# Практическая работа № 4\_5

# Вспомогательные библиотеки Python.

Оглавление

[Цель работы 1](#_Toc83033337)

[Задачи работы 1](#_Toc83033338)

[Перечень обеспечивающих средств 1](#_Toc83033339)

[Общие теоретические сведения 1](#_Toc83033340)

[Задание 5](#_Toc83033341)

[Контрольные вопросы 6](#_Toc83033342)

[Требования к отчету 6](#_Toc83033343)

[Литература 6](#_Toc83033344)

## Цель работы

Изучение некоторых вспомогательных библиотек Python.

## Задачи работы

1. Изучить основы библиотеки numpy.
2. Изучить основы библиотеки pandas.
3. Изучить основы библиотеки matplotlib.

## Перечень обеспечивающих средств

1. ПК.
2. Учебно-методическая литература.
3. Задания для самостоятельного выполнения.

## Общие теоретические сведения

**Библиотека numpy**

Numpy предоставляет удобный инструментарий для работы с большими, в том числе многомерными, массивами данных.

Основной тип данных (объект) – многомерный массив ndarray (от английского n‑dimentional array – n-мерный массив).

Ndarray можно создать напрямую из списка или кортежа с помощью метода numpy.array(), например:

|  |
| --- |
| l = [0, 1, 2]  t = (3, 4, 5)  a1 = numpy.array(l)  a2 = numpy.array(t) |

Аналогично создается многомерный ndarray:

|  |
| --- |
| l = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]  a = numpy.array(l) |

Также есть специальные методы для создания специфических массивов:

numpy.zeros(t) – создает массив, заполненный нулями, с размерностями, заданными кортежем t. Например, при t равном (3, 3) будет создан двумерный массив 3 на 3, а при t равном (2, 3, 4) — трехмерный массив 2 на 3 на 4 и т. п.

Аналогично, метод numpy.ones(t) создает массив, заполненный единицами, с размерностями, заданными кортежем t.

Метод numpy.empty(t) создает пустой массив с размерностями, заданными кортежем t.

Также есть метод numpy.eye(n), который создает двумерный квадратный массив n на n с единичной матрицей, т. е. по диагонали стоят единицы, а в остальных позициях — нули.

Основные свойства массива ndarray:

ndim – количество измерений или осей массива,

shape – кортеж с размерностью массива,

dtype – тип элемента в массиве.

С массивами одинаковой размерности можно производить обычные арифметические операции: сложение, умножение и т.д., они выполняются поэлементно.

Также можно выполнять арифметические операции с массивом и числом, при этом операция будет выполняться между каждым элементом массива и числом.

Для получения элементов массива можно использовать индексы и слайсы, аналогично тому, как это делается со списками и кортежами. При этом следующие записи эквиваленты: a[0][1], a[0, 1], a[(0, 1)].

Можно изменить размерность массива с помощью метода reshape(t), где t — кортеж, задающий новую форму, например:

|  |
| --- |
| a = numpy.array([[0, 1], [2, 3], [4, 5]])  a = a.reshape((2, 3))  a = a.reshape((6, 1))  a = a.reshape((1, 6)) |

**Библиотека pandas**

Pandas предоставляет инструментарий для работы с табличными данными.

Основные типы данных (объекты): Series и DataFrame.

Series – это одномерный массив с заданным явно или сформированным автоматически индексом. Больше всего он похож на словарь.

|  |
| --- |
| # Явно заданные индексы  s1 = pandas.Series({0:1, 2:3, 4:5})  s2 = pandas.Series([1, 3 , 5], index = [0, 2, 4])  # Автоматические индексы  s3 = pandas.Series([1, 3, 5]) |

DataFrame — это таблица, столбцами которой являются Series. DataFrame можно создать явно, например, из словаря, значениям в котором выступают списки:

|  |
| --- |
| df = pandas.DataFrame({1 : [1, 2, 3], 2 : [11, 12, 12], 3 : [21, 22, 23]}) |

Другой способ — создать DataFrame из csv-файла, используя метод read\_csv(*<Имя файла>*, *<Разделитель>*), где разделитель по умолчанию — запятая.

К отдельным столбцам можно обращаться по их ключам, например, df[1].

К отдельным строкам — с помощью свойства iloc, например, df.iloc[1].

Также возможна комбинация индексирования по строкам и столбцам одновременно, в том числе, с применением слайсинга: df.iloc[1, 1] или df.iloc[1, :].

В качестве индексов можно указывать условия на значения столбцов, например, df[df[1] > 1].

Добавить новый столбец в DataFrame можно использовать метод insert(<*Номер столбца*>, <*Имя столбца*>, <*Значения*>), где значения могут быть переданы в виде списка. Другой вариант — использовать нотацию словаря. Пример:

|  |
| --- |
| l = [31, 32, 33]  df.insert(3, 4, l)  df[4] = l |

Чтобы удалить столбец можно использовать метод drop(<*Имя столбца*>).

**Библиотека matplotlib**

Matplotlib предоставляет возможность визуального представления различных данных. Ниже мы рассмотрим основы интерфейса pyplot.

Отображение гистограммы выполняется с помощью метода matplotlib.pyplot.hist(), например:

|  |
| --- |
| x = list(range(100))  matplotlib.pyplot.hist(x) |

Отображение графика y от x выполняется с помощью метода matplotlib.pyplot.plot(), например:

|  |
| --- |
| x = numpy.array(range(100))  y = x \*\* 2  matplotlib.pyplot.plot(x, y) |

Также x и y могут быть заданы как имена словаря, numpy.ndarray, pandas.DataFrame т. п., например:

|  |
| --- |
| df = pandas.DataFrame({'Икс' : numpy.array(range(100)), 'Игрек' : numpy.array(range(100)) \*\* 2})  matplotlib.pyplot.plot('Икс', 'Игрек', data = df) |

Если передать только один числовой ряд, то он считается значениями y, а x определяется автоматически.

При необходимости график можно дополнить заголовком, названиями осей, а также ограничить значения, выводимые по обеим осям, следующими методами:

matplotlib.pyplot.title('*<Заголовок>*') - добавляет заголовок,

matplotlib.pyplot.xlabel('*<Ось X>*') - добавляет название оси X,

matplotlib.pyplot.ylabel('*<Ось Y>*') - добавляет название оси Y,

matplotlib.pyplot.xlim(*min\_x*, *max\_x*) – устанавливает значения оси X от min\_x до max\_x,

matplotlib.pyplot.ylim(*min\_y*, *max\_y*) – устанавливает значения оси Y от min\_y до max\_y,

matplotlib.pyplot.show() - отображает график со всеми установленными параметрами.

Пример:

|  |
| --- |
| df = pandas.DataFrame({'Икс' : numpy.array(range(100)), 'Игрек' : numpy.array(range(100)) \*\* 2})  matplotlib.pyplot.plot('Икс', 'Игрек', data = df)  plt.title('Парабола')  plt.xlabel('X')  plt.ylabel('Y')  plt.xlim(0, 10)  plt.ylim(0, 10)  plt.show() |

## Задание

**Часть 1**

* Сделайте форк репозитория <https://github.com/mosalov/NotebookWithLibs>
* Откройте сайт Binder: <https://mybinder.org/>.
* В поле «GitHub repository name or URL» укажите ссылку на свой репозиторий. Нажмите кнопку «launch», дождитесь открытия репозитория.
* Откройте (кликните) файл «[empty\_notebook.ipynb](https://hub.gke.mybinder.org/user/mosalov-emptyju-eebookforbinder-wu4mdwz8/notebooks/empty_notebook.ipynb#_blank)».
* Создайте квадратный двумерный массив 4 на 4, заполненный единицами.
* Преобразуйте его в массив, заполненный пятерками.
* Измените форму массива на 2 на 8.
* Выведите получившийся массив.
* Сохраните файл Jupyter notebook с названием «*Фамилия*\_Задание 4\_5\_1.ipynb» и загрузите его в созданный репозиторий.

**Часть 2**

* Вернитесь к файлу «[empty\_notebook.ipynb](https://hub.gke.mybinder.org/user/mosalov-emptyju-eebookforbinder-wu4mdwz8/notebooks/empty_notebook.ipynb#_blank)», открытому в Binder.
* Создайте пустой DataFrame.
* Добавьте в него столбец «Число» со значениями от 1 до 10.
* Добавьте в него столбец «Квадрат» со значениями, равными квадратам чисел в столбце «Число».
* Создайте новый DataFrame из строк первого DataFrame, в которых значение в столбце «Число» — чётное.
* Удалите из нового DataFrame столбец «Число».
* Выведите получившийся DataFrame.
* Сохраните файл Jupyter notebook с названием «*Фамилия*\_Задание 4\_5\_2.ipynb» и загрузите его в созданный репозиторий.

**Часть 3**

* Вернитесь к файлу «[empty\_notebook.ipynb](https://hub.gke.mybinder.org/user/mosalov-emptyju-eebookforbinder-wu4mdwz8/notebooks/empty_notebook.ipynb#_blank)», открытому в Binder.
* Создайте DataFrame из файла «winequality-white.csv», обратите внимание на разделитель значений в файле.
* Из полученного DataFrame создайте новый следующим образом:
  + возьмите только те строки, для которых значение в столбце «residual sugar» меньше 1,
  + возьмите только столбцы «density» и «pH»,
  + из результата возьмите только четные строки
* Выведите получившийся DataFrame.
* Сохраните файл Jupyter notebook с названием «*Фамилия*\_Задание 4\_5\_3.ipynb» и загрузите его в созданный репозиторий.

**Часть 4**

* Вернитесь к файлу «[empty\_notebook.ipynb](https://hub.gke.mybinder.org/user/mosalov-emptyju-eebookforbinder-wu4mdwz8/notebooks/empty_notebook.ipynb#_blank)», открытому в Binder.
* Используя DataFrame, полученный в предыдущем пункте, постройте гистограмму значений в столбце «pH».
* Используя DataFrame, полученный в предыдущем пункте, постройте график для значений в столбце «density».
* Сохраните файл Jupyter notebook с названием «*Фамилия*\_Задание 4\_5\_4.ipynb» и загрузите его в созданный репозиторий.

## Контрольные вопросы

1. Предложите структуру массива ndarray для хранения состояния кубика Рубика (https://ru.wikipedia.org/wiki/Кубик\_Рубика). Приведите пример заполненной структуры для собранного состояния кубика.
2. Постройте график гиперболы y = 1/x на отрезке от -10 до 10.

## Требования к отчету

Все файлы загрузите в свой репозиторий, созданный в практическом задании №1 по пути: «Notebook\_For\_AI\_Main/2021 Осенний семестр/Практическое задание 4\_5/» и сделайте пул-реквест.

## Литература

1. <https://pythonworld.ru/numpy/1.html>
2. <https://habr.com/ru/post/352678/>
3. <https://khashtamov.com/ru/pandas-introduction/>
4. <https://habr.com/ru/post/196980/>
5. https://nbviewer.jupyter.org/github/whitehorn/Scientific\_graphics\_in\_python/blob/master/P1%20Chapter%202%20Main%20graphical%20commands.ipynb