Decorators

Check In

Take some time to check in with each other and discuss how you are doing

What we’re going to learn

- What Decorators are
- Why we would want to use them
- How we can create basic decorators
- Some real world examples of decorators

Decorators

Decorators are a feature of Python that give us simple clean syntax to call functions and methods

A decorator acts as a function that takes another function

Decorators **extend the behavior of the function they take in but don’t explicitly modify it**

In what scenarios could this be useful?

Take a look at some of the places decorators are used

Functions

In Python functions are actually represented as an object

This means that you can easily pass functions as an argument to other functions (or return a function as a return value)

```
def say_hello(name):
 return f"Hello {name}"

def say_hello_to_Bei(hello_function, name):
 return hello_function(name)

print(say_hello_to_Bei(say_hello, "Bei"))
```

Defining Functions Inside of Other Functions

We can also define a function inside of another function

```
def outer_function():

    def inner_function():
        print("Hello!")

    return inner_function

inner_function = outer_function()

inner_function()
```

An Ugly Decorator

We can write code that behaves similarly to a decorator but doesn't have the nicer syntax

Students browse: repl.it/@MakeSchool/uglydecorator?lite=true

Decorators

Basically decorators wrap around other functions and modify their behavior without messing with the original

The syntax to get the decorator to work is a little weird so Python creators added syntactic sugar to help us get the functionality without the ugliness

Decorators

The @ symbol let's you tell Python to wrap a simpler function in a decorator that you or someone else wrote

Basic Decorator Components

- Take in a function as a parameter
- Define a wrapper function that uses the input function and adds stuff to it
- Return the wrapper function
- The @ symbol to indicate to Python to use the wrapper function defined in the decorator every time a decorated function is called

Decorators

Decorators can also handle functions with arguments and return values

Let’s take a look at a more complex example

Students browse: repl.it/@MakeSchool/complexdecorators?lite=true

Let’s work on a more practical example

Decorators Applied to Classes

In addition to decorating functions, we can also decorate methods of classes

Just add an @ and the name of your decorator above the method name and you are good to go!

Let’s look at an example using the decorator we just built

MAKE SCHOOL

Look here for even more info on decorators

An interesting breakdown of how decorators work in flask

Some other info on Decorators

MAKE SCHOOL

Shout Outs

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar