Decorators & call backs

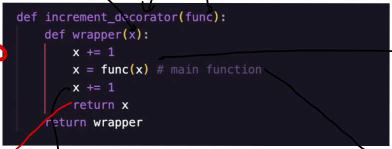
19th March, 2023

Decorators : They are like adding spices to a meal. They don't change the code. They just modify or improve it. They wrap themselves around the main code

input variable

name of decorator function (you decide)

another fxn as argument. Usually the fxn it will wrap around



```python
def increment_decorator(func):
    def wrapper(x):
        x += 1
        x = func(x) # main function
        x += 1
        return x
    return wrapper
```

Something to be done before the main fxn is called

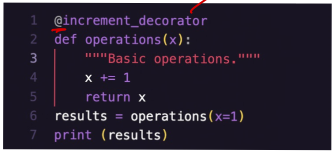
returns

Something to be done after main fxn

Calls main fxn

Using the decorator fxn

decorator fxn is applied to 'operations' fxn

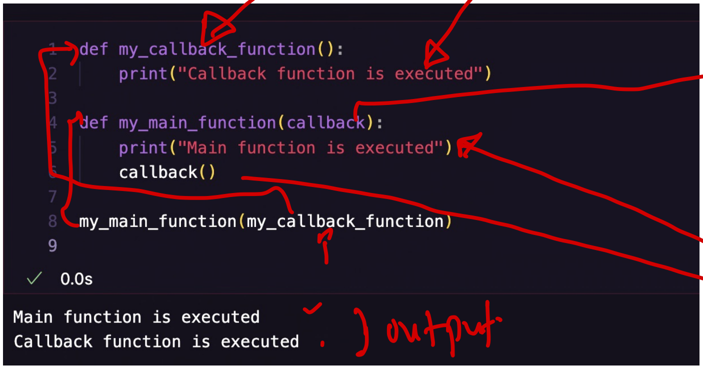


```python
@increment_decorator
def operations(x):
    """Basic operations."""
    x += 1
    return x
results = operations(x=1)
print (results)
```

When the operations (x) fxn is called, it will be wrapped in the 'wrapper' fxn created by the increment_decorator.

Callbacks

They run at specific point in the code. Just like asking someone to call you back after achieving something.



```python
def my_callback_function():
    print("Callback function is executed")

def my_main_function(callback):
    print("Main function is executed")
    callback()

my_main_function(my_callback_function)
```

```
✓ 0.0s
Main function is executed
Callback function is executed
```

Call back fxn
Does this
has a callback as argument
Do this & Do what's in the call_back
} output