Числовые типы, арифметические операции

На этом занятии мы поподробнее поговорим о представлении чисел и арифметических операциях над ними.

В Питоне имеются три базовых типа для представления чисел:

- int – для целочисленных значений;
- float – для вещественных;
- complex – для комплексных.

Мы затронем первые два: int и float. Первый целочисленный тип представляет собой, следующие числа:

0, 1, 2, 100, 6697959484, -1, -2, -7567658

Python поддерживает работу с очень большими числами, поэтому у вас, скорее всего не возникнет проблем с выходом за пределы диапазона.

Вещественные числа, то есть, дробные записываются через точку, например, так:

6.8, -5.567, 345.546, -65467.99

Здесь также довольно широкий диапазон значений, достаточный для большинства практических задач.

Основные арифметические операции

Пока такого понимания чисел будет вполне достаточно. Следующим шагом, нам с вами нужно научиться делать арифметические операции над ними. Что это за операции? Базовыми из них являются, следующие:

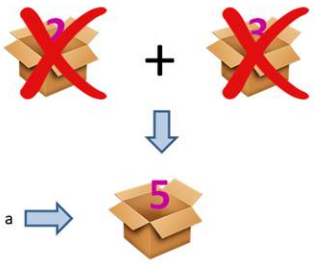
Оператор	Описание	Приоритет
+	сложение	2
-	вычитание	2
*	умножение	3
/, //	деление	3
%	остаток деления	3
**	возведение в степень	4

Давайте, я поясню их работу на конкретных примерах. Перейдем в консоль языка Python, чтобы выполнять команды в интерактивном режиме. Так будет удобнее для демонстрации возможностей вычислений. В самом простом варианте мы можем просто сложить два целых числа:

```
2+3
```

Получим результат 5. Но этот результат у нас нигде не сохраняется. Чтобы иметь возможность делать какие-либо действия с пятеркой, ее следует сохранить через переменную, например, вот так:

```
a = 2+3
```



Теперь a ссылается на объект с числом 5. Давайте разберемся, как работает эта строчка. Сначала в Python создаются два объекта со значениями 2 и 3. Оператор сложения берет эти значения, складывает их и формирует третий объект со значением 5. А, затем, через оператор присваивания, этот объект связывается с переменной a. В конце, если на объекты 2 и 3 не ссылаются никакие другие переменные, они автоматически удаляются из памяти сборщиком мусора.

Возможно, вас удивило, что при такой простой операции сложения двух чисел выполняется столько шагов. Но в Python реализовано все именно так. И это справедливо для всех арифметических операций. Мало того, раз операция сложения возвращает объект с результатом, то можно сделать и такое сложение из трех чисел:

```
b = 2+3+4
```

И так далее, можно записать сколько угодно операций сложения в цепочку.

Давайте теперь сложим целое число с вещественным:

```
c = 2 + 3.5
```

Очевидно, что результат получается тоже вещественным. Отсюда можно сделать вывод, что сложение целого числа с вещественным всегда дает вещественное значение.

```
d1 = 8 / 2
d2 = 3 / 6
```

А вот при делении двух любых чисел, мы всегда будем получать вещественное число (даже если числа можно разделить нацело):

Если же нам нужно выполнить деление с округлением к наименьшему целому, то это делается через оператор:

```
d3 = 7 // 2
```

На выходе получаем значение 3, так как оно является наименьшим целым по отношению к 3,5. Обратите внимание, что при делении отрицательных чисел:

```
d3 = -7 // 2
```

получим уже значение -4, так как оно наименьшее по отношению к -3,5. Вот этот момент следует иметь в виду, применяя данный оператор деления.

Следующий оператор умножения работает очевидным образом:

```
5 * 6
2 * 4.5
```

Обратите внимание, в последней операции получим вещественное значение 9.0, а не целое 9, так как при умножении целого на вещественное получается вещественное число.

Давайте теперь предположим, что мы хотим вычислить целый остаток от деления. Что это вообще такое? Например, если делить 10 : 3

то остаток будет равен 1. Почему так? Все просто, число 3 трижды входит в число 10 и остается значение 10 - 3·3 = 1. Для вычисления этого значения в Python используется оператор:

```
10 % 3
```

Если взять:

```
10 % 4
```

то получим 2. Я думаю, общий принцип понятен. Здесь есть только один нюанс, при использовании отрицательных чисел. Давайте рассмотрим четыре возможные ситуации: **-9%5 = 1, 9%-5 = -1, -9%-5 = -4**

Почему получаются такие значения? Первое, я думаю, понятно. Здесь 5 один раз входит в 9 и остается еще 4. При вычислении -9 % 5 по правилам математики следует взять наименьшее целое, делящееся на 5. Здесь – это значение -10. А, далее, как и прежде, вычисляем разность между наименьшим, кратным 5 и -9: -9 – (-10) = 1

При вычислении 9 % -5, когда делитель отрицательное число, следует выбирать наибольшее целое, кратное 5. Это значение 10. А, далее, также вычисляется разность: 9 – 10 = -1

В последнем варианте -9 % -5 следует снова выбирать наибольшее целое (так как делитель отрицателен), получаем -5, а затем, вычислить разность:-9 – (-5) = -4

Как видите, в целом, все просто, только нужно запомнить и знать эти правила. Кстати, они вам в дальнейшем пригодятся на курсе математики. Последняя арифметическая операция – это возведение в степень. Она работает просто:

```
2 ** 3  # возведение в куб
36 ** 0.5  # 36 в степени 1/2
```

Приоритеты арифметических операций

```
2 ** 3 ** 2 # 2^3^2 = 512
```

В последней строчке сначала 3 возводится в квадрат (получаем 9), а затем, 2 возводится в степень 9, получаем 512. То есть, оператор возведения в степень выполняется справа-налево. Тогда как все остальные арифметические операции – слева-направо. Давайте теперь посмотрим, что будет, если выполнить команду:

```
27 ** 1/3
```

Получим значение 9. Почему так произошло? Ведь кубический корень из 27 – это 3, а не 9? Все дело в приоритете арифметических операций (проще говоря, в последовательности их выполнения). Приоритет у оператора возведения в степень ** - наибольший. Поэтому здесь сначала 27 возводится в степень 1, а затем, 27 делится на 3. Получаем искомое значение 9.

Если нам нужно изменить порядок вычисления, то есть, приоритеты, то следует использовать круглые скобки:

```
27 ** (1/3)
```

Теперь видим значение 3. То есть, по правилам математики, сначала производятся вычисления в круглых скобках, а затем, все остальное в порядке приоритетов. Приведу еще один пример, чтобы все было понятно:

```
2 + 3 * 5     # 17
(2 + 3) * 5   # 25
```

То есть, приоритеты работают так, как нас учили на школьных уроках математики. Я думаю, здесь все должно быть понятно. Также не забывайте, что все арифметические операторы выполняются слева-направо (кроме оператора возведения в степень), поэтому в строчке:

```
32 / 4 * 2
```

сначала будет выполнено деление на 4, а затем, результат умножается на 2. В заключение этого занятия рассмотрим некоторые дополнения к арифметическим операторам. Предположим, что у нас имеются переменные:

```
i = 5
j = 3
i = i + 1
j = j - 2
print(i, j)
```

И, далее, мы хотим переменную i увеличить на 1, а j – уменьшить на 2. Используя существующие знания, это можно сделать, следующим образом:

```
i += 1
j -= 2
```

Результат будет прежним, но запись короче. Часто, в таких ситуациях на практике используют именно такие сокращенные операторы. То же самое можно делать и с умножением, делением:

```
i *= 3
j /= 4
print(i, j)
```