PYTHON

From Simple to Complex
With
Examples

AYESHA NOREEN

Bachelor's in Software Engineering,

Master's in Computer Science

from COMSATS University, Islamabad

NOTE!!!

In these notes Screenshots of practice examples and coding are added. The code files are also available in code folder that contain .ipynb files that are created on Jupyter notebook.

Chapter17
Generators

Generators are same as list but list when list is created it takes memory than store whole list in memory and generator generate one number every time which we require. so generator take less memory and have high performance as compare to list.

We use generators when we have to use data only once because it deletes data and use list when we have to manipulate with our data many times because list stores data.

Example

```
def nums(n):
    for i in range(1,n+1):
        yield i #yield keyword is used to generate generator
print(nums(10)) #it prints object of generator
numbers=nums(10)
numbers=list(nums(10))
print(numbers)
```

```
<generator object nums at 0x000001DDF7B270D0>
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

TODO Task

Take any number than print even numbers between o and number by using generator.

```
n=int(input("Enter any number: "))
def even(num):
    for i in range(1,num+1):
        if i%2==0:
            yield i
evens=even(n)
for e in evens:
    print(e)
#we can also do above function as
def even(num):
    for i in range(2,num+2,2): #here we use step argument it takes step of 2
            yield i
evens=even(n)
for e in evens:
    print(e)
```

```
Enter any number: 6
2
4
6
2
4
6
```

- **Generator Comprehension**

Generator comprehension is same as list comprehension but instead of square brackets parenthesis are used. e.g.

```
#Generator comprehension
square=(i**2 for i in range(1,11))
print(square) #it prints object of generator
for i in square: #it prints square of items from 1 to 10
    print(i)
```

```
<generator object <genexpr> at 0x000001DDF7B27760>
1
4
9
16
25
36
49
64
81
100
```

- # Generator VS List

List take more time as compare to generator and having less performance .List uses more memory than generator and we use list only when we have to use sequence of data many times and we use generator when we have to use sequence of data only one time.

```python
#list vs generator
import time
t1=time.time()
l=[i**2 for i in range(1000000)] #10 lac
t2=time.time()
t=t2-t1
print(f"This program took {t} sec to execute through list")
t1=time.time()
g=(i**2 for i in range(1000000)) #10 lac
t2=time.time()
t=t2-t1
print(f"This program took {t} sec to execute through generator")
```

```
This program took 5.331292629241943 sec to execute through list
This program took 0.0009996891021728516 sec to execute through generator
```