# Generator

Python generators are a simple way of creating iterators.

1st cell
**create**

① 
```
def gen-demo():
    yield "first statement"
    yield s"second statement"
    yield "third statement"
```

2nd cell
**object**
```
gen = gen demo()
for i in gen:
    print(i)
```

O/P
```
first statement
second   "
third    "
```

② 1st cell
```
def square(num):
    for i in range(1, num +1):
        yield i**2
```

2nd cell
```
gen = square (10)
print (next (gen))
print (next (gen))
print (next (gen))
for i in gen:
    print(i)
```

O/P
```
1
4
9
16
25
36
49
64
810
```

## Range Function using Generator

) 1st cell
```
def own_range (start, end):
        for i in range(start, end):
            yield i
```

2nd cell
```
for i in own_range(15,26):
    print(i)
```

```
15
16
17
18
19
20
21
22
23
24
25
```

## DECORATOR

A decorator is a function that takes another function as an argument and return a new function that modifies the behavior of the original function. The new function is often referred to as a "decorated" function.

Syntax
```
@ decorator_function
def my_function():
    pass
```

(1) Write a decorator function that double the result of a given function.
Test case: given number is 3 and 5, output will be 16

```
def double_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result*2
    return wrapper.

@double_result
def add(a,b)
    return a+b
result = add(3,5)
print(result)
```

<u>Output</u>

```
16
```

② Write a decorator function that checks
a given number is even before executing the
function. If the number is odd, print a messa
and skips executing the function. When numbe
is even, take square value of it.

```
def check_even(func):
    def wrapper(num):
        if num%2==0:
            return func(num)
        else:
            print(f"Skipping function execution f
                        odd number: {num}")
    return wrapper.

@check_even
def square(num):
    print(f"Square function execution fo
                      number:{num**2
square(5)
O/P Skipping ___________
  square(8)
  O/P Square ___________
```

## Error

1)
```
x = "My building no is :"
y = 67/B
z = x+y
print(z)
```
O/P
```
Name Error: name 'B' is not defined.
```

2)
```
x = "hello"
y = "python"
v = x+y
print(v)
```
O/P hellopython

3)
```
x = "My ~~name~~ roll no is :"
y = 12
z = x+y
print(z)
```
O/P
```
TypeError: can only concatenate str (not int) to str.
```

4)
```
x = "My roll no:"
y = 12
z = x+ str(y)
print(z)
```
O/P
```
My roll no : 12
```

5)
```
x = "hello"
y = int(x)
```
O/P
```
ValueError: invalid literal for int() with base 10 : 'hello'
```