09-30-2024 Python || Generator functions

```
def v1():
    yield 10;

print(v1())
```

Output:

> *<generator ovject b1 at 0x000001B31A6F2CF0>*

Trying to return multiple values from regular functions doesn't work

```
def v1():
    return 10;
    return 20;

print(v1())
```

Even if we try to return multiple values vy using a loop it does not work.

```
def v1():
    return 10
    return 20

for i in v1():
    print(i)
```

We can use generator functions to print these multiple return values.

- Yield is an identifier that we can use to make the program pause at a specific level

```
def v1():
    yield 10
    yield 20

for i in v1():
    print(i)
print(v1())

v = v1()
print(next(v))
print(next(v))
print(next(v))
```

Example of Generator Function.

We can use this generator function to generate an output out of an infinite loop step by step.

```python
def num1():
    n = 2
    while True:
        yield n
        n= n*n

n1 = num1()
print(next(n1))
print(next(n1))
print(next(n1))
print(next(n1))
print(next(n1))
for i in num1():
    print(i)
```

If we try to modify the function to return values, we won't be able to pause the output and the output directly throws all values, which requires more memory.

```python
def num1():
    n = 2
    while True:
        n = n*n
        print(n)

num1()
```

Decorators in Python

```python
def m():
    def i():
        print("I am from Inner")
    return i()


def normal():
    print("I am from Normal")

m()
normal()
```

Using Decorators in Python

```python
def m(func):
    def i():
        print("I am from Inner")
        func()
    return i
```

```python
def normal():
    print("I am from Normal")

m1 = m(normal)
m1()
```

Using Annotations in Python

```python
def m(func):
    def i():
        print("I am from Inner")
        func()
    return i

@m
def normal():
    print("I am from Normal")

normal()
```

```python
def divide(func):
    def inner(a,b):
        print("Divide value is :",a,"and",b)
        if(b==0):
            print("Cannot Divide by 0")
            return
        return func(a,b)
    return inner

@divide
def div(a,b):
    print(a/b)

div(10,2)
div(5,0)
```

Destructor functions in Python

```
```

Lambda functions in Python

- These are also called as anonymous functions.

```
```