



컴퓨팅 사고력을 키우는 SW 교육

파이썬

Chapter 06

제어문과 연관된 함수

- 01 range – 수열의 생성
- 02 리스트 항목과 인덱스 값을 동시에 얻는 방법
- 03 리스트 내장
- 04 반복문 작성시 도움이 되는 함수
- 05 효율적인 순회 방법



Section 01 range[수열의 생성]

range (1)

- **range**(['시작값', '종료값', '증가값'])

- 수열을 순회하는 이터레이터 객체를 반환 (단, 종료 값 제외)
- 종료값 필수 항목, 시작값과 증가값은 생략 가능하며, 이때는 각 0, 1이 할당

```
>>> list(range(10)) # 종료값만 있는 경우
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> list(range(5, 10)) # 시작값, 종료값만 있는 경우
```

```
[5, 6, 7, 8, 9]
```

```
>>> list(range(10, 0, -1)) # 시작값, 종료값, 증가값이 있는 경우
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
>>> list(range(10, 20, 2)) # 10부터 20까지 짝수만 출력
```

```
[10, 12, 14, 16, 18]
```

```
>>> for i in range(10, 20, 2):
```

```
    print(i)
```

```
10, 12, 14, 16, 18
```



Section 02 리스트 항목과 인덱스 값을 동시에 얻는 방법

리스트항목과 인덱스 값을 동시에 얻는 방법 (1)

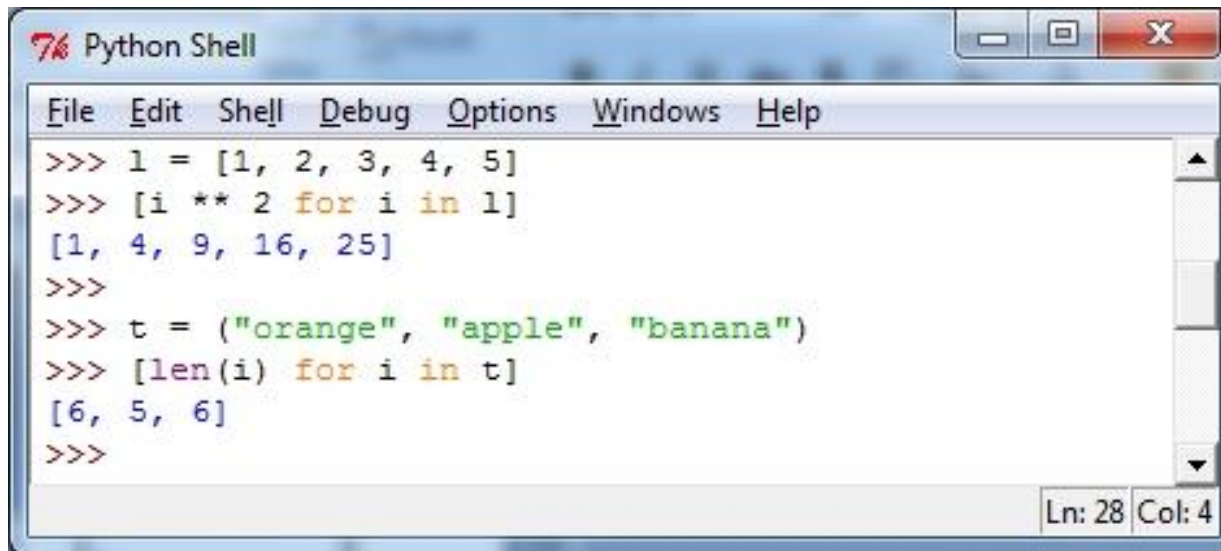
- `enumerate('시퀀스 타입 객체' [, ' 시작값' = 0])`
 - 함수를 실행하면 tuple 형태로 (인덱스, 시퀀스 객체의 아이템) 반환함
 - **시퀀스 타입 객체 : list, set, tuple, dictionary, string** 등의 객체를 말함
- ```
>>> L = [100, 15,5, "Apple"]
>>> for i in enumerate(L):
 print(i)
(0, 10)
(1, 15,5)
(2, 'Apple')
>>> L = [10, 15,5, "Apple"]
>>> for i, v in enumerate(L, 101):
 print(i)
101 10
102, 15,5
103, 'Apple'
```



## Section 03 리스트 내장

# 리스트 내장 (1)

- [**<표현식> for <아이템> in <시퀀스 객체> (if <조건식>)**]
  - 기존 시퀀스 객체를 이용하여 추가적인 연산을 통하여 새로운 리스트 객체를 생성

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following code and output:

```
>>> l = [1, 2, 3, 4, 5]
>>> [i ** 2 for i in l]
[1, 4, 9, 16, 25]
>>>
>>> t = ("orange", "apple", "banana")
>>> [len(i) for i in t]
[6, 5, 6]
>>>
```

The status bar at the bottom right indicates "Ln: 28 Col: 4".

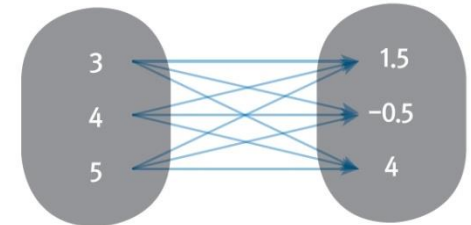
## 리스트 내장 (2)

- [**<표현식>** **for** **<아이템>** **in** **<시퀀스 객체>** (**if** **<조건식>**)]
  - 조건식을 이용하여 원본 객체에서 조건을 만족하는 아이템만 선별

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> t = ("orange", "apple", "banana", "kiwi")
>>> [i for i in t if len(i)>5]
['orange', 'banana']
>>>
>>>
>>>
Ln: 39 Col: 4
```

- 원본 리스트가 2개인 경우

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L_1 = [3, 4, 5]
>>> L_2 = [1.5, -0.5, 4]
>>> [x*y for x in L_1 for y in L_2]
[4.5, -1.5, 12, 6.0, -2.0, 16, 7.5, -2.5, 20]
>>> |
Ln: 43 Col: 4
```



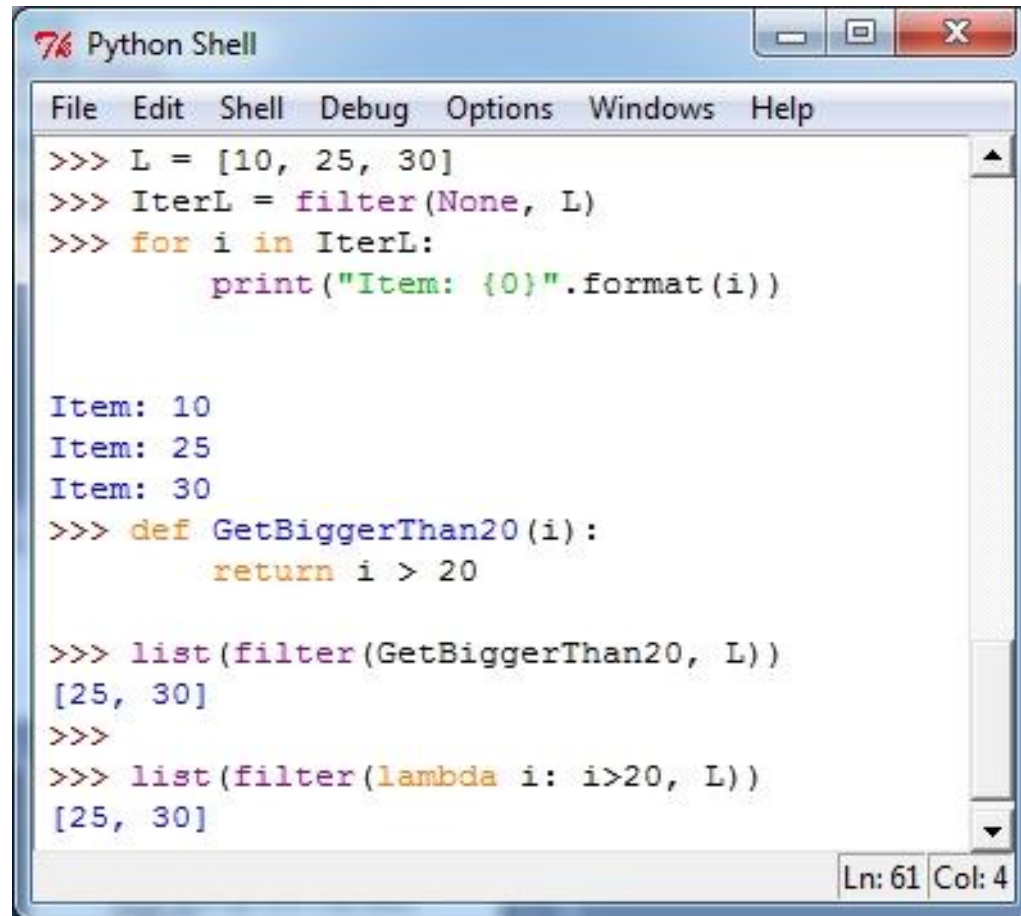




## Section 04 반복문 작성시 도움이 되는 함수

# 반복문 작성시 도움이 되는 함수 (1)

- **filter**(<function>|None, 시퀀스 객체)
  - 함수의 결과 값이 참인 시퀀스 객체의 이터레이터를 반환
  - None이 오는 경우 필터링하지 않음



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L = [10, 25, 30]
>>> IterL = filter(None, L)
>>> for i in IterL:
 print("Item: {0}".format(i))

Item: 10
Item: 25
Item: 30
>>> def GetBiggerThan20(i):
 return i > 20

>>> list(filter(GetBiggerThan20, L))
[25, 30]
>>>
>>> list(filter(lambda i: i>20, L))
[25, 30]
Ln: 61 Col: 4
```

## 반복문 작성시 도움이 되는 함수 (2)

- **zip**(시퀀스 객체1, 시퀀스 객체2, ..)
  - 함수의 리턴 값은 인자로 시퀀스 객체들의 각 원소 쌍들을 튜플 형태로 묶은 형태임

```
>>> X = [10, 20, 30]
>>> Y = ['A', 'B', 'C']
>>> for i in zip(X, Y):
 print("Item: {}".format(i))
```

```
Item: (10, 'A')
```

```
Item: (20, 'B')
```

```
Item: (30, 'C')
```

```
>>> X = [10, 20, 30]
>>> Y = ['A', 'B', 'C']
>>> RetList = list(zip(X, Y))
[(10, 'A'), (20, 'B'), (30, 'C')]
>>> X2, Y2 = zip(*RetList)
>>> X2
(10, 20, 30)
>>> Y2
('A', 'B', 'C')
```

## 반복문 작성시 도움이 되는 함수 (3)

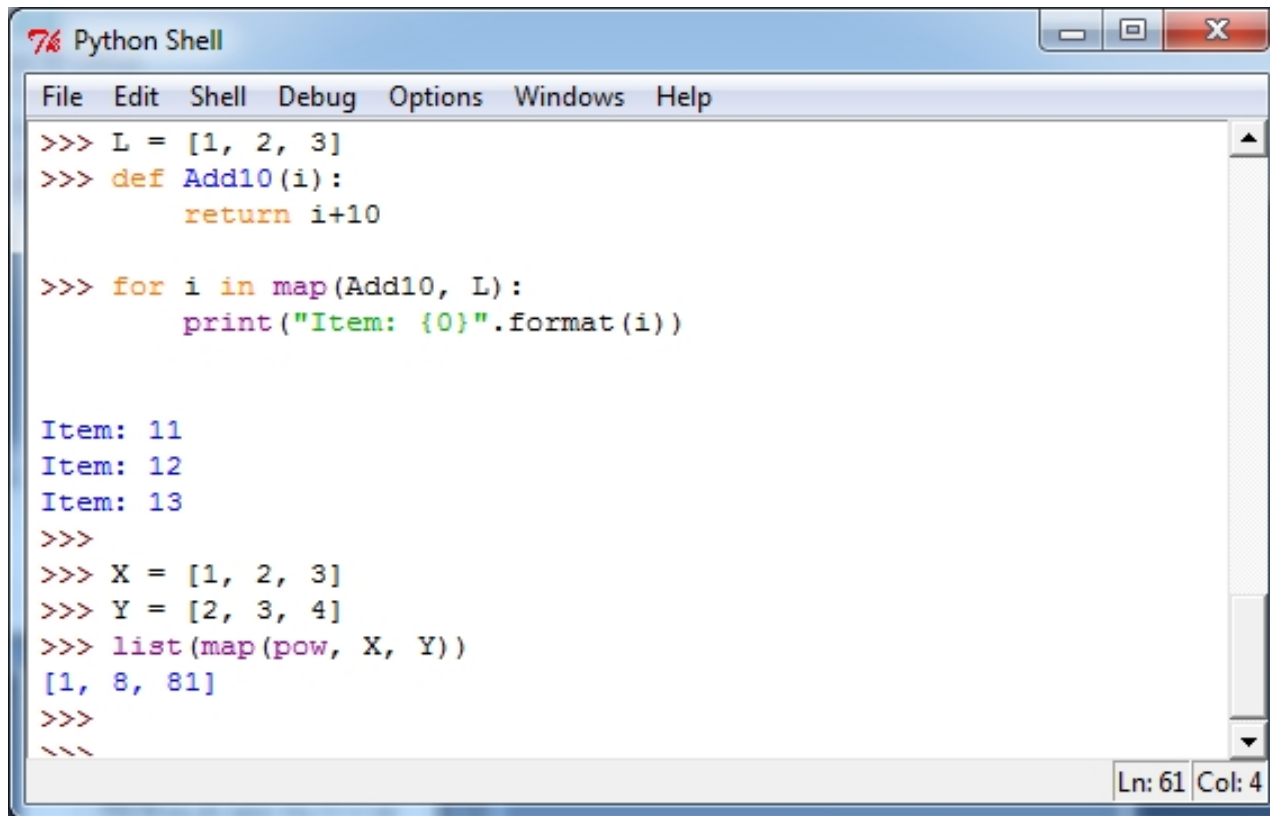
- **zip**(시퀀스 객체1, 시퀀스 객체2, ..)

```
>>> X = [10, 20, 30]
>>> Y = "ABC"
>>> Z = (1.5, 2.5, 3.5)
>>> RetList = list(zip(X, Y, Z))
[(10, 'A', 1.5), (20, 'B', 2.5), (30, 'C', 3.5)]
```

```
>>> X = [10, 20, 30]
>>> Y = "ABCDE"
>>> RetList = list(zip(X, Y))
[(10, 'A'), (20, 'B'), (30, 'C')] ← 짧은 쪽 기준으로 결합됨
```

# 반복문 작성시 도움이 되는 함수 (4)

- **map**(<function>, 시퀀스 객체, ...)
  - 시퀀스 객체를 순회하며 function의 연산을 수행
  - 함수의 인자수만큼 시퀀스 객체를 전달



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L = [1, 2, 3]
>>> def Add10(i):
>>> return i+10
>>> for i in map(Add10, L):
>>> print("Item: {0}".format(i))

Item: 11
Item: 12
Item: 13
>>>
>>> X = [1, 2, 3]
>>> Y = [2, 3, 4]
>>> list(map(pow, X, Y))
[1, 8, 81]
>>>
~>>>
```

Ln: 61 Col: 4



## Section 05 효율적인 순회 방법

# 효율적인 순회 방법 (1)

- 시퀀스형 자료를 순회하는 방법은 다음과 같습니다.

- 시퀀스 객체를 순회하며 function의 연산을 수행

```
>>> fruits = ['apple', 'Orange', 'Banana']
```

```
>>> for i in fruits:
```

```
 print(i)
```

```
Apple
```

```
Orange
```

```
Banana
```

- 함수의 인자수만큼 시퀀스 객체를 전달

```
>>> print("\n".join(l))
```

```
Apple
```

```
Orange
```

```
Banana
```

```
>>> print("\n".join(i for i in l))
```

```
Apple
```

```
Orange
```

```
Banana
```



Thank You

---