Язык программирования Питон (Python)

Типы и структуры данных

Числа

Числа в Python могут быть

- целыми (тип **`int`**),
- вещественными (тип **`float`**),
- комплексными (тип **`complex`**).

Замечание. Нет отдельного типа '**`long int`**' (длинное целое). Целые числа по умолчанию могут быть произвольной длины.

Встроенные функции для работы с числами:

– преобразование типов	**`int()`** **`float()`**
– определение типа	**`type()`**
– вычисление абсолютного значения	**`abs()`**
– возведение в степень	**`pow()`**
– выч. результата целочисленного деления и остатка	**`divmod()`**
– округление	**`round()`**

Все ***прочие функции*** для работы с числами (математические) требуют подключения модуля **`math`**.

Примеры:

```
int(-22.3) ->  -22;   abs(-3)   ->  3;   divmod(17, 5)     ->  (3, 2);

float(12)  ->  12.0;  pow(2, 3) ->  8;   round(100.0/6, 3) ->  16.667.
```

```
print(2 ** 1000)  ->
                     10715086071862673209484250490600018105614048117055336
                     07443750388370351051124936122493198378815695858127594
                     67291755314682518714528569231404359845775746985748039
                     34567774824230985421074605062371141877954182153046474
                     98358194126739876755916554394607706291457119647768654
                     21676604298316526243868372056680693 76
```

```
type(1)    ->   int              type(int(1.0))        -> int

type(1.0)  ->   float            type(-1.2+3.7j + 5)   ->  complex
```

Операции с числами

Операция	Описание
`x + y`	Сложение (сумма x и y)
`x - y`	Вычитание (разность x и y)
`x * y`	Умножение (произведение x и y)
`x/y`	Деление x на y (частное). *Внимание*! Если x и y целые, то результат всегда будет целым числом! (*в Python 2*) Для получения вещественного результата хотя бы одно из чисел должно быть вещественным. *Пример:* `100/8 -> 12,` а вот `100/8.0 -> 12.5`
`x//y`	Целочисленное деление (результат – целое число). Если оба числа в операции вещественные, получается вещественное число с дробной частью, равной нулю. *Пример:* `100//8 -> 12,` `100//12.0 -> 8.0`
`x%y`	Остаток от целочисленного деления x на y *Пример:* `10%4 -> 2`
`x**y`	Возведение в степень (x в степени y). *Примеры:* `2**3 -> 8,` `2.3**(-3.5) -> 0.05419417057580235`
`-x`	Смена знака числа

Операции с числами (продолжение)

Операция	Описание
`<<`	Сдвигает биты числа влево на заданное количество позиций. *Пример:* `2 << 2 -> 8.`
`>>`	Сдвигает биты числа вправо на заданное число позиций. *Пример:* `11 >> 1 -> 5.`
`&`	Побитовая операция И над числами *Пример:* `5 & 3 -> 1.`
`|`	Побитовая операция ИЛИ над числами *Пример:* `5 | 3 -> 7`
`^`	Побитовая операция ИСКЛЮЧАЮЩЕЕ ИЛИ *Пример:* `5 ^ 3 -> 6`
`~`	Побитовая операция НЕ *Пример:* `~5 -> 6`

Логические значения

Логические значения в Python представлены константами `True` (Истина)
`False` (Ложь).

Замечание: Тип `bool` – подтип целочисленного типа для обозначения логических величин.

Основные логические операции и выражения

Операция	Описание
`>`	Условие “больше” *Пример:* `5 < 3 -> False; 3 < 5 -> True`
`<`	Условие “меньше”
`<=`	Определяет, верно ли, что x меньше или равно y *Пример:* `x = 3; y = 6; x <= y -> True`
`>=`	Определяет, верно ли, что x больше или равно y
`==`	Условие равенства
`!=`	Условие неравенства *Пример:* `x = 2; y = 3; x != y -> True`

Основные логические операции и выражения (продолжение)

Операция	Описание
`not x`	Отрицание (условие x не выполняется) *Пример:* `x = True; not x -> False`
`x and y`	Логическое “И” (умножение). Чтобы выполнилось условие x and y, необходимо, чтобы одновременно выражения x и y были истинными. *Пример:* `x = True; y = False; x and y -> False`
`x or y`	Логическое “ИЛИ” (сложение). Чтобы выполнилось условие x or y, необходимо, чтобы истинным было хотя бы одно из выражений. *Пример:* `x = True; y = False; x or y -> True`
`x in A`	Проверка принадлежности элемента x множеству (структуре) A
`a < x < b`	Эквивалентно (x > a) and (x < b)

Примеры:

```
print((1 == 2) | (2 == 2))     -> True
print (1 < 2 < 3 < 4)          -> True
print (1 < 3 < 3 < 4)          -> False
'a' in 'abc'    -> True
'A' in 'abc'    -> False
```

```
x = 4
if 3 < x < 5:  # можно без скобок
    print('четыре')
```

Примеры: ***Неожиданный результат при сравнении вещественных чисел!***

```
0.1 + 0.1 == 0.2          -> True
0.1 + 0.1 + 0.1 == 0.3    -> False
```

Логическим высказыванием (предикатом) называют любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Примеры:

```
2 > 4 and 45 > 3   -> False
2 > 4 or 45 > 3    -> True
```

Замечание:

Для Python истинным или ложным может быть не только логическое высказывание, но и объект:

- любое ***число, не равное нулю***, или ***непустой объект*** интерпретируется как **`True`**.
- ***числа, равные нулю***, ***пустые объекты*** и специальный объект **`None`** интерпретируются как **`False`**.

Приоритет операций

Оператор	Описание
lambda	лямбда-выражение
or	Логическое “ИЛИ”
and	Логическое “И”
not x	Логическое “НЕ”
in, not in	Проверка принадлежности
is, is not	Проверка тождественности
<, <=, >, >=, !=, ==	Сравнения
|	Побитовое “ИЛИ”
^	Побитовое “ИСКЛЮЧИТЕЛЬНО ИЛИ”
&	Побитовое “И”
<<, >>	Сдвиги
+, -	Сложение и вычитание
*, /, //, %	Умножение, деление, целочисленное деление и остаток от деления
+x, -x	Положительное, отрицательное
~x	Побитовое НЕ
**	Возведение в степень
x.attribute	Ссылка на атрибут
x[индекс]	Обращение по индексу
x[индекс1:индекс2]	Вырезка
f(аргументы ...)	Вызов функции
(выражения, ...)	Связка или кортеж [2]
[выражения, ...]	Список
{ключ:данные, ...}	Словарь

Структуры данных

Структуры данных используются ***для хранения связанных данных***.

В Python определены такие структуры данных (составные типы):

1. **Последовательности:** обеспечивают индексирование и проверку принадлежности

   – **изменяемые:** позволяют добавлять или удалять элементы этой последовательности.

      – ***списки:*** упорядоченные наборы объектов, возможно объекты разных типов.

         - *элементы списка разделяются запятыми и заключаются в квадратные скобки.*

   – **неизменяемые**

      – ***строки:*** последовательности символов.

         - *задаются при помощи одиночных или двойных кавычек.*
         - *можно указывать «многострочные» строки с использованием тройных кавычек.*

      – ***кортежи***: упорядоченный набор объектов, возможно объекты разных типов.

         - *обозначаются указанием элементов, разделённых запятыми;*
         - *по желанию их можно ещё заключить в круглые скобки.*

Структуры данных (*продолжение*)

2. Отображения (***словари***) – устанавливают связи (ассоциации) “уникальный ключ – значение”.

- *пары ключ-значение указываются в словаре следующим образом:*

```
d = {key1 : value1, key2 : value2 }.
```

3. ***Множества*** – неупорядоченные наборы простых объектов.

- *присутствие объекта в наборе важнее порядка или того, сколько раз данный объект там встречается*
- *можно осуществлять проверку принадлежности, определять, является ли данное множество подмножеством другого множества, находить пересечения множеств и так далее*
- *задаются при помощи ключевого слова* `set`

Пример:

```
bri = set(['Бразилия', 'Россия', 'Индия'])
'Индия' in bri    -> True
'США' in bri    -> False
bric = bri.copy()
bric.add('Китай')
bric.issuperset(bri)   -> True
bri.remove('Россия')
bri & bric   -> {'Бразилия', 'Индия'}
```