III B.Tech.

Computer Science & Engineering

CSE304: PYTHON PROGRAMMING WITH WEB FRAMEWORKS

Generator Functions, Generator Expressions and Factory Functions

By

Mrs. S. KAMAKSHI, AP-III / CSE

School of Computing

Generator Functions

- Coded as normal def statements but use yield statements to return results one at a time, suspending and resuming their state between each call
- Function Definition:

```
def gensquares(N):
    for i in range(N):
        yield i ** 2            # Resume here later
```

- Function call:

```
for i in gensquares(5):          # Resume the function
    print(i, end=' : ')           # Print last yielded value
```

```
0 : 1 : 4 : 9 : 16 :
x = gensquares(2)
 x
<generator object gensquares at 0x000000000292CA68>
 next(x)
0
next(x)
1
```

Generator Expressions

- **S**imilar to the comprehensions, but they return an object that produces one item at a time on demand instead of building all items at once
- It has single iteration reference
  - G1 = (x*x for x in range(10))

    for x in G1:

        print(x,end=':')
- Creating set, list, tuple, dictionary from generator expression
  - set (x*x for x in range(10))
  - list (x*x for x in range(10))
  - tuple (x*x for x in range(10))
  - dict((x,x*x) for x in range(10)

Nested Generator Expressions

- (x * 2 for x in (abs(x) for x in (−1, −2, 3, 4)))

- dict((x, x*2) for x in (x.lower() for x in ('ABCxyz')))
  - {'a': 'aa', 'b': 'bb', 'c': 'cc', 'x': 'xx', 'y': 'yy', 'z': 'zz'}

- L1 = [1, 2, 3, 4]
- L2 = [10, 20, 30, 40]
- L3 = [100, 200, 300, 400]
- nested_expr = ((x,(y,z)) for x, y, z in zip(L1, L2, L3))
- D1 = dict(nested_expr)
- D1

Factory Functions

- Returning the generator function object from a nested function
- Example

```
def table(N):
    def term(x):        # Nested function
        for i in range(1,N):
            yield i, x, x*i
    return term
table_N = table(17)

Sixth_table=table_N(6)
for x, y, z in Sixth_table:
    print (x, 'x', y, '=', z)

Second_table = table_N(2)
for x, y, z in Second_table:
    print (x, 'x', y, '=', z)
```

Fibonacci Sequence using Factory Function

```
def fibo(N):
    f0 = 0
    f1 = 1
    print(f0, ',', f1, end=', ')
    def next_num():
        nonlocal f0, f1
        for i in range(2, N+1):
            f2 = f0 + f1
            yield f2
            f0 = f1
            f1 = f2
    return next_num
get_next = fibo(20)
for x in get_next():
    print(x, end=', ')
```