Python Numbers

Summary: in this tutorial, you'll learn about Python numbers and how to use them in programs.

Python supports integers (https://www.pythontutorial.net/advanced-python/python-integers/) , floats (https://www.pythontutorial.net/advanced-python/python-float/) , and complex numbers. This tutorial discusses only integers and floats.

Integers

The integers (https://www.pythontutorial.net/advanced-python/python-integers/) are numbers such as -1, 0, 1, 2, 3, .. and they have type `int` .

You can use Math operators like +, -, *, and / to form expressions that include integers. For example:

```
>>> 20 + 10
30
>>> 20 - 10
10
>>> 20 * 10
200
```

```
>>> 20 / 10
2.0
```

To calculate exponents, you use two multiplication symbols (`**`). For example:

```
>>> 3**3
27
```

To modify the order of operations, you use the parentheses `()` . For example:

```
>>> 20 / (10 + 10)
1.0
```

Floats

Any number with a decimal point is a floating-point number (https://www.pythontutorial.net/advanced-python/python-float/) . The term float means that the decimal point can appear at any position in a number.

In general, you can use floats like integers. For example:

```
>>> 0.5 + 0.5
1.0
>>> 0.5 - 0.5
0.0
>>> 0.5 / 0.5
1.0
>>> 0.5 * 0.5
0.25
```

The division of two integers always returns a float:

```
>>> 20 / 10
2.0
```

If you mix an integer and a float in any arithmetic operation, the result is a float:

```
>>> 1 + 2.0
3.0
```

Due to the internal representation of floats, Python will try to represent the result as precisely as possible. However, you may get the result that you would not expect. For example:

```
>>> 0.1 + 0.2
0.30000000000000004
```

Just keep this in mind when you perform calculations with floats. And you'll learn how to handle situations like this in later tutorials (https://www.pythontutorial.net/advanced-python/python-rounding/) .

Underscores in numbers

When a number is large, it'll become difficult to read. For example:

```
count = 10000000000
```

To make the long numbers more readable, you can group digits using underscores, like this:

```
count = 10_000_000_000
```

When storing these values, Python just ignores the underscores. It does so when displaying the numbers with underscores on screen:

```
count = 10_000_000_000
print(count)
```

Output:

```
10000000000
```

The underscores also work for both integers and floats.

> Note that the underscores in numbers has been available since Python 3.6

Summary

- Python supports common numeric types including integers, floats, and complex numbers.
- Use the underscores to group numbers for the large numbers.