session_one___00_numbers

September 18, 2019

1 Numbers

1.1 Types of numbers

Python has various ”types” of numbers. We’ll mainly focus on integers and floating point numbers.

Integers are whole numbers, positive or negative. For example: 10 and -96.

Floating point numbers have a decimal point in them, or use an exponential (e) to define the number. For example 11.0 and -10.25. 5E4 (5 times 10 to the power of 4) is also an example of a floating point number in Python.

Now, lets get started with some basic arithmetic.

1.1.1 Addition (+)

```
[1]: 2 + 55
```

```
[1]: 57
```

```
[2]: 52.36 + 98.2
```

```
[2]: 150.56
```

1.1.2 Subtraction (-)

```
[3]: 53 - 25
```

```
[3]: 28
```

```
[4]: 24 - 48
```

```
[4]: -24
```

```
[5]: 2.65 - 2.5
```

```
[5]: 0.14999999999999999
```

1.1.3 Multiplication (*)

```
[6]: 2 * 6
```

```
[6]: 12
```

```
[7]: 2.4 * 56.34
```

```
[7]: 135.216
```

```
[8]: -2 * 5
```

```
[8]: -10
```

1.1.4 Division (/)

```
[9]: 4 / 2
```

```
[9]: 2.0
```

```
[10]: 25 / 3
```

```
[10]: 8.333333333333334
```

```
[11]: 4.85 / 2.5
```

```
[11]: 1.94
```

If you see carefully, the division of two integers returned a float and not an integer. Keep this point in mind. To obtain the an integer result, we use the floor division operator (which is explained below).

1.1.5 Floor Division (//)

Floor division returns the floor of the result, or the result of the division rounded down to the nearest integer.

`5 / 2` results in `2.5` and `2.5` rounded down to the nearest integer is `2`.

Note that, `2.5` when rounded up results in `3`.

Hence, `5 // 2` results in `2` and not `3`.

```
[12]: 5 // 2
```

```
[12]: 2
```

```
[13]: 83 // 3
```

```
[13]: 27
```

```
[14]: 2.5 // 2
```

```
[14]: 1.0
```

```
[15]: 4.856 // 3
```

```
[15]: 1.0
```

1.1.6 Modulo (%)

The modulo operator returns the remainder when division is performed between two numbers.

```
[16]: 5 % 2
```

```
[16]: 1
```

```
[17]: 2.5 % 2
```

```
[17]: 0.5
```

```
[18]: 2 % 1.5
```

```
[18]: 0.5
```

```
[19]: 22.6 % 1.5
```

```
[19]: 0.10000000000000142
```

1.1.7 Exponent (**)

```
[20]: 5 ** 2
```

```
[20]: 25
```

```
[21]: 25 ** 25
```

```
[21]: 88817841970012523233890533447265625
```

You can also compute roots using the exponent operator!

```
[22]: 4 ** 0.5
```

```
[22]: 2.0
```

```
[23]: 16 ** 0.25
```

```
[23]: 2.0
```

You can use parentheses to specify the order in which an arithmetic expression is computed.

```
[24]: # First multiplication, then addition!
      4 * 2 + 2 * 2
```

```
[24]: 12
```

```
[25]: # Here, we are using parentheses to perform addition first, then multiplication
      4 * ( 2 + 2 ) * 2
```

```
[25]: 32
```

There is also a third number type in python: `complex`.

`complex` numbers are of the form `(a + bj)` where `a` and `b` are integers are floating point numbers.

All the arithmetic operations are defined on `complex` numbers as well.

```
[26]: (2 + 3j) + (1 + 5j)
```

```
[26]: (3+8j)
```

```
[27]: (5 + 3j) - (0 + 5j)
```

```
[27]: (5-2j)
```

```
[28]: (1 + 3j) * (1 + 5j)
```

```
[28]: (-14+8j)
```

```
[29]: (10 + 9j) / (10 + 9j)
```

```
[29]: (1+0j)
```

That's all about numbers in Python!