Generators

```
#Ex1 Find the Range of 5

print(range(5)) # range(0, 5)
```

```
#Ex2 Find the range of 5
r = range(5)
for i in r:
    print(i, end = " ") # 0 1 2 3 4
```

```
#Ex3 Multiply the range using forloop

def d1(n):
    lst = []
    for i in range(n):
        lst.append(i ** 2)
    return lst

for j in d1(5):
    print(j, end = ' ') # 0 1 4 9 16

# i = n **2, i = 0 ** 2 = 0
# i = n **2, i = 1 ** 2 = 1
# i = n **2, i = 2 ** 2 = 4
# i = n **2, i = 3 ** 2 = 9
# i = n **2, i = 4 ** 2 = 16
```

```python
#Ex4 Multiply the range using forloop with list comprehensions

def d1():
    for x in [n ** 2 for n in range(5)]:
        print(x, end = " ") # 0 1 4 9 16

d1()


# i = n **2, i = 0 ** 2 = 0
# i = n **2, i = 1 ** 2 = 1
# i = n **2, i = 2 ** 2 = 4
# i = n **2, i = 3 ** 2 = 9
# i = n **2, i = 4 ** 2 = 16
```

```python
#Ex5 Multiply the range using forloop with map function

def d1():
    for x in map((lambda n : n **2), range(5)):
        print(x, end = " ") # 0 1 4 9 16

d1()


# i = n **2, i = 0 ** 2 = 0
# i = n **2, i = 1 ** 2 = 1
# i = n **2, i = 2 ** 2 = 4
# i = n **2, i = 3 ** 2 = 9
# i = n **2, i = 4 ** 2 = 16
```

```python
# Ex6 Using Generators
def d1(n):
    for i in range(n):
        yield i ** 2 # i = n **2, i = 0 ** 2 = 0

d = d1(5)
print(d) # <generator object d2 at 0x000001F92F687120>
print(type(d)) # <class 'generator'>

# next() method returns the next item from the iteration
print(next(d)) # 0
print(next(d)) # 1
print(next(d)) # 4
print(next(d)) # 9
print(next(d)) # 16
# print(next(d)) # StopIteration

for i in d1(5): # loading all sequence at a time
    print(i, end = " ") # 0 1 4 9 16

# i = n **2, i = 0 ** 2 = 0
# i = n **2, i = 1 ** 2 = 1
# i = n **2, i = 2 ** 2 = 4
# i = n **2, i = 3 ** 2 = 9
# i = n **2, i = 4 ** 2 = 16
```

Output

```
<generator object d1 at 0x0000023397A97120>
<class 'generator'>
0
1
4
9
16
0 1 4 9 16
```