**Лекция 4**

**Библиотека Matplotlib**

Практически все задачи, связанные с построением графиков, можно решить, используя возможности, которые предоставляет модуль pyplot. В настоящее время среди пользователей принято импорт модуля pyplot производить следующим образом:

import matplotlib.pyplot as plt

**Построение графиков**

Основным элементом изображения, которое строит pyplot, является фигура (Figure), на неё накладывается одно или более поле с графиками, оси координат, текстовые надписи и т.д. Для построения графика используется функция plot(). В самом минимальном варианте её можно использовать без параметров:

import matplotlib.pyplot as plt

%matplotlib inline

plt.plot()

В результате будет выведено пустое поле (см. рисунок 1)

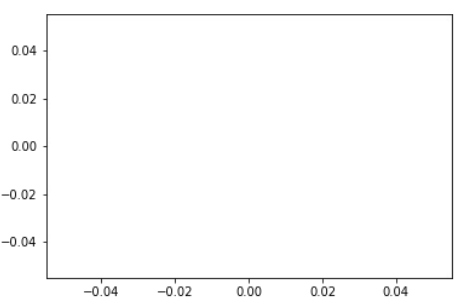


Рисунок 1 — Пустое поле

Если в качестве параметра функции plot() передать список, то значения из этого списка будут отложены по оси ординат (ось y), а по оси абсцисс (ось x) будут отложены индексы элементов массива:

plt.plot([1, 7, 3, 5, 11, 1])

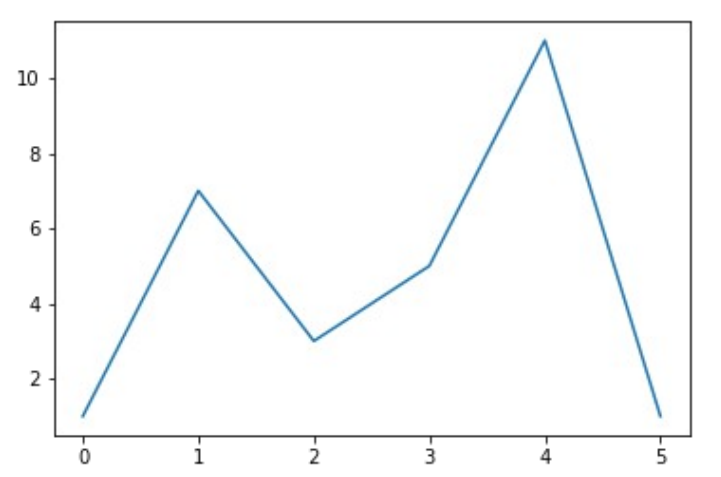


Рисунок 2 — Линейный график, построенный по значениям для оси Y

Для того чтобы задать значения по осям X и Y, необходимо в plot() передать два списка:

plt.plot([1, 5, 10, 15, 20], [1, 7, 3, 5, 11])

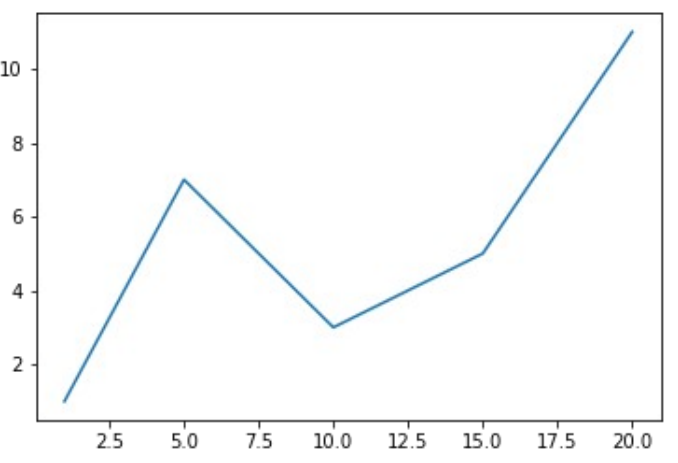


Рисунок 3 — Линейный график, построенный по значениям для осей Y и X

**Текстовые надписи на графике**

Наиболее часто используемые текстовые надписи на графике это:

• наименования осей;

• наименование самого графика;

• текстовые примечания на поле с графиком;

• легенда.

**Наименование осей**

Для задания подписи оси x используется функция xlabel(), оси y — ylabel(). Основными параметрами функций xlabel() и ylabel() являются:

• xlabel (или ylabel): str

◦ Текст подписи.

• labelpad: численное значение либо None; значение по умолчанию:

None

◦ Расстояние между областью графика, включающим оси, и текстом подписи.

Функции xlabel() и ylabel() дополнительно принимают в качестве аргументов параметры конструктора класса matplotlib.text.Text (далее Text), вот некоторые из них:

• fontsize или size: число либо значение из списка: {'xx-small',

'x-small', 'small', 'medium', 'large', 'x-large', 'xxlarge'}

◦ Размер шрифта.

fontstyle: значение из списка: {'normal', 'italic',

'oblique'}

◦ Стиль шрифта.

• fontweight: число в диапазоне от 0 до 1000 либо значение из

списка: {'ultralight', 'light', 'normal', 'regular',

'book', 'medium', 'roman', 'semibold', 'demibold',

'demi', 'bold', 'heavy', 'extra bold', 'black'}

◦ Толщина шрифта.

• color: один из доступных способов задания цвета

◦ Цвет текста подписи.

Пример использования:

plt.xlabel('Day', fontsize=15, color='blue')

Нами были рассмотрены только некоторые из аргументов функций xlabel() и ylabel()

**Заголовок графика**

Для задания заголовка графика используется функция title():

plt.title('Chart price', fontsize=17)

Из её параметров отметим следующие:

• label: str

◦ Текст заголовка.

• loc: значение из набора: {'center', 'left', 'right'}

◦ Выравнивание заголовка.

Для функции title() также доступны параметры конструктора класса Text, некоторые из них приведены в описании аргументов функций xlabel() и ylabel().

**Текстовое примечание**

За размещение текста на поле графика отвечает функция text(). Первый и второй её аргументы — это координаты позиции, третий — текст надписи, пример использования:

plt.text(1, 1, 'type: Steel')

Для более тонкой настройки внешнего вида текстового примечания используйте параметры конструктора класса Text.

**Легенда**

Легенда будет размещена на графике, если вызвать функцию legend(), в рамках данного раздела мы не будем рассматривать аргументы этой функции.

Разместим на графике рассмотренные выше текстовые элементы:

x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

plt.plot(x, y, label='steel price')

plt.title('Chart price', fontsize=15)

plt.xlabel('Day', fontsize=12, color='blue')

plt.ylabel('Price', fontsize=12, color='blue')

plt.legend()

plt.grid(True)

plt.text(15, 4, 'grow up!')

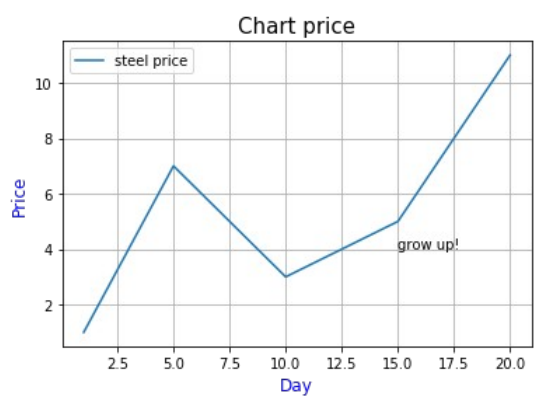


Рисунок 4 — Текстовые надписи на графике

**Работа с линейным графиком**

Линейный график строится с помощью функции plot(), его внешний вид можно настроить непосредственно через аргументы указанной функции,

plt.plot(x, y, color='red')

Либо можно воспользоваться функцией setp():

plt.setp(color='red', linewidth=1)

**Стиль линии графика**

Стиль линии графика задаётся через параметр linestyle, который может принимать значения из таблицы 1.

Таблица 1 — Стили линии линейного графика

Изображение выглядит как стол

Автоматически созданное описание

Стиль линии можно передать сразу после списков с координатами без

указания, что это параметр linestyle:

x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

plt.plot(x, y, '--')

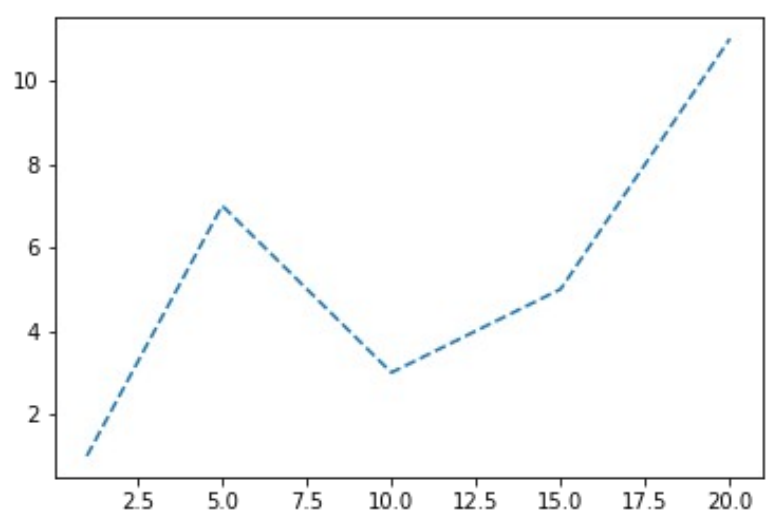


Рисунок 5 — Линия с примененным стилем

Другой вариант — это воспользоваться функцией setp():

x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

line = plt.plot(x, y)

plt.setp(line, linestyle='--')

Для того, чтобы вывести несколько графиков на одном поле, необходимо передать соответствующие наборы значений в функцию plot().

Построим несколько наборов данных и выведем их, задав различные стили линиям:

x = [1, 5, 10, 15, 20]

y1 = [1, 7, 3, 5, 11]

y2 = [i\*1.2 + 1 for i in y1]

y3 = [i\*1.2 + 1 for i in y2]

y4 = [i\*1.2 + 1 for i in y3]

plt.plot(x, y1, '-', x, y2, '--', x, y3, '-.', x, y4, ':')

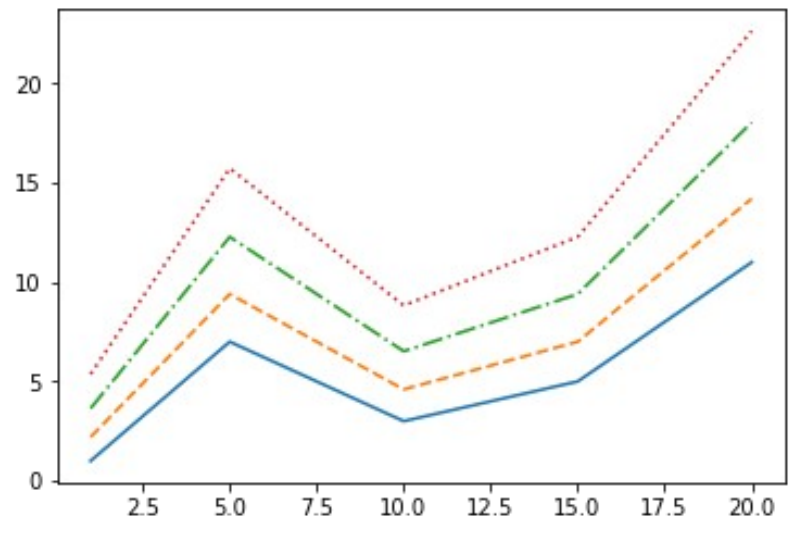


Рисунок 6 — Несколько графиков, построенных одной функцией plot()

Тот же результат можно получить, вызвав plot() отдельно для каждого набора данных. Если вы хотите представить графики изолированно друг от друга (каждый на своём поле), то используйте для этого функцию subplot() :

plt.plot(x, y1, '-')

plt.plot(x, y2, '--')plt.plot(x, y3, '-.')

plt.plot(x, y4, ':')

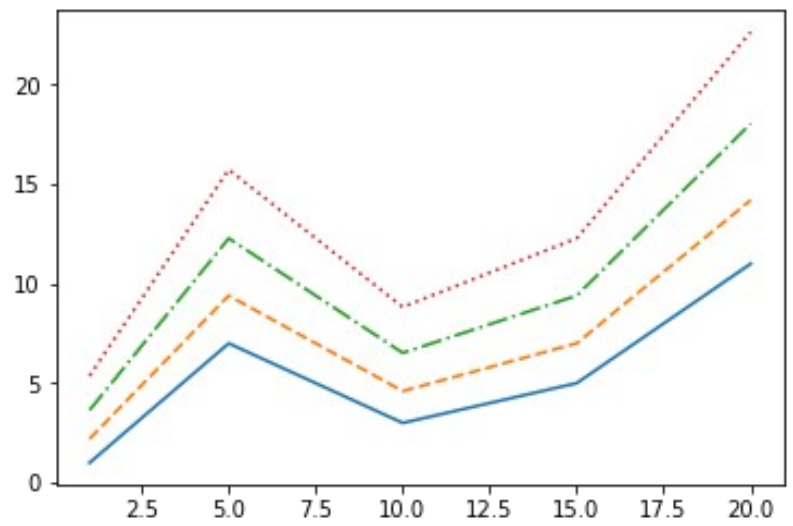


Рисунок 7 — Несколько графиков, построенных разными функциями plot()