

COMP
110

Operator Overloads,
Default Parameters,
and Union Types

Operator Overloads

- You can write magic methods to give operators meaning!
- Think about operators you use on numbers that you'd like to use on other objects, e.g. $+$, $-$, $*$, $/$, $<$, $<=$, etc...
- This is called **operator overloading**

Default Parameters

Say I define:

```
def add(x: int, y: int) -> int:  
    | return x + y
```

Then I call it with `add(x, y)` for any ints x and y.

But what if I wanted the option to ALSO call `add(x)` and just *assume* $y = 1$?

Default Parameters

Change it to:

```
def add(x: int, y: int = 1) -> int:  
    | return x + y
```

I can still call `add(x, y)` for any ints `x` and `y` and get `x + y`.

But I can ALSO call `add(x)` and it'll return `x + 1`.

You can do this for as many or as few parameters as you like!

Union Types

Now that I have:

```
def add(x: int, y: int = 1) -> int:  
    |  
    |     return x + y
```

Say I want this function to work for ints *or* floats...

I can express this using Union:

```
def add(x: int | float, y: int | float = 1) -> int | float:  
    |  
    |     return x + y
```

Goals + Solutions

- Make some parameters optional with default values

Default Parameters

- Have parameters that can take more than one different type

Union Types