Generator

- ✓ Generator are functions we can create our own iterations
- ✓ Generator will return sequence of values
- ✓ In Generators we use yield keyword to return elements instead of return keyword
- ✓ In generator we cannot include return keyword, if we do it, then it will terminate the function.
- ✓ The common diff b/w yield and return is
  yield returns a value and pauses the execution while maintaining the internal states,
  Whereas return keyword returns a value and terminates the execution of the function

```
# Ex1 Using return keyword
def d1():
   return 10
   return 20
   return 30

print(d1()) # 10
print(type(d1)) # <class 'function'>
```

```
# Ex2 Using return keyword
def d2():
   return 10, 20, 30

print(d2()) # (10, 20, 30)
print(type(d2)) # <class 'function'>
```

```
# Ex3 Using yield keyword
def d3():
    yield 10
    yield 20
    yield 30

print(d3()) # <generator object d3 at 0x0000024F7B887120>
print(type(d3)) # <class 'function'>

d = d3()
next() method returns the next item from the iteration
print(next(d)) # 10
print(next(d)) # 20
print(next(d)) # 30
# print(next(d)) # StopIteration

for i in d3():
    print(i, end=' ') # 10 20 30
```

```
# Ex4 Generator stops executing bcoz return terminated the function.
def d4():
    yield 10
    return
    yield 20

d = d4()
print(d) # <generator object d4 at 0x000001D6C5B17120>
print(next(d)) # 10
print(next(d)) # StopIteration
```

```
# Ex 5 forloop Using generator function
def d5(n):
    for i in range(n):
        yield  i

d = d5(2)
print(next(d)) # 0 sending first value
print(next(d)) # 1 sending second value
print(next(d)) # error
```

```
0
1
Traceback (most recent call last):
  File "D:\Github\PythonWorkspace\Day18_Comprehensions\G7.py", line 52, in
<module>
    print(next(d)) # error
StopIteration
```

Use forloop

```
for i in d5(2):
    print(i, end = " ")
```

output

```
0 1
```

Note:

The advantage of the generator over the iterator is that elements are generated dynamically. Since the next item is generated only after the first is consumed, it is more efficient than the iterator