ЛАБОРАТОРНАЯ РАБОТА №2
ОСНОВЫ РАБОТЫ С ИЗОБРАЖЕНИЯМИ

Цель: Изучение и освоение основных операций обработки изображений с использованием выбранной библиотеки или инструмента.

2.1 Установка

Варианты установки *Matplotlib*

Существует два основных варианта установки этой библиотеки: в первом случае 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.

Установка *Matplotlib* через менеджер *pip*

Второй вариант – это воспользоваться менеджером *pip* и установить *Matplotlib* самостоятельно, для этого введите в командной строке вашей операционной системы следующие команды:

```
>python -m pip install -U pip
>python -m pip install -U matplotlib
```

Первая из них обновит ваш *pip*, вторая установит *matplotlib* со всеми необходимыми зависимостями.

Проверка установки

Для проверки того, что все у вас установилось правильно, запустите интерпретатор *Python* и введите в нем следующее:

```
>>> import matplotlib
```

После этого можете проверить версию библиотеки (она скорее всего будет отличаться от приведенной ниже):

```
>>> matplotlib.__version__
'3.0.3'
```

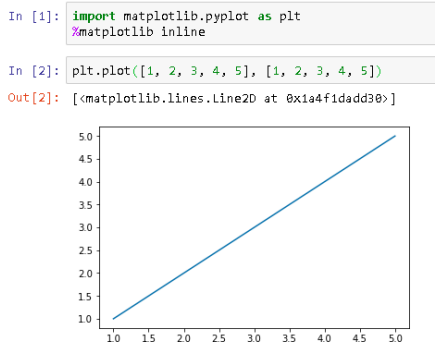
2.2 Быстрый старт

Перед тем как углубиться в дебри библиотеки *Matplotlib*, для того, чтобы появилось интуитивное понимание принципов работы с этим инструментом, рассмотрим несколько примеров, изучив которые уже сможете использовать библиотеку для решения своих задач.

Если работаете в *Jupyter Notebook* для того, чтобы получать графики рядом с ячейками с кодом необходимо выполнить специальную *magic* команду после того, как импортируете *matplotlib*::

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Результат работы выглядеть будет так, как показано на рисунке ниже.

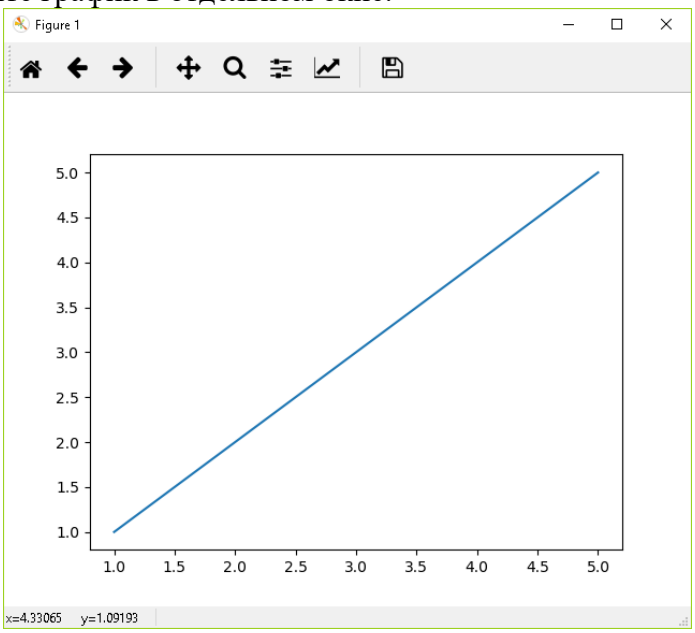


Если пишете код в *.py* файле, а потом запускаете его через вызов интерпретатора *Python*, то строка %matplotlib inline вам не нужна, используйте только импорт библиотеки.

Пример, аналогичный тому, что представлен на рисунке выше, для отдельного *Python* файла будет выглядеть так:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])
plt.show()
```

В результате получите график в отдельном окне.



Далее не будем останавливаться на особенностях использования *magic* команды, просто запомните, если вы используете *Jupyter notebook* при работе с *Matplotlib* вам обязательно нужно включить %matplotlib inline.

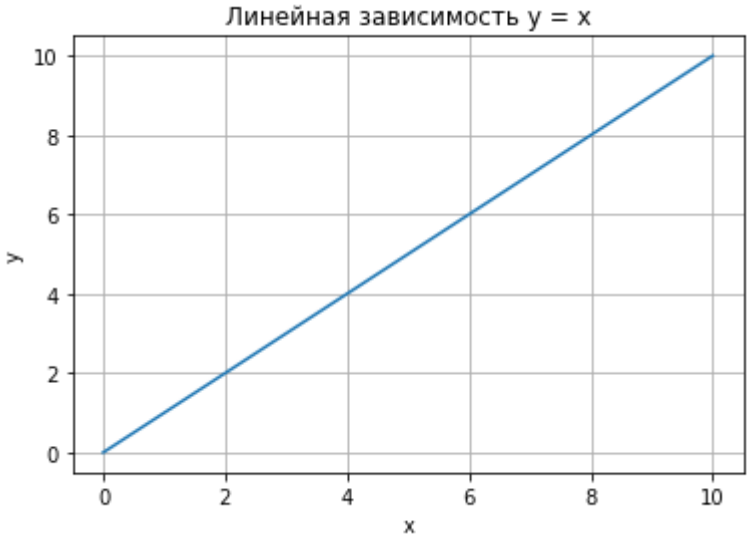
2.3 Построение графика

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

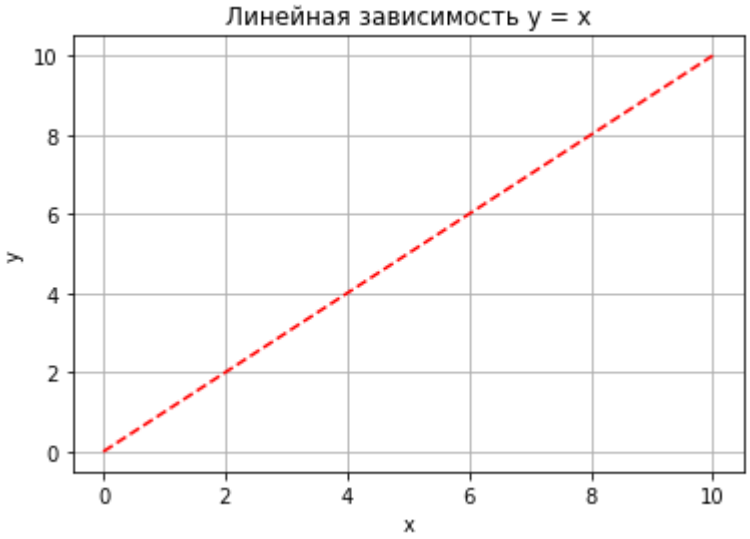
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y) # построение графика
```

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.

```
# 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
```

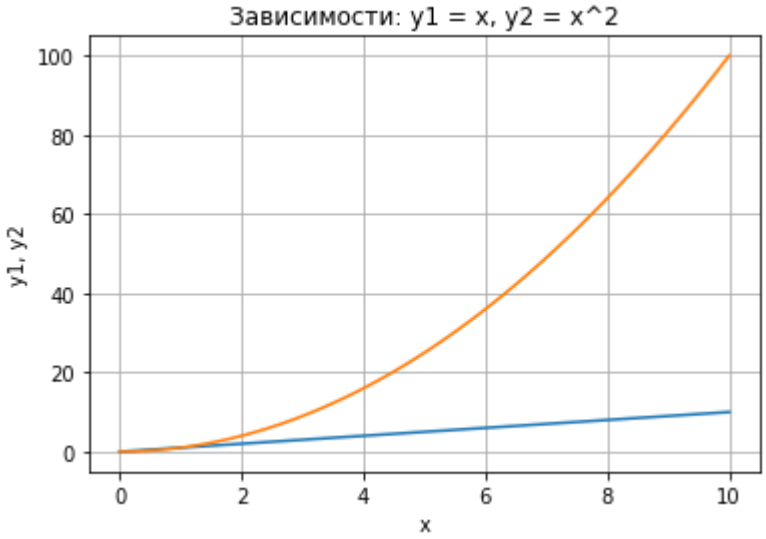


2.4 Несколько графиков на одном поле

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



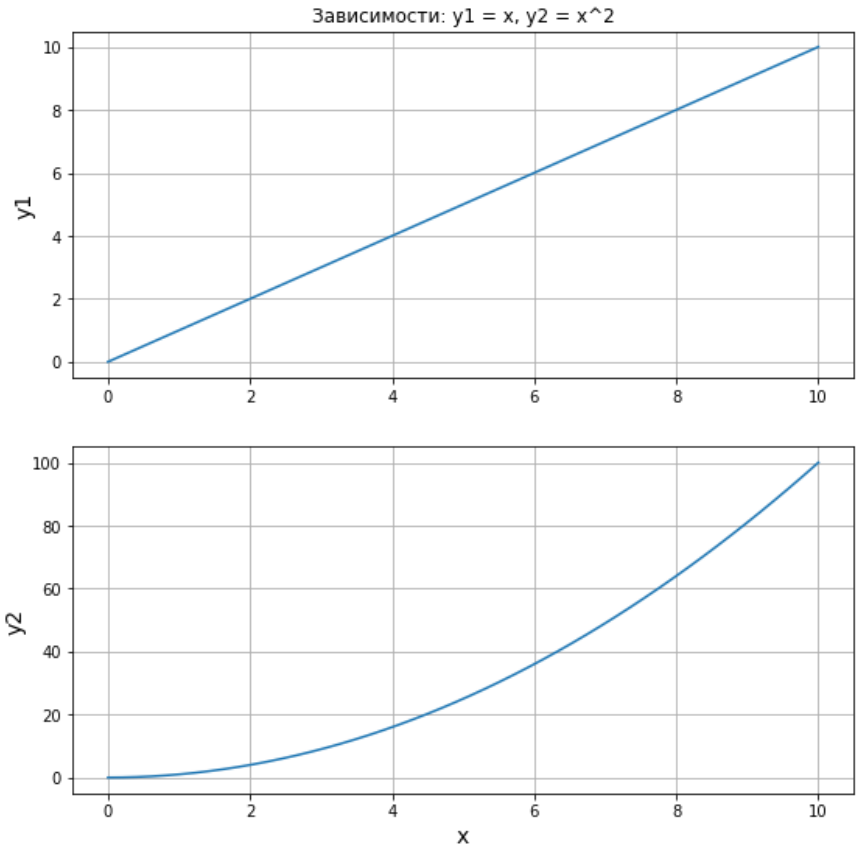
В приведенном примере в функцию *plot()* последовательно передаются два массива для построения первого графика и два массива для построения второго, при этом, как вы можете заметить, для обоих графиков массив значений независимой переменной *x* один и то же.

2.5 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).

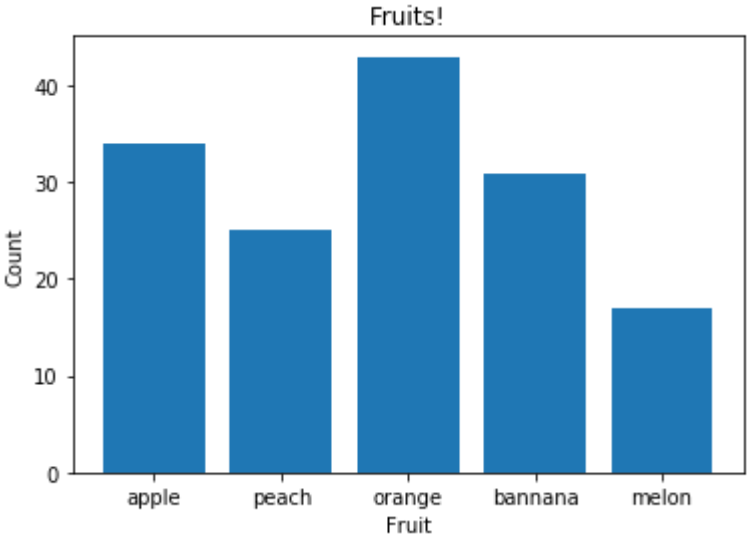
Остальные функции уже вам знакомы, дополнительно мы использовали параметр *fontsize* для функций *xlabel()* и *ylabel()*, для задания размера шрифта.

2.6 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 использовали функцию *bar()*.