**Типы данных языка python. Операции над ними**

В этой главе рассматриваются основные типы данных языка Python и операции над ними.

**Целые числа. Тип int**

В языке Python для представления целых чисел используется тип int. Литералы целых чисел по умолчанию записываются в десятичной системе счисления. При работе с целыми числами используются математические операторы. Пусть даны два целых числа X и Y, тогда:

* X + Y – складывает числа X и Y;
* X - Y – вычитает число Y из X;
* X \* Y – умножает числа X и Y;
* X / Y – делит число X на Y – результатом является значение типа float (число с плавающей точкой);
* X // Y – делит число X на Y, при этом результат округлен до ближайшего целого числа в менышую сторону;
* X % Y – возвращает остаток от деления X на Y;
* X \*\* Y – возводит X в степень Y;
* - X – изменяет знак числа X, если X не равен 0;

Для работы с целыми числами в языке Python предусмотрены и встроенные функции, которые являются аналогами математических функций:

* abs(X) – возвращает модуль числа X;
* pow(X,Y,[Z]) – альтернативен записи (X\*\*Y)%Z;

Для операторов (+,-,/,//,%,\*\*) существует альтернативная укороченная запись, имеющая общий вид: x op= y, которая раскрывается в x = x op y. Например x = x \* y, можно записать как: x\*=y.

a = 4  
a = a \* 2  
print(a)

Вывод на консоль:

8

a = 4  
a \*= 2  
print(a)

Вывод на консоль:

8

**Логический тип. Bool**

В Python логический тип представлен типом данных bool. Переменная типа bool может принимать два возможных значения: False (ложь) и True (истина). В языке Python имеются три логических оператора:

* X and Y – результатом является True, тогда и только тогда, когда X и Y – являются True;
* X or Y – результатом является False, тогда и только тогда, когда X и Y – являются False;
* not X – возвращает противоположное X значение;

В Python разрешено рассматривать любое значение как истинное или ложное. Например, в логических выражения число 0 представляет False, а все остальное как True.

print(bool(2))  
print(bool(1))  
print(bool(0))

Вывод на консоль:

True

True

False

Логические значения могут трактоваться как числа, т.е. True – 1, а False – 0. Значение типа int можно привести к типу bool, с помощью функции bool().

print(int(True))  
print(int(False))

Вывод на консоль:

1

0

**Числа с плавающей точкой. Тип float**

В языке Python для представления десятичных дробей используется тип float. Такие дроби в программировании называются как «Числа с плавающей точкой». Все числовые операторы и функции типа int применимы для типа float. Для типа float существует дополнительная встроенная функция round(), которая округляет число с плавающей точкой до заданной точности. Число типа int можно привести к типу float с помощью функции float(), а число типа float привести к типу int с помощью функции int(), при этом дробная часть будет отброшена. При выполнении операции с числами типа float, возможна ситуация, когда полученное значение - NaN (не число) или бесконечность.

**Строки. Тип str**

Строки в Python представляют из себя набор символов, заключённый в одинарные (' ') или двойные (" ") кавычки, при этом строки “Hello World” и ‘Hello World’ – это одно и то же. Применив кавычки одного типа в роли ограничителей, автоматически запрещается их использование внутри строки. Строки в языке Python представлены типом данных str. Любой ранее рассмотренный тип данных можно привести к типу str с помощью функции str(). Строки можно привести к типу int или float с помощью соответствующих функции int() и float(). Однако, на такое приведение накладываются дополнительные ограничения:

* Строку можно привести к типу int, тогда и только тогда, когда строка содержит числовые литералы.
* Строку можно привести к типу float, тогда и только тогда, когда строка содержит числовые литералы и один разделитель десятичной части.

В строках для представления специальных (например, непечатаемых) символов используются escape-последовательности, состоящие из обратной косой черты, за которой следует один или несколько символов. Например, \n – символ перевода строки. Доступные escape-последовательности:

|  |  |
| --- | --- |
| **\b** | Backspace |
| **\f** | Перевод страницы |
| **\n** | Новая строка |
| **\r** | Возврат каретки |
| **\t** | Горизонтальная табуляция |
| **\v** | Вертикальная табуляция |
| **\'** | Одиночная кавычка |
| **\"** | Двойная кавычка |
| **\\** | Обратная косая черта |
| **\?** | Литерал вопросительного знака |

Строки поддерживают обычные операторы сравнения <,<=,==,!=,>,>=. Эти операторы выполняют побайтовое сравнение строк. Для строк определены операторы +, \* и in. Оператор + возвращает склеенную строку.

Например, “Hello” +” world” вернет “Hello world”:

print("Hello" + " world")

Вывод на консоль:

Hello world

Умножение целого числа на строку вернет, строку содержащую размноженную копию исходной строки. Например, “Hello” \* 3, вернет “HelloHelloHello”:

print("Hello" \* 3)

Вывод на консоль:

HelloHelloHello

Оператор in cчитается истиной (true), если находит переменную в заданной последовательности, и ложью (false) в противном случае. Например: ‘e’ in ‘Hello’ – вернет True, а ‘w’ in ‘Hello’ – вернет False.

s = 'Hello'  
print('e' in s)  
print('w' in s)

Вывод на консоль:

True

False

Для обращения к отдельным элементам строки используется оператор []. Например “Hello”[1] вернет “e”, но с помощью оператора [] нельзя изменять значение элемента, потому что строки в Python являются неизменяемыми.

s = 'Hello'  
print(s[1])

Вывод на консоль:

e

С помощью оператора [] можно получить не только один элемент, а сразу несколько подряд идущих элементов. Для этого требуется в операторе [] указать диапазон, при этом последний элемент диапазона не будет включен в последовательность. Например, “Hello”[0:2] вернет “He”. Если в срезе опущен первый символ, значит, он равен нулю; если опущен последний символ – он равен длине строки. Например, “Hello” [0:2] == “Hello” [:2] == “He”, “Hello” [3:] == “lo”.

s = 'Hello'  
print(s[0:2])  
print(s[0:2])  
print(s[3:])

Вывод на консоль:

He

He

lo

Для типа str в языке Python определено большое число встроенных функции, рассмотрим наиболее популярные из них. Пусть s = “Мама мыла раму”, тогда:

* find() – находит подстроку в строке – возвращает позицию вхождения строки, либо -1. Пример: s.find(‘мыла’) вернет 5.
* split() – разбивает строку на последовательности по указанному разделителю. Разделитель по умолчанию – пробел.  
  Пример: s.split() вернет список последовательностей [“Мама”,”мыла”,”раму”]
* join() – обратная к функции split(). Объединяет через указанный разделитель. Разделитель по умолчанию – пробел.  
  Пример: s.split().join(“!”) вернет строку “Мама!мыла!раму”.
* ……..

Так как строки являются последовательностями, то с помощью встроенной функции len() можно получить количество символов в строке. Например, len(“Hello”) вернет 5.

**Приоритет операторов.**

|  |  |
| --- | --- |
| Оператор | Описание |
| **\*\*** | Возведение в степень |
| **\* / % //** | Умножение, деление, деление по модулю, целочисленное деление. |
| **+ -** | Сложение и вычитание. |
| **>> <<** | Побитовый сдвиг вправо и побитовый сдвиг влево. |
| **<= < > >=** | Операторы сравнения |
| **<> == !=** | Операторы равенства |
| **= %= /= //= -= += \*= \*\*=** | Операторы присваивания |
| **is is not** | Тождественные операторы |
| **in not in** | Операторы членства |
| **not or and** | Логические операторы |