

스크립트 프로그래밍

10 리스트

2016 2학기 (02분반)

강승우

학습 목표

- 리스트를 생성할 수 있다 (§10.2.1).
- 시퀀스(sequence)에 대한 공통 연산을 활용할 수 있다 (§10.2.2).
- 리스트에 대해 len, min, max, sum, random.suffle 함수를 사용할 수 있다 (§10.2.3).
- 인덱스 변수를 사용하여 리스트 원소에 접근할 수 있다 (§10.2.4).
- 슬라이싱 연산자 [start : end]를 사용하여 리스트의 부분 리스트를 얻을 수 있다 (§10.2.5).
- 리스트에서 +(연결, concatenation), *(반복, repetition), in/not in 연산자를 사용할 수 있다 (§10.2.6).
- for 루프를 사용하여 리스트의 원소들을 순회할 수 있다 (§10.2.7).
- 비교 연산자를 사용하여 두 리스트의 내용을 비교할 수 있다 (§10.2.8).
- 리스트 컴프리헨션을 사용하여 리스트를 생성할 수 있다 (§10.2.9).
- 리스트의 append, count, extend, index, insert, pop, remove, reverse, sort 메소드를 호출할 수 있다 (§10.2.10).
- str의 split 메소드를 사용하여 문자열을 분할하여 리스트로 구성할 수 있다 (§10.2.11절).
- 콘솔에서 읽은 데이터로 리스트를 만들 수 있다 (§10.2.12).
- 애플리케이션 개발에서 리스트를 활용할 수 있다 (§10.3-10.5).
- 리스트의 내용을 다른 리스트로 복사할 수 있다 (§10.6).
- 리스트 인자와 반환 리스트를 포함한 함수를 개발하고 호출할 수 있다 (§10.7-10.9).
- 선형 검색 (§10.10.1)과 이진 검색 (§10.10.2)을 사용하여 원소를 검색할 수 있다.
- 선택 정렬을 사용하여 리스트의 원소들을 정렬할 수 있다 (§10.11.1).
- 삽입 정렬을 사용하여 리스트의 원소들을 정렬할 수 있다 (§10.11.2).

list 클래스

- 파이썬에서 연속적인 데이터를 저장하기 위해 사용하는 클래스
 - 가변 크기
 - int, str, float 등 어떤 타입의 데이터로 리스트로 만들 수 있음
 - 서로 다른 타입의 데이터를 한 리스트에 담을 수 있음
- 리스트 생성
 - `list1 = list()` # 빈 리스트 생성
 - `list2 = list([2, 3, 4])` # 원소 2, 3, 4를 가진 리스트 생성
 - `list3 = list(["red", "green", "blue"])` # 문자열을 가진 리스트 생성
 - `list4 = list(range(3, 6))` # 원소 3, 4, 5를 가진 리스트 생성
 - `list5 = list("abcd")` # 문자 a, b, c, d를 가진 리스트 생성
 - `list1 = []` # `list()` 와 동일
 - `list2 = [2, 3, 4]` # `list([2, 3, 4])` 와 동일
 - `list3 = ["red", "green"]` # `list(["red", "green"])` 와 동일

list - 시퀀스 타입

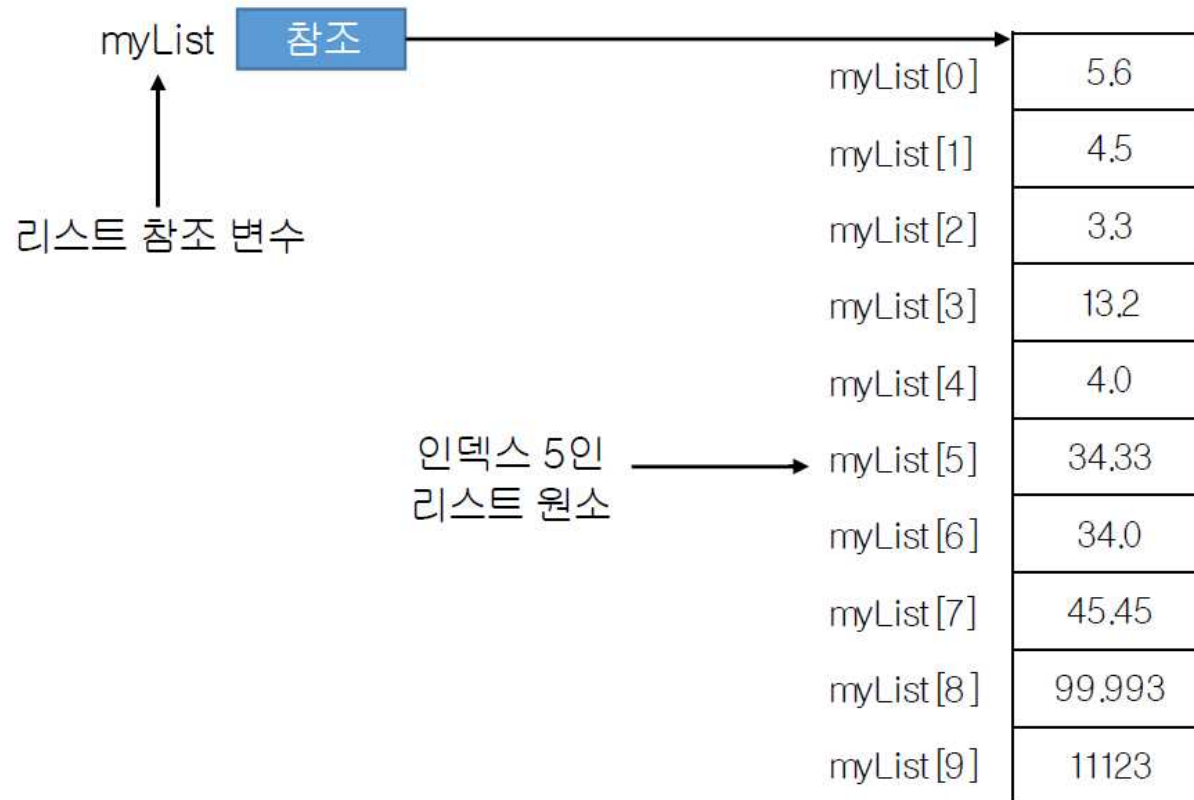
- 파이썬에 기본 제공되는 시퀀스(Sequence) 타입
 - str
 - list
 - tuple
- 시퀀스 타입에 제공되는 공통 연산 가능
 - (str 클래스에서 이미 대부분 봤음)
 - In / not in
 - + / *
 - 인덱싱 list1[i] / 슬라이싱 list1[i : j]
 - len(list1) / min(list1) / max(list1) / sum(list1) / random.shuffle(list1)
 - for loop
 - <, <=, >, >=, ==, !=

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list1 = [2, 3, 4, 1, 32]
>>> len(list1)
5
>>> max(list1)
32
>>> min(list1)
1
>>> sum(list1)
42
>>> import random
>>> random.shuffle(list1)
>>> list1
[4, 2, 3, 1, 32]
>>> 3 in list1
True
>>> 5 in list1
False
>>> list1 * 3
[4, 2, 3, 1, 32, 4, 2, 3, 1, 32, 4, 2, 3, 1, 32]
>>> |
```

Ln: 22 Col: 4

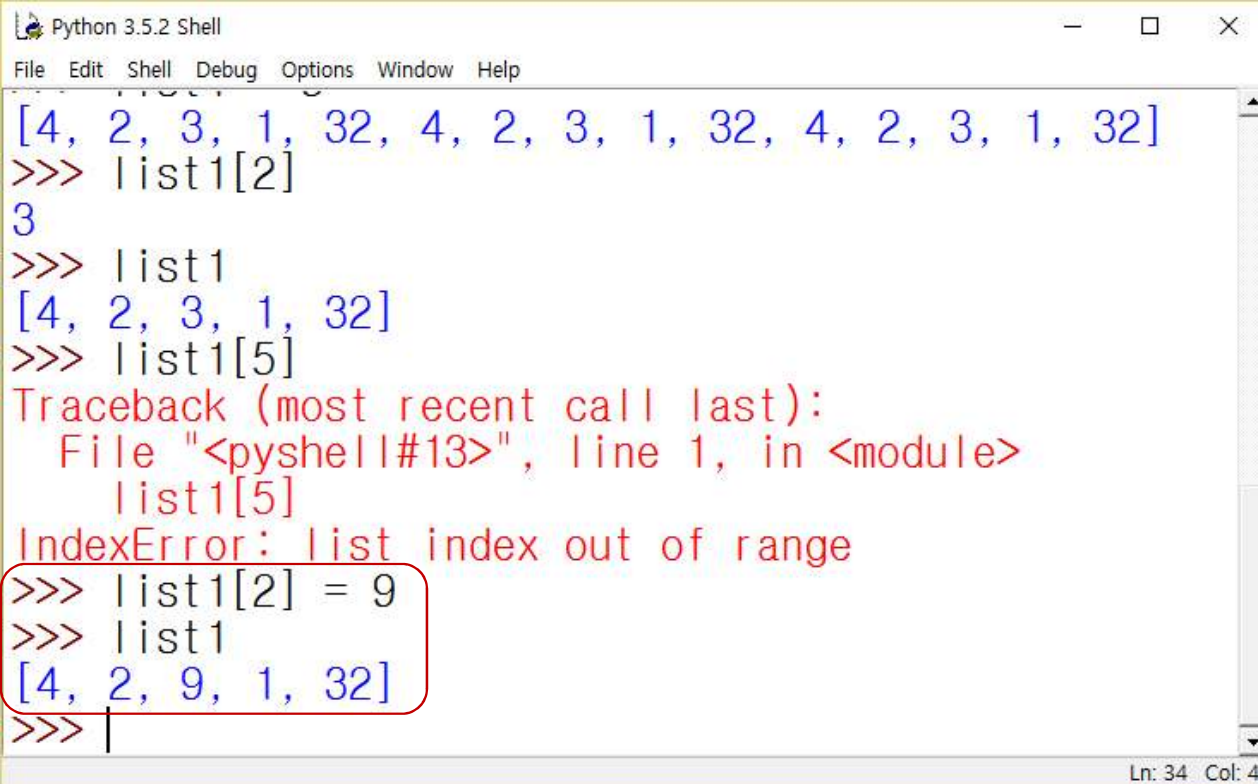
list 인덱스 연산

```
myList = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45, 99.993, 11123]
```



리스트는 변경 가능 (Mutable)

- 변경 가능 시퀀스 타입
 - list
- 변경 불가능 시퀀스 타입
 - str
 - tuple
- 오른쪽과 같이 인덱스 연산을 통해 리스트 원소의 값을 변경할 수 있음



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
[4, 2, 3, 1, 32, 4, 2, 3, 1, 32, 4, 2, 3, 1, 32]
>>> list1[2]
3
>>> list1
[4, 2, 3, 1, 32]
>>> list1[5]
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    list1[5]
IndexError: list index out of range
>>> list1[2] = 9
>>> list1
[4, 2, 9, 1, 32]
>>> |
```

Ln: 34 Col: 4

list 슬라이싱 l[start : end : step]

```
*Python 3.5.2 Shell*
File Edit Shell Debug Options Window Help
>>> list1
[4, 2, 9, 1, 32]
>>> list1[1: 4]
[2, 9, 1]
>>> list1[-4: -2]
[2, 9]
>>> list1[-4 + len(list1) : -2 + len(list1)]
[2, 9]
>>> list1[1:4:2]
[2, 1]
>>> list1[-2: -4]
[]
>>> list1[-2: -4: -1]
[1, 9]
>>> |
```

Ln: 46 Col: 4

연결 연산자(+), 반복 연산자(*), in / not in

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list1
[4, 2, 9, 1, 32]
>>> list2
[3, 3]
>>> list1 + list2
[4, 2, 9, 1, 32, 3, 3]
>>> list2 * 3
[3, 3, 3, 3, 3, 3]
>>> 4 in list2
False
>>> 3 not in list2
False
>>> 1 in list1
True
>>> |
```

Ln: 61 Col: 4

for 루프로 원소 순회

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help

>>> list1
[4, 2, 9, 1, 32]
>>> for l in list1:
    print(l)

4
2
9
1
32
>>> for i in range(0, len(list1), 2):
    print(list1[i])

4
9
32
>>> |
```

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help

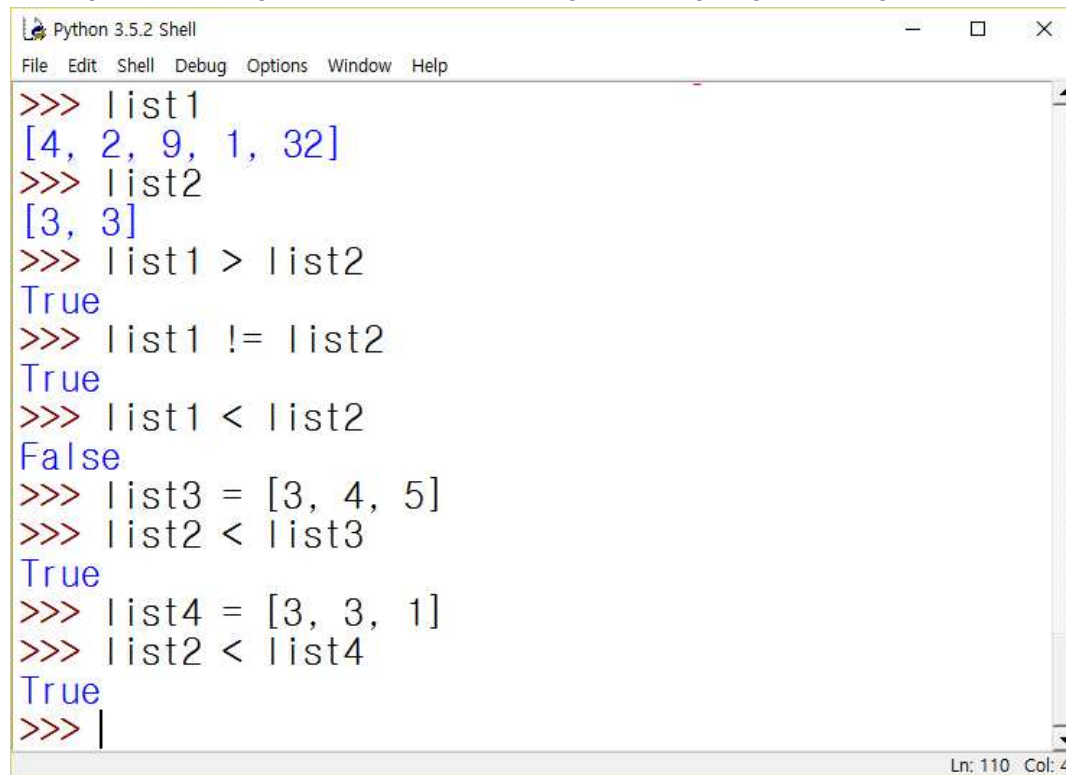
>>> i = 0
>>> while i <= len(list1):
    print(list1[i])
    i += 1

4
2
9
1
32
Traceback (most recent call last):
  File "<pyshell#41>", line 2, in <module>
    print(list1[i])
IndexError: list index out of range
>>> |
```

- 루프로 리스트 원소 순회 시 주의점
참조하는 인덱스가 범위를 넘어가지 않도록
해야 함

list 비교 (<, <=, >, >=, ==, !=)

- 두 개의 리스트를 비교하기 위해서는 둘 모두 동일한 타입의 데이터를 담고 있어야 함
 - 리스트에 있는 원소의 순서대로 비교



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list1
[4, 2, 9, 1, 32]
>>> list2
[3, 3]
>>> list1 > list2
True
>>> list1 != list2
True
>>> list1 < list2
False
>>> list3 = [3, 4, 5]
>>> list2 < list3
True
>>> list4 = [3, 3, 1]
>>> list2 < list4
True
>>> |
```

Ln: 110 Col: 4

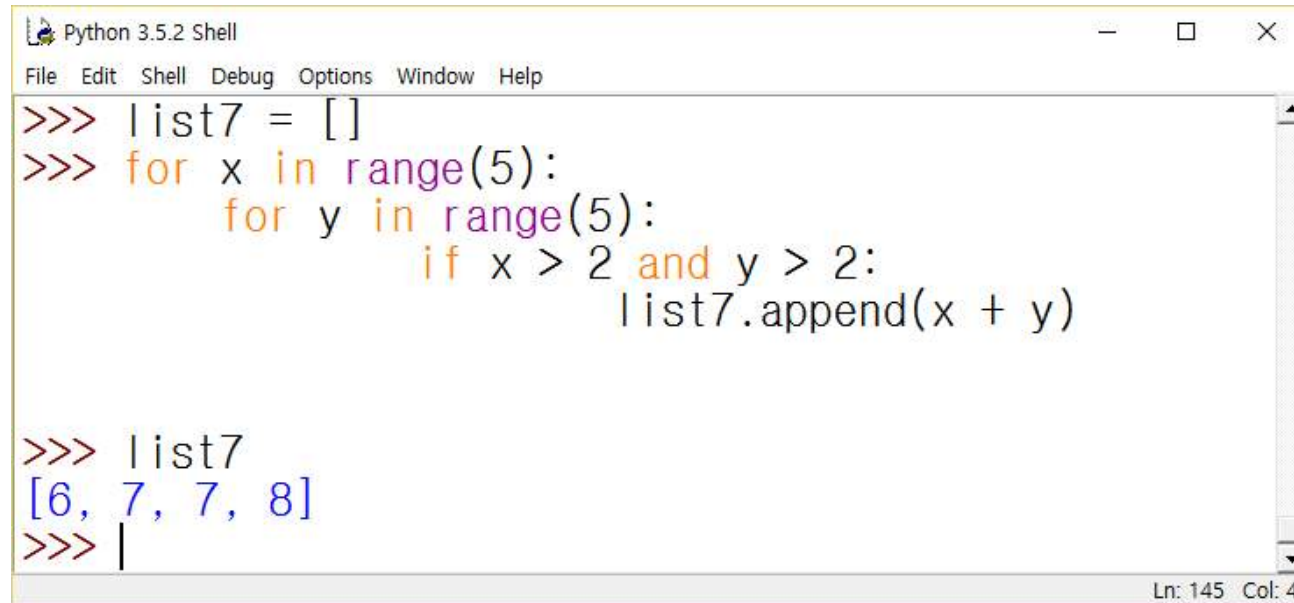
list comprehension

- 리스트 안에 표현식을 내포하여 그 표현식의 결과로 리스트를 만드는 것
 - [expression for item1 in iterable1 if condition1
for item2 in iterable2 if condition2
...
for itemN in iterableN if conditionN]
- 중첩 for 문을 이용한 표현식으로 리스트를 생성하는 것과 동일

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help

>>> list1 = [x for x in range(5)]
>>> list1
[0, 1, 2, 3, 4]
>>> list2 = [x for x in range(5) if x > 2]
>>> list2
[3, 4]
>>> list3 = [x * y for x in range(5)
              for y in range(5)]
>>> list3
[0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 0, 2, 4, 6, 8, 0, 3
, 6, 9, 12, 0, 4, 8, 12, 16]
>>> list4 = [0.5 * x for x in list1]
>>> list4
[0.0, 0.5, 1.0, 1.5, 2.0]
>>> list5 = [x + y for x in range(5) if x > 2
              for y in range(5) if y > 3]
>>> list5
[7, 8]
>>> list6 = [x + y for x in range(5) if x > 2
              for y in range(5) if y > 2]
>>> list6
[6, 7, 7, 8]
>>> |
```

list comprehension으로 만든 list6와
동일한 리스트를 중첩 for 문을 써서
만든다면?



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list7 = []
>>> for x in range(5):
    for y in range(5):
        if x > 2 and y > 2:
            list7.append(x + y)

>>> list7
[6, 7, 7, 8]
>>> |
```

Ln: 145 Col: 4

list comprehension 연습

- 아래 리스트가 있을 때 각 원소에 3을 곱한 리스트를 만들어보자
 - $a = [1, 2, 3, 4]$
- 홀수에만 3을 곱하여 리스트를 만들어보자

list 메소드

- `append(x)`
 - 원소 `x`를 리스트의 끝에 추가
- `count(x)`
 - 리스트 내에서 원소 `x`의 개수 반환
- `extend(lis)`
 - 리스트 `lis`의 모든 원소를 리스트에 추가
- `index(x)`
 - 리스트에서 원소 `x`가 첫 번째로 나온 위치의 인덱스 값 반환
- `insert(ind, x)`
 - `ind`에 해당하는 인덱스 위치에 원소 `x`를 삽입
- `pop(i)`
 - 인덱스 위치 `i`에 있는 원소를 제거하고 그 원소를 반환, `i`가 없으면 마지막 원소
- `remove(x)`
 - 리스트에서 원소 `x`를 제거, 두 개 이상 있다면 첫 번째로 나오는 원소 `x`를 제거
- `reverse()`
 - 리스트 내의 원소를 역순으로 만들
- `sort()`
 - 리스트 내의 원소를 오름차순으로 정렬


```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list7
[6, 7, 7, 8]
>>> list7.append(3)
>>> list7
[6, 7, 7, 8, 3]
>>> list7.count(7)
2
>>> list7.extend([1, 2])
>>> list7
[6, 7, 7, 8, 3, 1, 2]
>>> list7.index(1)
5
>>> list7.index(7)
1
>>> list7.insert(1, 9)
>>> list7
[6, 9, 7, 7, 8, 3, 1, 2]
>>> |
```

Ln: 164 Col: 4

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> list7.pop(0)
6
>>> list7
[9, 7, 7, 8, 3, 1, 2]
>>> list7.pop()
2
>>> list7
[9, 7, 7, 8, 3, 1]
>>> list7.remove(1)
>>> list7
[9, 7, 7, 8, 3]
>>> list7.reverse()
>>> list7
[3, 8, 7, 7, 9]
>>> list7.sort()
>>> list7
[3, 7, 7, 8, 9]
>>>
```

Ln: 181 Col: 4

문자열을 리스트로 분할

- str 클래스의 split() 메소드 이용
 - split() 메소드에 구분자를 넣어줄 수 있음

A screenshot of a Python 3.5.2 Shell window. The window has a title bar with the text 'Python 3.5.2 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python commands and their outputs, demonstrating the use of the split() method. The commands are: 1. names = "Mary Bob Alice Tom" (names are green, quotes are black). 2. nlist = names.split() (nlist is red, split is green). 3. nlist (output is blue: ['Mary', 'Bob', 'Alice', 'Tom']). 4. nlist = names.split(' ') (split is green). 5. nlist (output is blue: ['Mary', 'Bob', 'Alice', 'Tom']). 6. "11/03/2016".split('/') (date is green, split is green). 7. Output (blue: ['11', '03', '2016']). The window ends with a prompt >>> and a cursor. At the bottom right of the window, the status bar shows 'Ln: 190 Col: 4'.

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> names = "Mary Bob Alice Tom"
>>> nlist = names.split()
>>> nlist
['Mary', 'Bob', 'Alice', 'Tom']
>>> nlist = names.split(' ')
>>> nlist
['Mary', 'Bob', 'Alice', 'Tom']
>>> "11/03/2016".split('/')
['11', '03', '2016']
>>> |
Ln: 190 Col: 4
```

리스트 입력하기

- 10개의 숫자를 입력 받아 리스트에 추가하기

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> lst = []
>>> for i in range(10):
    lst.append(eval(input()))

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

```
listinput.py - D:/GitRepo/ScriptProgramming_2016-2/ch10/listinput.py (3.5.2)
File Edit Format Run Options Window Help
s = input("한 행에 공백으로 구분된 숫자 10개를 입력하세요: ")
items = s.split()
lst = [eval(x) for x in items]
print(lst)
```

리스트 복사

- `list1 = [1, 2]`
- `list1`과 동일한 리스트 `list2`를 만들고자 할 때
- `list2 = list1` 이라고 하면 어떻게 될까?

- 아래 코드 실행 결과는?

```
list1 = list(range(1, 10, 2))
```

```
list2 = list1
```

```
list1[0] = 111
```

```
print(list1, list2)
```

리스트 복사

- `list2 = [x for x in list1]`
- `list2 = [] + list1`

- 아래 코드 실행 결과는?

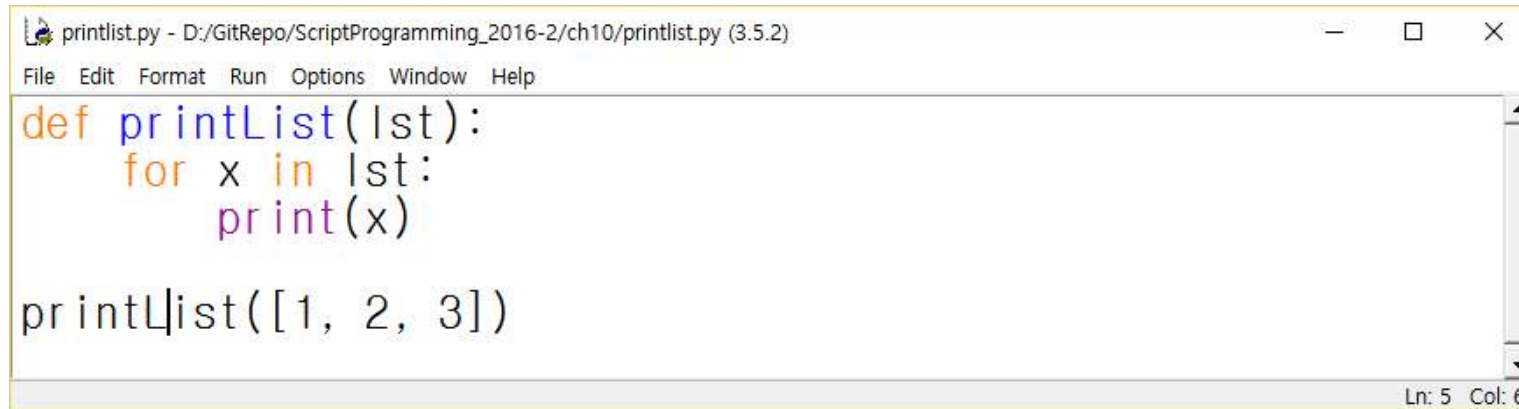
```
list1 = list(range(1, 10, 2))
```

```
list2 = [] + list1
```

```
list1[0] = 111
```

```
print(list1, list2)
```

함수에 리스트 전달하기



```
def printList(lst):  
    for x in lst:  
        print(x)  
  
printList([1, 2, 3])
```

Ln: 5 Col: 6

- 실행 결과



```
>>>  
RESTART: D:/GitRepo/ScriptProgramming_2016-2/ch10/  
/printlist.py  
1  
2  
3  
>>>
```

Ln: 233 Col: 46

함수에 리스트 전달하기

```
PassListArgument.py - D:\GitRepo\ScriptProgramming_2016-2\ch10\PassListArgument.py (3.5.2)
File Edit Format Run Options Window Help
def main():
    x = 1 # x는 int 변수이다.
    y = [1, 2, 3] # y는 리스트이다.

    m(x, y) # x, y 인자를 사용하여 m을 호출한다.

    print("x is", x)
    print("y[0] is", y[0])

def m(number, numbers):
    number = 1001 # number에 새로운 값을 할당한다.
    numbers[0] = 5555 # numbers[0]에 새로운 값을 할당한다.

main() # main 함수를 호출한다.
```

Ln: 1 Col: 0

- 실행 결과는?

리스트를 기본 인자로 사용

```
DefaultListArgument.py - D:\GitRepo\ScriptProgramming_2016-2\ch10\DefaultListArgument.py (3.5.2)
File Edit Format Run Options Window Help
def add(x, lst = []):
    if x not in lst:
        lst.append(x)

    return lst

def main():
    list1 = add(1)
    print(list1)

    list2 = add(2)
    print(list2)

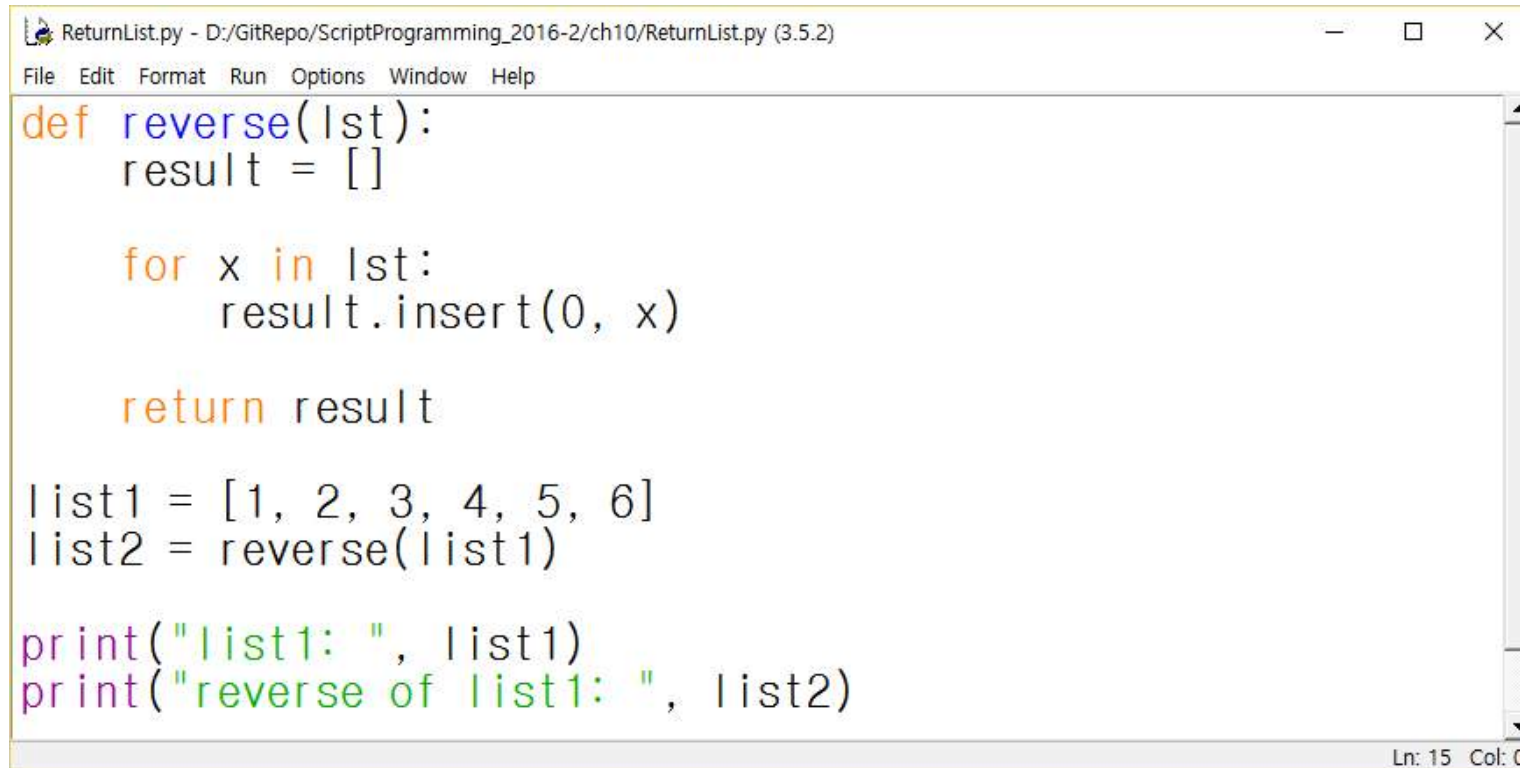
    list3 = add(3, [11, 12, 13, 14])
    print(list3)

    list4 = add(4)
    print(list4)

main()
Ln: 21 Col: 0
```

- 실행 결과는?

함수에서 리스트 반환하기



```
ReturnList.py - D:/GitRepo/ScriptProgramming_2016-2/ch10/ReturnList.py (3.5.2)
File Edit Format Run Options Window Help

def reverse(lst):
    result = []

    for x in lst:
        result.insert(0, x)

    return result

list1 = [1, 2, 3, 4, 5, 6]
list2 = reverse(list1)

print("list1: ", list1)
print("reverse of list1: ", list2)
```

Ln: 15 Col: 0

- 실행 결과는?

사례 연구: 문자 빈도수 세기

- 100개의 소문자를 랜덤하게 생성하고 이를 chars라는 문자 리스트에 할당
 - RandomCharacter 모듈의 getRandomLowerCaseLetter() 함수 이용
 - 코드 6.11에 정의
- chars 리스트에 있는 각 문자의 개수를 센다
 - 이를 위해 26개의 int 값을 갖는 counts라는 리스트를 생성한다
 - counts의 각 원소는 각 문자의 개수를 저장한다
즉, counts[0]은 chars 리스트에 있는 문자 a의 개수,
counts[1]은 문자 b의 개수, ...

사례 연구: 문자 빈도수 세기

```
def main():  
    # 문자 리스트를 생성한다.  
    chars = createList()  
  
    # 리스트를 출력한다.  
    print("소문자:")  
    displayList(chars)  
  
    # 각 문자의 빈도수를 센다.  
    counts = countLetters(chars)  
  
    # 빈도수를 출력한다.  
    print("각 문자의 빈도수는:")  
    displayCounts(counts)
```

```
# 문자 리스트를 생성한다.  
def createList():  
    # 빈 리스트를 생성한다.  
  
    # 소문자를 랜덤하게 생성하고 리스트에 추가한다.  
  
    # 리스트를 반환한다.
```

```
# 문자 리스트를 출력한다.  
def displayList(chars):  
    # 리스트에 포함된 문자를 한 행에 20개씩 출력한다.  
    for i in range(len(chars)):
```

사례 연구: 문자 빈도수 세기

```
# 각 문자의 빈도수를 센다.  
def countLetters(chars):  
    # 0으로 초기화된 26개의 정수 리스트를 생성한다.  
  
    # 리스트의 각 소문자를 센다.  
  
    # 빈도수가 저장된 리스트 counts를 반환한다.  
    return counts
```

```
# 빈도수를 출력한다.  
def displayCounts(counts):  
    for i in range(len(counts)):  
        if (i + 1) % 10 == 0:  
            print(counts[i], chr(i + ord('a')))  
        else:  
            print(counts[i], chr(i + ord('a')), end = ' ')
```