Decorators

Decorators are higher order functions, because they *take a function as an argument or they return a function*

So, a decorator in Python adds additional responsibilities/functionalities to a function dynamically without modifying a function.

Note: To understand Decorators you need know what is Higher Order Function and Closures

Assigning a function to variable, get function name using __name__

```
def d1(func):
    print(func)

d1("Hello Py") # Hello Py
d = d1
print(d) # <function d1 at 0x00000013C558CF70>
print(d.__name__) # d1
d("Hello Java") # Hello Java
```

```
# without Decorator
def d1(func):
    def d2():
        return "Hi", func()
    return d2

def d3():
    return "Hello Py"

d = d1(d3)
print(d) # <function d1.<locals>.d2 at 0x000000A2F0B81280>
print(d.__name__) # d2
print(d()) # ('Hi', 'Hello Py')
```

```
# with decorator
def d1(func):
    def d2():
        return "Hi", func()
    return d2

# with decorator
@d1
def d3():
    return "Hello Py"
print(d3()) # ('Hi', 'Hello Py')

# without decorator
# def d3():
#     return "Hello Py"
# d = d1(d3)
# print(d) # <function d1.<locals>.d2 at 0x000000A2F0B81280>
# print(d.__name__) # d2
# print(d()) # ('Hi', 'Hello Py')
```

```
# With Decorator  and Without Decorator using Parameters
def d1(func):
    def d2(a,b):
        return func(a*b)
    return d2

@d1 # d = d1(d4) --> d(10,10) --> 100
def d4(c):
    return c
print(d4(10,10)) # 100

def d5(d):
    return d
d = d1(d5)
print(d(5,5)) # 25
```

```
# with and without decorator using parameters
def d1(func):
    def d2(userName, passWord):
        return func(userName, passWord)
    return d2

@d1 # d = d1(d4) # d("sai", "kiran"))
def d3(uName, pWord):
    return uName, pWord
print(d3('hari', 'vinod'))
print(d3('manoj', 'jagadesh'))

def d4(uName, pWord):
    return uName, pWord
d = d1(d4)
print(d("sai", "kiran"))
```

```
('hari', 'vinod')

('manoj', 'jagadesh')

('sai', 'kiran')
```

```
# Function as argument
def d1(text):
    return text.upper()

def d2(text):
    return text.lower()

def d3(name):
    result = name("Python")
    print(result)

d3(d1) # PYTHON
d3(d2) # python
```