In [2]:
```
#Example for normal function calling
def addexclamation(function):
    def add():
        func = function()
        return func +" !!!"
    return add
def sentence():
       return "hello all"
msg = addexclamation(sentence)
print(msg())
```

```
hello all !!!
```

Out[2]:
```
'#Example of using decorators on same above example\ndef addexclamation(functio
n):\n def add():\n func = function()\n return func +" !!!"\n return add\n@addex
clamation\ndef sentence():\n return "hello all"\nprint(sentence())'
```

In [13]:
```
#Example of using decorators on same above example
def addexclamation(function):
    def add():
        func = function()
        return func +" !!!"
    return add
@addexclamation
def sentence():
     return "hello all"
print(sentence())
```

```
hello all !!!
```

In [18]:
```
#Applying multiple decorators on same callable
def addstar(func):
    def star():
        return "*"+func()+"*"
    return star
@addstar
@addexclamation
def sentence():
    return "hello all"
print(sentence())
```

```
*hello all !!!*
```

In [19]:
```
#Example for decorator with arguments
def args_function(func):
    def getargs(arg1,arg2):
        print(arg1,arg2)
        func(arg1,arg2)
    return getargs
@args_function
def decorator_with_args(num1,num2):
    print("arguments are {} and {}".format(num1,num2))
decorator_with_args(5,6)
```

```
5 6
arguments are 5 and 6
```

In [25]:
```
def decor(func):
    def inner(name):
        print("First Decor(decor) Function Execution")
        func(name)
    return inner

def decor1(func):
    def inner(name):
        print("Second Decor(decor1) Execution")
        func(name)
    return inner
@decor1
@decor
def wish(name):
    print("Hello",name,"Good Morning")
wish("Durga")
```

```
Second Decor(decor1) Execution
First Decor(decor) Function Execution
Hello Durga Good Morning
```

In [ ]: