Арифметические операции

С.В. Лемешевский (`sergey.lemeshevsky@gmail.com`)

Институт математики НАН Беларуси

on

Язык Python , благодаря наличию огромного количества библиотек для решения разного рода вычислительных задач, является конкурентом таким пакетам как Matlab и Octave. Запущенный в интерактивном режиме, он, фактически, превращается в мощный калькулятор. В этой главе речь пойдет об арифметических операциях, доступных в данном языке.

Как было сказано в предыдущей главе, посвященной типам и модели данных Python, в этом языке существует три встроенных числовых типа данных:

- целые числа (`int`);
- вещественные числа (`float`);
- комплексные числа (`complex`).

Если в качестве операндов некоторого арифметического выражения используются только целые числа, то результат тоже будет целое число. Исключением является операция деления, результатом которой является вещественное число. При совместном использовании целочисленных и вещественных переменных, результат будет вещественным.

Содержание

1. Арифметические операции с целыми и вещественными числами

Все эксперименты будем проводить в Python, запущенном в интерактивном режиме.

Сложение. Складывать можно непосредственно сами числа ...

```
>>> 3 + 2
5
```

либо переменные, но они должны предварительно быть проинициализированы.

```
>>> a = 3
>>> b = 2
>>> a + b
5
```

Результат операции сложения можно присвоить другой переменной...

```
>>> a = 3
>>> b = 2
>>> c = a + b
>>> print (c)
5
```

либо ей же самой, в таком случае можно использовать полную или сокращенную запись, полная выглядит так:

```
>>> a = 3
>>> b = 2
>>> a = a + b
>>> print (a)
5
```

сокращенная так:

```
>>> a = 3
>>> b = 2
>>> a += b
>>> print (a)
5
```

Все перечисленные выше варианты использования операции сложения могут быть применены для всех нижеследующих операций.

Вычитание.

```
>>> 4 - 2
2
```

```
>>> a = 5
>>> b = 7
>>> a - b
-2
```

Деление.

```
>>> 9 / 3
3.0
```

```
>>> a = 7
>>> b = 4
>>> a / b
1.75
```

Получение целой части от деления.

```
>>> 9 // 3
3
```

```
>>> a = 7
>>> b = 4
>>> a // b
1
```

Получение дробной части от деления.

```
>>> 9 % 5
4
```

```
>>> a = 7
>>> b = 4
>>> a % b
7
```

Возведение в степень.

```
>>> 5**4
625
```

```
>>> a = 4
>>> b = 3
>>> a**b
64
```

2. Работа с комплексными числами

Для создания комплексного числа можно использовать функцию `complex(a, b)`, в которую, в качестве первого аргумента, передается действительная часть, в качестве второго – мнимая. Либо записать число в виде `a+bj`.

Рассмотрим несколько примеров.

Создание комплексного числа.

```
>>> z = 1+2j
>>> print(z)
(1+2j)
```

```
>>> x = complex(3, 2)
>>> print (x)
(3+2j)
```

Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень.

```
>>> x+z
(4+4j)
```

```
>>> x-z
(2+0j)
```

```
>>> x*z
(-1+8j)
```

```
>>> x/z
(1.4-0.8j)
```

```
>>> x**z
(-1.1122722036363393-0.012635185355335208j)
```

```
>>> x**3
(- 9+46j)
```

У комплексного числа можно извлечь действительную и мнимую части.

```
>>> x = 3+2j
>>> x.real
3.0
```

```
>>> x.imag
2.0
```

Для получения комплексносопряженного число необходимо использовать метод `conjugate()`.

```
>>> x.conjugate()
(3-2j)
```

3. Битовые операции

В Python доступны битовые операции, их можно производить над целыми числами.

Побитовое И (AND).

```
>>> p = 9
>>> q = 3
>>> p & q
1
```

Побитовое ИЛИ (OR).

```
>>> p | q
11
```

Побитовое Исключающее ИЛИ (XOR).

```
>>> p^q
10
```

Инверсия.

```
>>> ~p
-10
```

Сдвиг вправо и влево.

```
>>> p<<1
18
```

4. Представление чисел в других системах счисления

В своей повседневной жизни мы используем десятичную систему исчисления, но при программирования, очень часто, приходится работать с шестнадцатеричной, двоичной и восьмеричной.

Представление числа в шестнадцатеричной системе.

```
>>> m = 124504
>>> hex(m)
'0x1e658'
```

Представление числа в восьмеричной системе.

```
>>> oct(m)
'0o363130'
```

Представление числа в двоичной системе.

```
>>> bin(m)
'0b11110011001011000'
```

5. Библиотека (модуль) `math`

В стандартную поставку Python входит библиотека math , в которой содержится большое количество часто используемых математических функций.

Для работы с данным модулем его предварительно нужно импортировать.

```
import math
```

Рассмотрим наиболее часто используемые функции.

Функция `math.ceil(x)`. Возвращает ближайшее целое число большее, чем `x`.

```
>>> math.ceil(3.2)
4
```

Функция `math.fabs(x)`. Возвращает абсолютное значение числа.

```
>>> math.fabs(-7)
7.0
```

Функция `math.factorial(x)`. Вычисляет факториал `x`.

```
>>> math.factorial(5)
120
```

Функция `math.floor(x)`. Возвращает ближайшее целое число меньшее, чем `x`.

```
>>> math.floor(3.2)
3
```

Функция `math.exp(x)`. Вычисляет `e**x`.

```
>>> math.exp(3)
20.08553692318766
```

Функция `math.log2(x)`. Логарифм по основанию `2`.

```
>>> math.log2(8)
3.0
```

Функция `math.log10(x)`. Логарифм по основанию 10.

```
>>> math.log10(1000)
3.0
```

Функция `math.log(x[, base])`. По умолчанию вычисляет логарифм по основанию `e`, дополнительно можно указать основание логарифма.

```
>>> math.log(5)
1.609437912434100
```

```
>>> math.log(4, 8)
0.666666666666666
```

Функция `math.pow(x, y)`. Вычисляет значение `x` в степени `y`.

```
>>> math.pow(3 , 4)
81.0
```

Функция `math.sqrt(x)`. Корень квадратный от `x`.

```
>>> math.sqrt(25)
5.0
```

Тригонометрические функции, их мы оставим без примера.

- `math.cos(x)`
- `math.sin(x)`
- `math.tan(x)`
- `math.acos(x)`
- `math.asin(x)`
- `math.atan(x)`

И напоследок пару констант.

- `math.pi` — число π.
- `math.e` — число e.

Помимо перечисленных, модуль `math` содержит ещё много различных функций, за более подробной информацией можете обратиться ена официальный сайт (`https://docs.python.org/3/library/math.html`).