

Decorators in Python

In Python, functions are the first class objects, which means that –

- Functions are objects; they can be referenced to, passed to a variable and returned from other functions as well.
- Functions can be defined inside another function and can also be passed as argument to another function.
- Decorators are very powerful and useful tool in Python since it allows programmers to modify the behaviour of function or class.
- Decorators allow us to wrap another function in order to extend the behaviour of wrapped function, without permanently modifying it.
- In Decorators, functions are taken as the argument into another function and then called inside the wrapper function.

Consider below application which demonstrate concept of Decorators

```
print("---- Marvellous Infosystems by Piyush Khairnar----")

print("Demonstration of Decorators")

# sub function is defined which accept two numbers and return subtraction.
def sub(a,b):
    print(a-b)

# SmartSub is our decorator which accept function as argument
def SmartSub(fptr):
    # Define inner function which swap the numbers depends on its value
    def inner(a,b):
        if a<b:
            a,b = b,a
        # Inner function calls our sub function and return.
        return fptr(a,b)

    # Return inner function
    return inner

sub = SmartSub(sub)

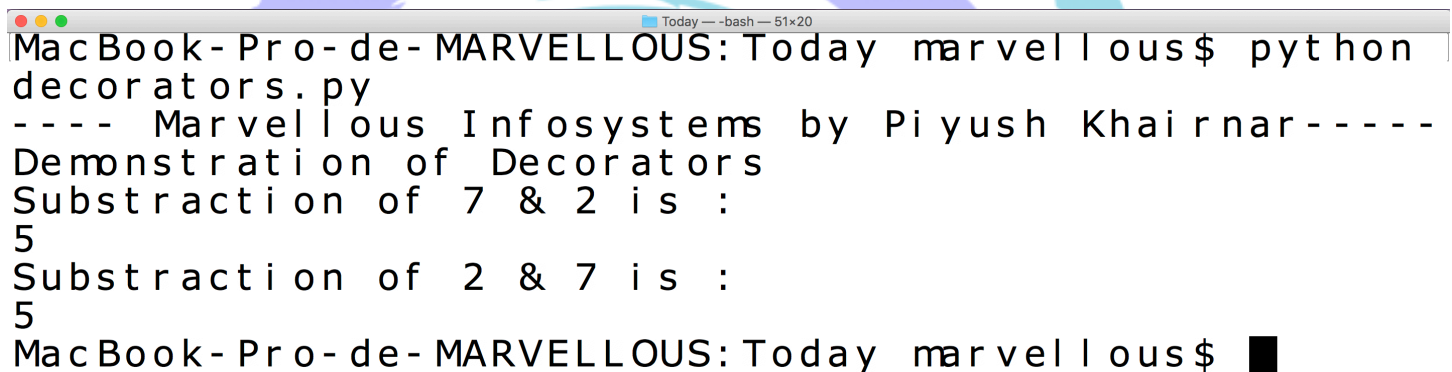
# Now we can call sub function which is our decorated function.

print("Substraction of 7 & 2 is : ")
sub(7,2)

print("Substraction of 2 & 7 is : ")
sub(2,7)
```

- In above application sub() function is already defined.
- Sub() function accepts two arguments and return its subtraction.
- If first argument is greater than the second argument then subtraction is positive and in other case it is negative.
- If we want subtraction always positive then we have to check the arguments.
- But for that we have to modify the sub() function to swap the arguments if first is smaller than the second.
- As our sub() function is already defined we can't change that function.
- For that purpose we use decorators.
- In above application SmartSub is our decorator.
- Decorator accepts our sub() function as an argument.
- Inside decorator we define one another function as inner().
- This inner() function swap the parameters depends in the values and in return call sub function.
- Our decorator return inner function so that we can indirectly call it.

Output of Above application



```
MacBook-Pro-de-MARVELLOUS:Today marvellous$ python
decorators.py
---- Marvellous Infosystems by Piyush Khairnar ----
Demonstration of Decorators
Subtraction of 7 & 2 is :
5
Subtraction of 2 & 7 is :
5
MacBook-Pro-de-MARVELLOUS:Today marvellous$
```