

decorators

modify or enhance functions without changing their definition

Implemented as callables
that take and return other
callables

decorators

modify or enhance functions without changing their definition

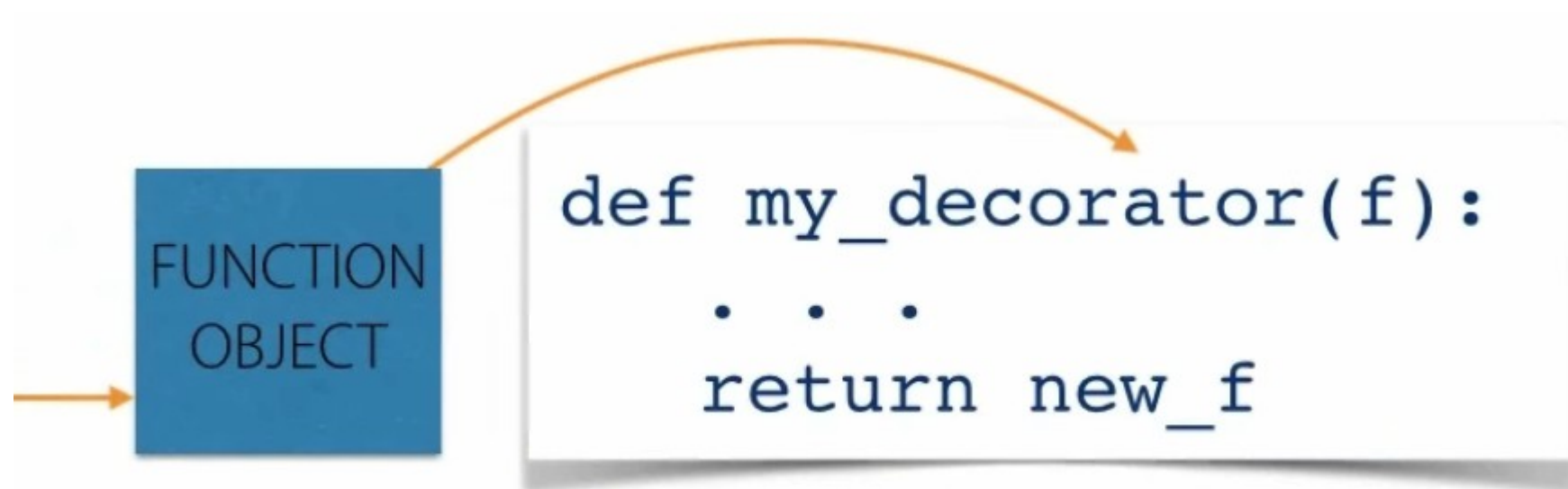
```
@my_decorator  
def my_function():  
    . . .
```

```
@my_decorator  
def my_function(x, y):  
    return x + y
```

```
@my_decorator  
def my_function(x, y):  
    return x + y
```

An orange arrow originates from the 'return x + y' line of the function definition and points to a blue box labeled 'FUNCTION OBJECT'.

FUNCTION
OBJECT



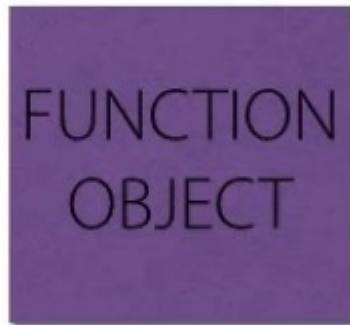


```
@my_decorator  
def my_function(x, y):  
    return x + y
```



FUNCTION
OBJECT

```
def my_function(x, y):
```

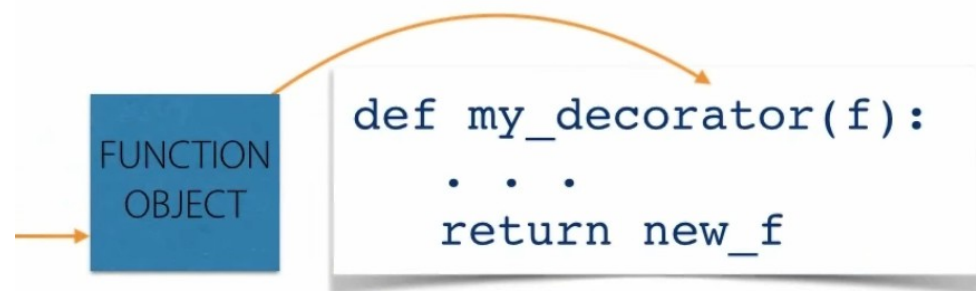


FUNCTION
OBJECT


```
@my_decorator
def my_function(x, y):
    return x + y
```

```
@my_decorator
def my_function(x, y):
    return x + y
```

FUNCTION
OBJECT



```
@my_decorator
def my_function(x, y):
    return x + y
```

FUNCTION
OBJECT

```
def my_function(x, y):
```

FUNCTION
OBJECT

- Replace, enhance, or modify existing functions
- Does not change the original function definition
- Calling code does not need to change
- Decorator mechanism uses the modified function's original name