Типы данных. Операции над числами

Аннотация

В этом уроке мы познакомимся с типами данных, узнаем, какие бывают числовые типы, и научимся с ними работать.

1.Повторение

На прошлом уроке мы рассмотрели условный оператор, который позволяет выполнять различные ветки кода в зависимости от заданных условий. Научились составлять сложные условия при помощи операций `not`, `and` и `or`.

2.Типы данных

Пока единственным типом данных, с которым мы работали, были строки. Теперь нам предстоит рассмотреть целые и вещественные числа. У каждого элемента данных, который встречается в программе, есть свой тип. (В случае с Python более правильный термин — «класс объекта», но об этом мы будем говорить гораздо позже).

Например, «привет» — это строка, а вот 15.3 — это число (дробное). Даже если данные не записаны прямо в программе, а получаются откуда-то еще, у них есть совершенно определенный тип. Например, на место `input()` всегда подставляется строка, а `2 + 2` даст именно число 4, а не строку "4".

Пользователь может ввести с клавиатуры какие-то цифры, но в результате `input()` вернет строку, состоящую из этих цифр. Если мы попытаемся, например, прибавить к этой строке 1, получим ошибку.

Давайте попробуем это сделать:

```
a = input()

print(a + 1)
```

Сохраните и запустите программу. Введите любое число и посмотрите, что получится.

Ошибка возникнет потому, что в переменную `a` у нас попадает строка, а в функции `print` мы пытаемся сложить эту строку из переменной `a` и число 1. Исправьте программу так, чтобы она работала.

А если нам надо работать с числами? Мы пока будем рассматривать целые и вещественные числа.

Когда речь идет о числовых данных, они записываются **без кавычек**.

А для вещественных чисел, чтобы отделить дробную часть от целой, используют **точку**.

На прошлом занятии мы складывали две строки:

```
print('10' + '20')
```

И получали результат — строку "1020".

Давайте попробуем в этом примере убрать кавычки. В таком случае речь пойдет уже не о строках, а о двух целых числах.

И результатом функции `print(10 + 20)` будет целое число 30.

А если мы попробуем сложить два вещественных числа `print(10.0 + 20.0)`, результатом будет вещественное число 30.0.

Попробуйте предположить, что будет, если сложить вещественное число и целое число `print(10.0 + 20)`. Почему?

3.Операции над числами

Мы выполняли сложение двух чисел внутри функции `print`, но мы можем переменным давать нужные значения и выполнять действия над переменными.

Давайте напишем программу, которая задаст нужные значения двум переменным (10 и 20), потом вычислит их сумму, положит это значение в третью переменную и выведет на экран полученный результат. Допишите начальные строки, чтобы программа решала поставленную задачу:

```
...
print(summ)
```

Обратите внимание: если в качестве имени переменной для суммы взять `sum`, оно выделяется цветом. Это означает, что такое имя знакомо среде и принадлежит какой-то функции, в качестве имени переменной его лучше не

использовать.

Как складывать два числа, мы научились. Еще числа можно вычитать, умножать, делить, возводить в степень, получать целую часть от деления и остаток от деления нацело. Давайте разберем эти операции на примерах.

```
print(30 - 10)
print(30.0 - 10)
print(3 * 3)
```

С вычитанием и умножением все понятно, они аналогичны сложению.

Возведение в степень обозначается двумя звездочками **, которые должны записываться без разделителей.

```
print(9 ** 2)
```

Обратите внимание: результат деления всегда вещественный, даже если мы делим два целых числа, которые делятся нацело.

```
print(10 / 2)
```

Попробуйте поделить на 0. Посмотрите, как будет выглядеть ошибка деления на 0.

4.Целочисленное деление

Для реализации целочисленного деления существуют два действия: деление нацело и остаток от деления нацело. Получение целой части от деления обозначается как удвоенный знак деления, а остатка от деления нацело — %.

Давайте подробнее разберем эти операции. Что будет выведено в результате этих действий?

```
print(10 // 3, 10 % 3)
print(10 // 5, 10 % 5)
print(10 // 11, 10 % 11)
```

Допустим, вам известны результаты a // b, a % b и число b, напишите формулу, как найти число a?

Давайте проверим вашу формулу:

```
a = 10
b = 3
print(…А сюда напишем формулу…)
```

Обратите внимание на порядок выполнения действий в вашей формуле. Целочисленное деление имеет тот же приоритет, что и обычное деление, значит, будет выполняться раньше, чем вычитание и сложение. Для изменения приоритета выполнения операций используются скобки, все так же, как в математике.

А теперь, немного разобравшись с этими операциями, попробуйте предположить, что выведется на экран после выполнения следующего куска кода:

```
print(10 // 3, 10 % 3)
```

```
print(-10 // 3, -10 % 3)
```

Определите, что будет выведено на экран?

```
a = 4
```

```
b = 15
```

```
c = b / 5 * 3 - a
```

```
print(c)
```

Для решения задач этого урока нам потребуется получать числа, вводимые пользователем. Но при вводе мы всегда получаем строку. Значит, эту строку нужно преобразовать в число. Подробнее с преобразованием типов данных мы познакомимся на следующем уроке, а пока получить целое или вещественное число из введенной строки можно так:

```
a = int(input())  # в переменную записали целое число
b = float(input())  # в переменную записали вещественное число
```