Основные типы данных в Python – числа и строки. Числа бывают двух типов: целые (integer, int) и вещественные (float). Присвоить переменной число можно следующим образом:

**a = 324**

**b = 24**

**c = a - b**

**print('a - b is', c)**

Попробуйте запустить эту программу.

Здесь переменные a, b и с содержат целочисленные значения, а выражение 'a-b is' является строкой.

Результат выполнения этой программы будет таким:

**a - b is 300**

Вы можете использовать различные арифметические операции при работе с числами:

Сложение: 2 + 3 = = 5

Вычитание: 5 - 2 = = 3

Умножение: 3 \* 4 = = 12

Деление: 15 / 3 = = 5.0

Остаток от деления: 18 % 5 = = 3

Возведение в степень: 2 \*\* 3 = = 8

Важно: В Python 3.x результатом деления целого числа на другое целое число будет число с плавающей запятой. Для выполнения целочисленного деления используйте следующую запись: 7 // 2 = = 3.

В Python символ "=" означает “присвоить значение переменной”. Для обозначения равенства используется двойной знак равенства "= =".

Строк представляет собой последовательность букв, цифр и других знаков. Создать строку можно следующим образом:

**a = "Hello"**

**b = 'World'**

**print(a,b)**

Обратите внимание: строку можно выделять как обычными кавычками, так и двойными.

Результат выполнения этой программы будет таким:

**Hello World**

Операции со строками отличаются от тех же операций с числами, например:

**a = 'Rosalind'**

**b = 'Franklin'**

**c = '!'**

**print(a + ' ' + b + c\*3)**

Результат выполнения этой программы будет таким:

**Rosalind Franklin!!!**

Таким образом, операция сложения “склеивает” строки, а операция умножения повторяет строку нужное количество раз подряд.

В Python есть типы данных, которые позволяют хранить сразу несколько значений. Простейшим подобным типом является список. Создать список можно следующим образом:

**list\_name = [value\_1, value\_2, …, value\_N]**

Элементами списка могут быть значения любого типа данных: целые числа, числа с плавающей запятой, строки и др. Даже список может быть элементом другого списка!

У каждого элемента списка есть свой индекс (то есть адрес в этом списке), например:

**tea\_party = ['Rabbit', 'Bear', 'Fox', 'Turtle']**

**print(tea\_party[2])**

Результат выполнения этой программы:

**Fox**

Обратите внимание: Программа напечатала Fox, а не Bear, т.к. в Python нумерация начинается не с 1, а с 0.

Элементы списка можно изменять:

**tea\_party[1] = 'Lion'**

**print(tea\_party)**

Результат:

**['Rabbit', 'Lion', 'Fox', 'Turtle']**

Также, с помощью функции append() можно добавлять элементы в конец списка:

**tea\_party.append('Mouse')**

**print(tea\_party)**

Результат:

**['Rabbit', 'Lion', 'Fox', 'Turtle', 'Mouse']**

Чтобы получить часть списка нужно использовать конструкцию list\_name[a:b], которая выведет элементы списка с индексами от a до b-1. Например:

**tea\_party[1:3] = = ['Lion', 'Fox']**

Если индекс **a** не указан, то результат будет начинаться с элемента под индексом 0; если не указан индекс **b** – результат будет заканчиваться последним элементом списка. Например:

**tea\_party[:2] = = ['Rabbit', 'Lion']**

**tea\_party[3:] = = ['Turtle', 'Mouse']**

Также, можно использовать отрицательные индексы. Индекс -1 соответствует последнему элементу списка, -2 – предпоследнему и так далее.

Python позволяет работать со строками так же, как и со списками, например:

**a = 'flimsy'**

**b = 'miserable'**

**c = b[0:1] + a[-4:]**

**print c**

Результат:

**mimsy**

Условный оператор в Python выглядит следующим образом:

**a = 42**

**if a < 10:**

**print('the number is less than 10')**

**else:**

**print('the number is greater or equal to 10')**

Важно: обратите внимание на пунктуацию и отступы – они обязательны!

Блок else можно опустить, например:

**if a + b == 4:**

**print('printed when a + b equals four')**

**print('always printed')**

Чтобы повторить команду несколько раз, необходимо использовать один из операторов цикла. Цикл while выглядит следующим образом:

**greetings = 1**

**while greetings <= 3:**

**print('Hello! ' \* greetings)**

**greetings = greetings + 1**

Эта программа напечатает следующее (команды в цикле выполнятся 3 раза):

**Hello!**

**Hello! Hello!**

**Hello! Hello! Hello!**

Будьте осторожны! Неправильное использование цикла while может вызвать его бесконечное выполнение, например:

**greetings = 1**

**while greetings <= 3:**

**print('Hello! ' \* greetings)**

**greetings = greetings + 0 # Bug here**

Обратите внимание: комментарии выделяются символом #.

Для выполнения команды для каждого элемента списка можно использовать цикл for:

**names = ['Alice', 'Bob', 'Charley']**

**for name in names:**

**print('Hello, ' + name)**

Чтобы повторить команду ровно N раз, используйте функцию range(N), которая создает список с элементами от 0 до N-1:

**n = 10**

**for i in range(n):**

**print(i)**

Попробуйте запустить следующую программу:

**print(range(5, 12))**

Python содержит команды для чтения из файла и записи в файл. Чтобы считать данные из файла, его сначала нужно открыть. Это делает функция open(file, mode), где file – путь к файлу, а mode – режим открытия (‘r’ – для чтения, ‘w’ – для записи, при этом содержимое файла стирается, ‘a’ – для записи, при этом данные записываются в конец файла).

Пример:

**f = open('input.txt', 'r')**

Для получения данных из файла можно использовать команду f.read(). Эта команда возвращает всё содержимое файла f в виде одной строки.

Команда f.readline() возвращает одну строку из файла, причем каждая строка (кроме последней) содержит в конце символ “\n” – символ переноса строки.

Обратите внимание: Каждый вызов команды f.readline() возвращает последнюю непрочитанную строку файла.

Команда f.readlines() возвращает список, элементами которого являются все строки из файла.

Для считывания файла можно использовать и цикл for:

**for line in f:**

**print(line)**

Если данные в строке разделены пробелами, или другими символами, удобно использовать функцию split(char), где char – символ-разделитель:

**print('Beautiful is better than ugly'.split(' '))**

Результат:

**['Beautiful', 'is', 'better', 'than', 'ugly.']**

Чтобы открыть файл на запись:

**f = open('output.txt', 'w')**

Для записи в файл:

**f.write('Any data you want to write into file')**

Чтобы записать в файл данные, не являющиеся строкой, необходимо их конвертировать в строку – это делает функция str(data):

**inscription = ['Rosalind Elsie Franklin ', 1920, 1958]**

**s = str(inscription)**

**f.write(s)**

Также, можно использовать цикл:

**for i in inscription:**

**f.write(str(i) + '\n')**

Данная программа запишет в файл каждый элемент списка в отдельную строку.

При завершении записи, файл необходимо закрыть (не забывайте об этом!):

**f.close()**

Помимо списков, в Python есть такой тип данных, как словарь. Словарь похож на список, но вместо автоматических числовых индексов использует заданные пользователем, причем индексом может служить любой тип данных:

**phones = {'Zoe':'232-43-58', 'Alice':'165-88-56'}**

**print(phones['Zoe'])**

Результат:

**232-43-58**

Изменение и добавление элементов происходит аналогично спискам:

**phones['Zoe'] = '658-99-55'**

**phones['Bill'] = '342-18-25'**

**print(phones)**

Результат:

**{'Bill': '342-18-25', 'Zoe': '658-99-55', 'Alice': '165-88-56'}**

Обратите внимание: Словарь выделяется фигурными скобками, а список – квадратными.

Обратите внимание: Порядок элементов в словаре не регламентирован.

Индексы в словарях чувствительны к регистру:

**d = {} #** Создаем пустой словарь

**d['key'] = 1**

**d['Key'] = 2**

**d['KEY'] = 3**

**print(d)**

Результат:

**{'KEY': 3, 'Key': 2, 'key': 1}**

Чтобы проверить, присутствует ли нужный индекс в словаре, используйте следующую запись:

**if 'Peter' in phones:**

**print("We know Peter's phone")**

**else:**

**print("We don't know Peter's phone")**

Результат:

**We don't know Peter's phone**

Чтобы удалить элемент из словаря, используйте оператор del:

**phones = {'Zoe':'232-43-58', 'Alice':'165-88-56'}**

**del phones['Zoe']**

**print(phones)**

Результат:

**{'Alice': '165-88-56'}**

В Python можно создать свою функцию:

**def func(a,b):**

**return a+b**

Проверим, работает ли она:

**c = 3**

**d = 4**

**print(func(c,d))**

Результат:

**7**

В Python любой файл с программой называется модулем. При написании программы можно использовать функции и переменные из сторонних модулей. Для этого сторонний модуль нужно вызвать. Это выполняется с помощью оператора import:

**import math #** вызываем модуль math

**a = math.sin(math.pi) #** используем функцию sin() и переменную pi из модуля math

**print(a)**

Модули могут объединяться в группы, называемые библиотеками модулей. Чтобы вызвать модуль из библиотеки, используйте конструкцию from bib\_name import module\_name:

**from matplotlib import pyplot** # вызываем модуль pyplot из библиотеки matplotlib

**pyplot.show()** # рисуем пустой график

Другой способ:

**import matplotlib.pyplot**

**matplotlib.pyplot.show()**

При вызове библиотек и модулей можно давать им псевдонимы:

**import numpy as np** # вызываем модуль numpy

**a = np.zeros((2,2))** # создаем матрицу 2 на 2, состоящую из нулей

**print(a)**