דקורטורים

פונקציה פנימית

```
def outer(x):
        def inner(y):
                return “inner returns “ + y
        return inner(“outer  ” + x)

print(outer(“sent arg”))
```

פונקציה שמקבלת פונקציה (כבר היה...)

```
def definer(function):
        return function(“arg”)
print(definer(lambda x: x*2))
```

להזכירכם זה לא אותו דבר כמו

```
def nested(arg):
        return arg + “arg”
print(nested(f(“x”))
```

פונקציה עוטפת

נגדיר פונקציה עם מעטפת

```
def my_decorator(function):
	def wrapper(x):
		print(“we have a wrapper”)
		function(x)
		print(“and still in wrapper”)
	return wrapper

def some_func(arg):
	return “some function”
my_decorator = my_decorator(some_func)
my_decorator(5)
```

או בקיצור @decorator

```
@my_decorator
def some_func():
        return “some function”

>>> some_func()
We have a wrapper
some function
and still in wrapper
```