**Язык программирования Python**

В 1991 году Гвидо ван Россумом был создан язык программирования Python. Назван он был в честь любимого шоу создателя– «Летающий цирк Монти Пайтона» («Monty Python's Flying Circus»). Существует 3 версии Python. В настоящее время самым актуальным является Python 3.

Python – это интерпретируемый, универсальный и передовой язык программирования, который используется в следующих областях:

* анализ данных;
* написание прикладных программных систем;
* разработка графических интерфейсов;
* написание сетевого программного обеспечения;
* взаимодействие с базами данных.

Принципы языка Python, они же философия, к которой призывают стремиться «носители» языка, сформулированы в The Zen of Python:

● Красивое лучше, чем уродливое.

● Явное лучше, чем неявное.

● Простое лучше, чем сложное.

● Сложное лучше, чем запутанное.

● Плоское лучше, чем вложенное.

● Разреженное лучше, чем плотное.

● Читаемость имеет значение.

● Особые случаи не настолько особые, чтобы нарушать правила.

● При этом практичность важнее безупречности.

● Ошибки никогда не должны замалчиваться.

● Если они не замалчиваются явно.

● Встретив двусмысленность, отбрось искушение угадать.

● Должен существовать один и желательно только один очевидный способ сделать это.

● Хотя он поначалу может быть и не очевиден, если вы не голландец.

● Сейчас лучше, чем никогда.

● Хотя никогда зачастую лучше, чем прямо сейчас.

● Если реализацию сложно объяснить – идея плоха.

● Если реализацию легко объяснить – идея, возможно, хороша.

● Пространства имен – отличная штука! Будем делать их больше!

Данная философия подходит к разработке на любом языке, и нередко разработчики на других языках ссылаются на The Zen of Python.

К особенностям Python относится *динамическая типизация*. Типы объектов во время выполнения программы строго не определены, они могут изменяться. Так же для определения переменных не нужно объявлять их тип.

Для выделения блоков кода используется «:».

Python позволяет писать операции на одной строке с помощью разделителя «;», но делать так не рекомендуется, такой код достаточно сложно читать.

Однострочный комментарий в Python начинается с #, а многострочный выделяется с помощью ’’’…’’’ или ”””…”””.

Python поддерживает функциональное программирование.

Существуют два режима запуска Python:

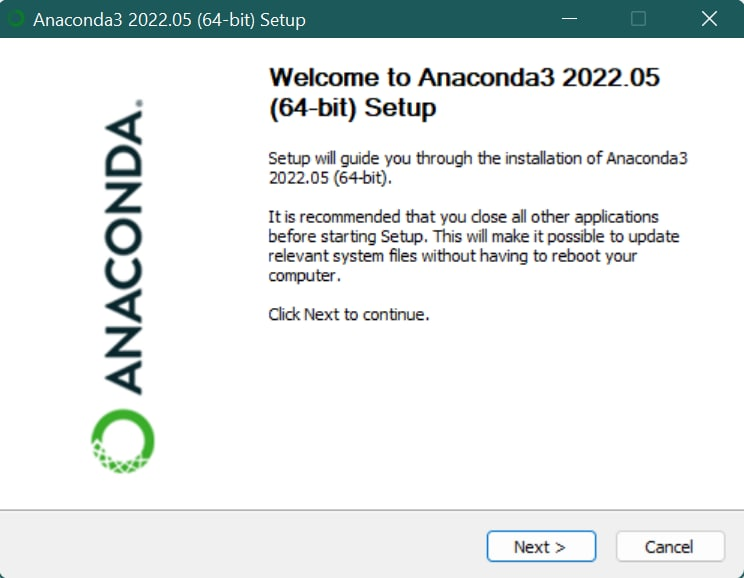
1. Стандартный режим. Необходимо сохранить файл с кодом с расширением .py, затем подать этот файл на вход интерпретатору для запуска кода.
2. Интерактивный режим. Результат работы команд виден сразу.

**Установка Python**

Один из самых простых способов установки Python – через дистрибутив Anaconda. Он содержит не только Python, но и необходимые библиотеки.

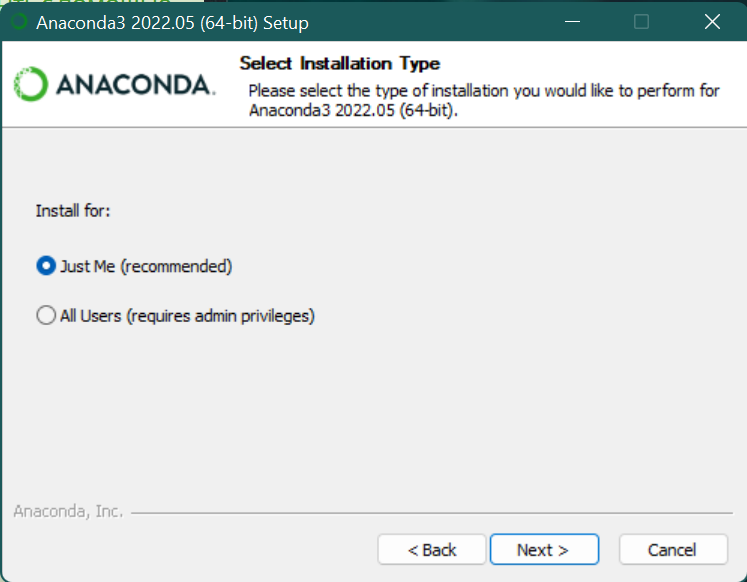
Скачать Anaconda можно тут: <https://www.anaconda.com/>

Далее необходимо установить Anaconda.

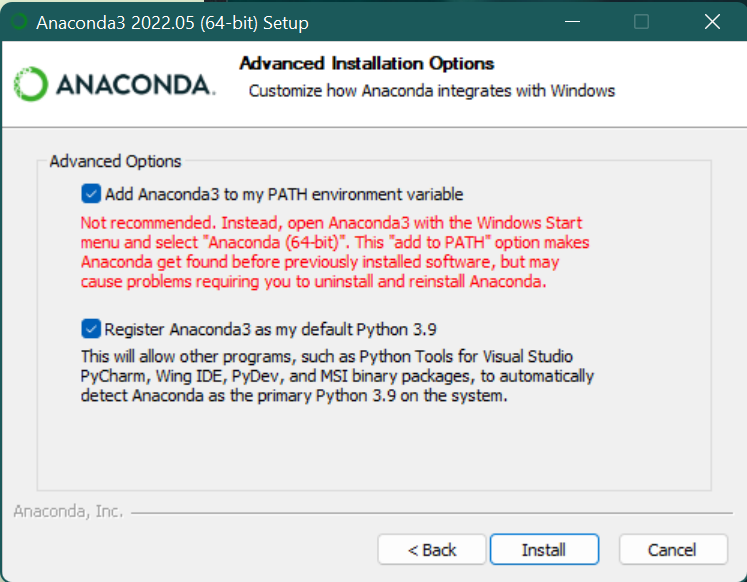


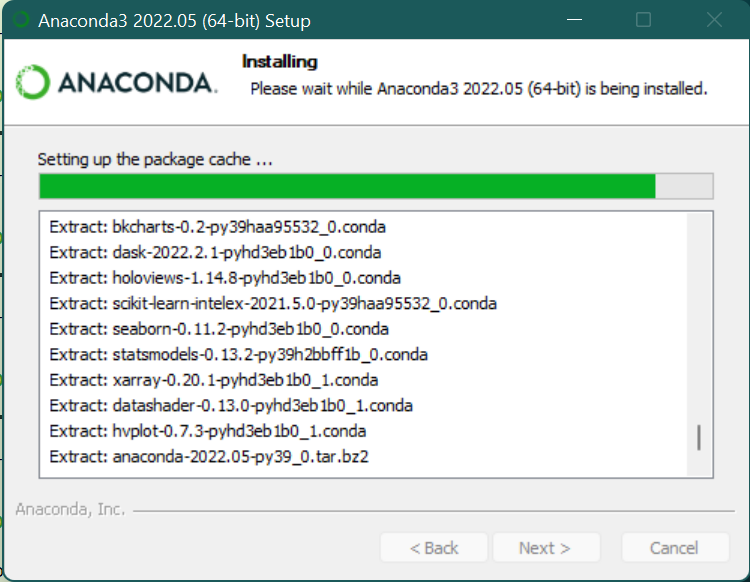


Необходимо ознакомиться с лицензией и нажать I agree.

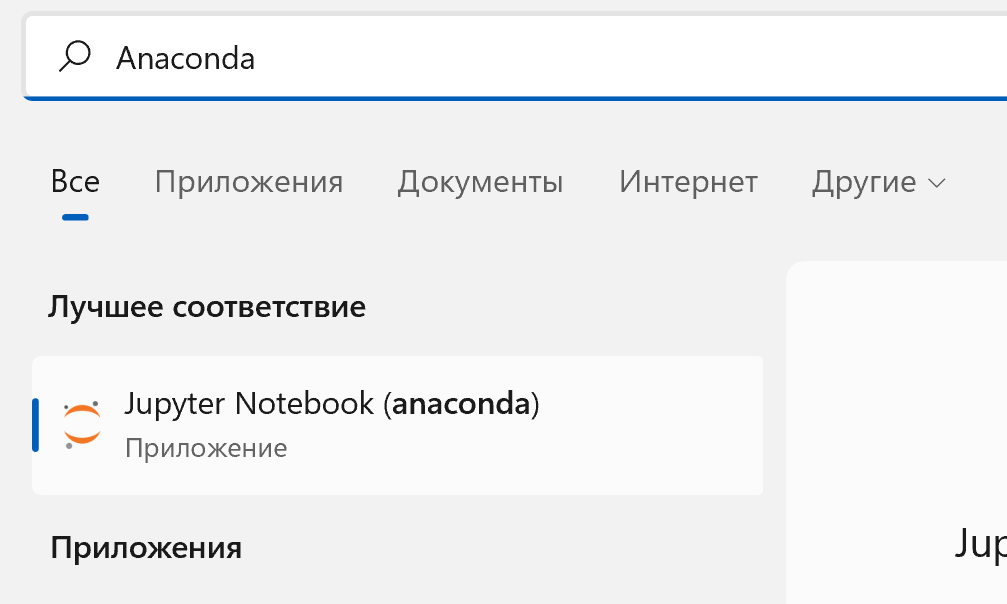


Далее предлагается внести изменения в окружение для того, чтобы Python мог по умолчанию запускаться из Anaconda. Выбрана первая галочка для того, чтобы в пункте PATH был прописан путь до расположения Anaconda. В дальнейшем это позволит писать команды в консоли Windows (cmd) команды для установки/удаления/обновления библиотек Python, например pip install/conda install

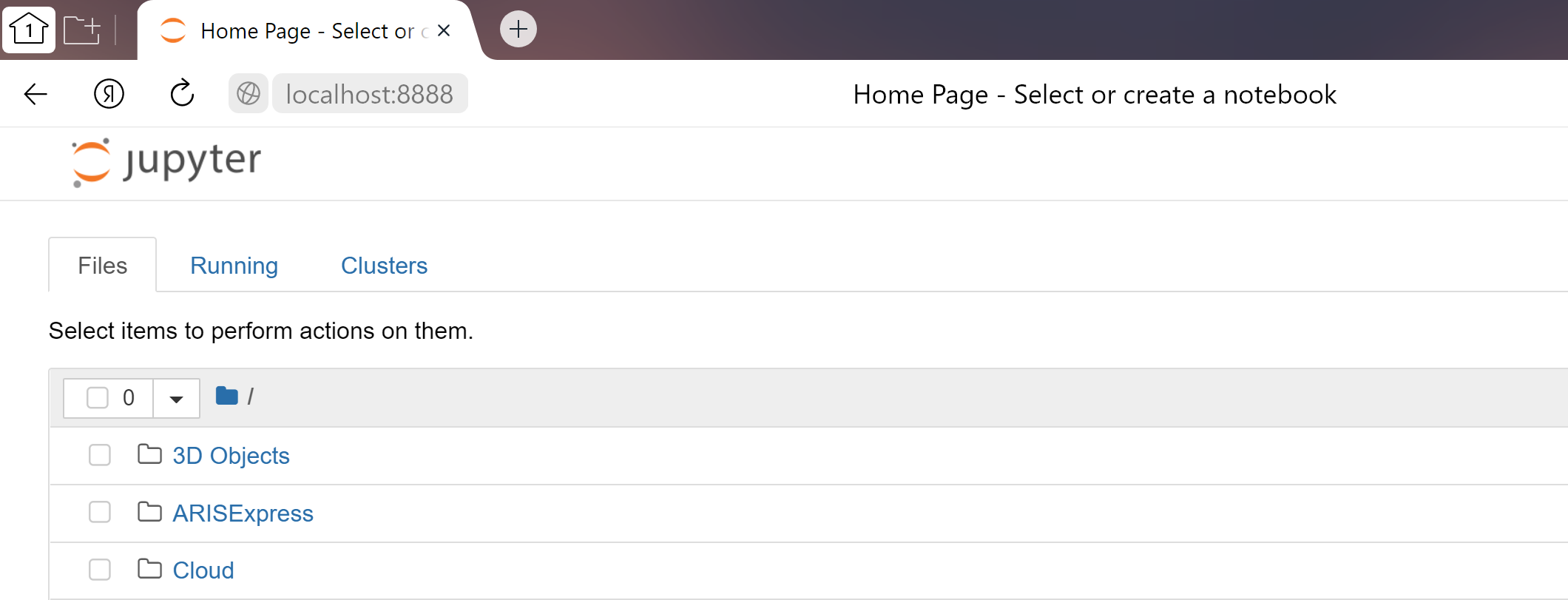




Проверим, работает ли Jupyter Notebook.



Должна открыться новая вкладка в браузере.



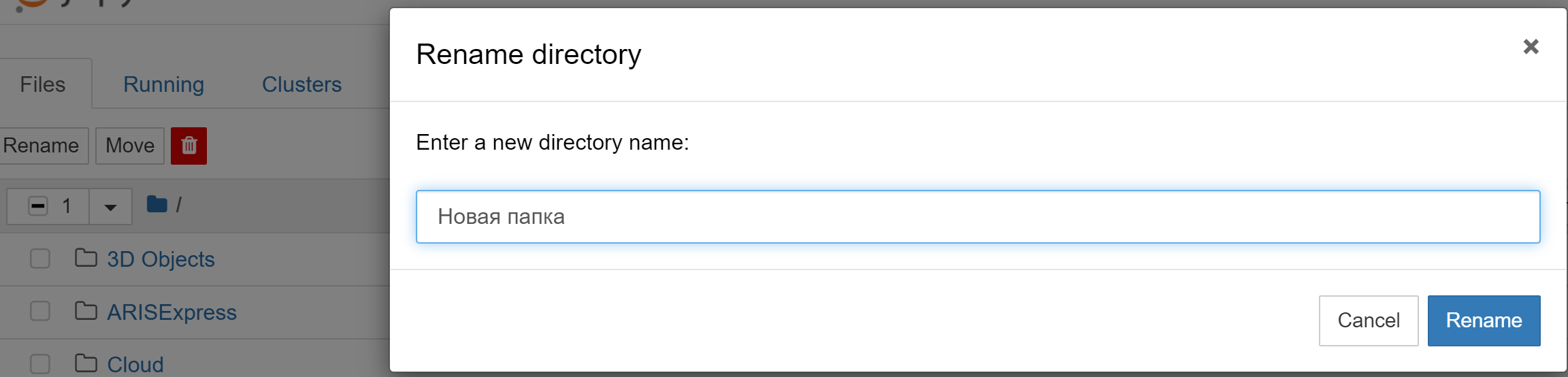
**Работа с Jupyter Notebook**

Создадим новую папку.

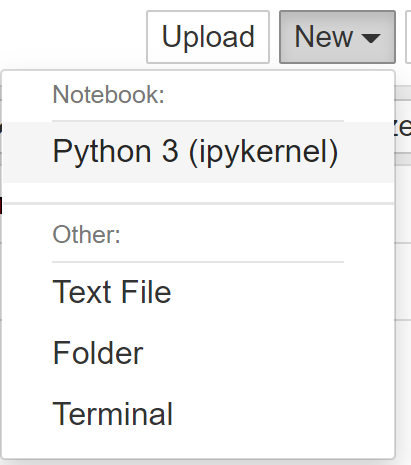


Для того, чтобы переименовать папку необходимо выбрать ее и в левом верхнем углу выбрать «Rename».

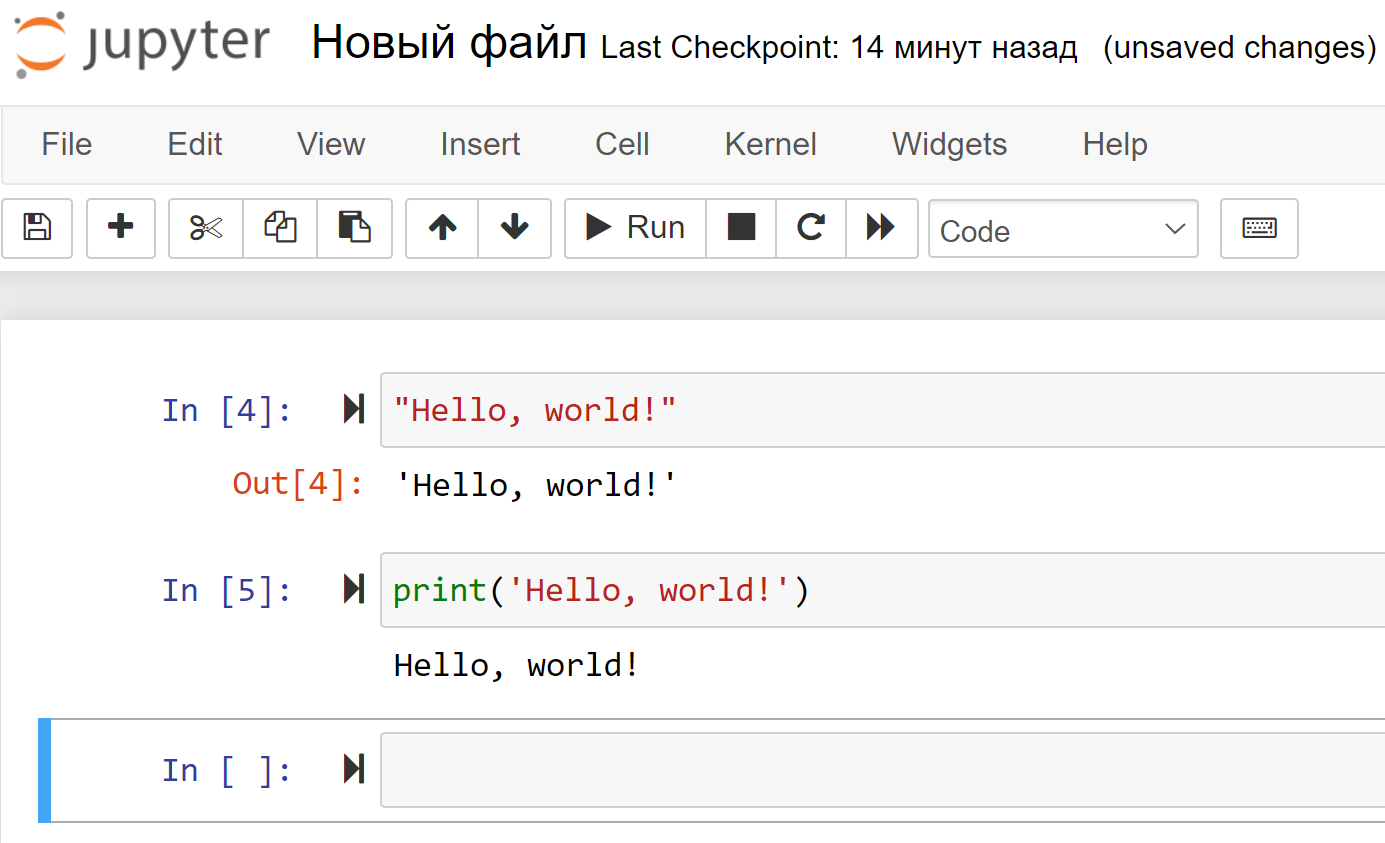




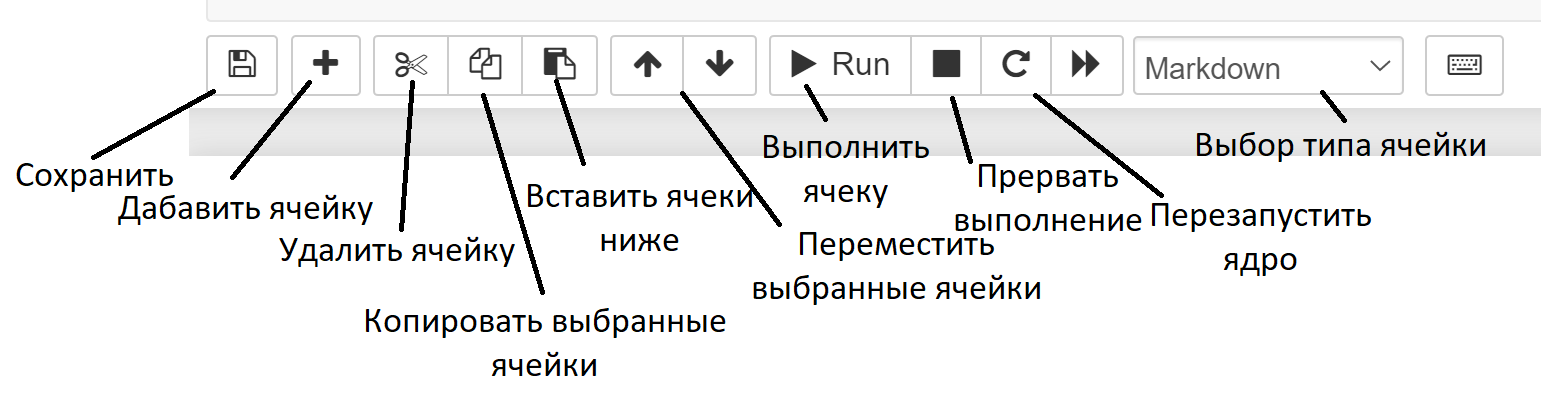
Создадим новый файл IPython. Файл с расширением ipynb.



Выведем предложение «Hello, world!»

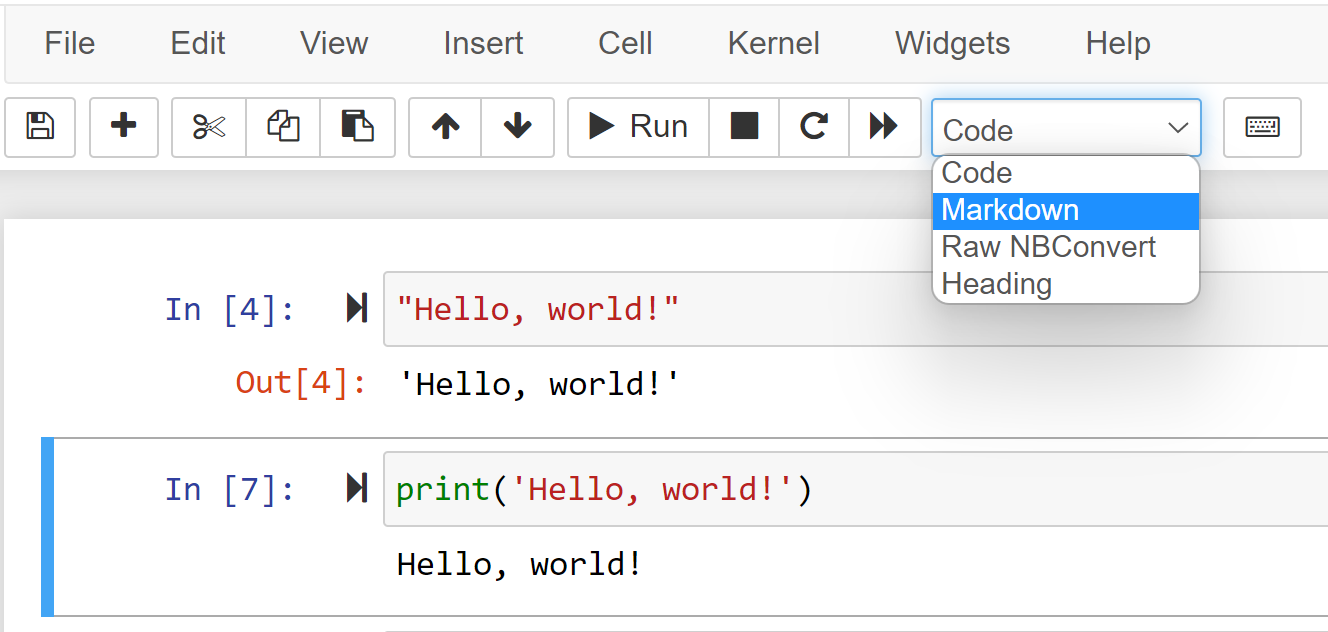


Для запуска кода в ячейке необходимо использовать комбинацию клавиш «Shift+Enter».

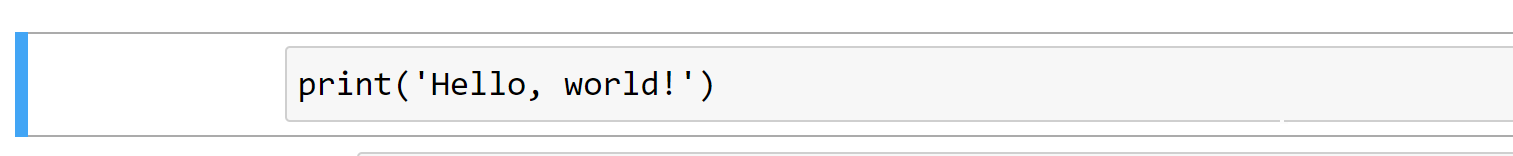


Для того, чтобы выделить несколько ячеек, необходимо с зажатым «Shift» их выбрать.

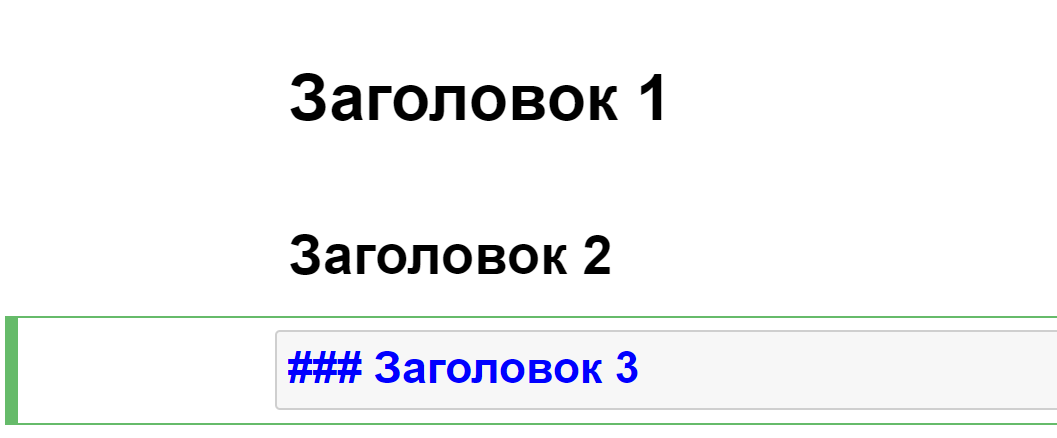
Существует два вида ячеек в Jupyter Notebook: ячейка для выполнения кода (Code) и ячейка для написания текста (Markdown). Тип ячейки всегда можно изменить, например:



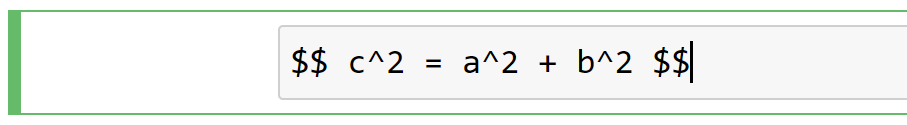
Результат:

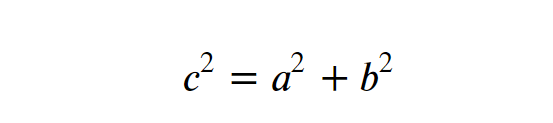


Для того, чтобы сделать заголовок в текстовой ячейке, необходимо использовать #. Чем больше #, тем меньше будет уровень заголовка. Пример:



В ячейках для текста также можно вставлять формулы с помощью знака $$, например:





**Переменные в Python**

Переменная — именованная типизированная ячейка памяти.

Список имен, которыми нельзя называть переменные, эти названия зарезервированы в Python:

● and

● assert

● break

● class

● continue

● def

● del

● except

● False

● for

● from

● global

● if / elif / else

● import

● in

● is

● lambda

● list

● None

● nonlocal

● not

● or

● pass

● raise

● return

● True

● try / finally

● while

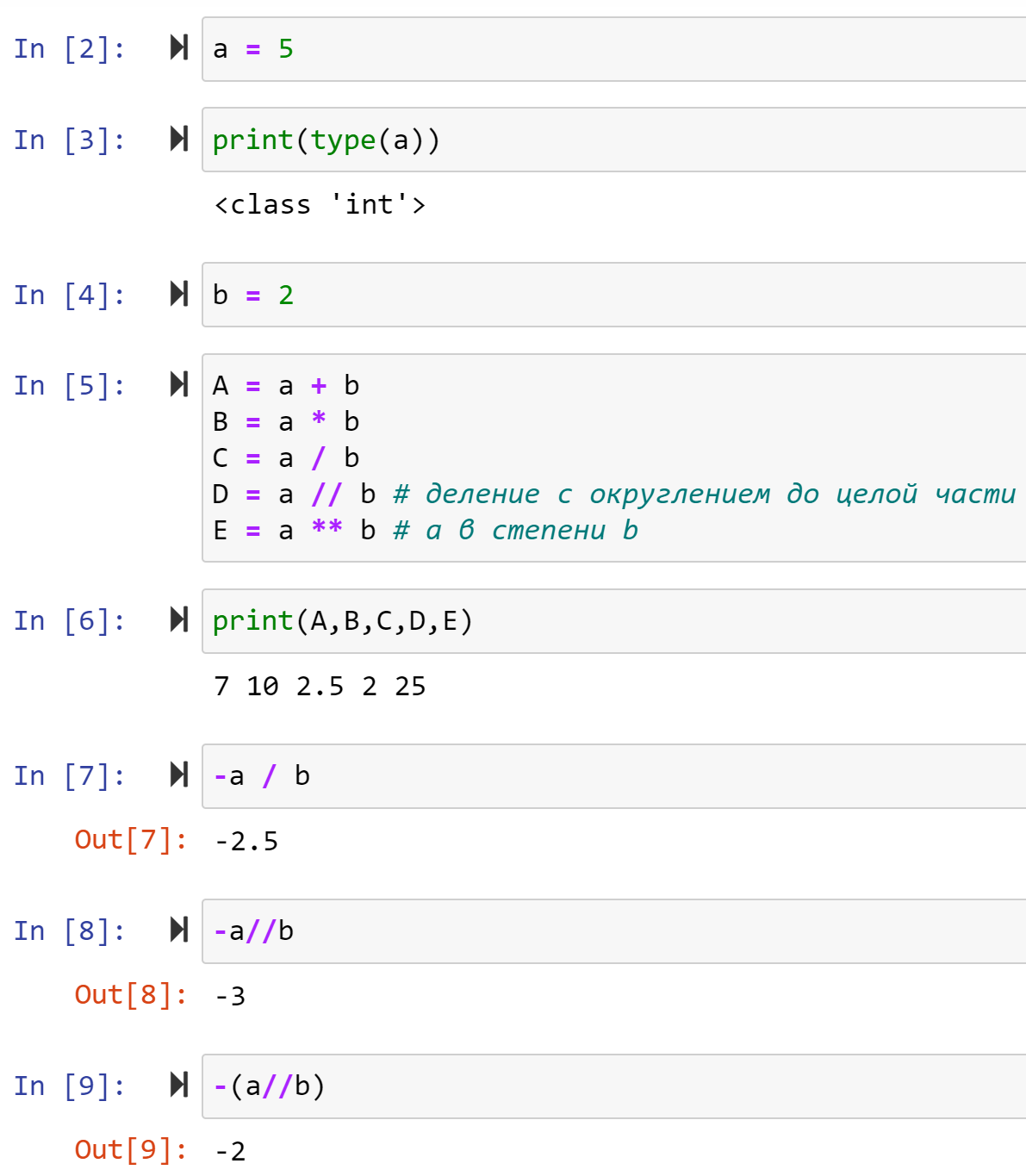
● with / as

● yield

**Типы переменных Python**

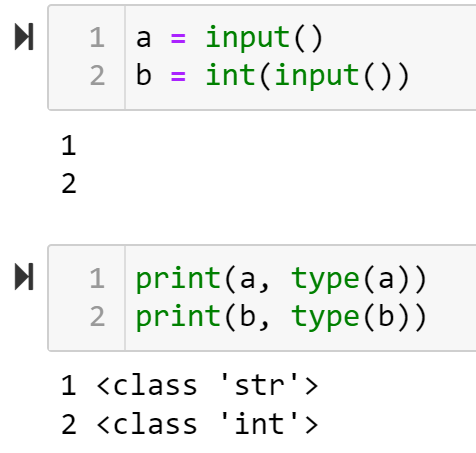
В Python 3 есть следующие базовые типы переменных: bool, str, float, int, complex.

*Целочисленные переменные (int) и основные операции с ними:*



При целочисленном делении отрицательного целого числа -5 на 2 получаем -3, идет округление в меньшую сторону. Чтобы получить такой же результат, как и в случае с положительными числами, необходимо сначала поделить положительные числа, а затем добавить минус (In [9]).

Метод input() позволяет ввести данные от пользователя, пример:

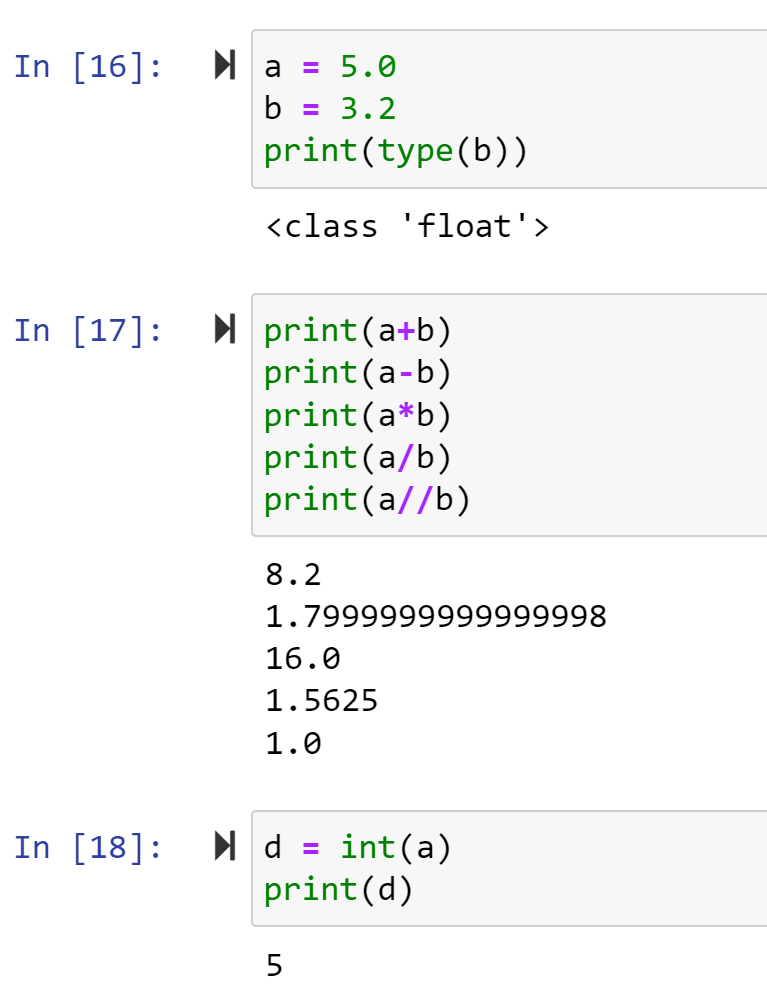


Если нет явного указания типа данных ввода, то по умолчанию – это str.

*Комплексные переменные (complex):*

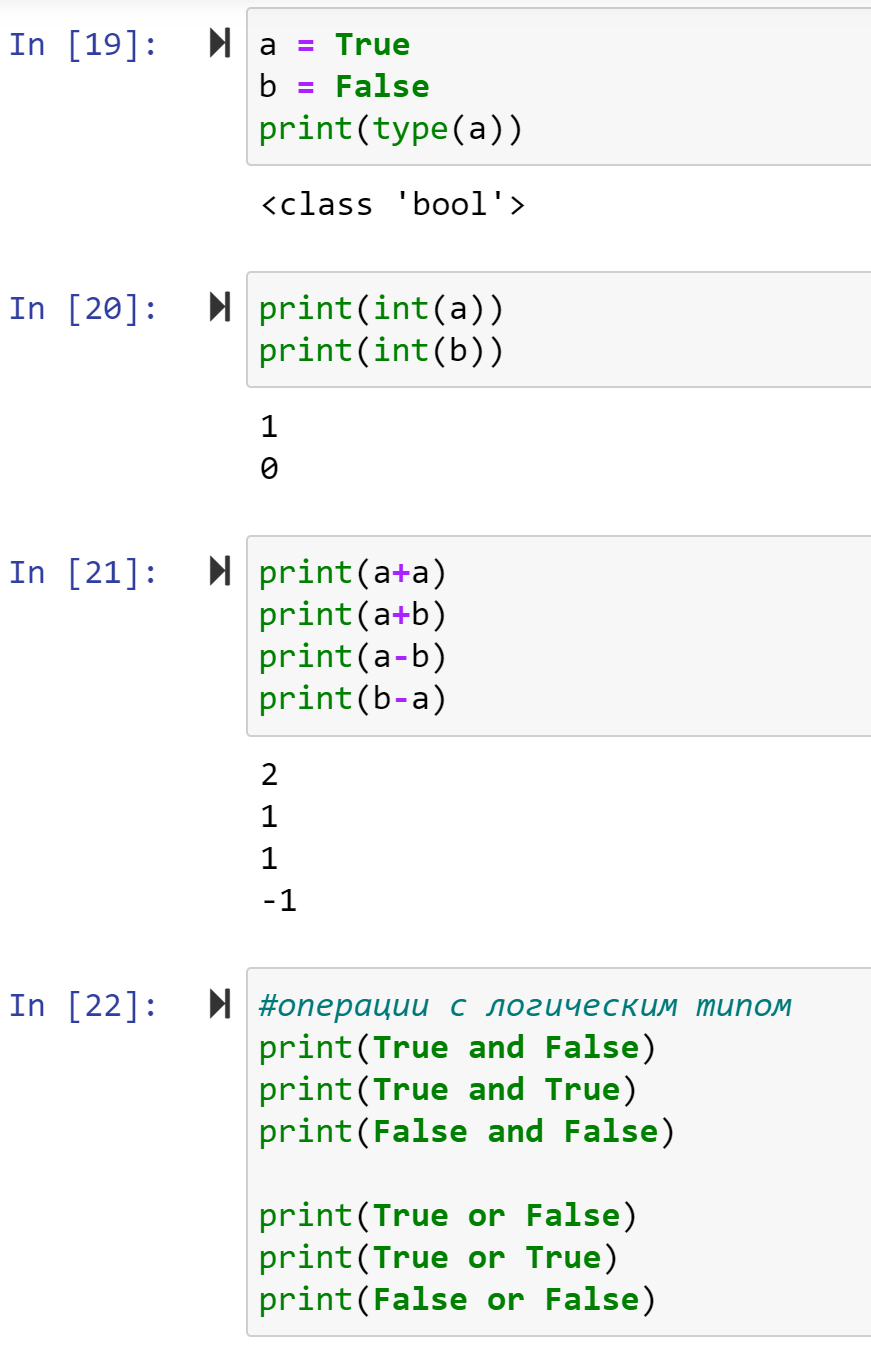


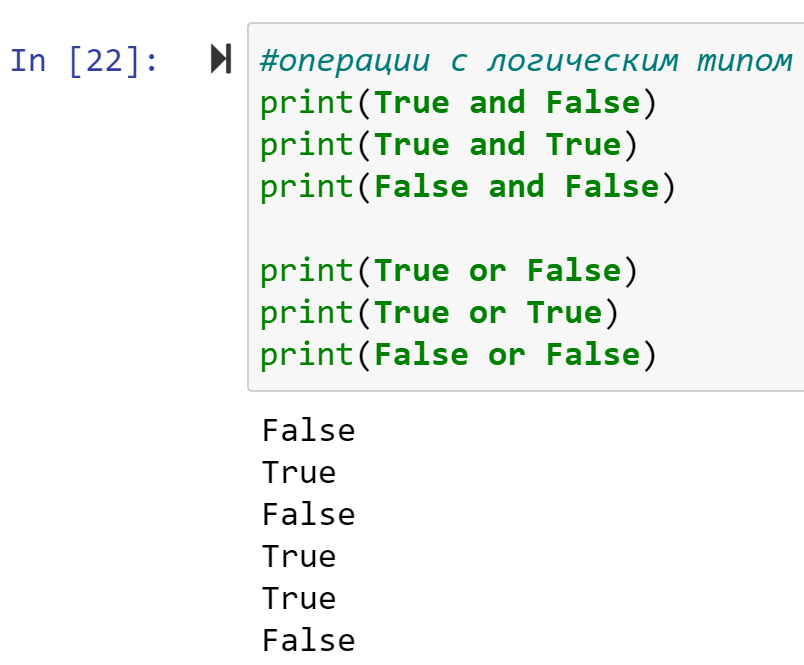
*Вещественные числа (float):*

**

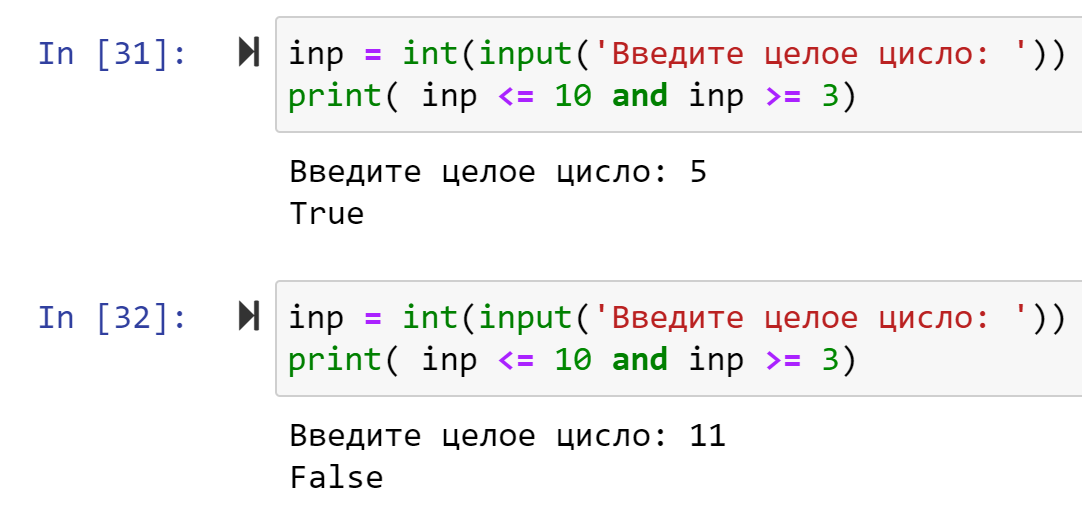
*Логический тип (bool):*

Логический тип (True или False) можно привести к целочисленному типу.





Проверка числа, подающегося на вход:

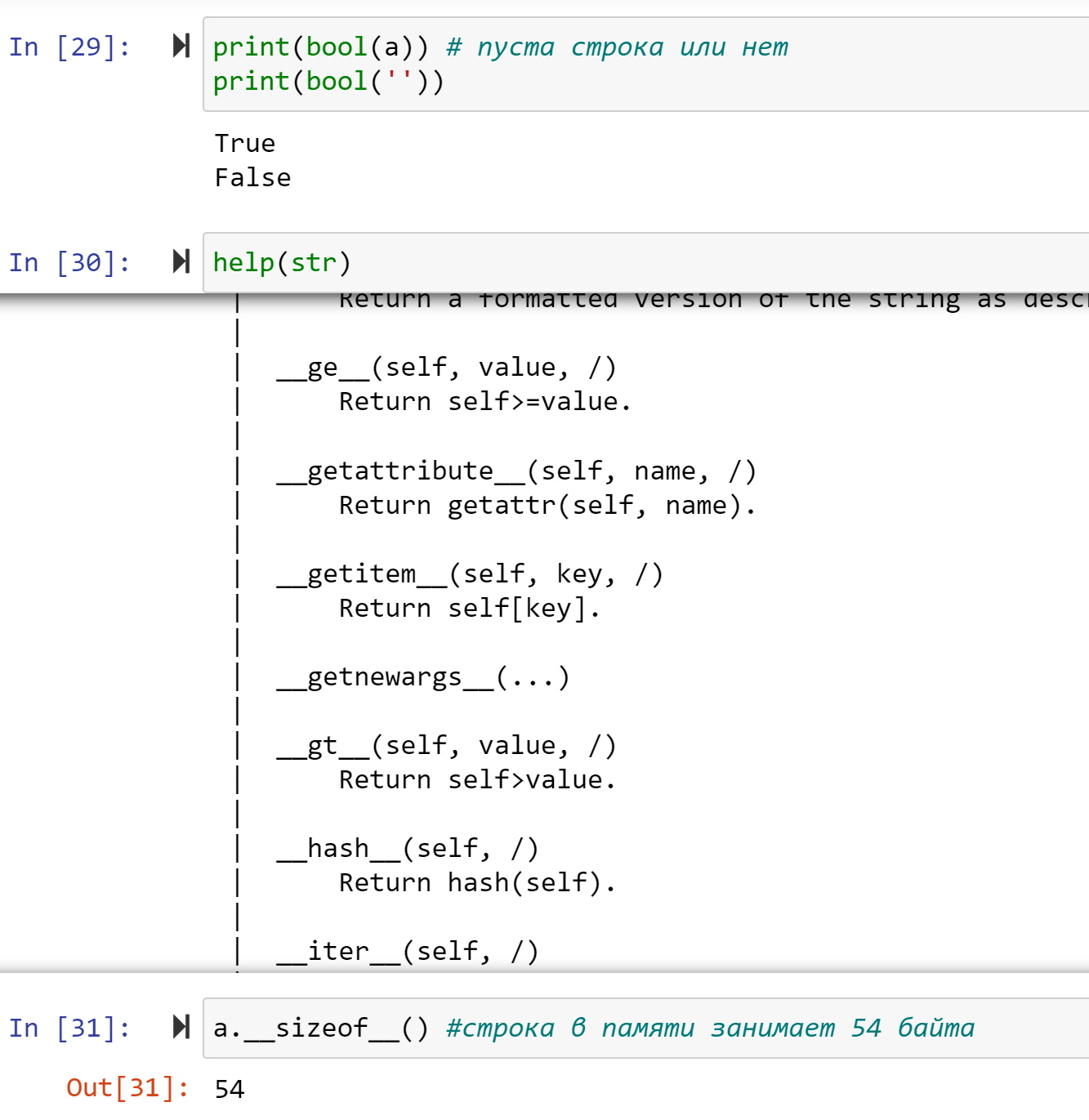


Пустая строка определяется нулевой длиной, а при преобразовании к логическому типу (bool) пустая строка превращается в False. Если строка имеет длину, отличную от нуля, то при преобразовании к логическому типу она превращается в True.

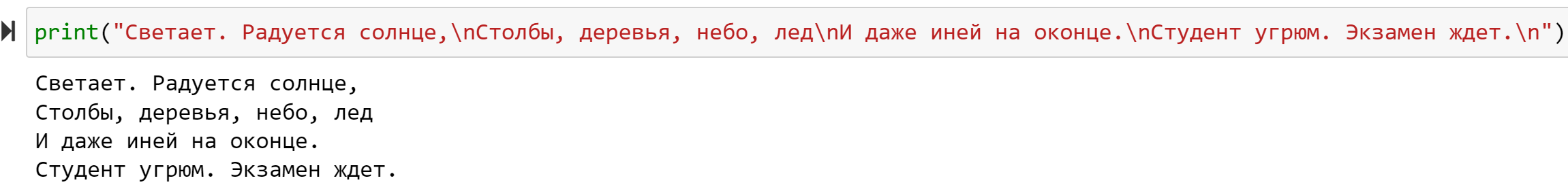
*Строковый тип (str):*

В Python строка – это последовательность символов, фиксация которой начинается с 0. Строки являются неизменяемыми.

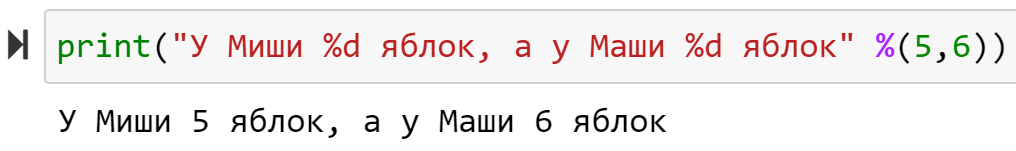




\n – символ перевода на новую строку.



Python поддерживает форматированный вывод строк с помощью оператора форматирования %. Например:



* Явное преобразование типов — преобразование через вызов функций типов str(), int(), float().
* Неявное преобразование типов — преобразование во время исполнения кода в зависимости от контекста.

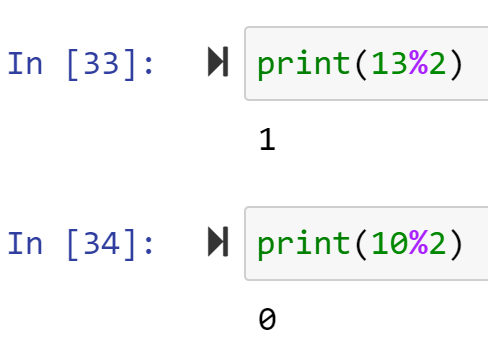
Чтобы понимать, в каком порядке выполняются операторы, нужно знать приоритеты выполнения операторов. Они перечислены в таблице ниже, где 1 — наибольший приоритет, 4 — наименьший приоритет.

|  |  |
| --- | --- |
| **Оператор** | **Приоритет** |
| **x\*\*y** | 1 |
| **-x** | 2 |
| **x\*y** | 3 |
| **x/y** | 3 |
| **x%y** | 3 |
| **x//y** | 3 |
| **x+y** | 4 |
| **x-y** | 4 |

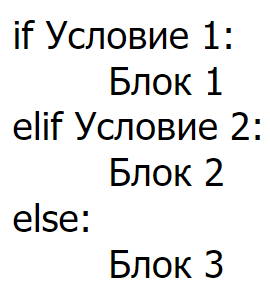
Если выражение содержит несколько операторов с разными приоритетами, то сначала будут выполняться операторы с более высоким приоритетом.

*Условный оператор (if, else, elif):*

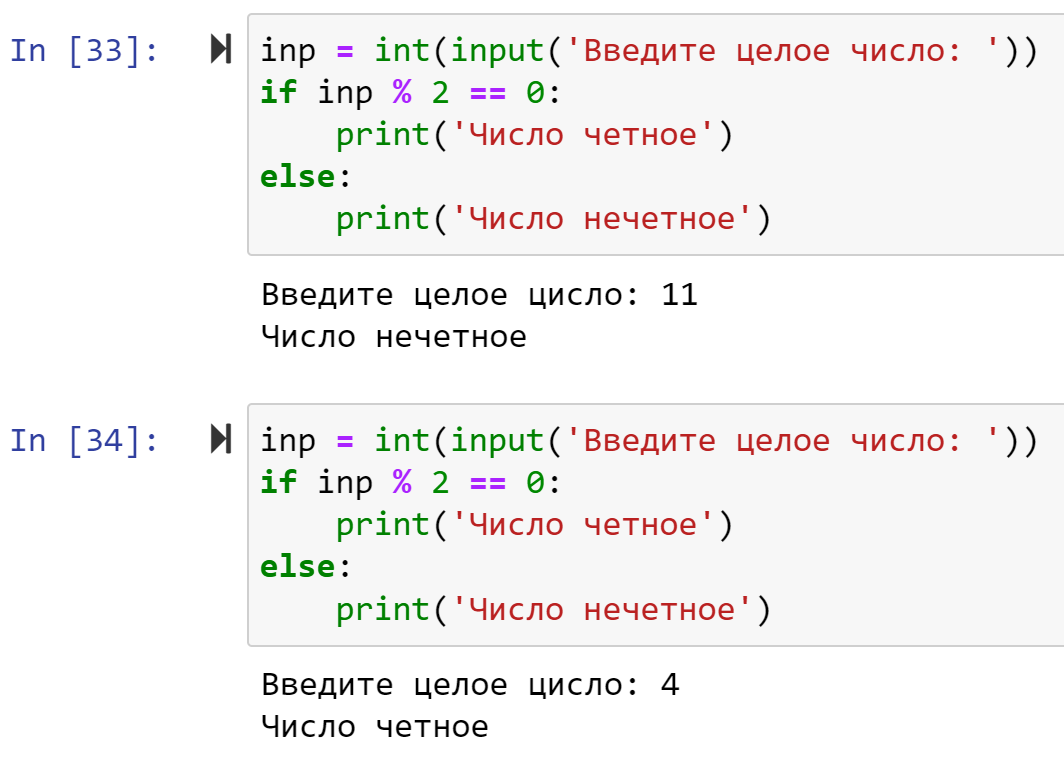
Классическая задача – проверка числа на четность. Для этого используется знак %, который позволяет посчитать остаток от деления, например:



Пример реализации условия в Python. В начале пишется ключевое слово if, затем блок кода, который должен выполниться при выполненном условии. Затем, если условие не выполняется, но есть еще одно условие, то пишется ключевое слово elif, после чего снова идет блок кода. Если же ни одно из условий не выполняется, то пишется ключевое слово else и соответствующий ему блок кода. В Python блоки кода выделяются отступом.



Пример:

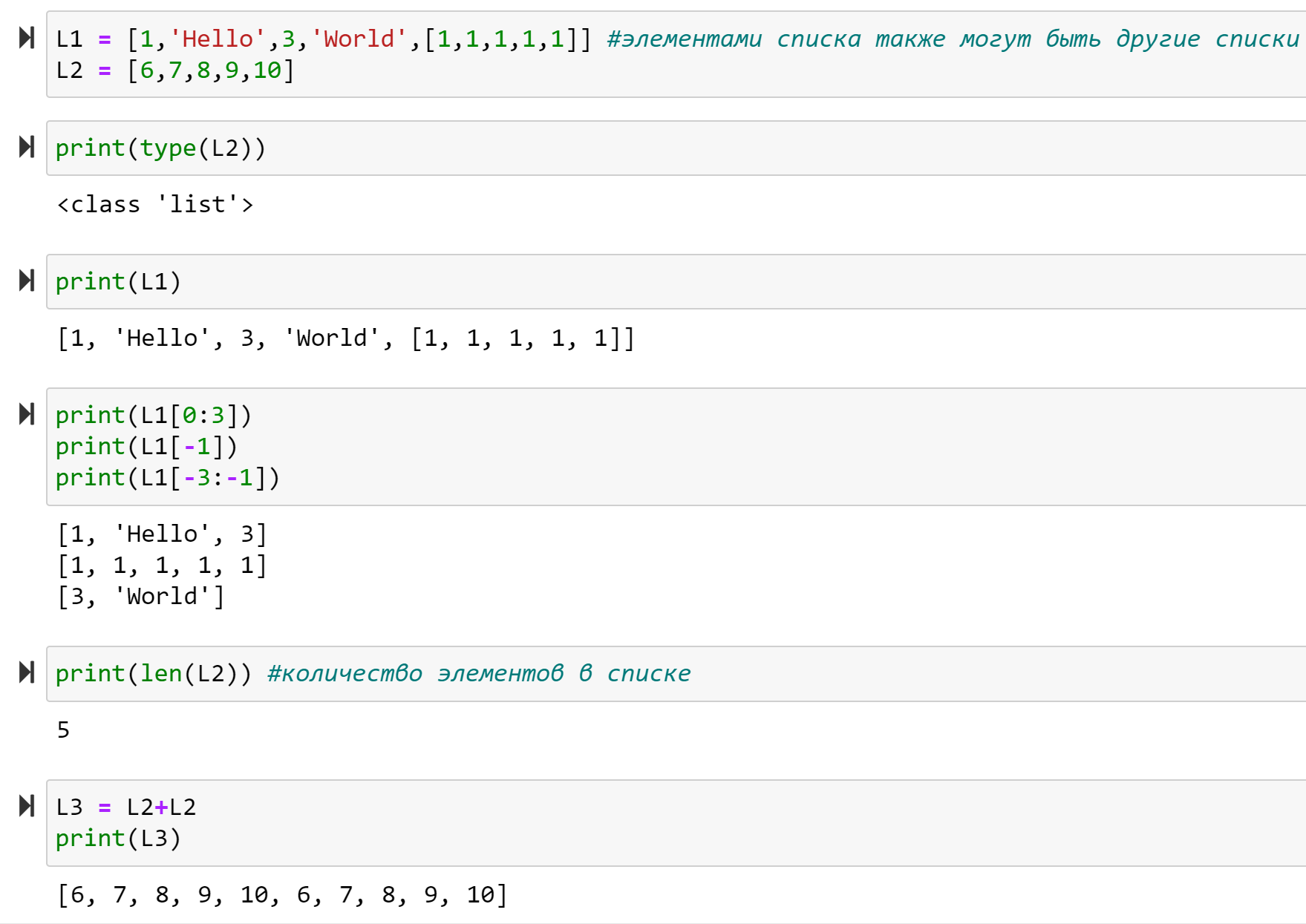


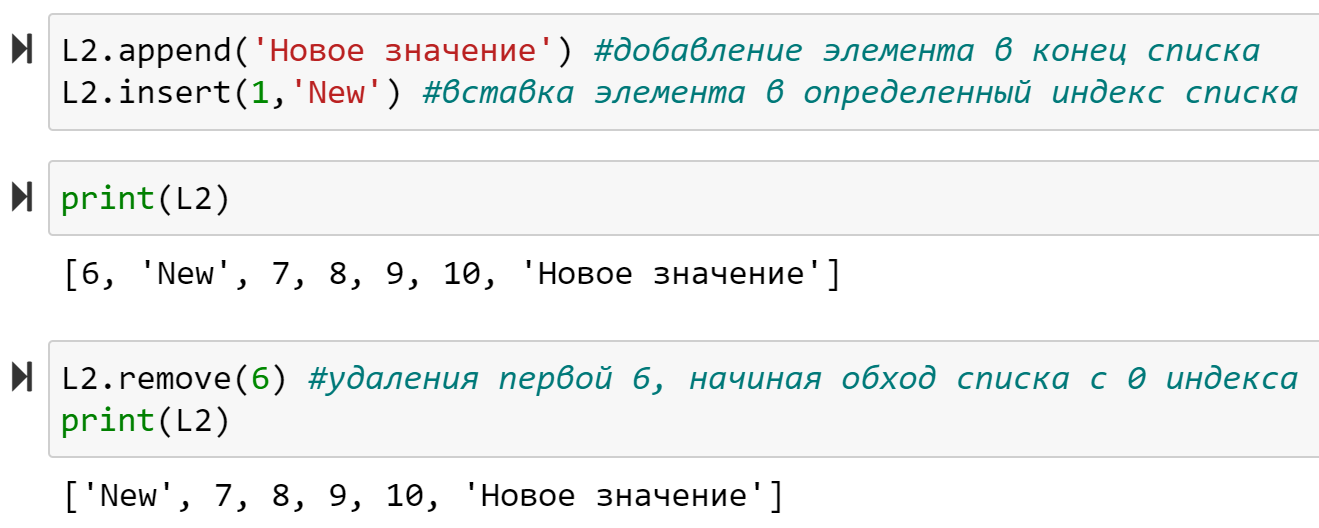
**Структуры данных Python**

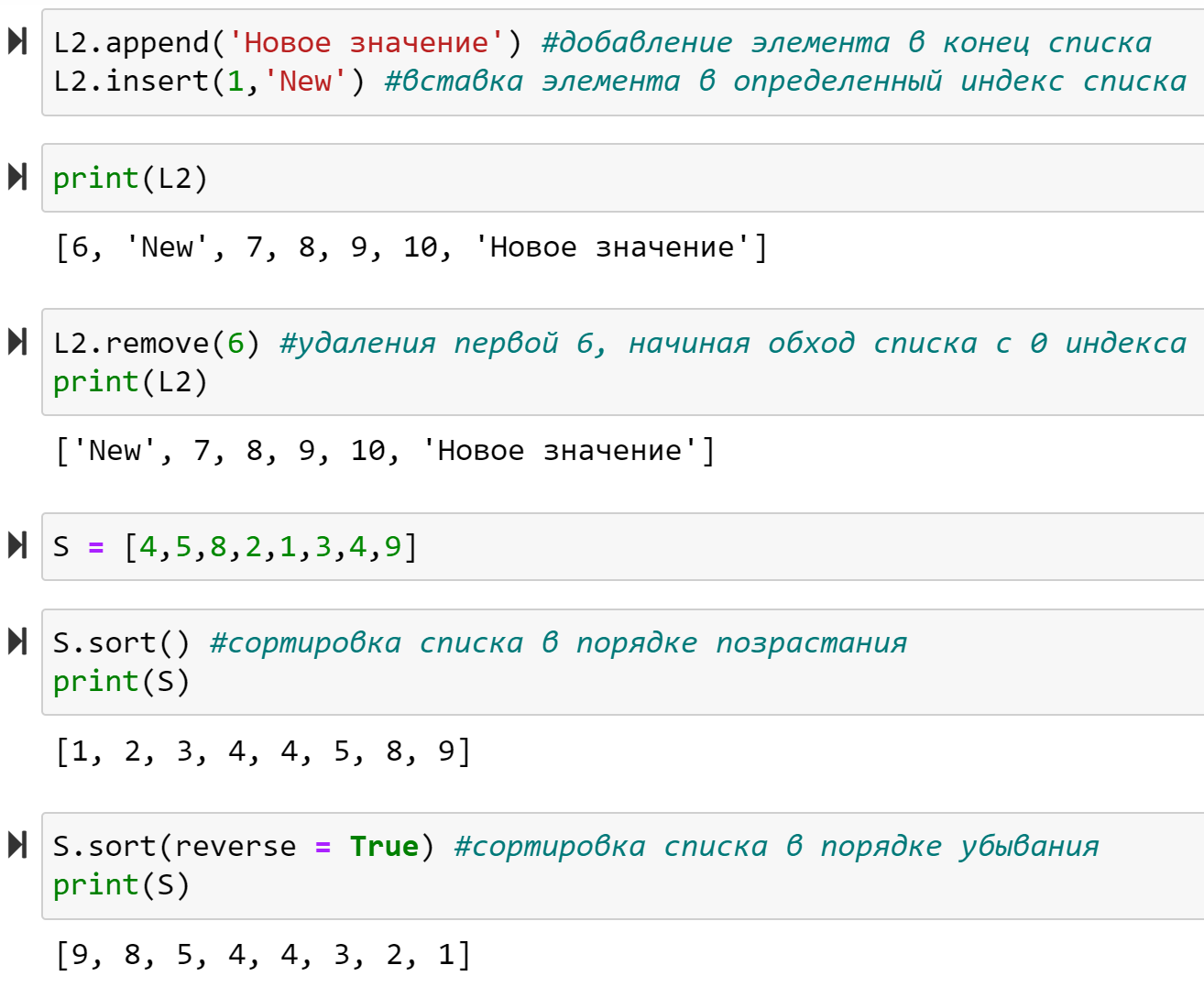
В Python все структуры данных являются объектами. У них есть методы и поля. Структуры данных Python: список (list), кортеж (tuple), множество (set), словарь (dict).

*Список (list).*

Список – это изменяемая структура данных. Может хранить в себе элементы разных типов. Обращение к элементу списка происходит с помощью [индекса], начиная с 0. Чтобы обратиться к элементу с конца списка, то используется -1, -2 и т.д. Объявление списка происходит с помощью [] или list(). Пример:

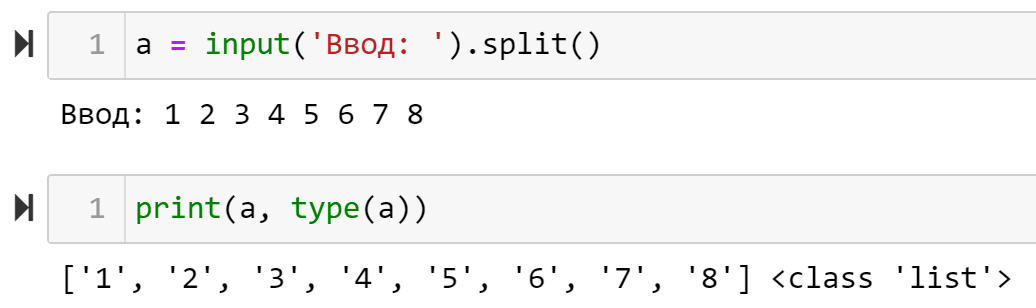








Можно получить список с помощью метода split() при вводе от пользователя, например:

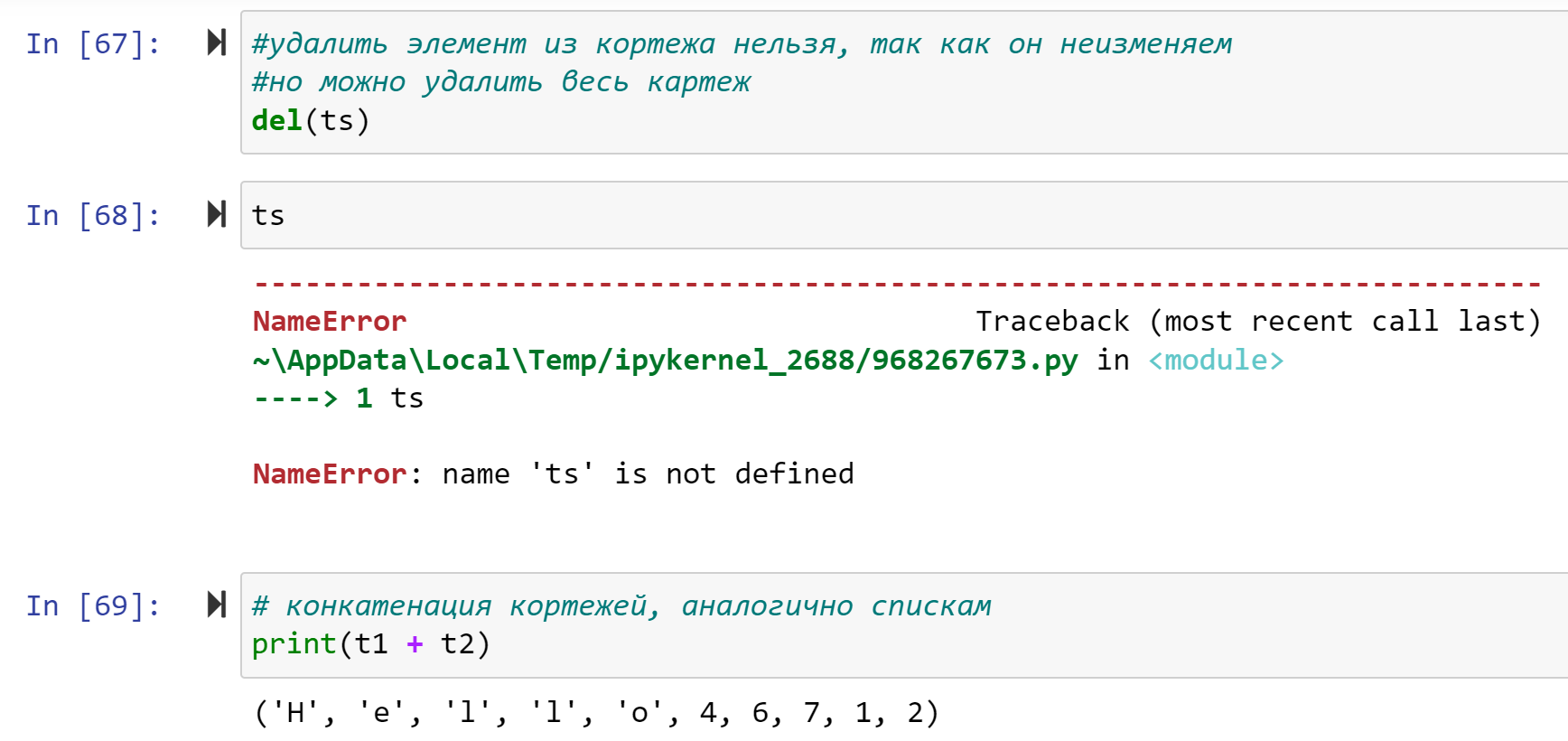


*Кортеж (tuple).*

Кортеж – неизменяемая структура данных. Аналогично списку, может хранить элементы разных типов. Кортеж объявляется через () или tuple().

Кортежи работают быстрее списков за счет своей неизменяемости. Они хранятся в памяти особым образом, поэтому операции с элементами кортежа выполняются быстрее. Кортежи используются, когда есть требования к безопасности.







*Множество (set).*

Множество – изменяемая структура данных, которая имеет требование к уникальности элементов. Множество удобно использовать для удаления повторяющихся элементов.

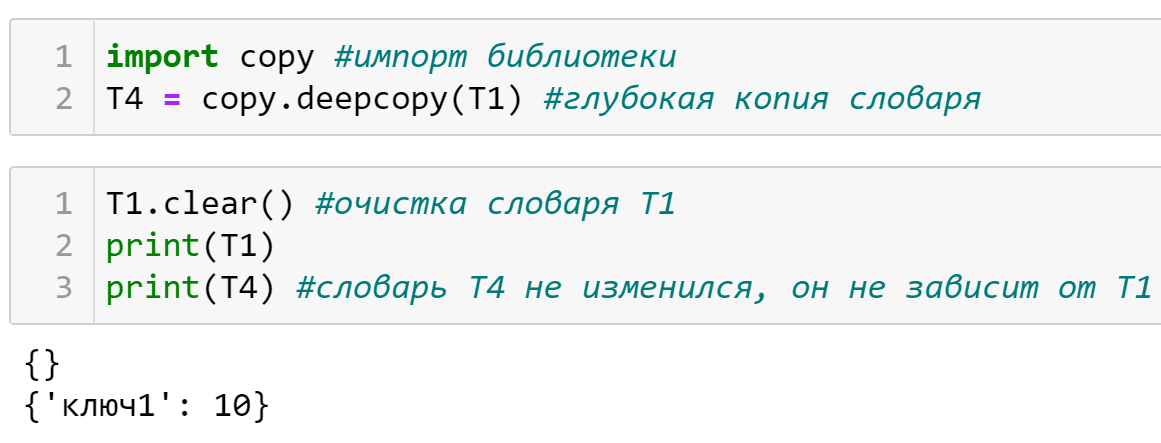


*Словарь (dict).*

Словарь – ассоциативный массив, который хранит пары ключ:значение. Объявляется через {} при указании {ключ1:значение1, ключ2:значение2}. Ключ – уникальный и неизменяемый, поэтому tuple может использоваться в качестве ключа.



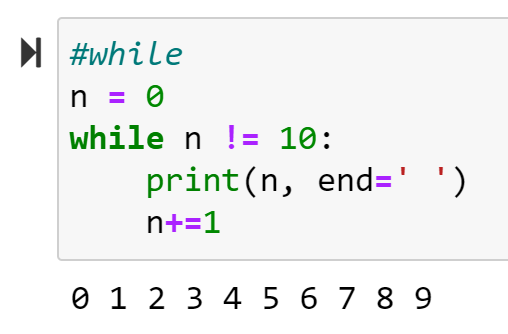
****

****

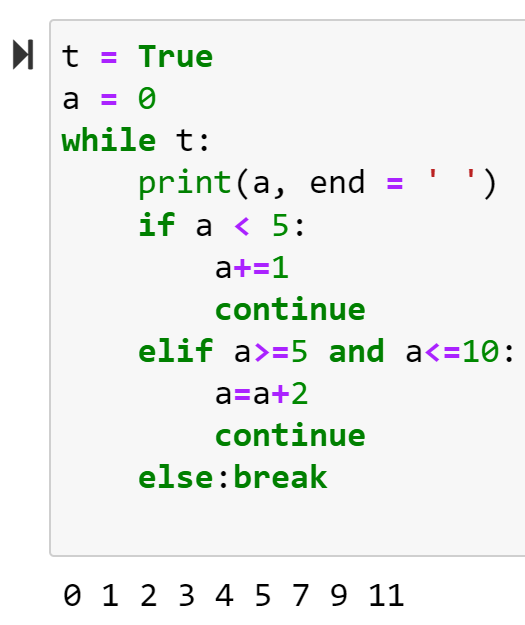
**Циклы, функции Python и работа с библиотеками**

*Цикл while*

Циклический оператор while используется достаточно редко, но все же бывает полезен. Определенный блок кода выполняется, пока выполняется заданное условие. Пример:

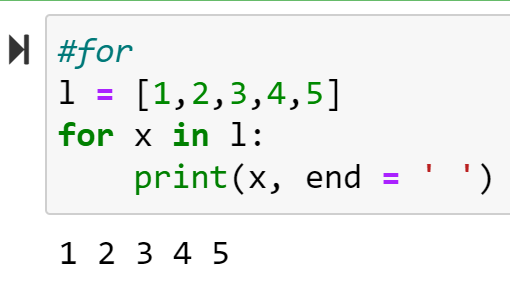


end в выводе позволяет указать, какой знак будет стоять после вывода, по умолчанию – это \n (переход на новую строку). У циклов есть continue и break. Если в цикле используется continue, то цикл будет переходить сразу к следующей итерации, не выполняя код ниже. В случае break – происходит полное прерывание цикла. Пример:

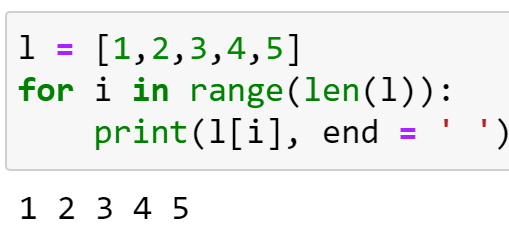


*Цикл for*

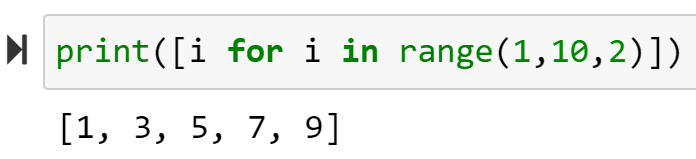
Цикл с использованием ключевого слова for – это прохождение по элементам некой последовательности. Например, необходимо вывести элементы, хранящиеся в списке:



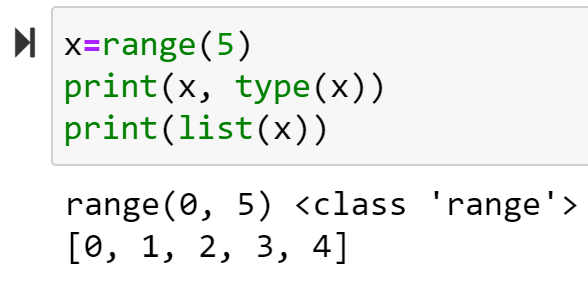
Второй способ – это вывод элементов списка по их индексу:



range – это генератор чисел, в данном случае он генерирует индексы от 0 до длины списка-1, то есть 0,1,2,3,4, затем вывод происходит через обращение к элементу списка по индексу. Примеры использования генератора:



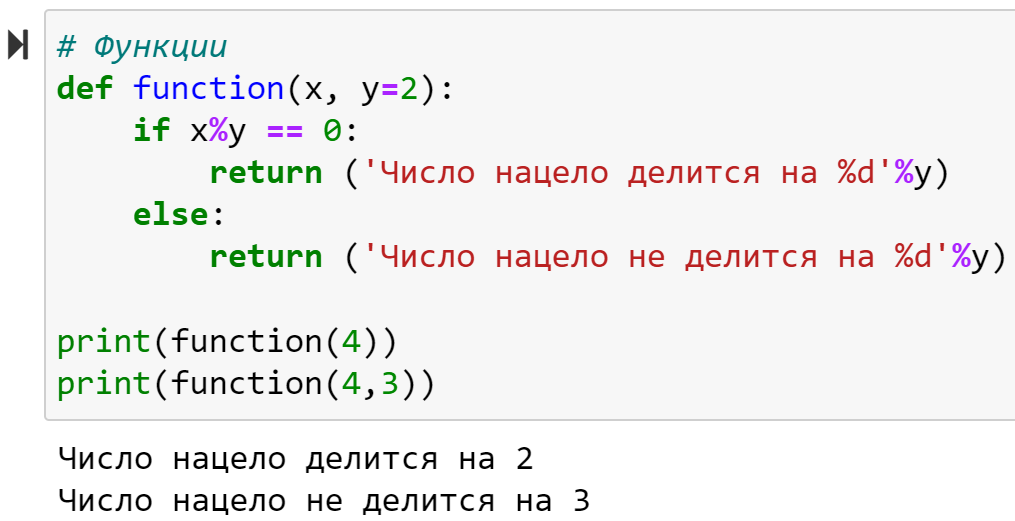
range(start, stop, step). start – начало, stop – конечное значение (обратите внимание, что stop не входит в интервал генерации чисел), step – шаг.



*Функции*

Функция – блок кода, который выполняется при его вызове в ходе выполнения программы.

Функция в Python начинается с ключевого слова def, далее идет имя функции и (). В () указываются аргументы функции. Сначала идут аргументы, который не даны по умолчанию, затем аргументы, которые даны по умолчанию. Тело функции выделяется отступом. Пример:

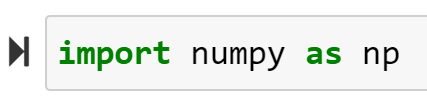


Ключевое слово return – это результат, который возвращает функция.

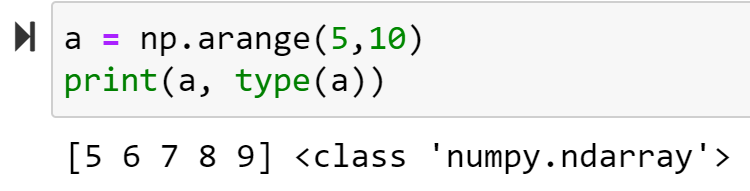
*Библиотека numpy*

numpy это библиотека для работы с матрицами и векторами.

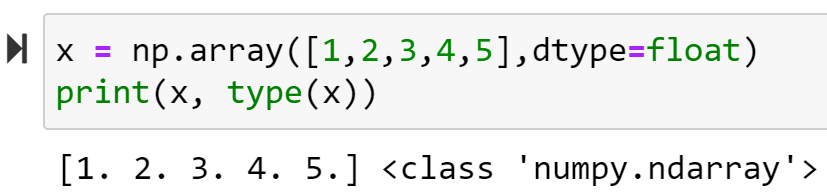
Для работы с библиотекой ее необходимо импортировать с помощью команды import.



Генерация массива в numpy:

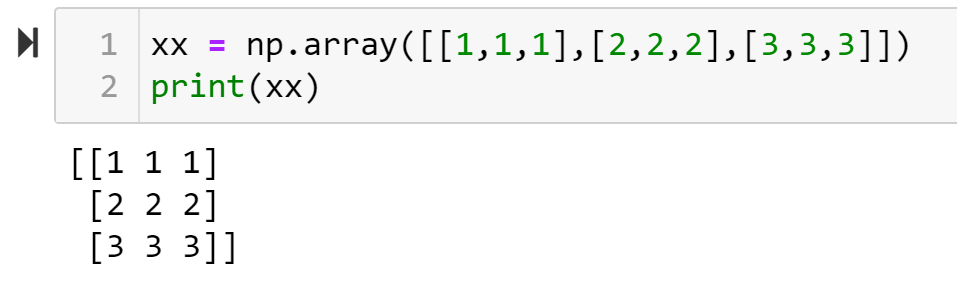


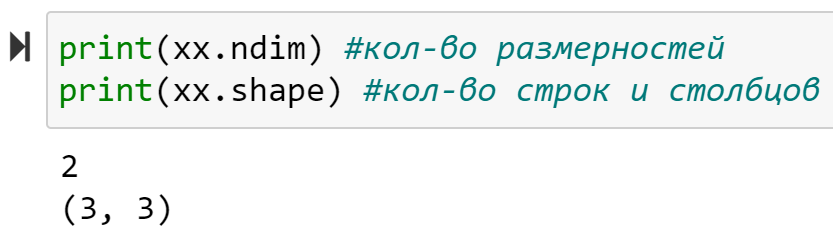
Можно привести список к numpy массиву:



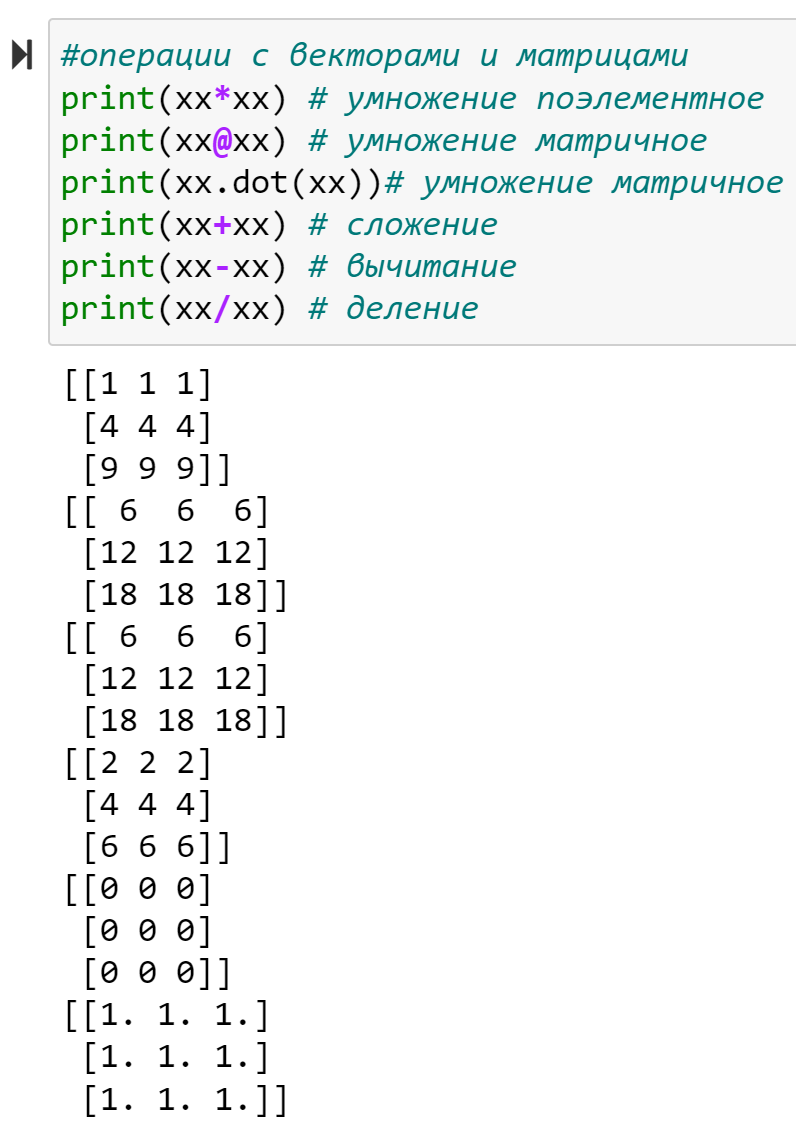
dtype позволяет указать тип хранимых элементов, в данном случае – это вещественные числа.

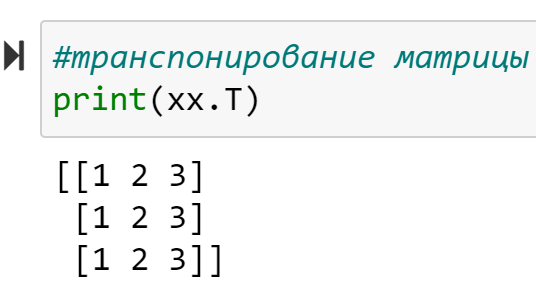
Двумерные (и не только) массивы:



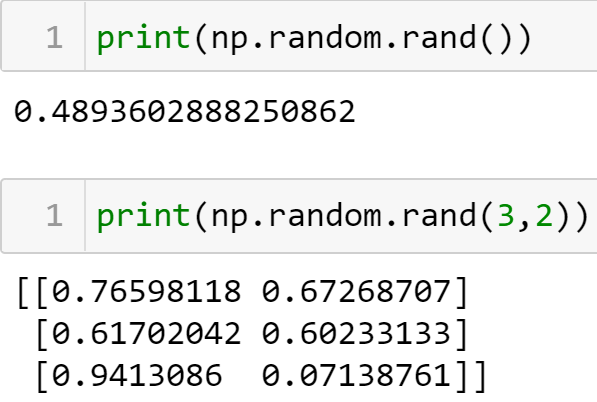








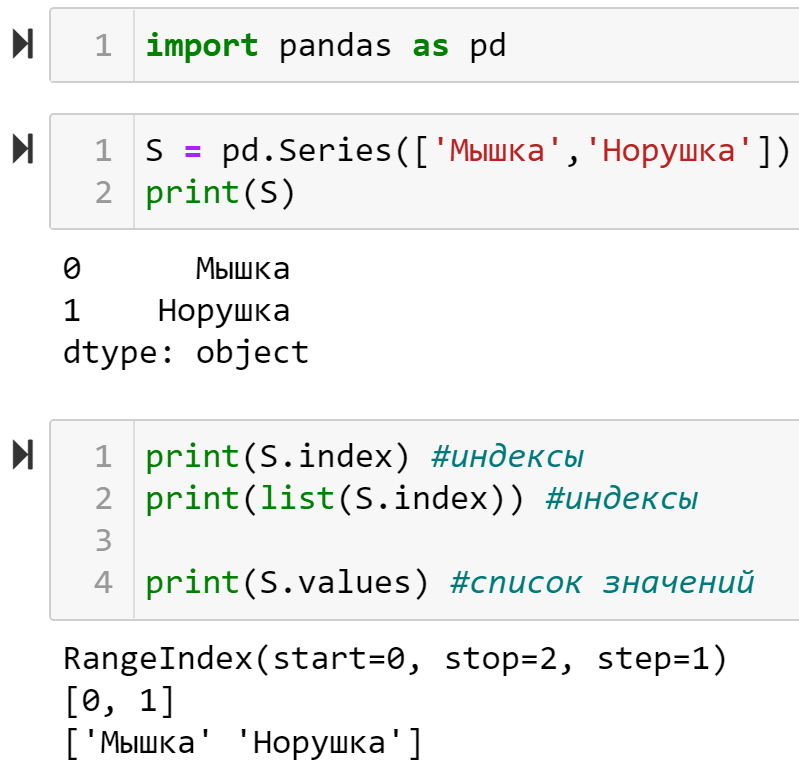
np.random.rand() – генерация случайных значений из равномерного распределения от 0 до 1.

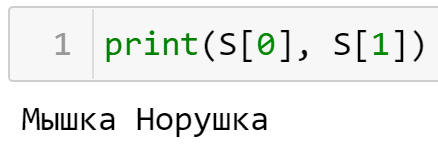


np.random.rand() генерация случайных значений из нормального распределения с дисперсией, равной 1, и средним, равным 0.

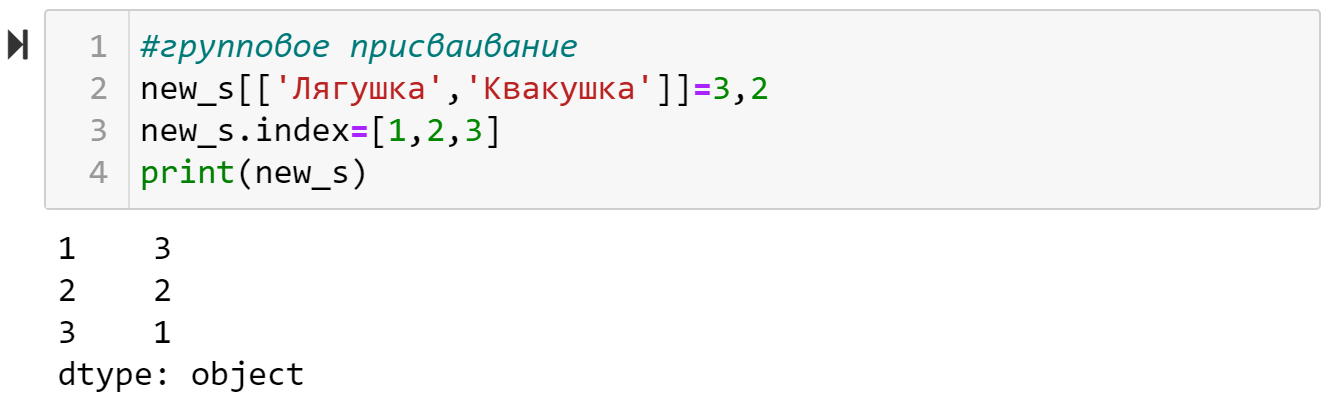
*Библиотека pandas*

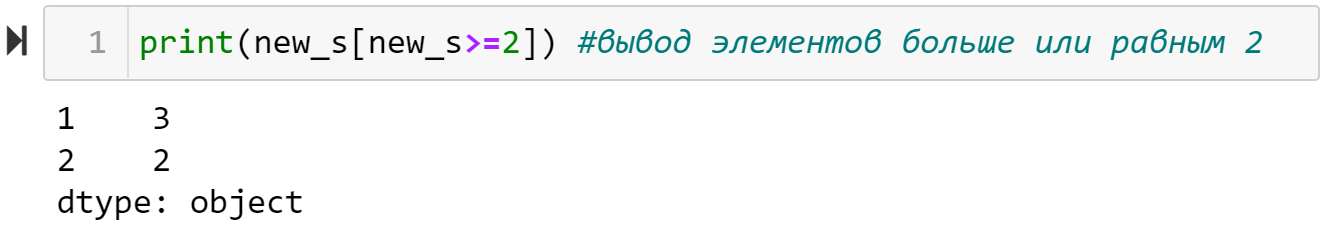
Pandas – это библиотека для обработки и анализа данных. В pandas есть две структуры объекта: DataFrame и Series. Series – это аналог одномерного массива, который имеет индексы, по сути, это ассоциативный массив, как словарь. DataFrame имеет табличную структуру, столбцы у DataFrame – это Series.





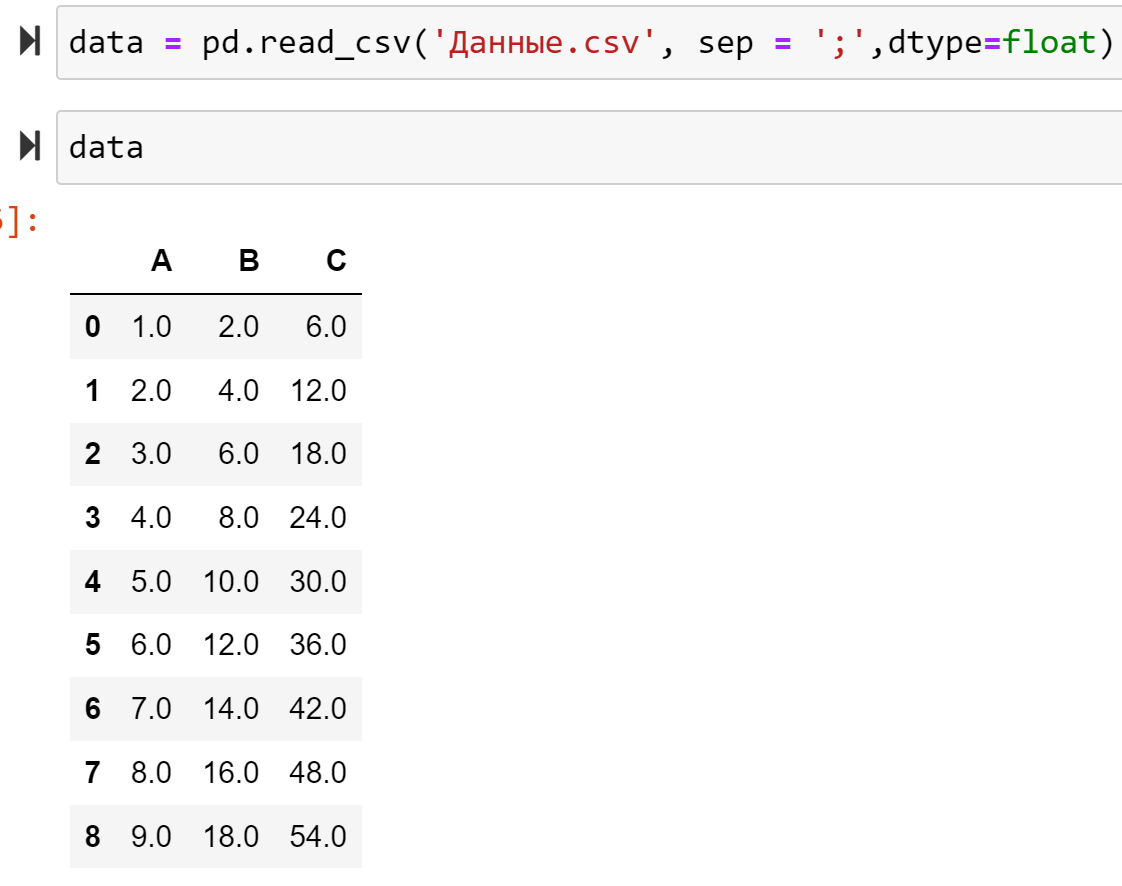






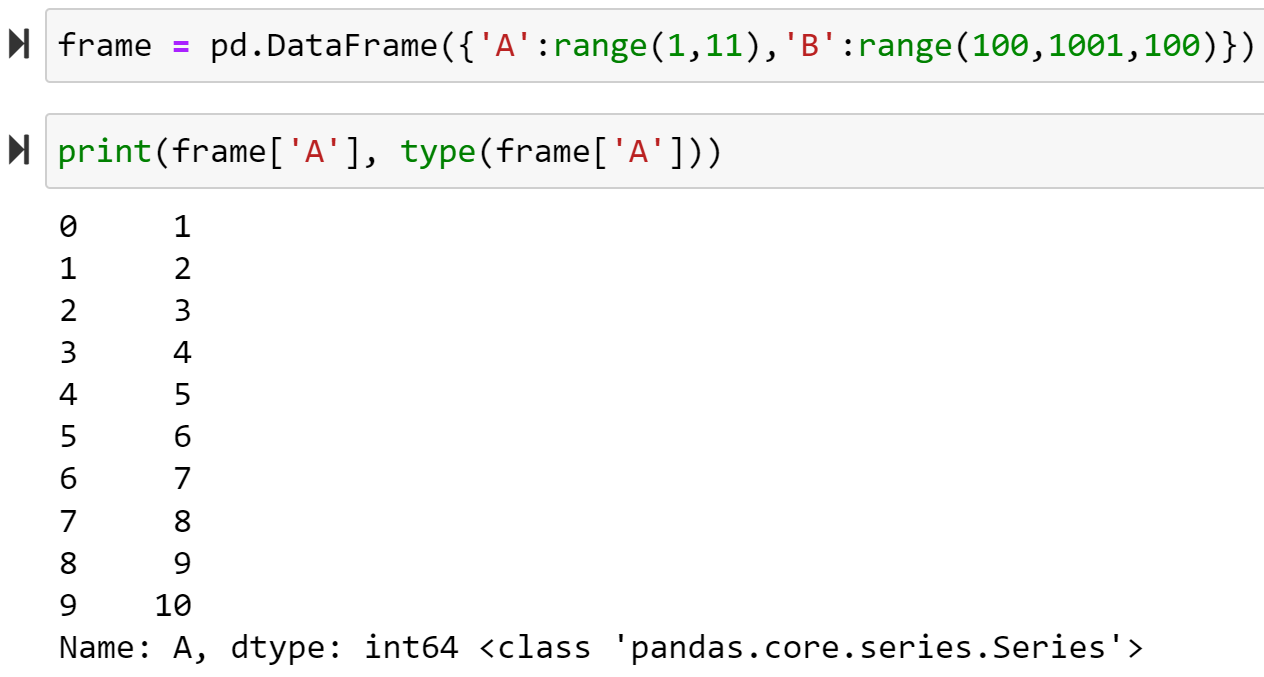
**DataFrame**

Загрузка данных из файла .csv в DataFrame.

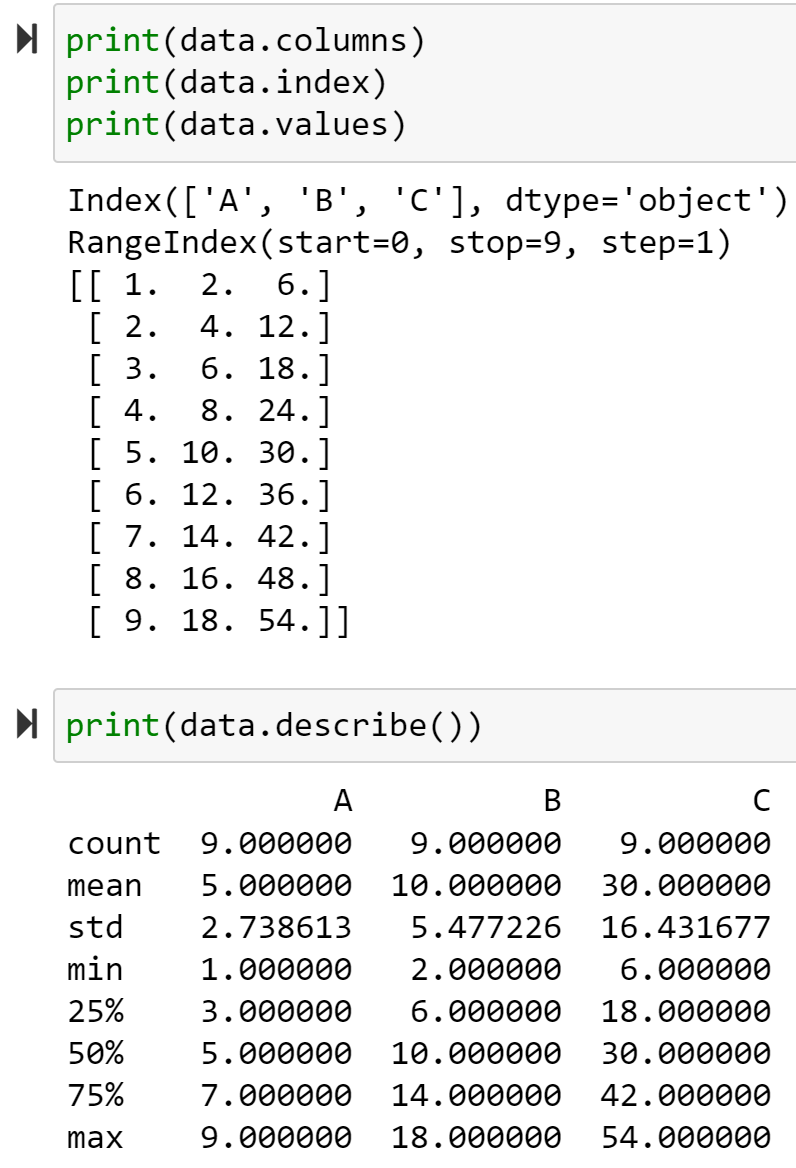


Аргумент sep позволяет указать разделитель. Также доступно чтение из txt (read\_txt) файла и xlsx файла (read\_excel).

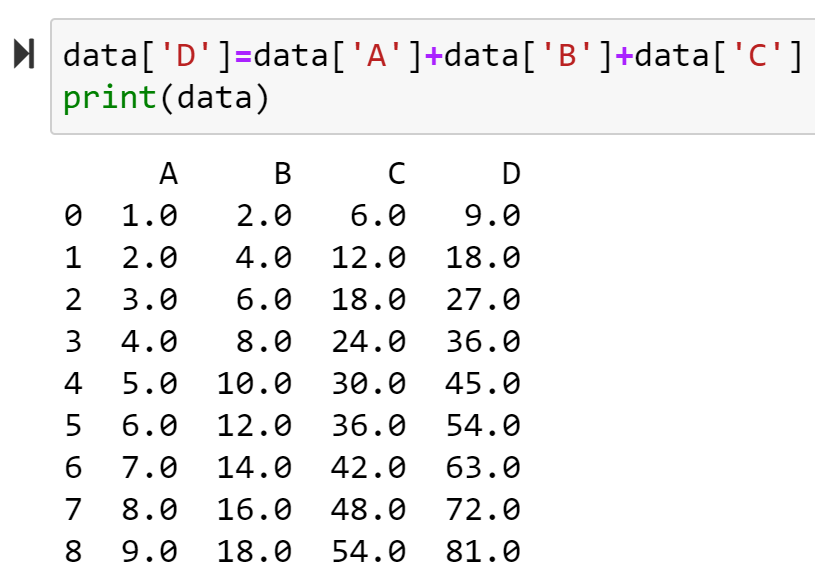
Задать столбцы и значения можно вручную:

****

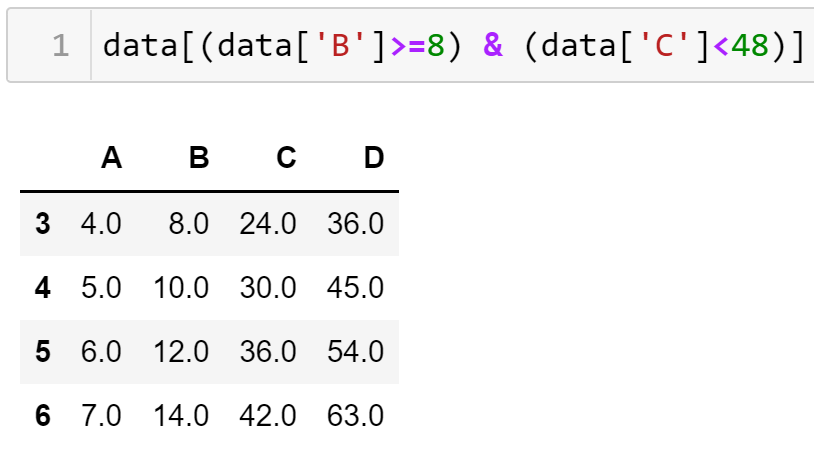
Полезным является метод describe(), который позволяет узнать основные статистические показатели по столбцам датафрейма.

****

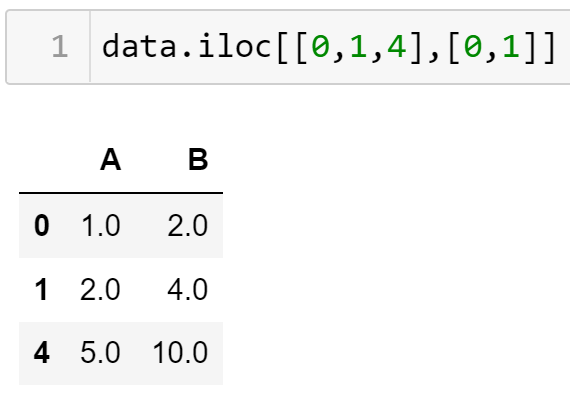
Можно создать новый столбец, используя известные данные:



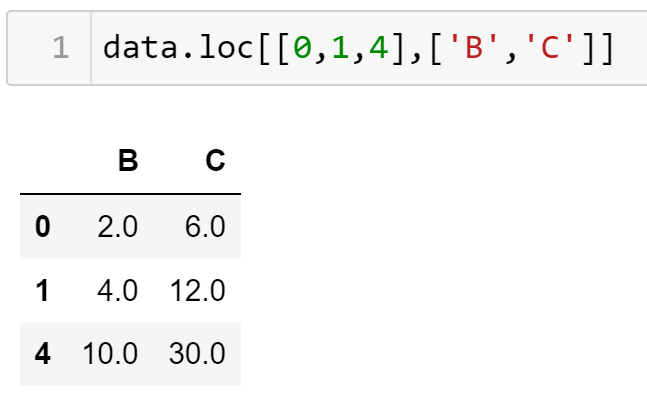
Значения в DataFrame или Series можно отфильтровать:



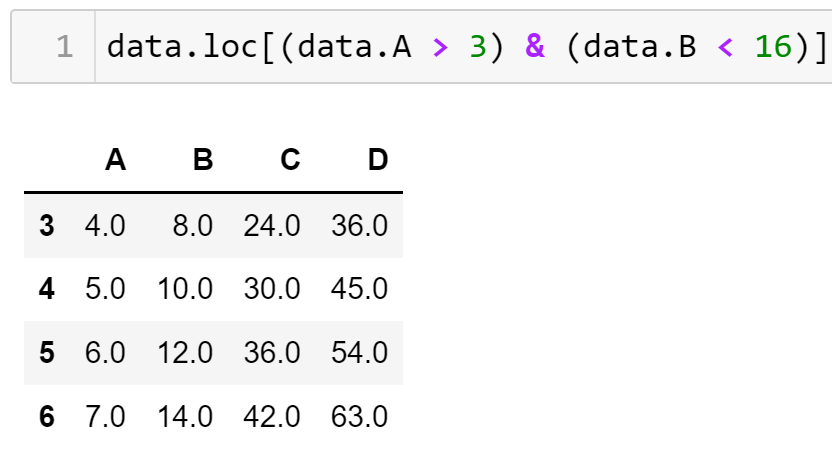
Обращение к датафрейму, метод iloc – индексация по позиции:



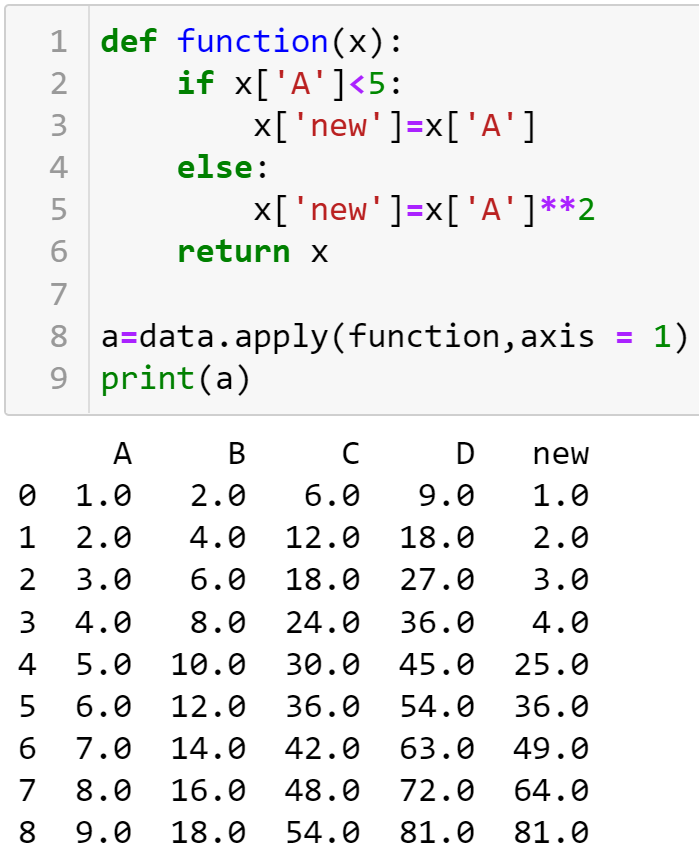
Метод loc – индексация на основе имени:



Так же с помощью loc можно фильтровать данные, например, выведем данные, где значение признака А > 3 и значение признака В < 16:



Метод apply() позволяет применить к серии или датафрейму некоторую функцию, например, создание нового признака, если значение в столбце A меньше 5 или больше:



axis = 1 – это проход по строкам, axis = 0 – это проход по столбцам, например:

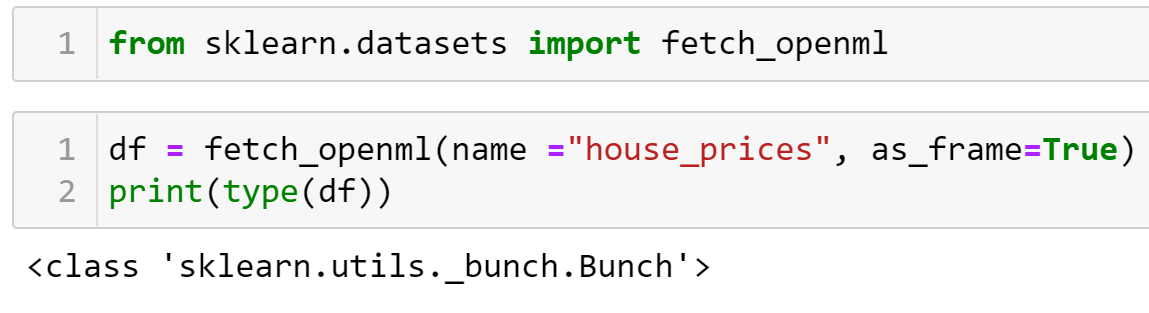


*Библиотека scikit-learn.*

scikit-learn – это библиотека для эффективного предиктивного анализа данных. Она содержит в себе различные алгоритмы машинного обучения, инструменты для предобработки данных, готовые наборы данных для исследований и тд.

Готовые датасеты можно взять в модуле dataset. У каждого датасета есть функция для его загрузки.

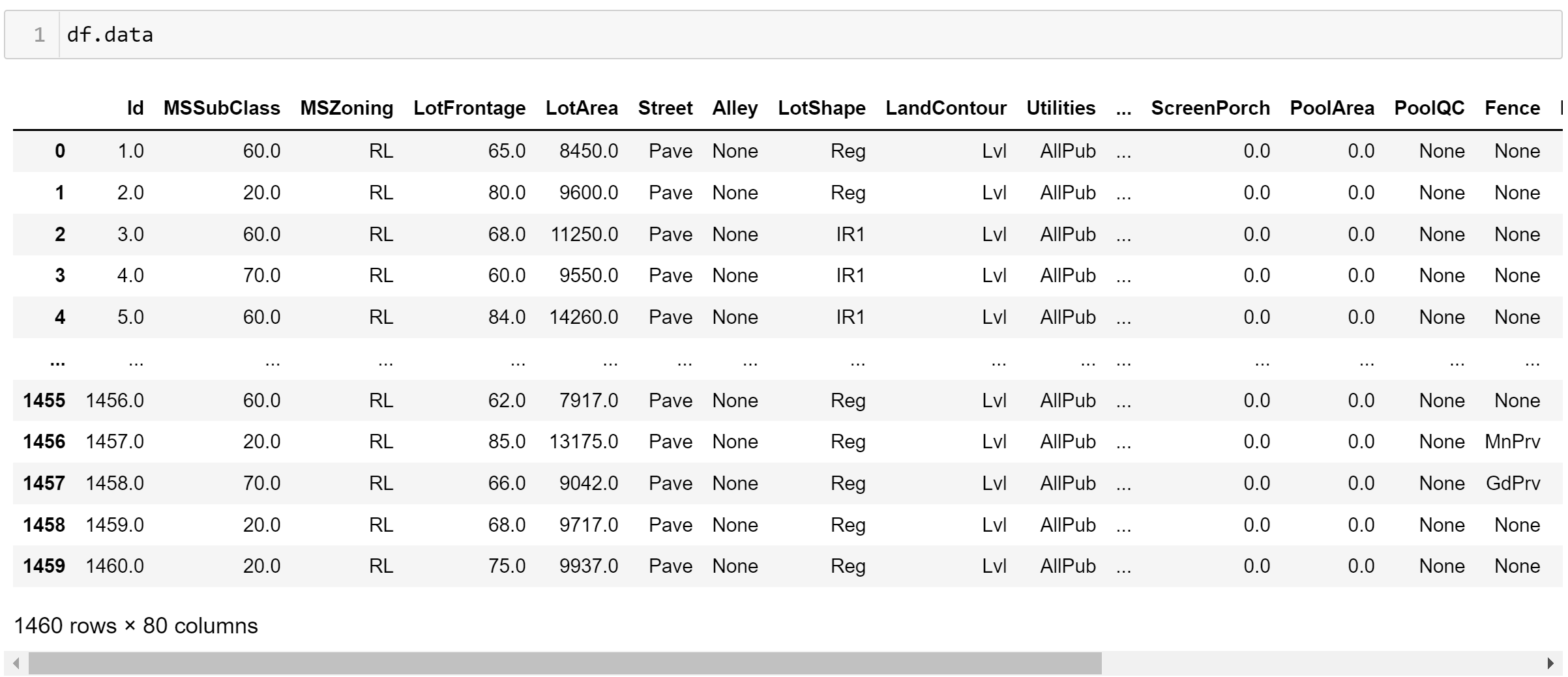
Рассмотрим пример загрузки датасета Ames housing:



В результате df – это объект Bunch, который представляет собой аналог словаря в библиотеке Scikit-Learn. Соответственно, у него есть ключи.



data – это таблица признаков. target – это значение целевой переменной. Этот набор данных используется для прогнозирования стоимости дома.



Подробнее данную библиотеку рассмотрим в следующих практических работах.

**Практическая работа**

1. Установить Python, если это не было сделано ранее.
2. Написать программу, которая вычисляет площадь фигуры, параметры которой подаются на вход. Фигуры, которые подаются на вход: треугольник, прямоугольник, круг. Результатом работы является словарь, где ключ – это название фигуры, а значение – это площадь.
3. Написать программу, которая на вход получает два числа и операцию, которую к ним нужно применить. Должны быть реализованы следующие операции: +, -, /, //, abs – модуль, pow или \*\* – возведение в степень.
4. Напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и после этого выводит сумму квадратов всех считанных чисел.
5. Напишите программу, которая выводит последовательность чисел, длинною N, где каждое число повторяется столько раз, чему оно равно. На вход программе передаётся неотрицательное целое число N. Например, если N = 7, то программа должна вывести 1 2 2 3 3 3 4. Вывод элементов списка через пробел – print(\*list).
6. Даны два списка:

А = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2]

В = [‘a’, ’b’, ’c’, ’c’, ’c’, ’b’, ’a’, ’c’, ’a’, ’a’, ’b’, ’c’, ’b’, ’a’]

Создать словарь, в котором ключи – это содержимое списка В, а значения для ключей словаря – это сумма всех элементов списка А в соответствии с буквой, содержащийся на той же позиции в списке В. Пример результата программы: {‘a’ : 10, ‘b’ : 15, ‘c’ : 6}.

Скачать и загрузить данные о стоимости домов в калифорнии, используя библиотеку sklearn.



1. Получить полный датасет с помощью «data.frame» и записать его в переменную.
2. Использовать метод info().
3. Узнать, есть ли пропущенные значения, используя isna().sum().
4. Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек, используя метод loc().
5. Узнать максимальное и минимальное значения медианной стоимости дома.
6. Используя метод apply(), вывести на экран название признака и его среднее значение.
7. Составить отчет о проделанной работе. В отчете должен быть представлен код и результаты его выполнения с выводами.

**1.\***

Дан текст на английском языке. Необходимо закодировать его с помощью азбуки Морзе, где каждой букве соответствует последовательность точек и тире. Например, буква «g» превратится в строку «--.». В переменной morze для удобства хранится словарь соответствия латинских букв коду Морзе.

morze = {'a': '.-', 'b': '-…', 'c': '-.-.', 'd': '-..',

'e': '.', 'f': '..-.', 'g': '--.', 'h': '….',

'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..',

'm': '--', 'n': '-.', 'o': '---', 'p': '.--.',

'q': '--.-', 'r': '.-.', 's': '…', 't': '-',

'u': '..-', 'v': '…-', 'w': '.--', 'x': '-..-',

'y': '-.--', 'z': '--..'}

**На входе**: В одной строке вам дан текст, который состоит из латинских букв и пробелов.

**На выходе**:

Выведите каждое слово исходного текста, закодированное азбукой Морзе. Количество строк в ответе должно совпадать с количеством слов в исходном тексте. Между закодированными буквами ставится ровно один пробел. Например, слово «Help» превратится в «.... . .-.. .--.». Строчные и заглавные буквы кодируются одинаково.

Например:

Ignition sequence start

Перевод

.. --. -. .. - .. --- -.

… . --.- ..- . -. -.-. .

… - .- .-. -

**2.\***

В некотором городе открывается новая служба по доставке электронных писем. Необходимо наладить систему регистрации новых пользователей.

Регистрация должна работать следующим образом: если новый пользователь хочет зарегистрироваться на сайте, то он должен послать системе запрос name со своим именем. Система должна определить, существует ли уже такое имя в базе данных. Если такого имени не существует, то оно заносится в базу данных системы и пользователю возвращается ответ "ОК", подтверждающий успешную регистрацию. А если пользователь с таким именем уже существует, то система должна сформировать новое имя и выдать его пользователю в качестве подсказки, при этом сама подсказка также добавляется в базу данных. Новое имя формируется следующим образом: к name последовательно приписываются числа, начиная с 1 (name1, name2 и так далее), и среди них находят такое наименьшее i, что namei еще не содержится в системе.

**Входные данные**

В первой строке входных данных задано число n (1 ≤ n ≤ 100000). Следующие n строк содержат запросы к системе. Каждый запрос представляет собой непустую строку длиной не более 32 символов, состоящую только из строчных букв латинского алфавита.

**Выходные данные**

В выходных данных должно содержаться n строк – ответы системы на запросы: "OK" в случае успешной регистрации, или подсказка с новым именем, если запрашиваемое уже занято.

Данные для проверки:

|  |  |
| --- | --- |
| Вход: | Выход: |
| 3  b  b  b | OK  b1  b2 |
| 10  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp  bhnqaptmp | OK  bhnqaptmp1  bhnqaptmp2  bhnqaptmp3  bhnqaptmp4  bhnqaptmp5  bhnqaptmp6  bhnqaptmp7  bhnqaptmp8  bhnqaptmp9 |
| 10  fpqhfouqdldravpjttarh  fpqhfouqdldravpjttarh  fpqhfouqdldravpjttarh  fpqhfouqdldravpjttarh  fpqhfouqdldravpjttarh  fpqhfouqdldravpjttarh  jmvlplnrmba  fpqhfouqdldravpjttarh  jmvlplnrmba  fpqhfouqdldravpjttarh | OK  fpqhfouqdldravpjttarh1  fpqhfouqdldravpjttarh2  fpqhfouqdldravpjttarh3  fpqhfouqdldravpjttarh4  fpqhfouqdldravpjttarh5  OK  fpqhfouqdldravpjttarh6  jmvlplnrmba1  fpqhfouqdldravpjttarh7 |

**3.\***

Необходимо создать программу обработки запросов пользователей к файловой системе компьютера. Над каждым файлом можно производить следующие действия: запись – w ("write"), чтение – r ("read"), запуск – x ("execute").

**Входные данные**

На вход программе подаются следующие параметры: число n – количество файлов в файловой системе. В следующих n строках содержится информация с именами файлов и допустимыми действиями (w, x, r), разделенных пробелами. Далее идет число m – количество запросов к файлам вида «операция файл» (обозначение операции: "write", "read", "execute").

**Выходные данные**

Для каждого допустимого запроса программа должна возвращать OK, для недопустимого – Access denied.

Данные для проверки:

|  |  |
| --- | --- |
| Вход: | Выход: |
| 3  python.exe x  book.txt r w  notebook.exe r w x  5  read python.exe  read book.txt  write notebook.exe  execute notebook.exe  write book.txt | Access denied  OK  OK  OK  OK |
| 3  root.html r w x  main.py x  login.txt w r  4  read root.html  write main.py  execute main.py  execute login.txt | OK  Access denied  OK  Access denied |
| 2  1.txt  2.txt  2  write 1.txt  execute 2.txt | Access denied  Access denied |