**1.1 Типы данных, переменный, арифметически операторы и математические функции**

**Введение в Python**

Программа на языке Python состоит из набора инструкций. Каждая инструкция помещается на новую строку. Например:

|  |  |
| --- | --- |
| 1  2 | print(2 + 3)  print("Hello") |

Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой. Например, в следующем случае мы получим ошибку, хотя код будет практически аналогичен приведенному выше:

|  |  |
| --- | --- |
| 1  2 | print(2 + 3)      print("Hello") |

Поэтому стоит помещать новые инструкции сначала строки. В этом одно из важных отличий пайтона от других языков программирования, как C# или Java.

Однако стоит учитывать, что некоторые конструкции языка могут состоять из нескольких строк. Например, условная конструкция **if**:

|  |  |
| --- | --- |
| 1  2 | if 1 < 2:      print("Hello") |

В данном случае если 1 меньше 2, то выводится строка "Hello". И здесь уже должен быть отступ, так как инструкция print("Hello") используется не сама по себе, а как часть условной конструкции if. Причем отступ, согласно [руководству по оформлению кода](https://www.python.org/dev/peps/pep-0008/), желательно делать из такого количество пробелов, которое кратно 4 (то есть 4, 8, 16 и т.д.) Хотя если отступов будет не 4, а 5, то программа также будет работать.

Таких конструкций не так много, поэтому особой путаницы по поводу где надо, а где не надо ставить пробелы, не должно возникнуть.

Регистрозависимость

Python - регистрозависимый язык, поэтому выражения print и Print или PRINT представляют разные выражения. И если вместо метода print для вывода на консоль мы попробуем использовать метод Print:

|  |  |
| --- | --- |
| 1 | Print("Hello World") |

то у нас ничего не получится.

Комментарии

Для отметки, что делает тот или иной участок кода, применяются комментарии. При трансляции и выполнении программы интерпретатор игнорирует комментарии, поэтому они не оказывают никакого влияния на работу программы.

Комментарии в Python бывают блочные и строчные. Все они предваряются знаком решетки (#).

Блочные комментарии ставятся в начале строки:

|  |  |
| --- | --- |
| 1  2 | # Вывод сообщения на консоль  print("Hello World") |

Строчные комментарии располагаются на той же строке, что и инструкции языка:

|  |  |
| --- | --- |
| 1 | print("Hello World")  # Вывод сообщения на консоль |

**Основные функции**

Python предоставляет ряд встроенных функций. Некоторые из них используются очень часто, особенно на начальных этапах изучения языка, поэтому рассмотрим их.

Основной функцией для вывода информации на консоль является функция print(). В качестве аргумента в эту функцию передается строка, которую мы хотим вывести:

|  |  |
| --- | --- |
| 1 | print("Hello Python") |

Если же нам необходимо вывести несколько значений на консоль, то мы можем передать их в функцию print через запятую:

|  |  |
| --- | --- |
| 1 | print("Full name:", "Tom", "Smith") |

В итоге все переданные значения склеятся через пробелы в одну строку:

Full name: Tom Smith

Если функция print отвечает за вывод, то функция **input** отвечает за ввод информации. В качестве необязательного параметра эта функция принимает приглашение к вводу и возвращает введенную строку, которую мы можем сохранить в переменную:

|  |  |
| --- | --- |
| 1  2 | name = input("Введите имя: ")  print("Привет", name) |

Консольный вывод:

Введите имя: Евгений

Привет Евгений

**Переменные и типы данных**

Переменная хранит определенные данные. Название переменной в Python должно начинаться с алфавитного символа или со знака подчеркивания и может содержать алфавитно-цифровые символы и знак подчеркивания. И кроме того, название переменной не должно совпадать с названием ключевых слов языка Python. Ключевых слов не так много, их легко запомнить: and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield.

Например, создадим переменную:

|  |  |
| --- | --- |
| 1 | name = "Tom" |

Здесь определена переменная name, которая хранит строку "Tom".

В пайтоне применяется два типа наименования переменных: camel case и underscore notation.

Camel case подразумевает, что каждое новое подслово в наименовании переменной начинается с большой буквы. Например:

|  |  |
| --- | --- |
| 1 | userName = "Tom" |

Underscore notation подразумевает, что подслова в наименовании переменной разделяются знаком подчеркивания. Например:

|  |  |
| --- | --- |
| 1 | user\_name = "Tom" |

И также надо учитывать регистрозависимость, поэтому переменные name и Name будут представлять разные объекты.

Переменная хранит данные одного из типов данных. В Python существует множество различных типов данных, которые подразделяются на категории: числа, последовательности, словари, наборы:

**boolean** - логическое значение True или False

**int** - представляет целое число, например, 1, 4, 8, 50.

**float** - представляет число с плавающей точкой, например, 1.2 или 34.76

**complex** - комплексные числа

**str** - строки, например "hello". В Python 3.x строки представляют набор символов в кодировке Unicode

**bytes** - последовательность чисел в диапазоне 0-255

**byte array** - массив байтов, аналогичен bytes с тем отличием, что может изменяться

**list** - список

**tuple** – кортеж

**set** - неупорядоченная коллекция уникальных объектов

**frozen set** - то же самое, что и set, только не может изменяться (immutable)

**dict** - словарь, где каждый элемент имеет ключ и значение

Python является языком с динамической типизацией. Он определяет тип данных переменной исходя из значения, которое ей присвоено. Так, при присвоении строки в двойных или одинарных кавычках переменная имеет тип str. При присвоении целого числа Python автоматически определяет тип переменной как int. Чтобы определить переменную как объект float, ей присваивается дробное число, в котором разделителем целой и дробной части является точка. Число с плавающей точкой можно определять в экспоненциальной записи:

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = 3.9e3  print(x)  # 3900.0  x = 3.9e-3  print(x)  # 0.0039 |

Число float может иметь только 18 значимых симолов. Так, в данном случае используются только два символа - 3.9. И если число слишком велико или слишком мало, то мы можем записывать число в подобной нотации, используя экспоненту. Число после экспоненты указывает степень числа 10, на которое надо умножить основное число - 3.9.

При этом в процессе работы программы мы можем изменить тип переменной, присвоив ей значение другого типа:

|  |  |
| --- | --- |
| 1  2  3  4  5 | user\_id = "12tomsmith438"  # тип str  print(user\_id)  user\_id = 234  # тип int  print(user\_id) |

С помощью функции **type()** динамически можно узнать текущий тип переменной:

|  |  |
| --- | --- |
| 1  2  3  4  5 | user\_id = "12tomsmith438"  print(type(user\_id))  # <class 'str'>  user\_id = 234  print(type(user\_id))  # <class 'int'> |

**Операции с числами**

**Арифметические операции**

Python поддерживает все распространенные арифметические операции:

**+** Сложение двух чисел:

print(6 + 2)  # 8

**-** Вычитание двух чисел:

print(6 - 2)  # 4

**\*** Умножение двух чисел:

print(6 \* 2)  # 12

**/** Деление двух чисел:

print(6 / 2)  # 3.0

**//** Целочисленное деление двух чисел:

print(7 / 2)  # 3.5

print(7 // 2)  # 3

Данная операция возвращает целочисленный результат деления, отбрасывая дробную часть

**\*\*** Возведение в степень:

print(6 \*\* 2)  # Возводим число 6 в степень 2. Результат - 36

**%** Получение остатка от деления:

print(7 % 2)  # Получение остатка от деления числа 7 на 2. Результат - 1

В данном случае ближайшее число к 7, которое делится на 2 без остатка, это 6. Поэтому остаток от деления равен 7 - 6 = 1

При последовательном использовании нескольких арифметических операций их выполнение производится в соответствии с их приоритетом. В начале выполняются операции с большим приоритетом. Приоритеты операций в порядке убывания приведены в следующей таблице.

Операции Направление

\*\* Справо налево

\* / // % Слева направо

+ - Слева направо

Пусть у нас выполняется следующее выражение:

number = 3 + 4 \* 5 \*\* 2 + 7

print(number)  # 110

Здесь начале выполняется возведение в степень (5 \*\* 2) как операция с большим приоритетом, далее результат умножается на 4 (25 \* 4), затем происходит сложение (3 + 100) и далее опять идет сложение (103 + 7).

Чтобы переопределить порядок операций, можно использовать скобки:

number = (3 + 4) \* (5 \*\* 2 + 7)

print(number)  # 224

Следует отметить, что в арифметических операциях могут принимать участие как целые, так и дробные числа. Если в одной операции участвует целое число (int) и число с плавающей точкой (float), то целое число приводится к типу float.

### Арифметические операции с присвоением

Ряд специальных операций позволяют использовать присвоить результат операции первому операнду:

**+=** Присвоение результата сложения

**-=** Присвоение результата вычитания

**\*=** Присвоение результата умножения

**/=** Присвоение результата от деления

**//=** Присвоение результата целочисленного деления

**\*\*=** Присвоение степени числа

**%=** Присвоение остатка от деления

Примеры операций:

number = 10

number += 5

print(number)  # 15

number -= 3

print(number)  # 12

number \*= 4

print(number)  # 48

### Функции преобразования чисел

### Ряд встроенных функций в Python позволяют работать с числами. В частности, функции int() и float() позволяют привести значение к типу int и float соответственно.

Например, пусть у нас будет следующий код:

first\_number = "2"

second\_number = 3

third\_number = first\_number + second\_number

Мы ожидаем, что "2" + 3 будет равно 5. Однако этот код сгенерирует исключение, так как первое число на самом деле представляет строку. И чтобы все заработало как надо, необходимо привести строку к числу с помощью функции int():

first\_number = "2"

second\_number = 3

third\_number = int(first\_number) + second\_number

print(third\_number) # 5

Аналогичным образом действует функция float(), которая преобразует в число с плавающей точкой. Но вообще с дробными числами надо учитывать, что результат операций с ними может быть не совсем точным. Например:

first\_number = 2.0001

second\_number = 5

third\_number = first\_number / second\_number

print(third\_number) # 0.40002000000000004

В данном случае мы ожидаем получить число 0.40002, однако в конце через ряд нулей появляется еще какая-то четверка. Или еще одно выражение:

print(2.0001 + 0.1)  # 2.1001000000000003

В этот случае для округления результата мы можем использовать функцию **round()**:

first\_number = 2.0001

second\_number = 0.1

third\_number = first\_number + second\_number

print(round(third\_number, 4))  # 2.1001

Первый параметр функции - округляемое число, а второй - сколько знаков после запятой должно содержать получаемое число.

### Представление числа

При обычном определении числовой переменной она получает значение в десятичной системе. Но кроме десятичной в Python мы можем использовать двоичную, восьмеричную и шестнадцатеричную системы.

Для определения числа в двоичной системе перед его значением ставится 0 и префикс **b**:

x = 0b101   # 101 в двоичной системе равно 5

Для определения числа в восьмеричной системе перед его значением ставится 0 и префикс **o**:

a = 0o11    # 11 в восьмеричной системе равно 9

Для определения числа в шестнадцатеричной системе перед его значением ставится 0 и префикс **x**:

y = 0x0a        # a в шестнадцатеричной системе равно 10

И с числами в других системах измерения также можно проводить арифметические операции:

x = 0b101   # 5

y = 0x0a    # 10

z = x + y   # 15

print("{0} in binary {0:08b}   in hex {0:02x} in octal {0:02o}".format(z))

Для вывода числа в различных системах исчисления используются функция format, которая вызывается у строки. В эту строку передаются различные форматы. Для двоичной системы "{0:08b}", где число 8 указывает, сколько знаков должно быть в записи числа. Если знаков указано больше, чем требуется для числа, то ненужные позиции заполняются нулями. Для шестнадцатеричной системы применяется формат "{0:02x}". И здесь все аналогично - запись числа состоит из двух знаков, если один знак не нужен, то вместо него вставляется ноль. А для записи в восьмеричной системе испольуется формат "{0:02o}".

Результат работы скрипта:

15 in binary 00001111 in hex 0f in octal 17