In [ ]:
```
# Decoarator -  function inside the other function we call as decorator functi
```

In [1]:
```
def greet(name):

    return name
```

In [2]:
```
greet("Hello, Morning")
```

Out[2]:
```
'Hello, Morning'
```

In [4]:
```
def greeting(name):

    def inside_function():
        return name
    return inside_function
```

In [7]:
```
g = greeting("Hello, Good morning")
print(g())
```

```
Hello, Good morning
```

In [8]:
```
def calculate(n1,n2):

    def calc():
        return (n1 * n2)
    return calc
```

In [9]:
```
c = calculate(10,10)
```

In [14]:
```
print(c())
```

```
100
```

In [17]:
```
def outsider(func):

    def indside():
        print("its decorated function")
        func()
    return indside

def order():
    print("I am the order function")

#decoarate function
d = outsider(order)
# call the decorated
d()
```

```
its decorated function
I am the order function
```

In [19]:
```
def outsider(func):

    def inside():
        print("its decorated function")
        func()
    return inside
```

In [20]:
```
@outsider
def order():
    print("I am the order function")
```

In [21]:
```
order()
```

```
its decorated function
I am the order function
```

In [24]:
```
@outsider
def cal():
    print(2+2)
```

In [25]:
```
cal()
```

```
its decorated function
4
```

In [26]:
```
def division(func):

    def inner_function(a,b):
        return func(a,b)

    return inner_function
```

In [27]:
```
@division
def divide(a,b):
    print(a/b)
```

In [29]:
```
divide(10,5)
```

```
2.0
```

Generator

In [30]:
```
def gen(n):

    i = 0
    while i < n:
        yield i
        i +=1
```

In [32]:
```
for j  in gen(10):
    print(j)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [48]:
```
generators = gen(10)
```

In [35]:
```
next(generators)
```

Out[35]:
```
0
```

In [36]:
```
next(generators)
```

Out[36]:
```
1
```

In [37]:
```
next(generators)
```

Out[37]:
```
2
```

In [51]:
```
for i in range(3):
    print(next(generators))
```

```
4
5
6
```

In [55]:
```
next(generators)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[55], line 1
----> 1 next(generators)

StopIteration:
```

In [ ]:
```
### Iterator
```

In [70]:
```
lst = [2,4,6,8,10]

# iter-->
# next-->

iters = iter(lst)
```

In [68]:
```
for i in iters:
    print(i)
```

```
2
4
6
8
10
```

In [71]:
```
next(iters)
```

Out[71]:
```
2
```

In [ ]:
```
## list comprehensive
```

In [79]:
```
names = ["john","bob","Tom","billy","Mark","john","john"]

namelist = []
for name in names:
    if 'john' in name:
        namelist.append(name)
namelist
```

Out[79]:
```
['john', 'john', 'john']
```

In [ ]:
```

```

In [ ]:
```
# Syntax

new = [ expression  for item in iterable if condition = True]
```

In [83]:
```
newnamelist = [x for x in names if 'john' in x ]
```

In [84]:
```
newnamelist
```

Out[84]:
```
['john', 'john', 'john']
```

In [94]:
```
names = ("john","bob","Tom","billy","Mark","john","john")

tup = (x for x in names if 'john' in x )
```

In [95]:
```
for i in tup:
    print(i)
```

```
john
john
john
```

In [96]:
```
tuple(i for i in (1,2,3))
```

Out[96]:
```
(1, 2, 3)
```

In [99]:
```
names = ("john","bob","Tom","billy","Mark","john","john")

tuple(x for x in names if 'b' in x )
```

Out[99]:
```
('bob', 'billy')
```

In [100]:
```
# Dict

keys = ['a','b','c']
values = [1,2,3]

mydict = { k:v for (k,v) in zip(keys,values)}
mydict
```

Out[100]:
```
{'a': 1, 'b': 2, 'c': 3}
```

In [ ]:
```

```