# SDEV140 - Introduction to Software Development

## Numeric Data Types in Python

### Integers (int)

Whole number, can be positive, negative, or zero.

https://docs.python.org/3/library/functions.html#int

### Floating Point (float)

Non-whole numbers with a decimal point. Due to the limitations of computer storage, all floating point should be considered an approximation.[1] [2]

https://docs.python.org/3/library/functions.html#float

### Complex Numbers (complex)

Numbers with both a real and imaginary part, can be represented by `3 + 2j` or `complex(3,2)`. Both parts are stored as floating point numbers.

https://docs.python.org/3/library/functions.html#complex

All three types of numbers can have mathematical operations performed on then, and can also be mixed in expressions. If numbers of two different types are being used, Python will "promote" the lower width number type to match the other. The result of the operation will be of the wider type.

> The width of numeric types goes from narrowest: int -> float -> complex: widest.

Python can be directed to convert a number of one type to another by using the int(), float(), and complex() constructors. When a floating point number is converted to an integer, it is done by truncating the decimal portion. No rounding is performed.

```
>>> int(5.99)
5
```

[1] As a basic demonstration, ask python to print the result of 10 / 3 or 0.1 + 0.2

[2] For a treatment of the subject beyond the scope of this class, see: https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

| Operator | Description | Notes |
|---|---|---|
| x + y | Addition | |
| x - y | Subtraction | |
| x * y | Multiplication | |
| x / y | Division | Returns float |
| x // y | Integer Division | Returns integer |
| x % y | Modulo (Remainder) | |
| -x | Negation | |
| x ** y | x to power of y | |

If you are ever curious about how the data type returned from an operation, you can check the type of any variable using the `type()` function.

```
>>> type(6 + 52)
<class 'int'>

>>> type(76.2 - 12)
<class 'float'>

>>> type(14 / 7)
<class 'float'>
```