Существует два основных варианта установки этой библиотеки: в первом случае вы устанавливаете пакет Anaconda, в состав которого входит большое количество различных инструментов для работы в области машинного обучения и анализа данных (и не только); во втором – установить Matplotlib самостоятельно, используя менеджер пакетов.

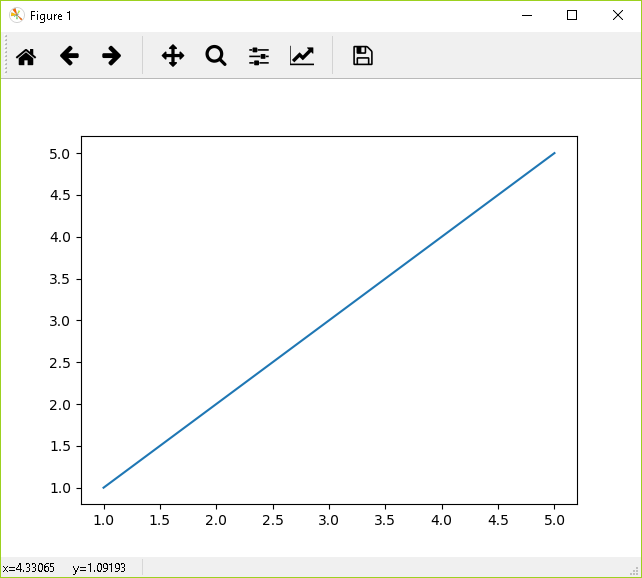
Пример простейшего использования библиотеки matplotlib будет выглядеть так:

**import** matplotlib.pyplot **as** plt

plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])

plt.show()

В результате получим график в отдельном окне.



**Построение графика**

Для начал построим простую линейную зависимость, дадим нашему графику название, подпишем оси и отобразим сетку. Код программы:

**import** numpy **as** np

# Независимая (x) и зависимая (y) переменные

x = np.linspace(0, 10, 50)

y = x

# Построение графика

plt.title("Линейная зависимость y = x") # заголовок

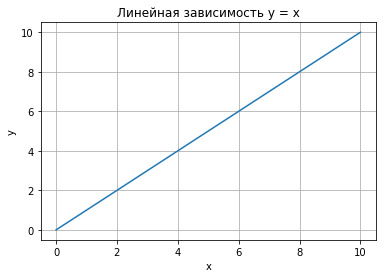
plt.xlabel("x") # ось абсцисс

plt.ylabel("y") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y) # построение графика

В результате получим следующий график:



Изменим тип линии и ее цвет, для этого в функцию *plot()*, в качестве третьего параметра передадим строку, сформированную определенным образом, в нашем случае это “r–”, где “r” означает красный цвет, а “–” – тип линии – пунктирная линия. Более подробно о том, как задавать цвет и какие типы линии можно использовать будет рассказано с одной из следующих глав.

# Построение графика

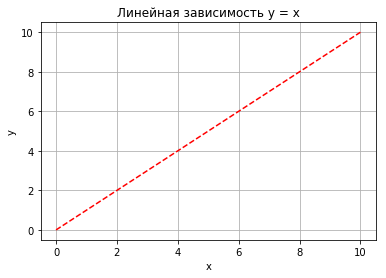
plt.title("Линейная зависимость y = x") # заголовок

plt.xlabel("x") # ось абсцисс

plt.ylabel("y") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y, "r--") # построение графика



**Несколько графиков на одном поле**

Построим несколько графиков на одном поле, для этого добавим квадратичную зависимость:

# Линейная зависимость

x = np.linspace(0, 10, 50)

y1 = x

# Квадратичная зависимость

y2 = [i\*\*2 **for** i **in** x]

# Построение графика

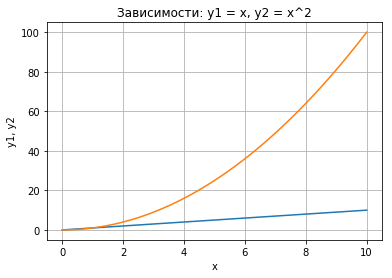
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок

plt.xlabel("x") # ось абсцисс

plt.ylabel("y1, y2") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y1, x, y2) # построение графика



В приведенном примере в функцию *plot()* последовательно передаются два массива для построения первого графика и два массива для построения второго, при этом, как вы можете заметить, для обоих графиков массив значений независимой переменной *x* один и то же.

**Несколько разделенных полей с графиками**

Третья, довольно часто встречающаяся задача – это отобразить два или более различных поля, на которых будет отображено по одному или более графику.

Построим уже известные нам две зависимость на разных полях.

# Линейная зависимость

x = np.linspace(0, 10, 50)

y1 = x

# Квадратичная зависимость

y2 = [i\*\*2 **for** i **in** x]

# Построение графиков

plt.figure(figsize=(9, 9))

plt.subplot(2, 1, 1)

plt.plot(x, y1) # построение графика

plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок

plt.ylabel("y1", fontsize=14) # ось ординат

plt.grid(**True**) # включение отображение сетки

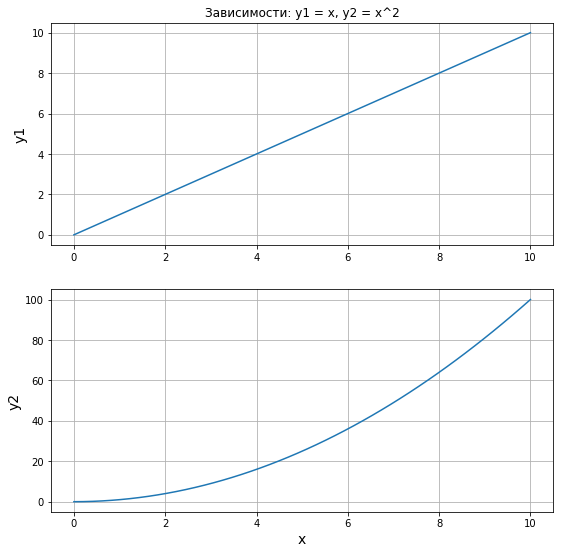
plt.subplot(2, 1, 2)

plt.plot(x, y2) # построение графика

plt.xlabel("x", fontsize=14) # ось абсцисс

plt.ylabel("y2", fontsize=14) # ось ординат

plt.grid(**True**) # включение отображение сетки



Здесь мы воспользовались новыми функциями:

*figure()* – функция для задания глобальных параметров отображения графиков. В нее, в качестве аргумента, мы передаем кортеж, определяющий размер общего поля.

*subplot()* – функция для задания местоположения поля с графиком. Существует несколько способов задания областей для вывода через функцию subplot() мы воспользовались следующим: первый аргумент – количество строк, второй – столбцов в формируемом поле, третий – индекс (номер поля, считаем сверху вниз, слева направо).

Остальные функции уже вам знакомы, дополнительно мы использовали параметр *fontsize*для функций *xlabel()* и *ylabel()*, для задания размера шрифта.

**Построение диаграммы для категориальных данных**

До этого мы строили графики по численным данным, т.е. зависимая и независимая переменные имели числовой тип. На практике довольно часто приходится работать с данными нечисловой природы – имена людей, название фруктов, и т.п.

Построим диаграмму на которой будет отображаться количество фруктов в магазине:

fruits = ["apple", "peach", "orange", "bannana", "melon"]

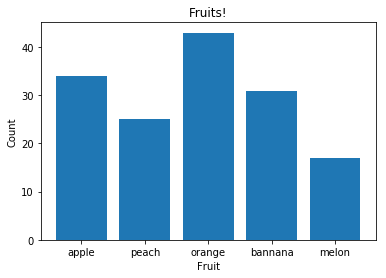
counts = [34, 25, 43, 31, 17]

plt.bar(fruits, counts)

plt.title("Fruits!")

plt.xlabel("Fruit")

plt.ylabel("Count")



Для вывода диаграммы мы использовали функцию *bar()*.

К этому моменту, если вы самостоятельно попробовали запустить приведенные выше примеры, у вас уже должно сформировать некоторое понимание того, как осуществляется работа с этой библиотекой.

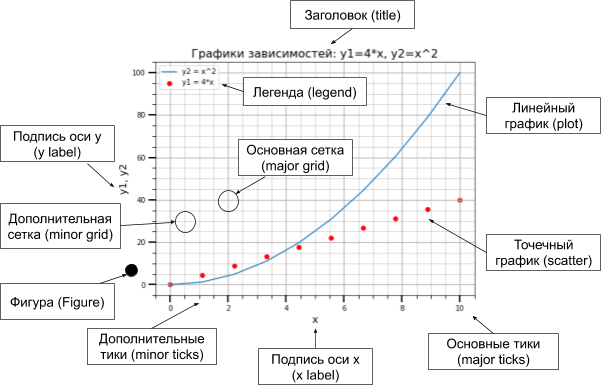
**Основные элементы графика**

Рассмотрим основные термины и понятия, касающиеся изображения графика, с которыми вам необходимо будет познакомиться, для того, чтобы в дальнейшем у вас не было трудностей при прочтении материалов из этого цикла статей и документации по библиотеке *matplotlib*.

Корневым элементом при построения графиков в системе *Matplotlib*является Фигура (*Figure*). Все, что нарисовано на рисунке выше является элементами фигуры. Рассмотрим ее составляющие более подробно.

**График**

На рисунке представлены два графика – линейный и точечный. *Matplotlib*предоставляет огромное количество различных настроек, которые можно использовать для того, чтобы придать графику вид, который вам нужен: цвет, толщина и тип линии, стиль линии и многое другое, все это мы рассмотрим в ближайших статьях.



**Оси**

Вторым, после непосредственно самого графика, по важности элементом фигуры являются оси. Для каждой оси можно задать метку (подпись), основные (*major*) и дополнительные (*minor*) тики, их подписи, размер и толщину, также можно задать диапазоны по каждой из осей.

**Сетка и легенда**

Следующими элементами фигуры, которые значительно повышают информативность графика являются сетка и легенда. Сетка также может быть основной (*major*) и дополнительной (*minor*). Каждому типу сетки можно задавать цвет, толщину линии и тип. Для отображения сетки и легенды используются соответствующие команды.

Ниже представлен код, с помощью которого была построена фигура, изображенная на рисунке выше:

**import** matplotlib.pyplot **as** plt

**from** *matplotlib.ticker* **import** (MultipleLocator, FormatStrFormatter,

AutoMinorLocator)

**import** numpy **as** np

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

y1 = 4\*x

y2 = [i\*\*2 **for** i **in** x]

fig, ax = plt.subplots(figsize=(8, 6))

ax.set\_title("Графики зависимостей: y1=4\*x, y2=x^2", fontsize=16)

ax.set\_xlabel("x", fontsize=14)

ax.set\_ylabel("y1, y2", fontsize=14)

ax.grid(which="major", linewidth=1.2)

ax.grid(which="minor", linestyle="--", color="gray", linewidth=0.5)

ax.scatter(x, y1, c="red", label="y1 = 4\*x")

ax.plot(x, y2, label="y2 = x^2")

ax.legend()

ax.xaxis.set\_minor\_locator(AutoMinorLocator())

ax.yaxis.set\_minor\_locator(AutoMinorLocator())

ax.tick\_params(which='major', length=10, width=2)

ax.tick\_params(which='minor', length=5, width=1)

plt.show()

Более подробную информацию Вы можете получить по ссылке https://devpractice.ru/.

Задание:

1. Пусть

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

, где

Построить на одном графике функцию (1) и приближенную к ней функцию разными цветами и стилями линий. Подписать оси, дать название графику. Добавить легенду на график. Отобразить детальную сетку

1. На одном холсте отобразить 10 графиков, соответствующих белому шуму (в каждый из моментов времени значение функции соответствует случайному значению в диапазоне (-1,1)).
2. Вывести гистограмму, отображающую выделение времени под те или иные занятия в течении дня.
3. Нарисовать график кривой: .
4. Отобразить график функции согласно варианту, интервалы определить экспериментальным путем: