации



Несколько графиков на рисунке

Функция subplot

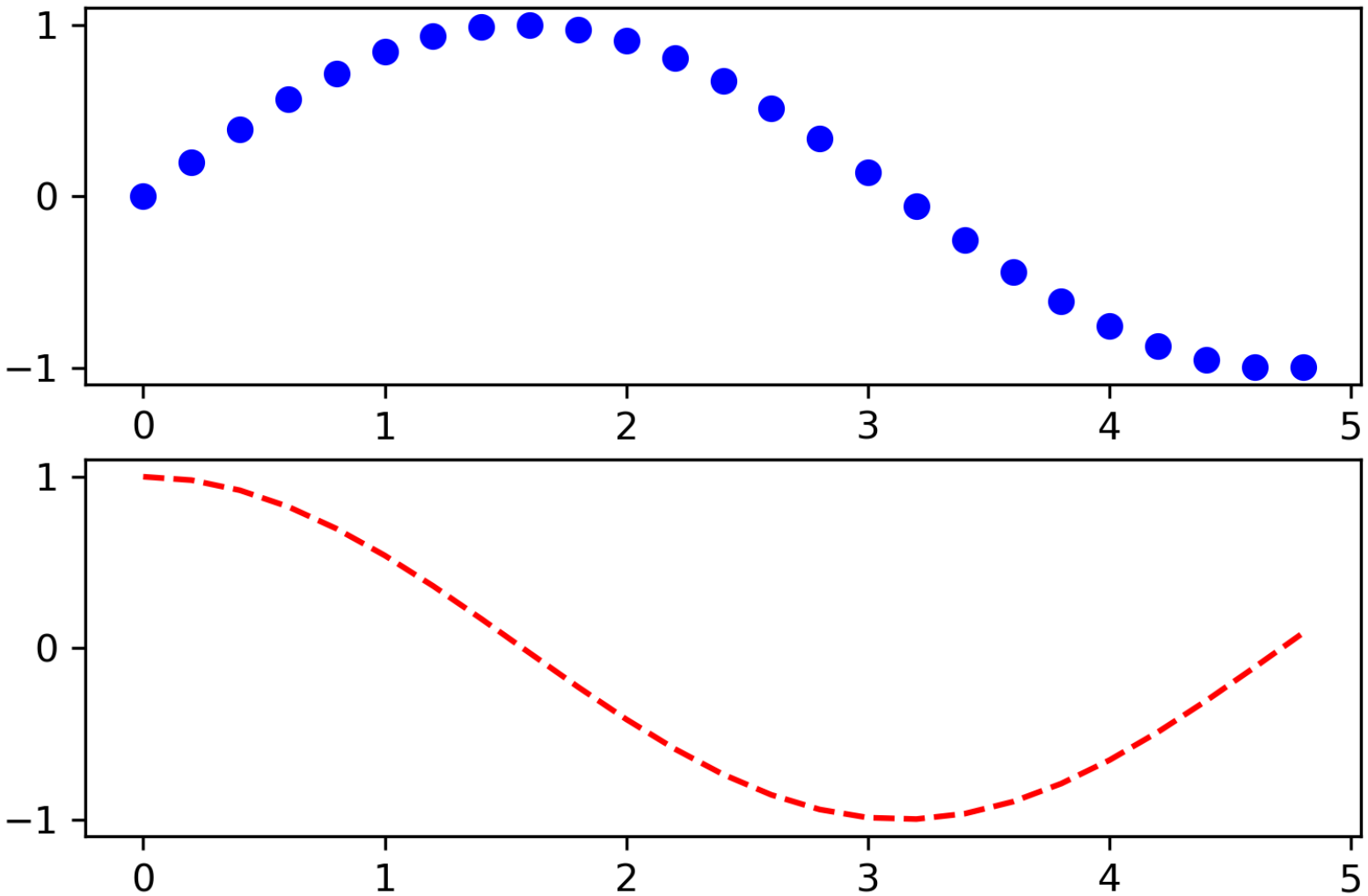
Для изображения на одном рисунке нескольких графиков используется функция subplot

```
1 t = np.arange(0.0, 5.0, 0.2)
2
3 plt.subplot(211)
4 plt.plot(t, np.sin(t), 'bo')
5
6 plt.subplot(212)
7 plt.plot(t, np.cos(t), 'r--')
```

subplot(211):

- 2 - количество строк
- 1 - количество столбцов
- 1 - позиция (построчно)

2x1

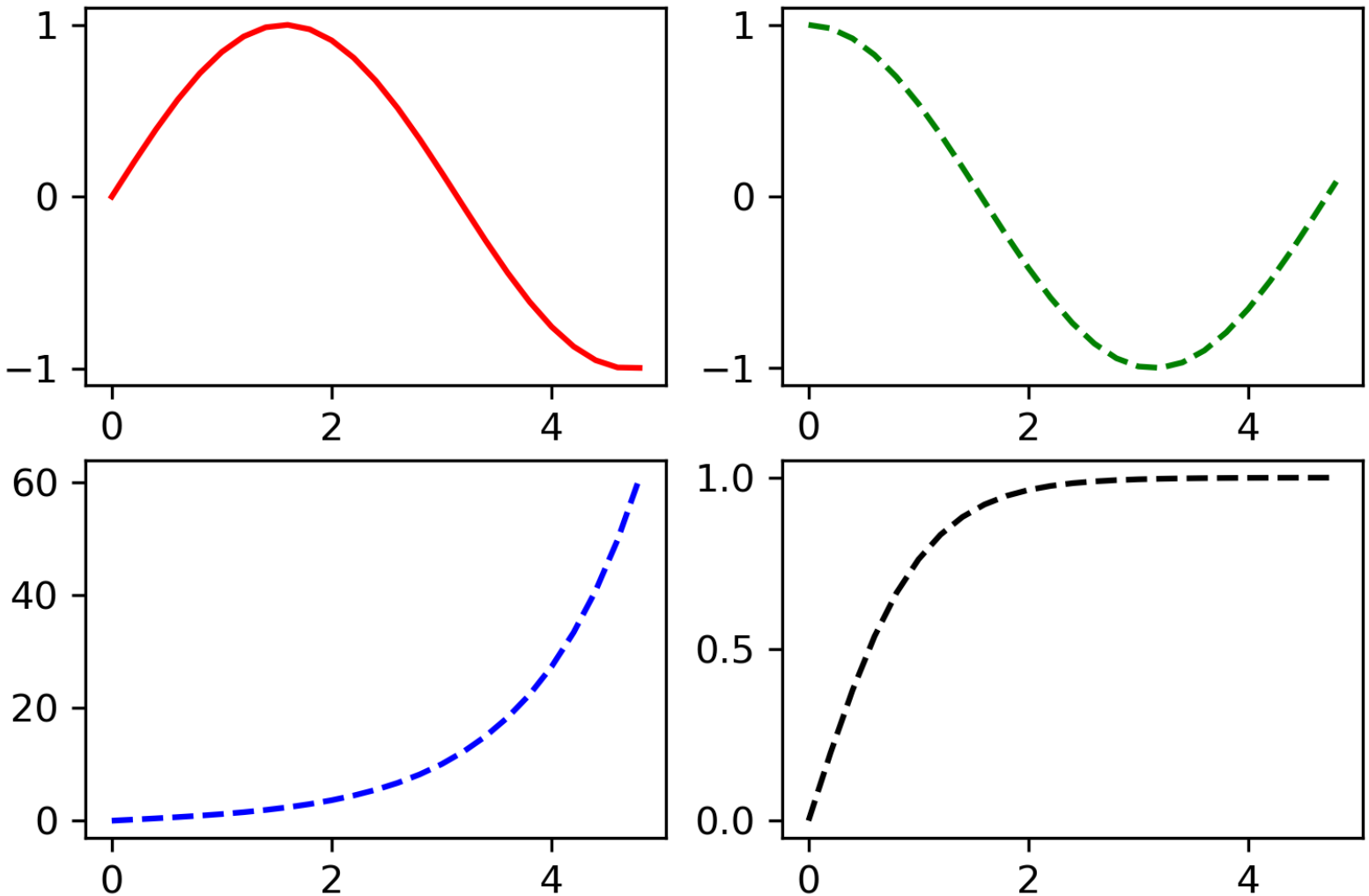


2x2

```
t = np.arange(0.0, 5.0, 0.2)

plt.subplot(221)
plt.plot(t, np.sin(t), 'r--')
plt.subplot(222)
plt.plot(t, np.cos(t), 'g--')
plt.subplot(223)
plt.plot(t, np.sinh(t), 'b--')
plt.subplot(224)
plt.plot(t, np.tanh(t), 'k--')
```

2x2



Графики по данным из файла

Данные из файла

Графики могут построены по данным из текстового файла.

data.txt

```
1 # Данные
2 0.00  1.000
3 0.01  1.252
4 0.02  2.253
5 0.03  3.542
6 0.04  1.725
```

data.txt

```
1 # Данные
2 0.00,1.000
3 0.01,1.252
4 0.02,2.253
5 0.03,3.542
6 0.04,1.725
```

Чтение данных из файла: numpy.loadtxt

```
data = np.loadtxt('data.txt', skiprows = 1)
print(data)
[[ 0.      1.    ]
 [ 0.01    1.252]
 [ 0.02    2.253]
 [ 0.03    3.542]
 [ 0.04    1.725]]
```

Первый столбец

```
data[:,0]

array([ 0.  ,  0.01,  0.02,  0.03,  0.04])
```

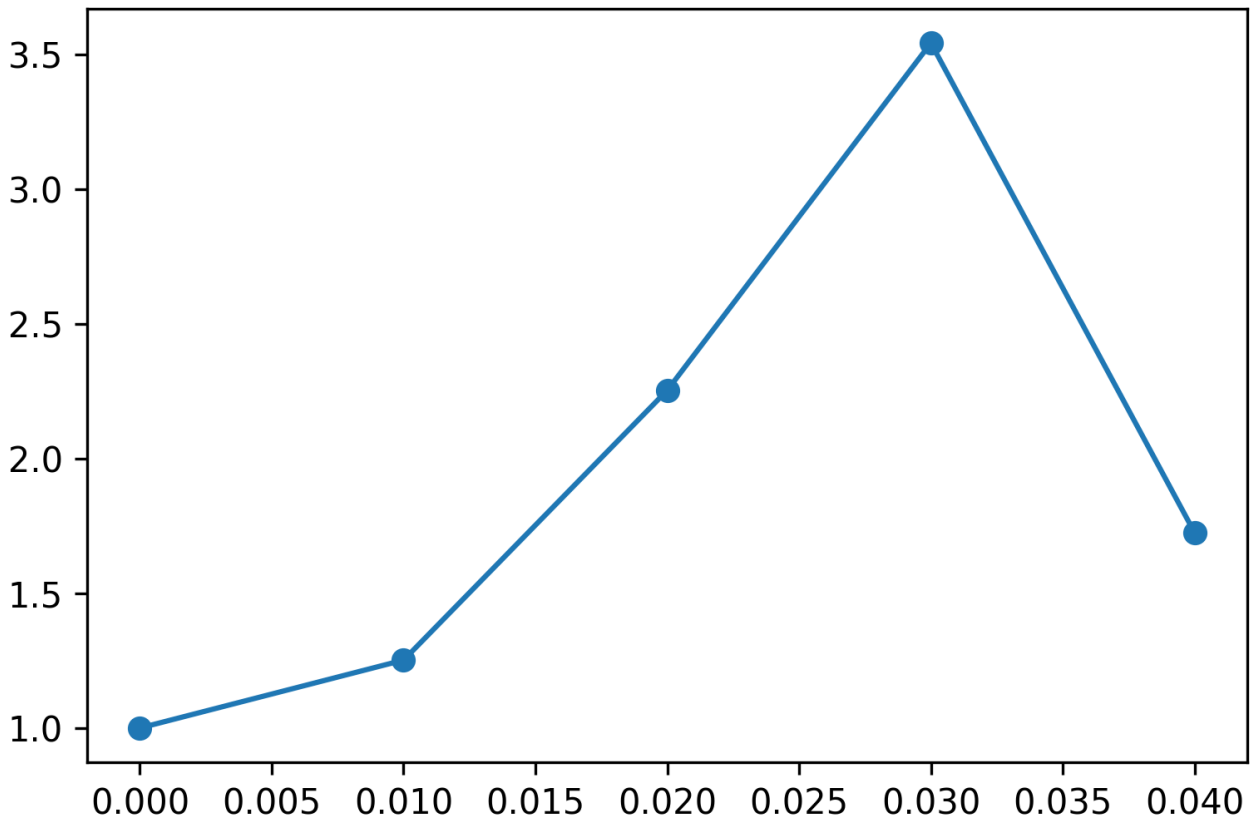
Второй столбец

```
data[:,1]

array([ 1.   ,  1.252,  2.253,  3.542,  1.725])
```

Построение графика

```
1 plt.plot(data[:,0],data[:,1],'-o')
```



Чтение данных из файла: numpy.loadtxt

Для данных, разделённых запятыми, при вызове функции loadtxt указывается разделитель

```
data = np.loadtxt('data.txt', skiprows = 1,
                  delimiter = ',')
print(data)
[[ 0.      1.    ]
 [ 0.01    1.252]
 [ 0.02    2.253]
 [ 0.03    3.542]
 [ 0.04    1.725]]
```

Источники

- https://matplotlib.org/
- Pyplot tutorial
- Научная графика а Python