

Арифметические операции.

Числа в python представлены такими типами данных, как **int** (от слова integer, “целый”) и **float** (от слова float, “плавающий”. Причем здесь плавающий? Вещественные числа принято называть числами с «плавающей точкой»).

Итак,

- **int**

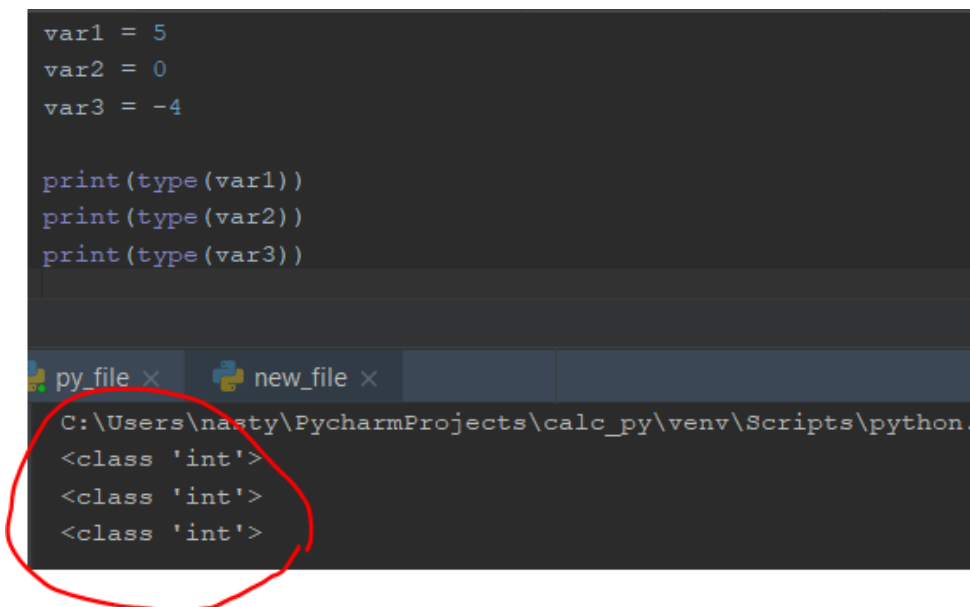
- **float**

Посмотрим это на примере.

Числа 5, 0 и -4 – целые. Так что всё верно, их тип **int**.

```
var1 = 5
var2 = 0
var3 = -4

print(type(var1))
print(type(var2))
print(type(var3))
```

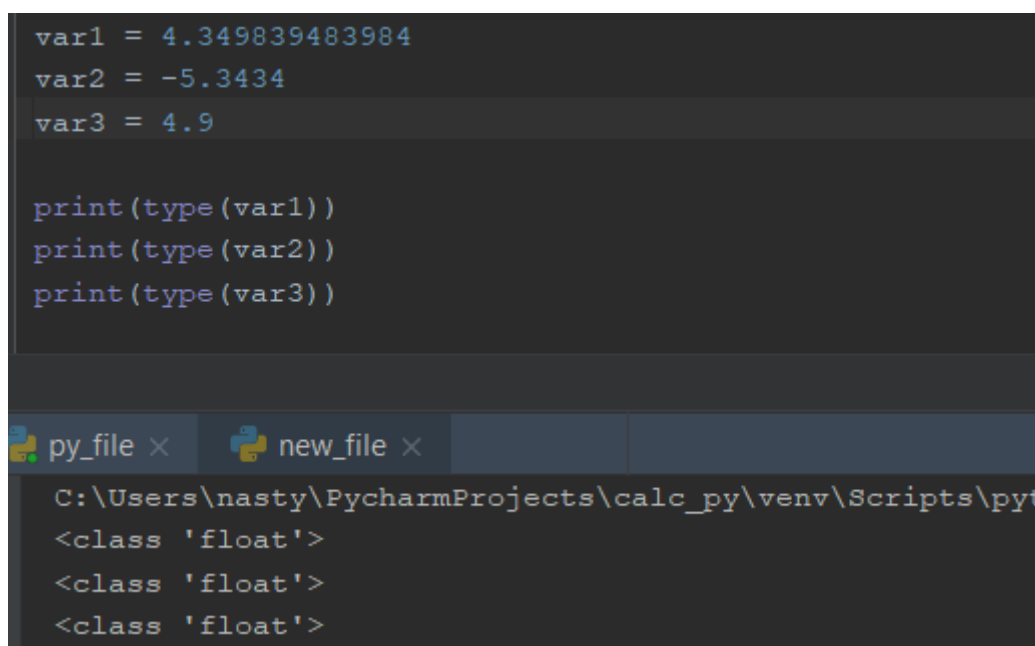


```
py_file x new_file x
C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.
<class 'int'>
<class 'int'>
<class 'int'>
```

Теперь посмотрим на вещественные числа. Дробную часть надо писать через ТОЧКУ (не запятую, а точку).

```
var1 = 4.349839483984
var2 = -5.3434
var3 = 4.9

print(type(var1))
print(type(var2))
print(type(var3))
```



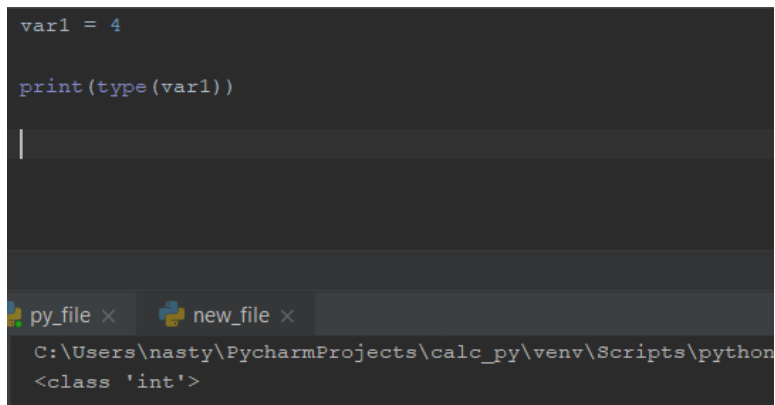
```
py_file x new_file x
C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\pyt
<class 'float'>
<class 'float'>
<class 'float'>
```

Один важный нюанс.

Это – целое число.

```
var1 = 4

print(type(var1))
```

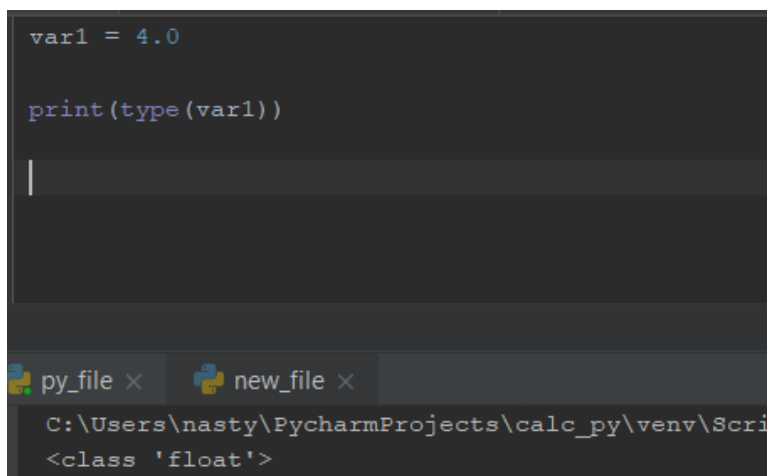


The screenshot shows a Python IDE with two tabs: 'py_file' and 'new_file'. The code in the 'py_file' tab is: `var1 = 4` and `print(type(var1))`. The output in the console is `<class 'int'>`.

А вот это – уже число типа float:

```
var1 = 4.0

print(type(var1))
```



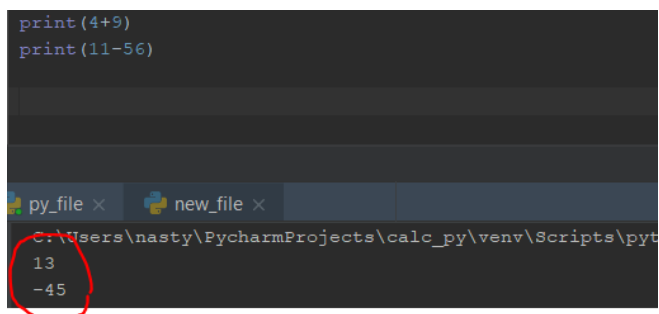
The screenshot shows a Python IDE with two tabs: 'py_file' and 'new_file'. The code in the 'py_file' tab is: `var1 = 4.0` and `print(type(var1))`. The output in the console is `<class 'float'>`.

Хотя мы понимаем, что 4 и 4.0 – это одно и то же... но нет. У второго числа есть дробная часть (пусть даже и нулевая, но она есть!).

Теперь поговорим о том, что с числами можно делать.

Складывать и вычитать:

```
print(4+9)
print(11-56)
```



The screenshot shows a Python IDE with two tabs: 'py_file' and 'new_file'. The code in the 'py_file' tab is: `print(4+9)` and `print(11-56)`. The output in the console is `13` and `-45`, which are circled in red.

Или можно сохранить значение в переменной и работать с переменными:

```
a = 56
b = 42

print(a + b)
print(a - b)
```

py_file x new_file x

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe C:/Users/nasty/PycharmProjects/calc_py/venv/Scripts/python.exe

98
14

Числа можно умножать:

```
a = 56
b = 42

print(a*b)
print(3*12)
```

py_file x new_file x

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe C:/Users/nasty/PycharmProjects/calc_py/venv/Scripts/python.exe

2352
36

Числа можно возводить в степень:

```
a = 5

print(a**2)
print(a**3)
print(a**4)

print(2**1)
print(2**3)
print(2**5)
```

py_file x new_file x

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe C:/Users/nasty/PycharmProjects/calc_py/venv/Scripts/python.exe

25
125
625
2
8
32

Числа можно делить. Но с делением не всё так просто.

1. «Обычное деление»:

```
a = 54
b = 9

print(a / b)
print(25/21)
```

py_file × new_file ×

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe
6.0
1.1904761904761905

2. Целочисленное деление:

```
a = 54
b = 10

print(a // b)
print(25//21)
```

py_file × new_file ×

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe
5
1

Как работает целочисленное деление.

В первом примере: 54 поделить на 10. Сколько раз 10 входит в число 54? 5 раз. Поэтому ответ 5.

Второй пример: 25 поделить на 21. Сколько раз число 21 входит в 25? Один раз. Поэтому ответ 1.

3. Деление с остатком.

```
a = 54
b = 10

print(a % b)
print(53 % 2)
```

py_file × new_file ×

C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\python.exe
4
1

Как работает деление с остатком.

Первый пример: сколько раз 10 входит в число 54? 5 раз ($5 * 10 = 50$). А сколько остается? 4. Эта четверка является остатком.

Второй пример: Сколько раз двойка входит в число 53? 26 раз ($26 * 2 = 52$). А сколько остается? Один. Ответ 1.

Разумеется, это не все возможные операции. Есть и вычисление корня. Вычисления значения синуса, косинуса и т.д.

Как это делать.

Вверху экрана пишем эти волшебные слова:

```
import math
```

Используя модуль `math`, можно записать таким образом возведение в степень (row сокращенно от слова `power` – степень):

```
import math

a = 5
b = 3

print(math.pow(a, b))
```

```
py_file × new_file ×
C:\Users\nasty\PycharmProjects\calc_py\venv\Scripts\p
125.0
```

Ссылка на тест:

<https://forms.gle/SuiVtaboNkEoT4YM6>