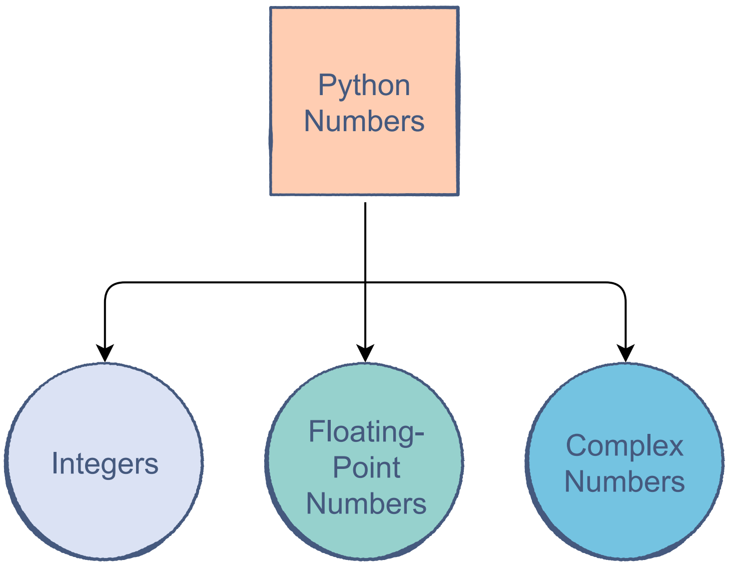
Numbers

This lesson provides an in-depth discussion about numbers in Python.

Python is one of the most powerful languages when it comes to manipulating numerical data.

It is equipped with support for several types of numbers, along with utilities for performing computations on them.

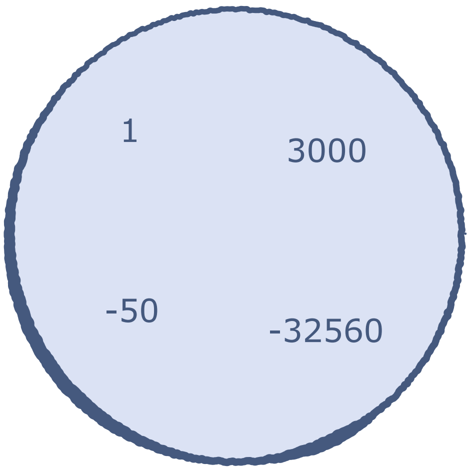
There are three main types of numbers in Python:



Integers

The integer data type is comprised of all the positive and negative whole numbers.

The amount of memory an integer occupies depends on its value. For example, `0` will take up *24 bytes* whereas `1` would occupy *28 bytes*.



Integers in Python.

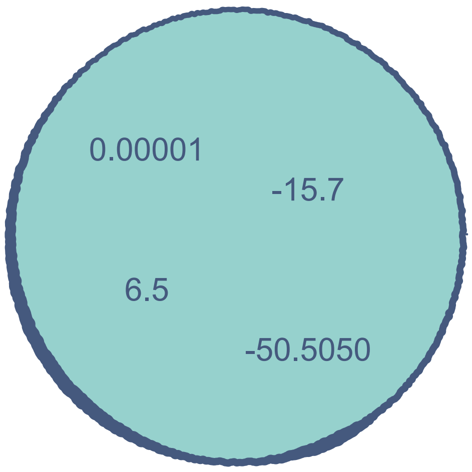
Here are some examples of integers:

```
print(10)  # A positive integer
print(-3000)  # A negative integer

num = 123456789  # Assigning an integer to a variable

print(num)
num = -16000  # Assigning a new integer
print(num)
```

Note: In Python, all negative numbers start with the `-` symbol.



Floating-point numbers in Python.

Floating Point Numbers

Floating-point numbers, or **floats**, refer to positive and negative decimal numbers.

Python allows us to create decimals up to a very high decimal place.

This ensures accurate computations for precise values.

A float occupies *24 bytes* of memory.

Below, we can find some examples of floats:

```
print(1.00000000005)  # A positive float
print(-85.6701)  # A negative float

flt_pt = 1.23456789
print(flt_pt)
```

In Python, `5` is considered to be an integer while `5.0` is a float.

Complex Numbers

Python also supports complex numbers, or numbers made up of a real and an imaginary part.

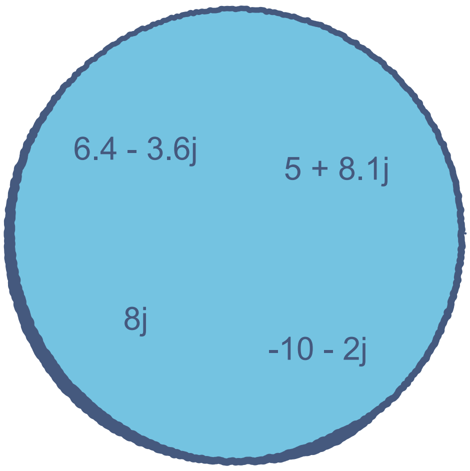
Just like the `print()` statement is used to print values, `complex()` is used to create complex numbers.

It requires two values. The first one will be the *real* part of the complex number, while the second value will be the *imaginary* part.

Here's the template for making a complex number:

```
complex(real, imaginary)
```

Let's see a few examples:



Complex numbers in Python.

```
print(complex(10, 20))  # Represents the complex number (10 + 20j)
print(complex(2.5, -18.2))  # Represents the complex number (2.5 - 18.2j)

complex_1 = complex(0, 2)
complex_2 = complex(2, 0)
print(complex_1)
print(complex_2)
```

> **Note**: In normal mathematics, the imaginary part of a complex number is denoted by `i`. However, in the code above, it is denoted by `j`. This is because Python follows the *electrical engineering* convention which uses `j` instead of `i`. Don't let that confuse you.

Complex numbers are useful for modelling physics and electrical engineering models in Python. While they may not seem very relevant right now, it never hurts to know!

to know.

A complex number usually takes up 32 bytes of memory.

The next data type we’ll study is the **Boolean**.