# → "Python Decorators"

In python decorator is a function that modifies the behaviour of another function without changing its source code. It is a way to add functionality to existing functions in a reusable & elegant manner.

A decorator is applied to a function using the @ symbol. followed by the decorator name, placed directly above the function definition. When the decorated function is called, the decorator function is executed first, modifying the function's behaviour before the original function is executed.

Eg:→

```
def my-decorator(func):
    def wrapper():
        print ("Before Calling the function.")
        func()
        print ("After calling the function.")
    return wrapper.

@my-decorator
def say-hello( ):
    print ("Hello!")

say-hello()
```

O/p:

```
Before Calling the function.
Hello!
After calling the function.
```

P.T.O

In this example, my-decorator takes the function say-hello. as an argument & returns a modified version of it called wrapper.
The @my-decorator syntax applies the decorator to say-hello, so when say-hello() is called, it is actually the wrapper function that gets executed.

Another Example :-

```
def my-decorator(some_function):
    def wrapper (num):
        print ("Inside wrapper to check odd/even")
        if num%2 ==0:
            ret ="Even"
        else:
            ret = "Odd")
        some_function(num)
        return ret
    ~~print~~print("wrapper function is called")
    return wrapper.

@my-decorator
def my-function(x):
    print("The number is =", x)

no=10
print("It is ", my-function(no))
```

O/p:
```
Wrapper function is called
Inside wrapper to check odd/Even
The number is= 10
It is Even
```