

Structured Types 1

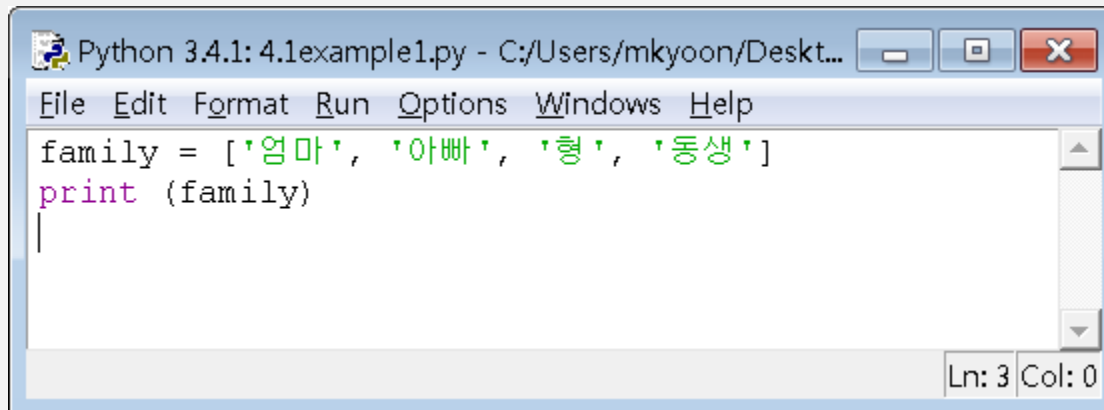
국민대 소프트웨어학부

수업목표

- What is a lists
- Creating a list
- Adding things to a list
- Getting items from a list
- Slicing a list
- Modifying/Deleting/Searching items
- Looping through a list
- Sorting a list
- Shallow and deep copy
- Lists of lists

What is a lists

- 리스트(list)는 아이템(item)들을 모은 순차 데이터 타입
- 아이템은 정수, 스트링, 다른 리스트 등 모든 종류의 데이터 가능
 - family = ['엄마', '아빠', '형', '동생']
 - numbers = [1, 3, 5, 7]
 - my_list = [5, 10, 21, 23, 'Hello', x, your_list]



The screenshot shows a Python 3.4.1 IDE window titled "Python 3.4.1: 4.1example1.py - C:/Users/mkyoon/Desktop...". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following text:

```
family = ['엄마', '아빠', '형', '동생']  
print (family)  
|
```

The status bar at the bottom right indicates "Ln: 3 Col: 0".

Creating a list

- “[] ” 사용
 - myList = []

```
# 리스트를 선언합니다.
```

```
infinite_members = [ "김성규", "장동우", "남우현", "이성열", "엘", "이성종" ]
```

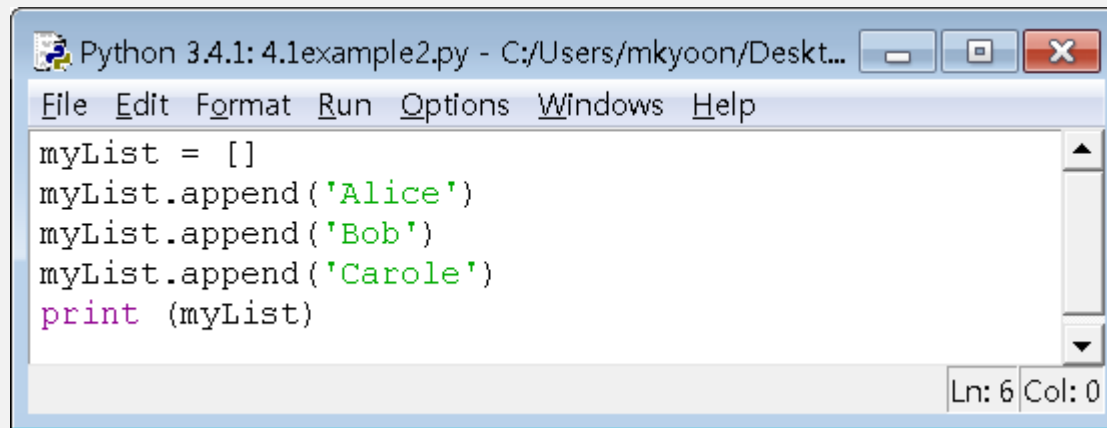
```
# 출력하기
```

```
print(infinite_members)
```

```
['김성규', '장동우', '남우현', '이성열', '엘', '이성종']
```

Adding things to a list

- append() 사용
 - 사용법: 리스트이름.append(item)
 - 리스트는 객체(object)이며, 리스트 객체는 append라는 함수(function)를 이용해서 아이템을 추가함
 - 객체, 함수 → 향후 설명



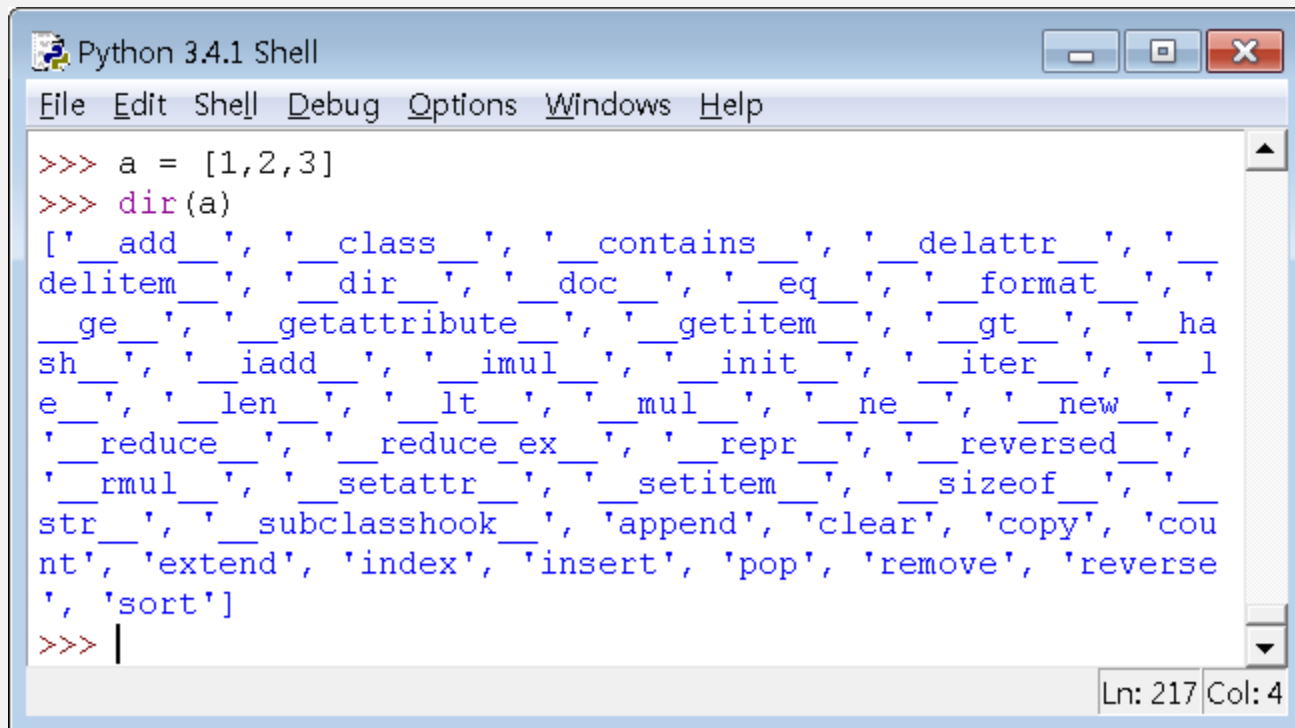
The screenshot shows a window titled "Python 3.4.1: 4.1example2.py - C:/Users/mkyoon/Desktop...". The window contains a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main text area contains the following Python code:

```
myList = []  
myList.append('Alice')  
myList.append('Bob')  
myList.append('Carole')  
print (myList)
```

The status bar at the bottom right indicates "Ln: 6 Col: 0".

Adding things to a list

- append() 사용
 - 객체, 함수 → 향후 설명
 - 객체가 어떤 함수와 속성을 가지고 있는지 확인하려면 dir() 함수 이용

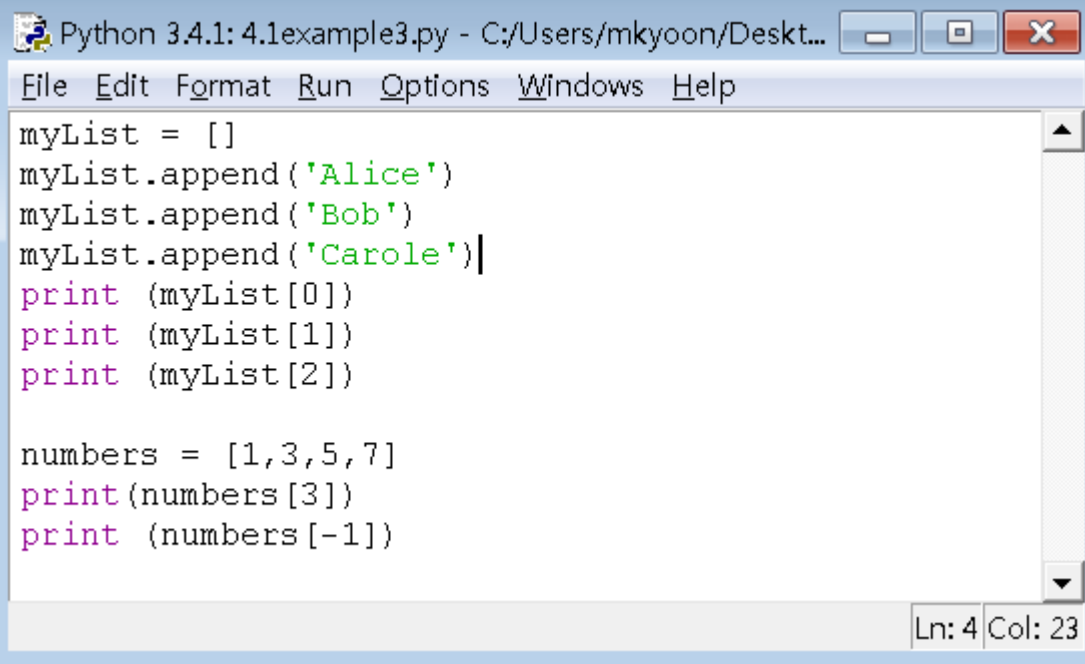


```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> a = [1,2,3]
>>> dir(a)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> |
```

Ln: 217 Col: 4

Getting items from a list

- 리스트에서 아이템 가져오기
 - 인덱스(index) 사용
 - 인덱스는 0부터 시작 → computer science 특징

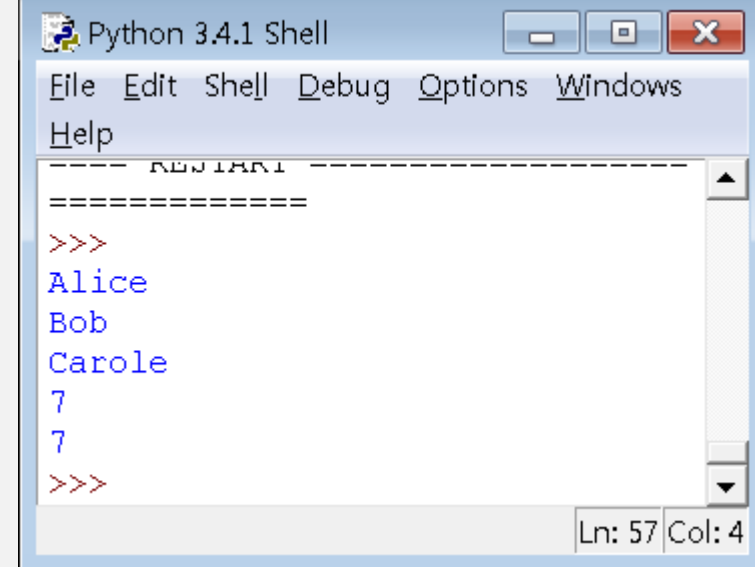


```
Python 3.4.1: 4.1example3.py - C:/Users/mkyoon/Desktop...
File Edit Format Run Options Windows Help

myList = []
myList.append('Alice')
myList.append('Bob')
myList.append('Carole')
print (myList[0])
print (myList[1])
print (myList[2])

numbers = [1,3,5,7]
print (numbers[3])
print (numbers[-1])

Ln: 4 Col: 23
```



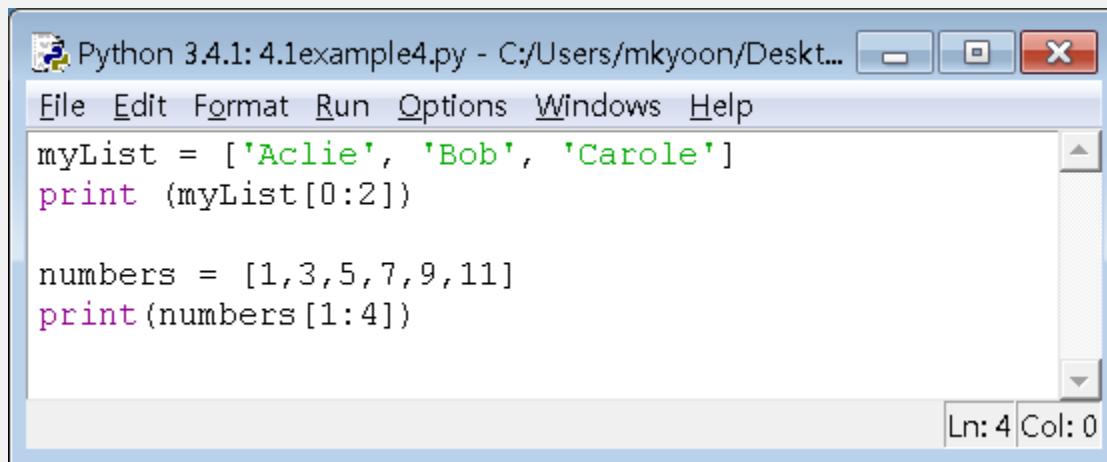
```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help

----- RESTART -----
=====
>>>
Alice
Bob
Carole
7
7
>>>

Ln: 57 Col: 4
```

Slicing a list

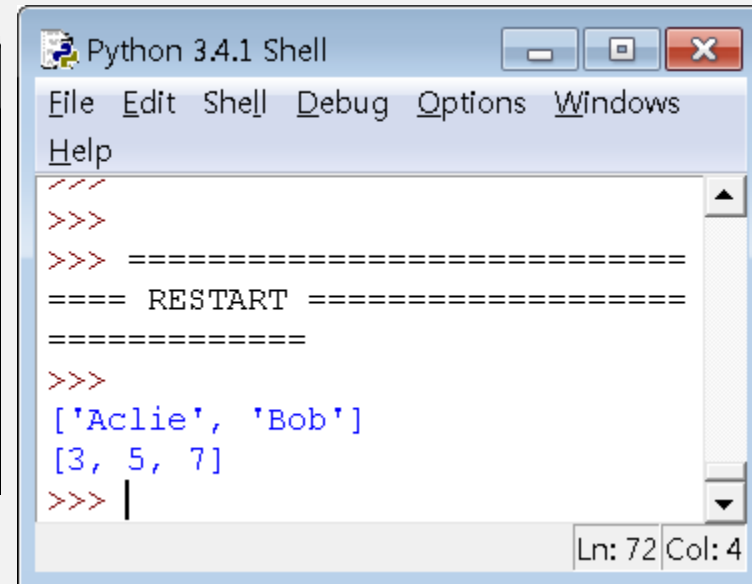
- 리스트 자르기
 - “리스트이름[a:b]” : a번째부터 (b-1)번째까지 아이템으로 구성된 부분 **리스트** (잘라진 리스트)가 결과값으로 얻어짐
 - 앞에서는 **아이템**이 결과값으로 얻어짐



```
Python 3.4.1: 4.1example4.py - C:/Users/mkyoon/Deskt...
File Edit Format Run Options Windows Help
myList = ['Aclie', 'Bob', 'Carole']
print (myList[0:2])

numbers = [1,3,5,7,9,11]
print (numbers[1:4])

Ln: 4 Col: 0
```



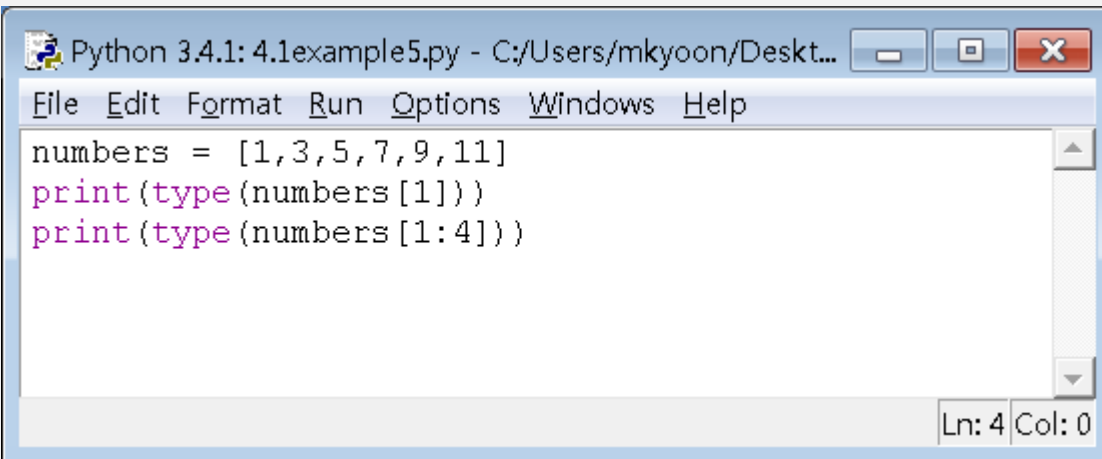
```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows
Help
///
>>>
>>> =====
==== RESTART =====
>>>
['Aclie', 'Bob']
[3, 5, 7]
>>> |

Ln: 72 Col: 4
```


Slicing a list

- 리스트 자르기

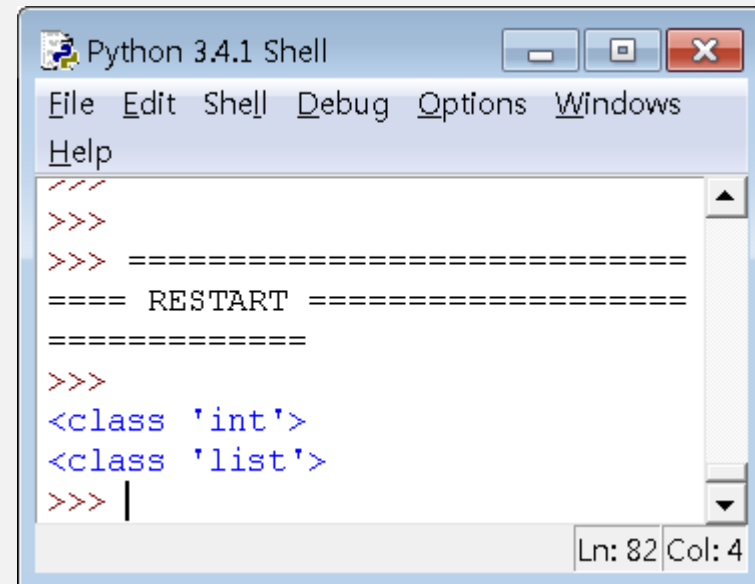
- “리스트이름[a:b]” : a번째부터 (b-1)번째까지 아이템으로 구성된 부분 **리스트** (잘라진 리스트)가 결과값으로 얻어짐
 - 앞에서는 **아이템**이 결과값으로 얻어짐



Python 3.4.1: 4.1example5.py - C:/Users/mkyoon/Deskt...

```
File Edit Format Run Options Windows Help
numbers = [1,3,5,7,9,11]
print(type(numbers[1]))
print(type(numbers[1:4]))
```

Ln: 4 Col: 0



Python 3.4.1 Shell

```
File Edit Shell Debug Options Windows Help
>>>
>>> =====
==== RESTART =====
>>>
<class 'int'>
<class 'list'>
>>> |
```

Ln: 82 Col: 4

Slicing a list

- 리스트 자르기
 - “리스트이름[a:]” : a번째부터 마지막 항목까지로 구성된 리스트
 - “리스트이름[:b]” : 처음부터 (b-1)번째 항목까지로 구성된 리스트
 - “리스트이름[:]” : 처음부터 마지막 항목까지로 구성된 리스트

결합

- 리스트 연산자
 - + : 두 리스트를 합친다.
 - * : 리스트를 숫자만큼 반복 한다.

```
graduate = [ "Jeong", "Lee", "Kim" ]  
undergraduate = [ "Yeongjae", "Myeong", "Han" ]  
  
all_lab_members = graduate + undergraduate  
  
print(all_lab_members)  
print(graduate * 2)  
print(undergraduate * 2)
```

리스트 크기

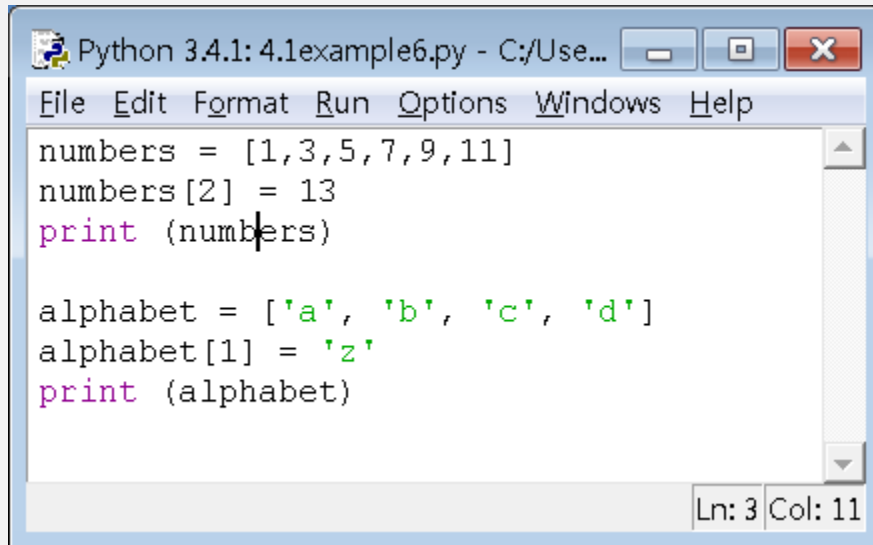
- `len(리스트)` : 리스트의 길이를 반환 한다.

```
graduate = [ "Jeong", "Lee", "Kim" ]
```

```
print("인원 수: ", len(graduate))
```

Modifying/Deleting/Searching items

- 수정하기 (Modify)
 - 인덱스를 사용해서 대상 아이템 변경

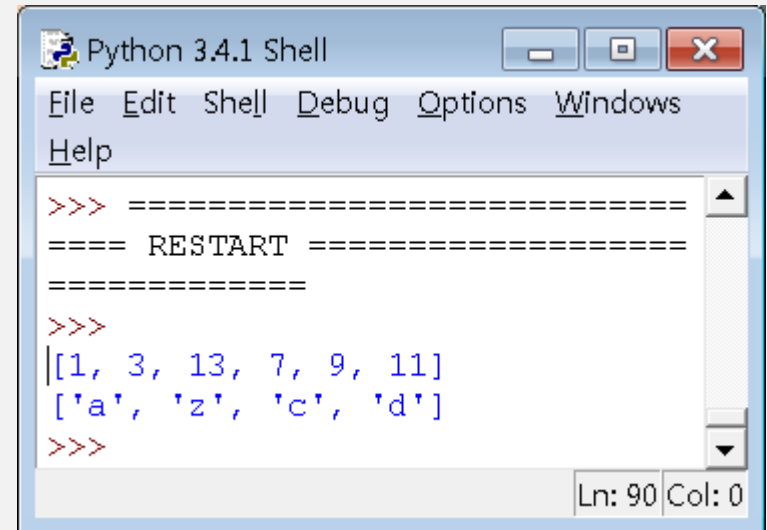


A screenshot of a Python 3.4.1 IDE window titled "Python 3.4.1: 4.1example6.py - C:/Use...". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor contains the following Python code:

```
numbers = [1,3,5,7,9,11]
numbers[2] = 13
print (numbers)

alphabet = ['a', 'b', 'c', 'd']
alphabet[1] = 'z'
print (alphabet)
```

The status bar at the bottom right shows "Ln: 3 Col: 11".



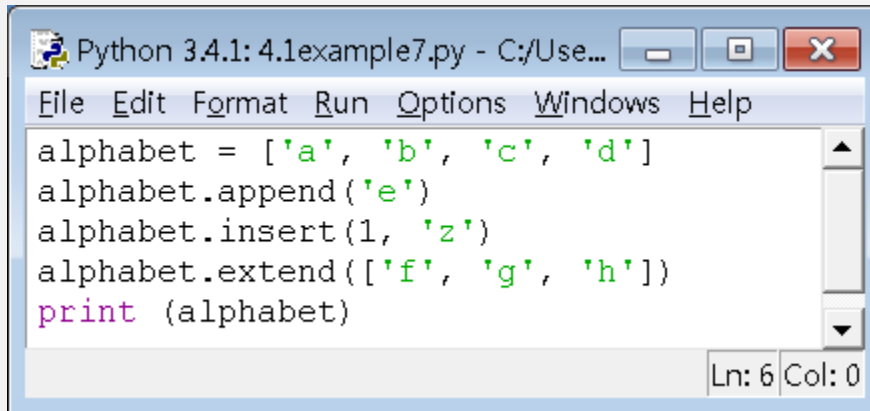
A screenshot of a Python 3.4.1 Shell window titled "Python 3.4.1 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The shell displays the output of the code from the previous window:

```
>>> =====
==== RESTART =====
>>>
[1, 3, 13, 7, 9, 11]
['a', 'z', 'c', 'd']
>>>
```

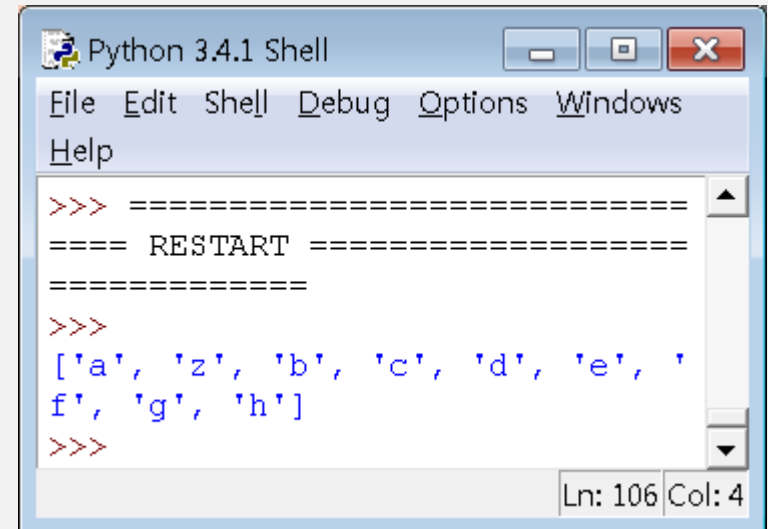
The status bar at the bottom right shows "Ln: 90 Col: 0".

Modifying/Deleting/Searching items

- 리스트에 아이터를 넣는 다양한 방법
 - `append()`: 리스트 뒤에 아이터 추가
 - `insert()`: 리스트 내 원하는 위치에 아이터 추가
 - `extend()`: 여러 아이터를 리스트 뒤에 추가



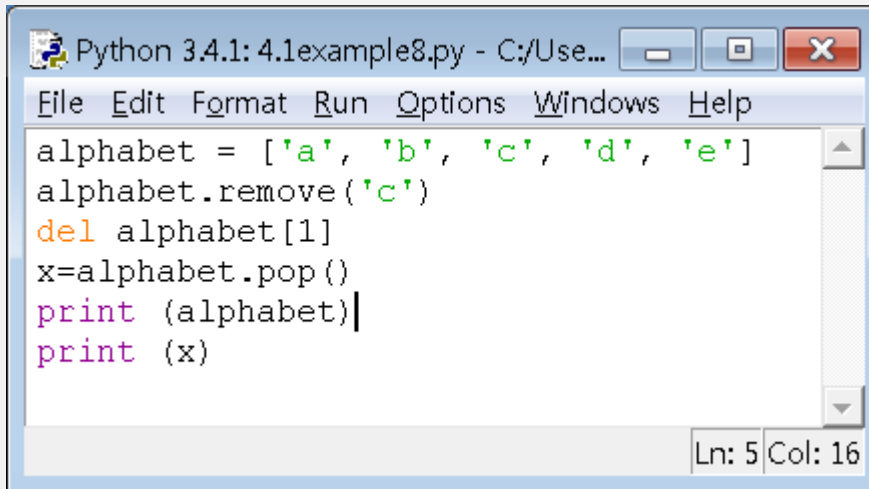
```
Python 3.4.1: 4.1example7.py - C:/Use...  
File Edit Format Run Options Windows Help  
alphabet = ['a', 'b', 'c', 'd']  
alphabet.append('e')  
alphabet.insert(1, 'z')  
alphabet.extend(['f', 'g', 'h'])  
print (alphabet)  
Ln: 6 Col: 0
```



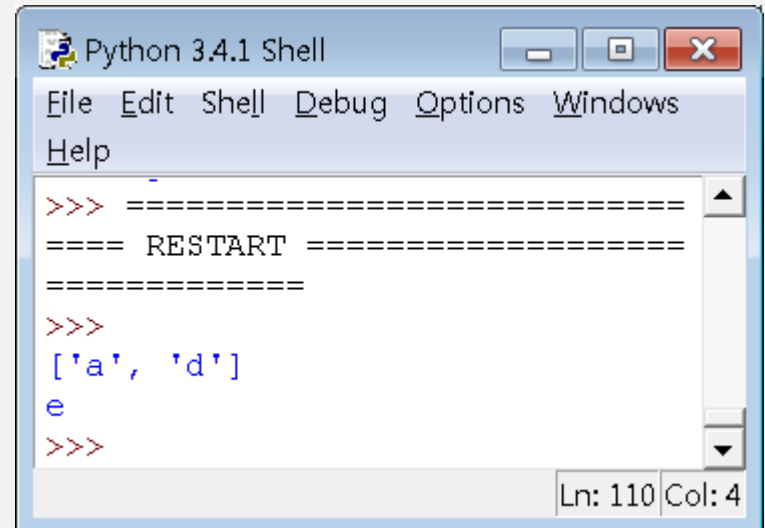
```
Python 3.4.1 Shell  
File Edit Shell Debug Options Windows Help  
>>> =====  
==== RESTART =====  
>>>  
['a', 'z', 'b', 'c', 'd', 'e', 'f', 'g', 'h']  
>>>  
Ln: 106 Col: 4
```

Modifying/Deleting/Searching items

- 리스트에서 아이템 삭제하기
 - <리스트>.remove(<값>) : 리스트의 특정 값을 제거
 - 해당 값이 복수 개인 경우 제일 먼저 발견되는 값 제거
 - del <리스트>[<인덱스>] : 리스트의 특정 인덱스에 있는 요소 제거
 - pop(): 리스트의 마지막 항목 리턴(return)
 - <리스트>.clear() : 리스트 내부의 모든 요소를 제거



```
Python 3.4.1: 4.1example8.py - C:/Use...
File Edit Format Run Options Windows Help
alphabet = ['a', 'b', 'c', 'd', 'e']
alphabet.remove('c')
del alphabet[1]
x=alphabet.pop()
print (alphabet)
print (x)
Ln: 5 Col: 16
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
-
>>> =====
==== RESTART =====
>>>
['a', 'd']
e
>>>
Ln: 110 Col: 4
```

Modifying/Deleting/Searching items

- 리스트에서 아이템 삭제하기

```
lab_members = [ "Jeong", "Jeong", "Lee", "Kim",  
                "Yeongjae", "Myeong", "Han"]
```

```
del lab_members[0]  
print(lab_members)
```

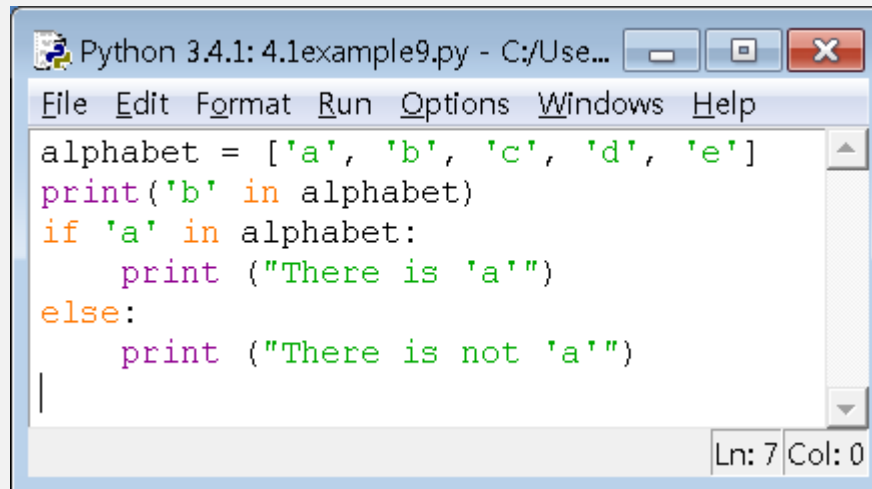
```
lab_members.pop(5)  
print(lab_members)
```

```
ant_series = [1, 1, 2, 3, 1, 2, 3, 1, 1, 1]
```

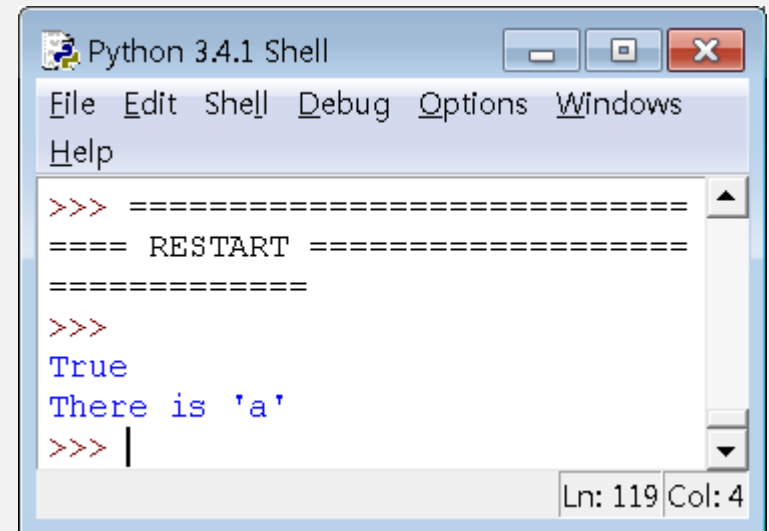
```
print("Before :", ant_series)  
ans_series.remove(2)  
print("After :", ant_series)
```


Modifying/Deleting/Searching items

- 리스트에서 찾기 (search)
 - 'in' 키워드: Boolean 값(True/False) 리턴



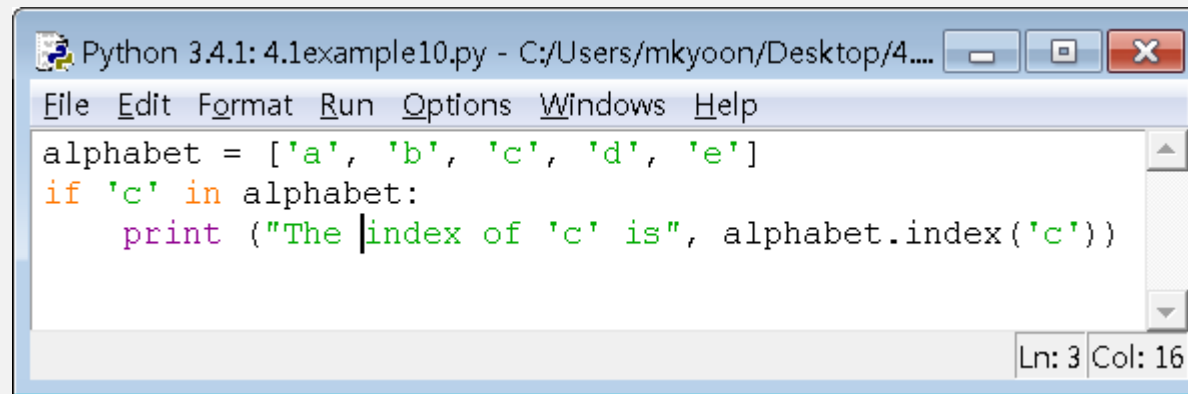
```
Python 3.4.1: 4.1example9.py - C:/Use...
File Edit Format Run Options Windows Help
alphabet = ['a', 'b', 'c', 'd', 'e']
print('b' in alphabet)
if 'a' in alphabet:
    print ("There is 'a'")
else:
    print ("There is not 'a'")
|
Ln: 7 Col: 0
```



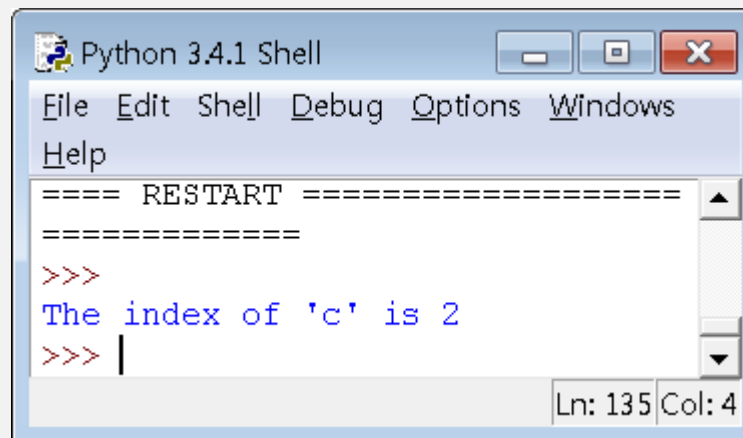
```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows
Help
>>> =====
==== RESTART =====
>>>
True
There is 'a'
>>> |
Ln: 119 Col: 4
```

Modifying/Deleting/Searching items

- 리스트에서 찾기 (search)
 - index() 함수: 아이템의 위치(인덱스값) 리턴



```
Python 3.4.1: 4.1example10.py - C:/Users/mkyoon/Desktop/4....  
File Edit Format Run Options Windows Help  
alphabet = ['a', 'b', 'c', 'd', 'e']  
if 'c' in alphabet:  
    print ("The index of 'c' is", alphabet.index('c'))  
Ln: 3 Col: 16
```



```
Python 3.4.1 Shell  
File Edit Shell Debug Options Windows Help  
==== RESTART =====  
>>>  
The index of 'c' is 2  
>>> |  
Ln: 135 Col: 4
```

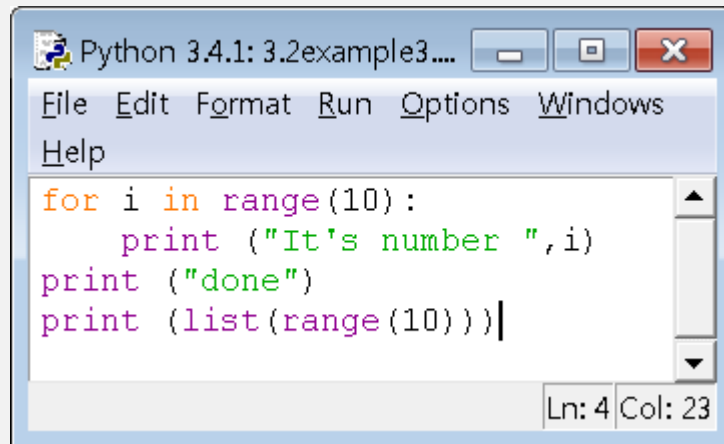
Modifying/Deleting/Searching items

- 리스트에서 찾기 (search)
 - index() 함수: 아이템의 위치(인덱스값) 리턴

```
infinite_members = [ "김성규", "장동우", "남우현", "이성열", "엘", "이성종" ]  
name = input("이름 >")  
if name in infinite_members :  
    idx = infinite_members.index(name)  
    print("{}는 인피니트의 {}번째 멤버입니다.".format(name, idx))  
else :  
    print("{}는 인피니트의 멤버가 아닙니다.".format(name))
```

Looping through a list

- for loop 다시 보기
 - range(10) → [0,1,2,3,4,5,6,7,8,9]
 - 직접 리스트를 생성하는 것은 아님

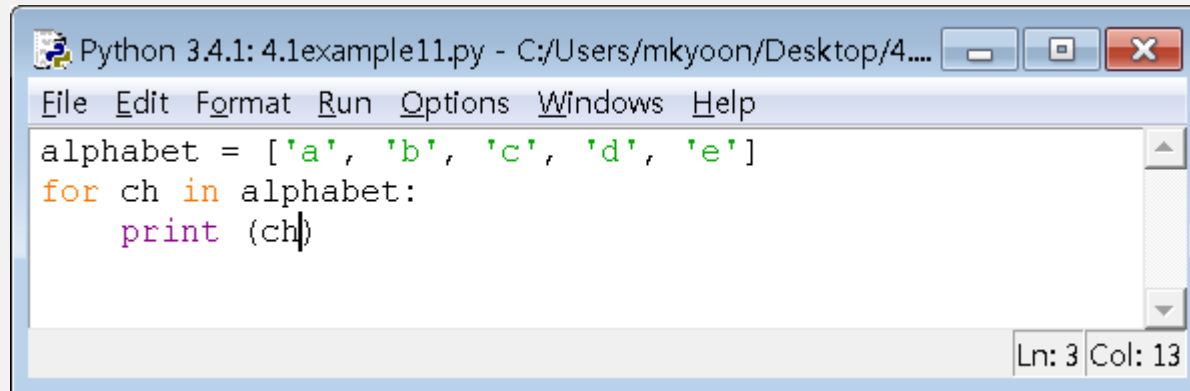


The image shows a screenshot of a Python 3.4.1 IDLE window. The title bar reads "Python 3.4.1: 3.2example3...". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor contains the following Python code:

```
for i in range(10):  
    print ("It's number ", i)  
print ("done")  
print (list(range(10)))
```

The status bar at the bottom right indicates "Ln: 4 Col: 23".

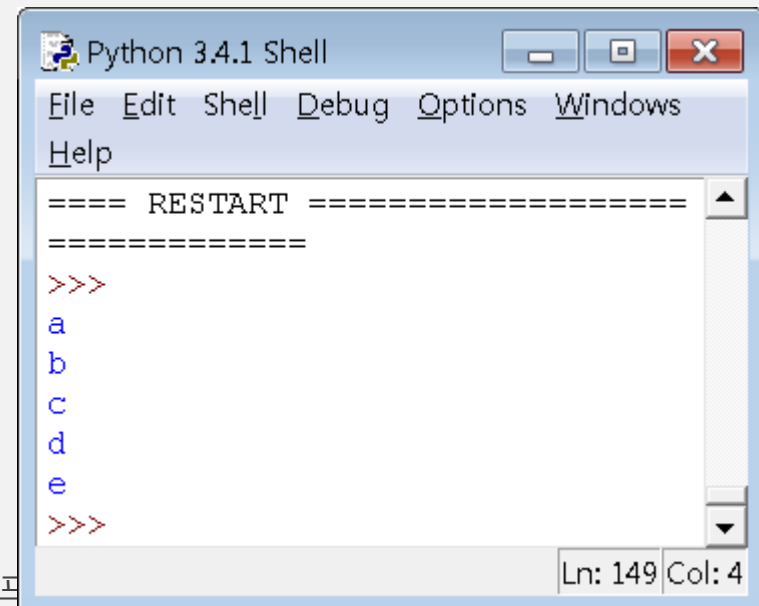
Looping through a list



A screenshot of a Python 3.4.1 IDE window titled "Python 3.4.1: 4.1example11.py - C:/Users/mkyoon/Desktop/4....". The window contains a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor shows the following Python code:

```
alphabet = ['a', 'b', 'c', 'd', 'e']
for ch in alphabet:
    print (ch)
```

The status bar at the bottom right indicates "Ln: 3 Col: 13".



A screenshot of a Python 3.4.1 Shell window titled "Python 3.4.1 Shell". The window contains a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The shell output shows the execution of the script:

```
==== RESTART =====
>>>
a
b
c
d
e
>>>
```

The status bar at the bottom right indicates "Ln: 149 Col: 4".

Loop in a list

- 리스트 안에 loop 문 작성 가능 (list comprehension)
 - [표현식 for 항목 in 반복가능객체]

```
a = [1,2,3,4]
result = []
for num in a:
    result.append(num*3)
print(result)
[3, 6, 9, 12]
```

=

```
a = [1,2,3,4]
result = [num * 3 for num in a]
print(result)
[3, 6, 9, 12]
```

- If 조건 추가 가능
 - [표현식 for 항목 in 반복가능객체 if 조건]

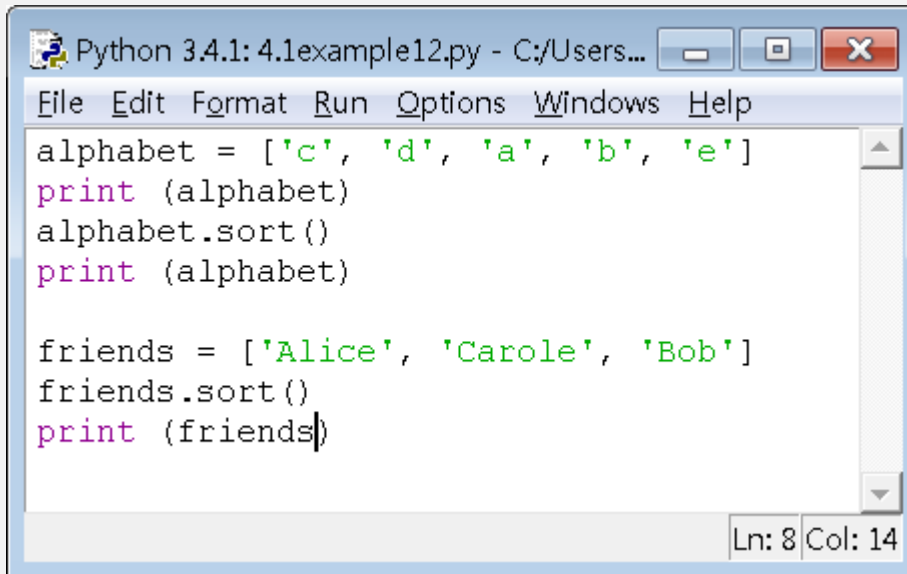
```
a = [1,2,3,4]
result = [num * 3 for num in a if num % 2 == 0]
print(result)
[6, 12]
```

Loop in a list

- 실습
 - List comprehension을 사용해서 x 를 생성하시오.
 - $X=\{x^2 \mid x \text{는 } 1 \text{ 이상 } 10 \text{ 이하의 자연수}\}$

Sorting a list

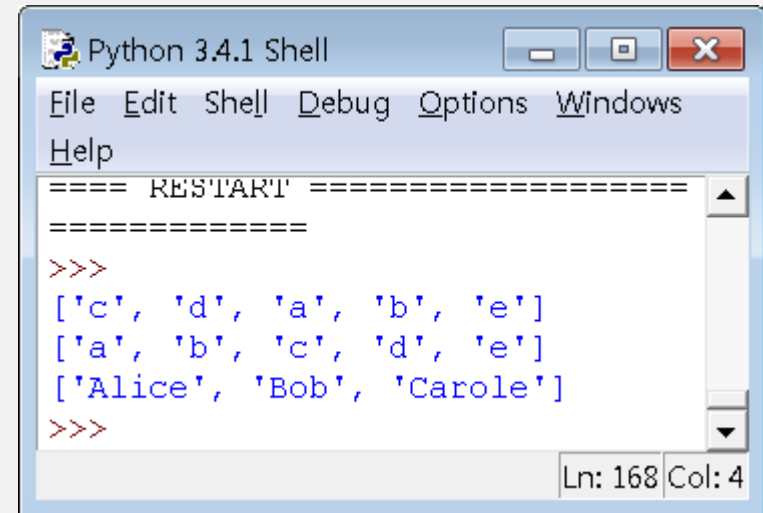
- 정렬 (sorting)
 - 작은 순서 (descending) 정렬 vs 큰 순서 (ascending) 정렬
 - 컴퓨터에게 “크다 작다”를 정해주어야 함!
 - 디폴트(default): 숫자 크기 순서, 사전 등장 순서
 - sort() 함수 사용
 - 정렬 순서를 반대로하려면, sort(reverse=True) 사용



```
Python 3.4.1: 4.1example12.py - C:/Users...
File Edit Format Run Options Windows Help
alphabet = ['c', 'd', 'a', 'b', 'e']
print (alphabet)
alphabet.sort()
print (alphabet)

friends = ['Alice', 'Carole', 'Bob']
friends.sort()
print (friends)
```

Ln: 8 Col: 14

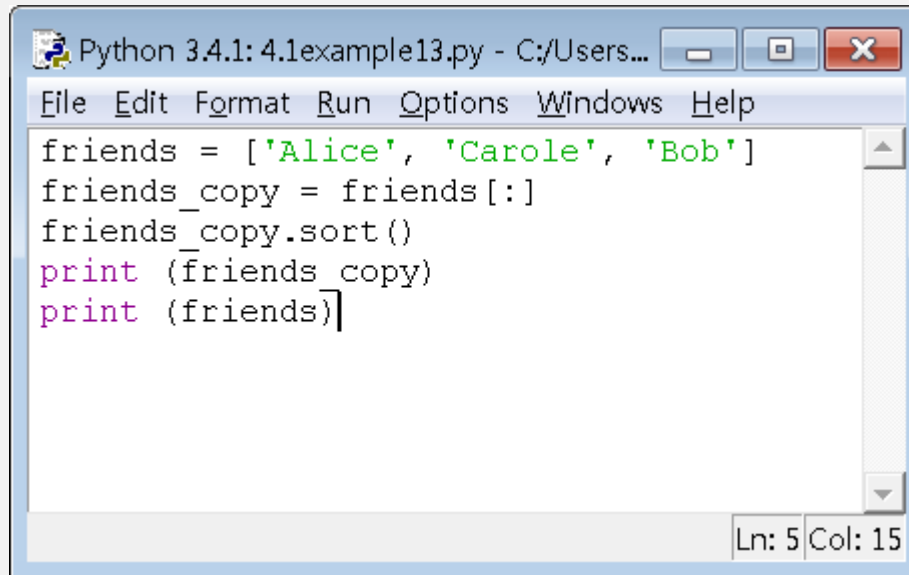


```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
==== RESTART: =====
>>>
['c', 'd', 'a', 'b', 'e']
['a', 'b', 'c', 'd', 'e']
['Alice', 'Bob', 'Carole']
>>>
```

Ln: 168 Col: 4

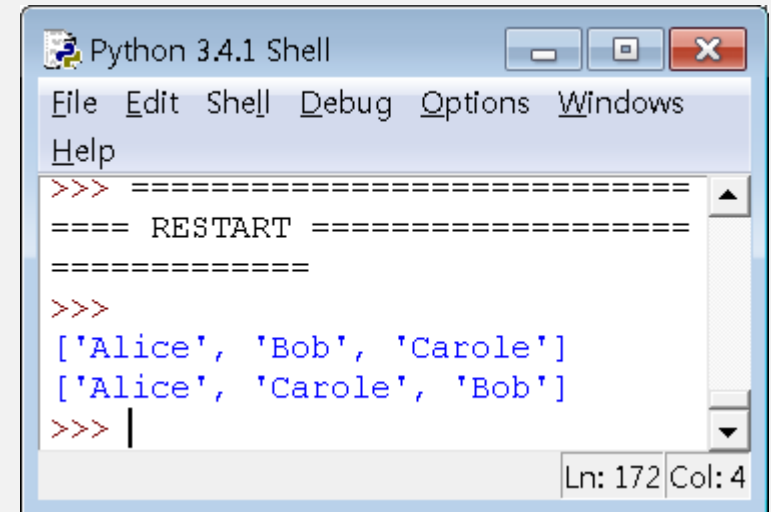
Sorting a list

- 정렬 (sorting)
 - sort() 적용하면 리스트 아이템의 순서가 변경됨에 주의해야 함
 - 원본 유지를 하고 싶으면 복사본을 사용해서 정렬해야 함



```
Python 3.4.1: 4.1example13.py - C:/Users...
File Edit Format Run Options Windows Help
friends = ['Alice', 'Carole', 'Bob']
friends_copy = friends[:]
friends_copy.sort()
print (friends_copy)
print (friends)
```

Ln: 5 Col: 15

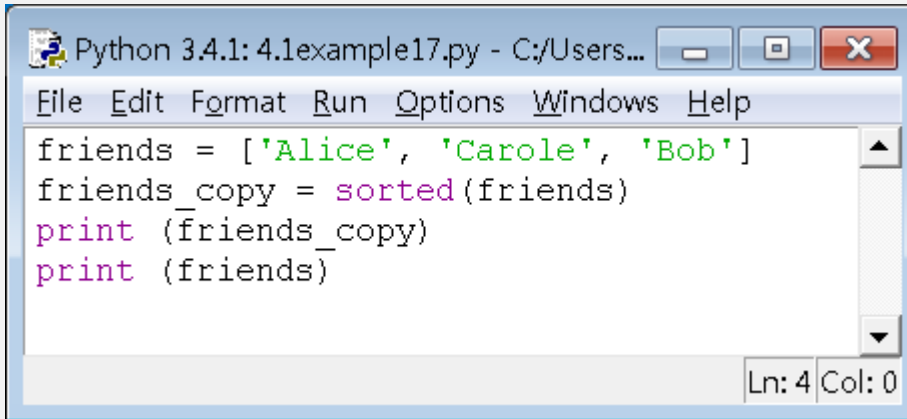


```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> =====
==== RESTART =====
>>>
['Alice', 'Bob', 'Carole']
['Alice', 'Carole', 'Bob']
>>> |
```

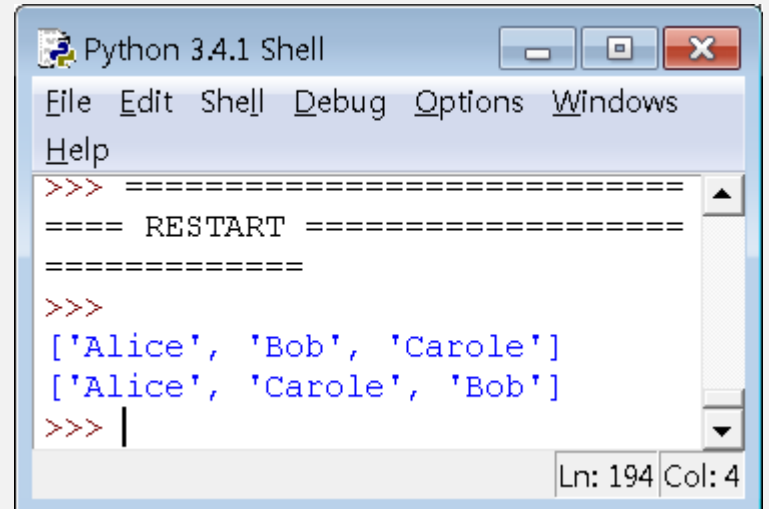
Ln: 172 Col: 4

Sorting a list

- 정렬 (sorting)
 - sorted() : 정렬된 복제 리스트 리턴
 - 원본 유지



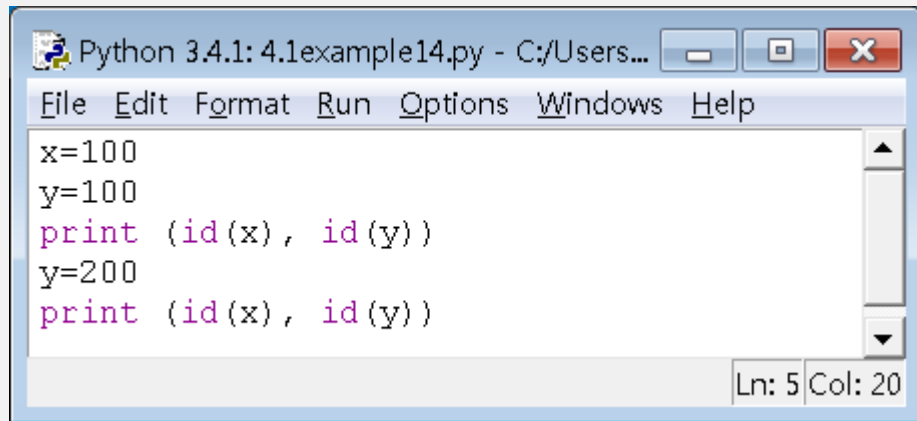
```
Python 3.4.1: 4.1example17.py - C:/Users...
File Edit Format Run Options Windows Help
friends = ['Alice', 'Carole', 'Bob']
friends_copy = sorted(friends)
print (friends_copy)
print (friends)
Ln: 4 Col: 0
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> =====
==== RESTART =====
>>>
['Alice', 'Bob', 'Carole']
['Alice', 'Carole', 'Bob']
>>> |
Ln: 194 Col: 4
```

Shallow and deep copy

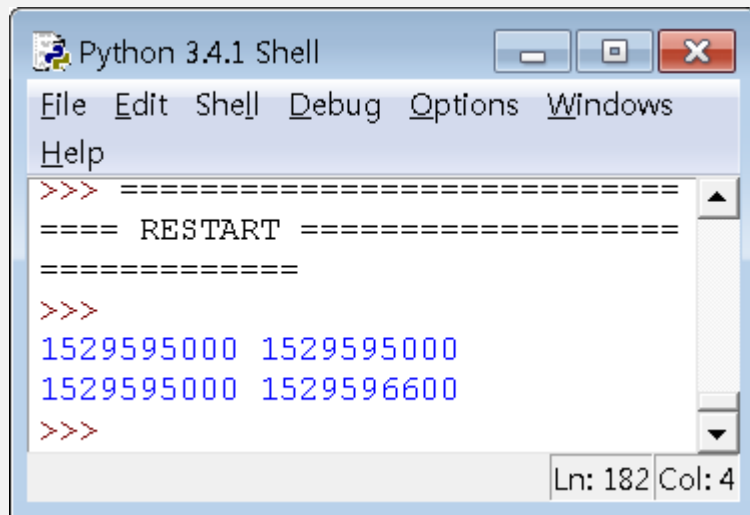
- 파이썬은 객체 지향 언어
 - 객체 id → 확인은 id() 함수 이용



Python 3.4.1: 4.1example14.py - C:/Users...

```
File Edit Format Run Options Windows Help
x=100
y=100
print (id(x), id(y))
y=200
print (id(x), id(y))
```

Ln: 5 Col: 20



Python 3.4.1 Shell

```
File Edit Shell Debug Options Windows Help
>>> =====
===== RESTART =====
>>>
1529595000 1529595000
1529595000 1529596600
>>>
```

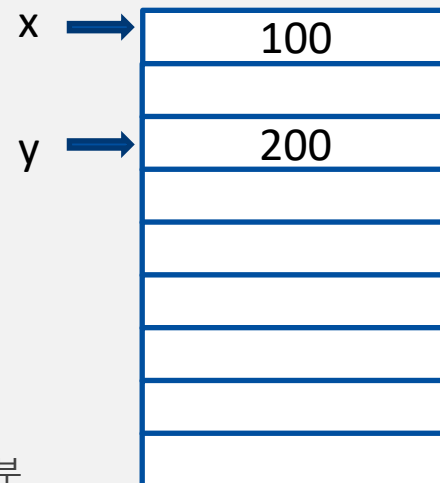
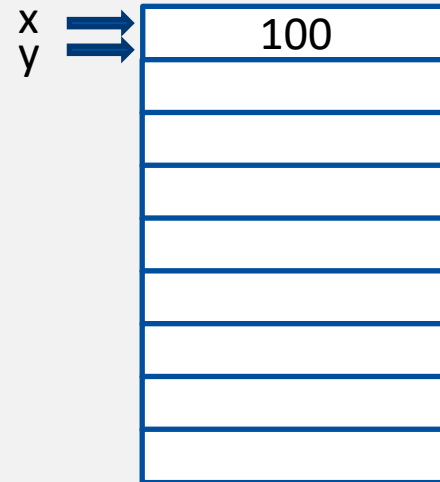
Ln: 182 Col: 4

Shallow and deep copy

- 파이썬은 객체 지향 언어
 - 객체 id → 확인은 id() 함수 이용

```
Python 3.4.1: 4.1example14.py - C:/Users...
File Edit Format Run Options Windows Help
x=100
y=100
print (id(x), id(y))
y=200
print (id(x), id(y))
Ln: 5 Col: 20
```

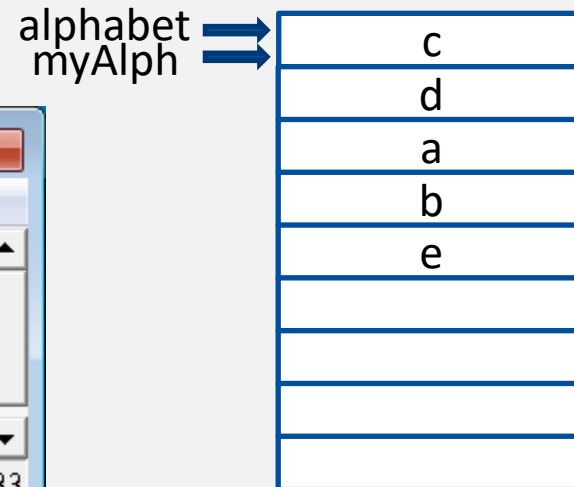
```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> =====
==== RESTART =====
>>>
1529595000 1529595000
1529595000 1529596600
>>>
Ln: 182 Col: 4
```



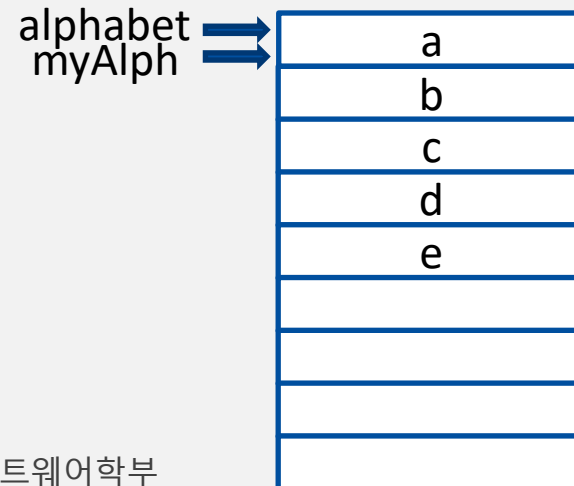
Shallow and deep copy

- 파이썬은 객체 지향 언어
 - 얕은 복사 (shallow copy)

```
Python 3.4.1: 4.1example15.py - C:/Users...
File Edit Format Run Options Windows Help
alphabet = ['c', 'd', 'a', 'b', 'e']
myAlpha = alphabet
print( id(alphabet), id(myAlpha))
alphabet.sort()
print( id(alphabet), id(myAlpha))
Ln: 5 Col: 33
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>> =====
==== RESTART =====
>>>
38497496 38497496
38497496 38497496
>>>
Ln: 186 Col: 4
```



Shallow and deep copy

- 파이썬은 객체 지향 언어
 - 깊은 복사 (deep copy)

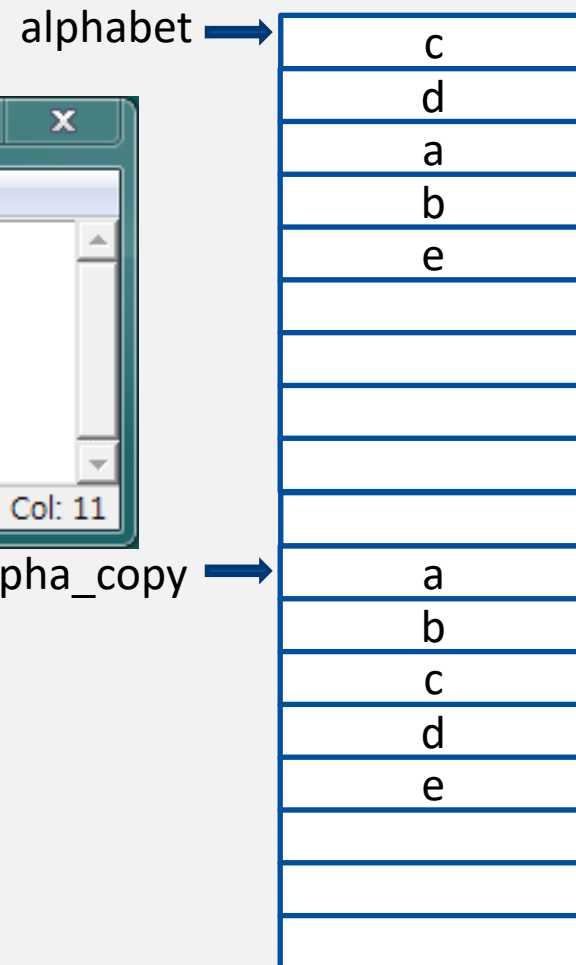
```
test.py - C:\Users\Wjerrick\Desktop\test.py ...
File Edit Format Run Options Window Help
import copy

alphabet = ['c', 'd', 'a', 'b', 'e']
alpha_copy = copy.deepcopy(alphabet)
print(id(alphabet), id(alpha_copy))
alpha_copy.sort()
print(id(alphabet), id(alpha_copy))

Ln: 1 Col: 11
```

```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows
Help
>>> =====
==== RESTART =====
>>>
49508304 31928528
49508304 31928528
>>>

Ln: 190 Col: 4
```



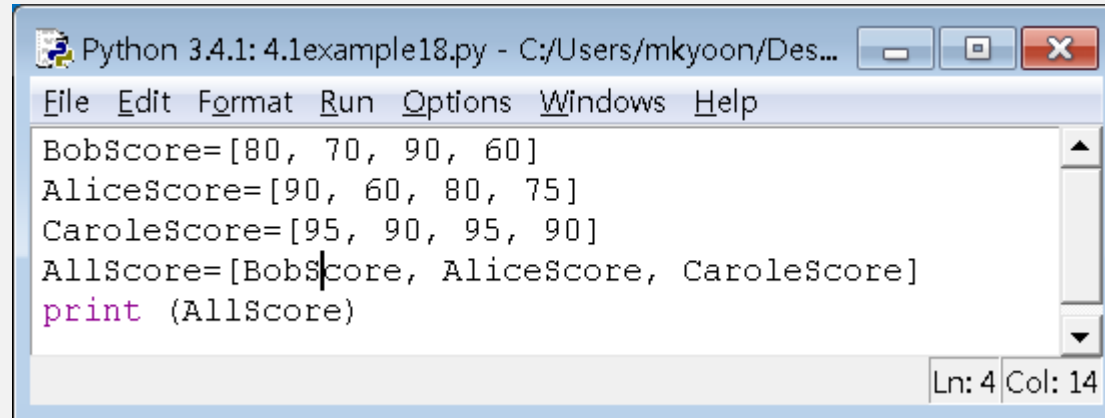
Lists of lists

- 리스트로 구성된 리스트
 - 다차원 자료구조
 - 행렬

	English	Math	Computer	Physics
Bob	80	70	90	60
Alice	90	60	80	75
Carole	95	90	95	90

Lists of lists

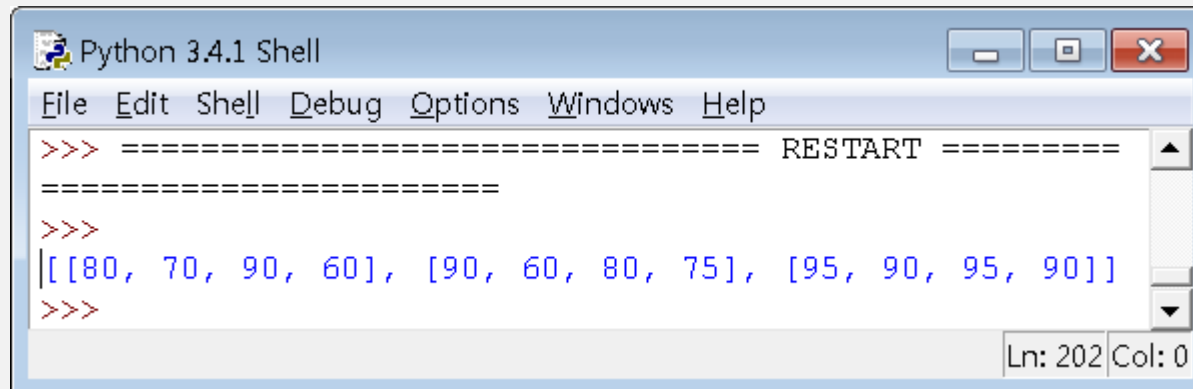
- 리스트로 구성된 리스트
 - 다차원 자료구조
 - 행렬



A screenshot of a Python 3.4.1 IDE window titled "Python 3.4.1: 4.1example18.py - C:/Users/mkyoon/Des...". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor contains the following Python code:

```
BobScore=[80, 70, 90, 60]
AliceScore=[90, 60, 80, 75]
CaroleScore=[95, 90, 95, 90]
AllScore=[BobScore, AliceScore, CaroleScore]
print (AllScore)
```

The status bar at the bottom right shows "Ln: 4 Col: 14".



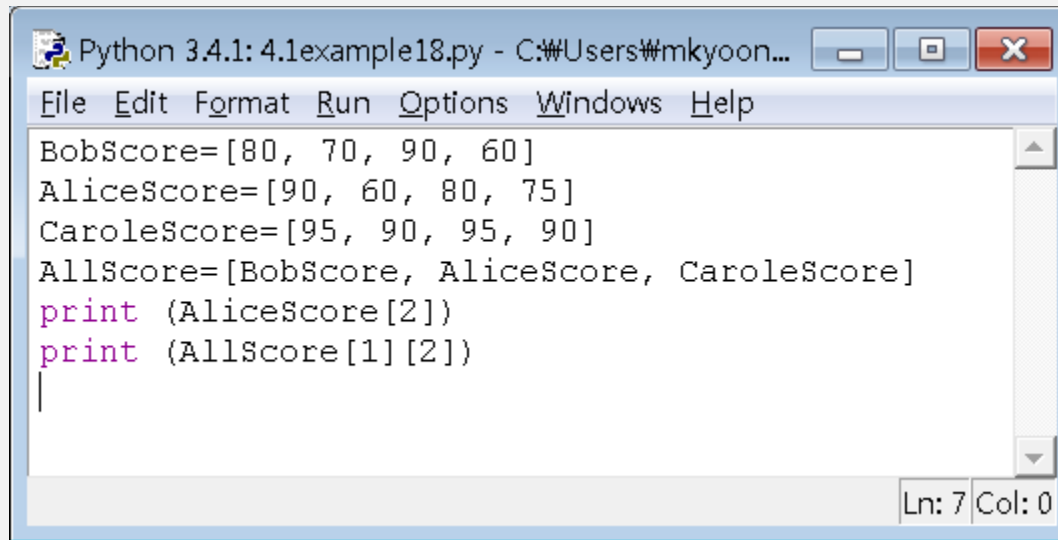
A screenshot of a Python 3.4.1 Shell window titled "Python 3.4.1 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The shell shows the output of the code from the previous window:

```
>>> ===== RESTART =====
>>>
[[80, 70, 90, 60], [90, 60, 80, 75], [95, 90, 95, 90]]
>>>
```

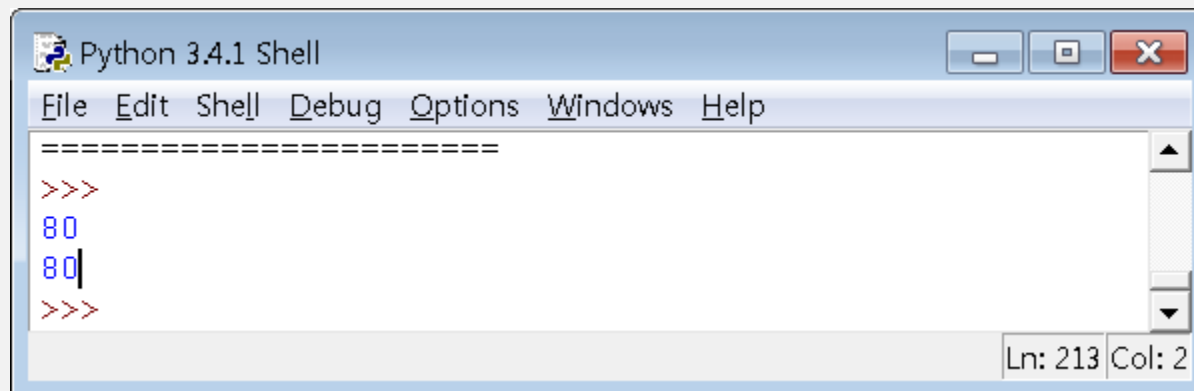
The status bar at the bottom right shows "Ln: 202 Col: 0".

Lists of lists

- 리스트로 구성된 리스트
 - 특정 값 하나에 접근하기: 내부 리스트의 인덱스 사용



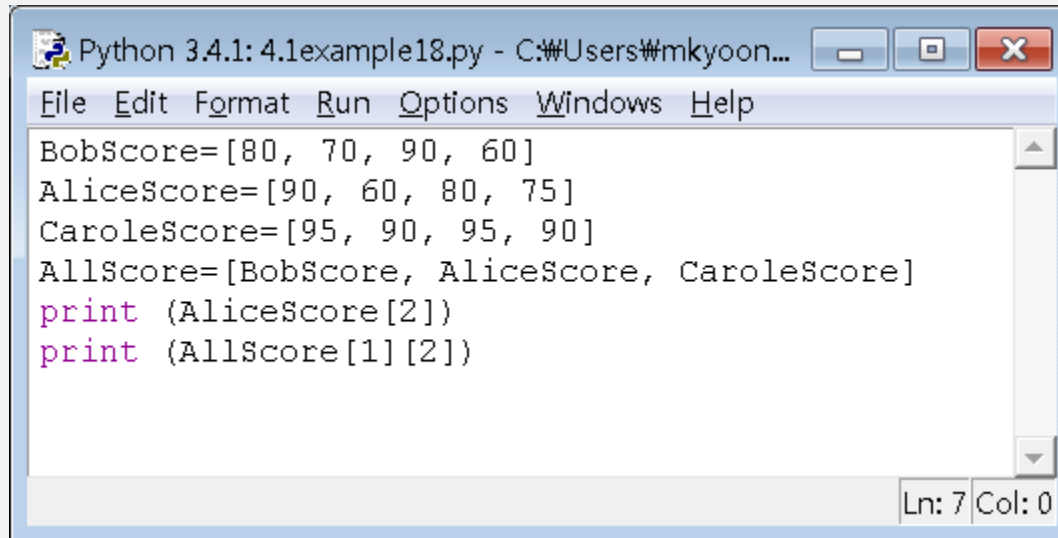
```
Python 3.4.1: 4.1example18.py - C:\Users\mkkyoon...
File Edit Format Run Options Windows Help
BobScore=[80, 70, 90, 60]
AliceScore=[90, 60, 80, 75]
CaroleScore=[95, 90, 95, 90]
AllScore=[BobScore, AliceScore, CaroleScore]
print (AliceScore[2])
print (AllScore[1][2])
|
Ln: 7 Col: 0
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
=====
>>>
80
80|
>>>
Ln: 213 Col: 2
```

Lists of lists

- 리스트로 구성된 리스트
 - 인덱싱

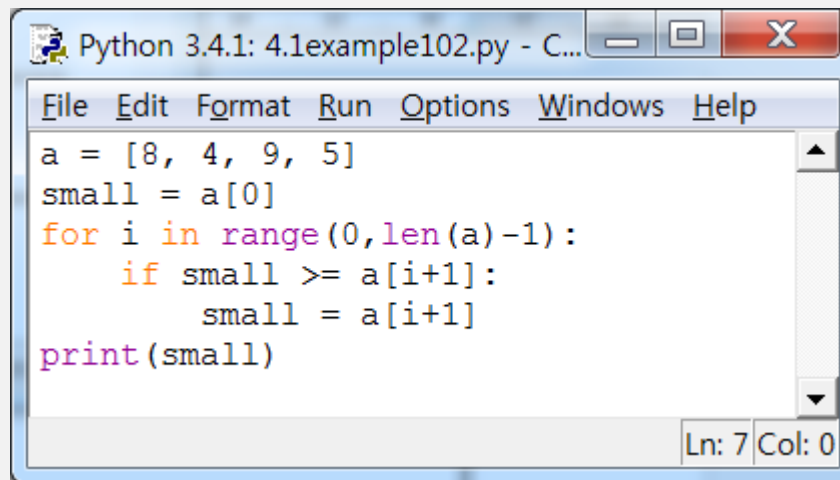


```
Python 3.4.1: 4.1example18.py - C:\Users\mkkyoon...
File Edit Format Run Options Windows Help
BobScore=[80, 70, 90, 60]
AliceScore=[90, 60, 80, 75]
CaroleScore=[95, 90, 95, 90]
AllScore=[BobScore, AliceScore, CaroleScore]
print (AliceScore[2])
print (AllScore[1][2])
Ln: 7 Col: 0
```

AllScore	English	Math	Computer	Physics
BobScore	[0][0]	[0][1]	[0][2]	[0][3]
AliceScore	[1][0]	[1][1]	[1][2]	[1][3]
CaroleScore	[2][0]	[2][1]	[2][2]	[2][3]

실습

- 정수로 구성된 리스트에서 가장 작은 수를 구하시오
 - `a=[8, 4, 9, 5]`
 - `small = a[0]`
 - `small`과 `a[1]`비교 → `a[1]`이 더 작으면 `small = a[1]`
 - `small`과 `a[2]`비교 → `a[2]`이 더 작으면 `small = a[2]`
 - `small`과 `a[3]`비교 → `a[3]`이 더 작으면 `small = a[3]`



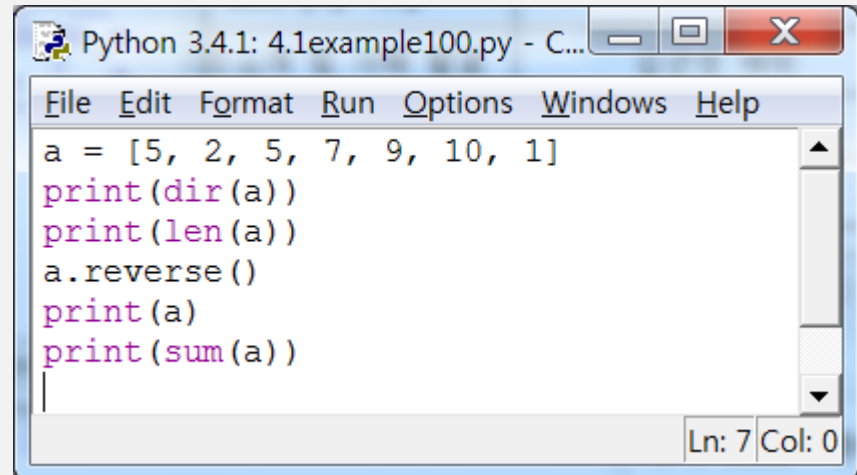
```
Python 3.4.1: 4.1example102.py - C...
File Edit Format Run Options Windows Help
a = [8, 4, 9, 5]
small = a[0]
for i in range(0, len(a)-1):
    if small >= a[i+1]:
        small = a[i+1]
print(small)
Ln: 7 Col: 0
```

실습

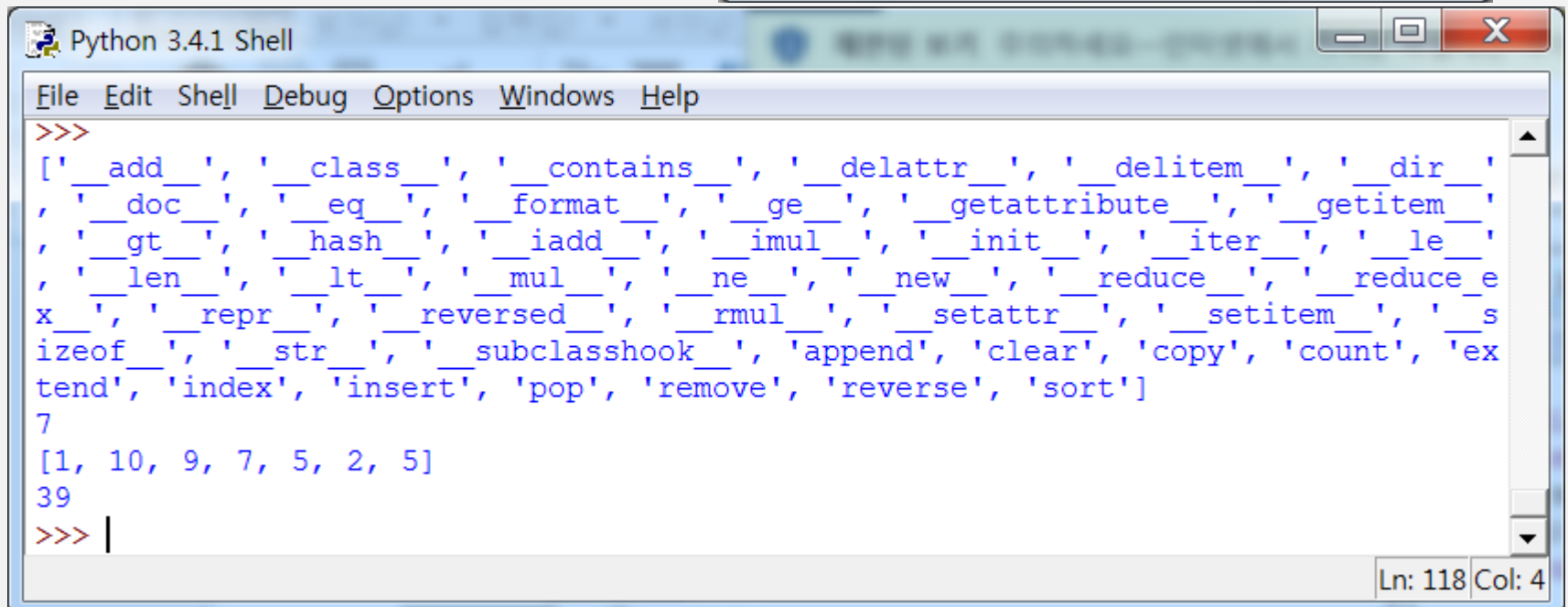
- 정수로 구성된 리스트에서 가장 큰 수를 구하시오

실습

- LIST
 - LIST 제공 함수 사용해 보기
 - sum(리스트)
 - 리스트.reverse()
 - min(리스트)
 - max(리스트)



```
Python 3.4.1: 4.1example100.py - C...
File Edit Format Run Options Windows Help
a = [5, 2, 5, 7, 9, 10, 1]
print(dir(a))
print(len(a))
a.reverse()
print(a)
print(sum(a))
|
Ln: 7 Col: 0
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
>>>
['_add_', '_class_', '_contains_', '_delattr_', '_delitem_', '_dir_',
'_doc_', '_eq_', '_format_', '_ge_', '_getattr_', '_getitem_',
'_gt_', '_hash_', '_iadd_', '_imul_', '_init_', '_iter_', '_le_',
'_len_', '_lt_', '_mul_', '_ne_', '_new_', '_reduce_', '_reduce_e
x_', '_repr_', '_reversed_', '_rmul_', '_setattr_', '_setitem_', '_s
izeof_', '_str_', '_subclasshook_', 'append', 'clear', 'copy', 'count', 'ex
tend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
7
[1, 10, 9, 7, 5, 2, 5]
39
>>> |
Ln: 118 Col: 4
```

실습

- 에라토스테네스의 체

- n 이 주어졌을 때, n 까지의 모든 소수를 출력하시오. 리스트를 사용하시오.

- N 까지 모든 소수를 찾는 방법 → 에라토스테네스의 체

- \sqrt{n} 이하의 소수를 찾는다

- » \sqrt{n} 사용 법

- `import math`

- `math.sqrt(n)`

- 찾아진 소수들의 배수를 제거한다

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

숙제

1. 버블정렬을 구현하시오
2. 구현한 버블정렬과 `sort()` 함수와의 처리 시간을 비교하시오
 1. `sort()` 함수는 파이썬이 제공하는 정렬 함수

숙제

- 버블정렬 구현

- n개의 정수로 구성된 리스트 a가 주어졌을 때 작은 값부터 순서대로 정렬하는 프로그램을 작성하시오(sort()와 동일 기능)

- a=[8,4,9,5]

- [0..3] 중 가장 큰 수를 찾아 a[3]로 이동시킴

- » 8과 4비교 → 순서 교환 a=[4,8,9,5]

- » 8과 9비교 → 교환 없음 a=[4,8,9,5]

- » 9와 5비교 → 순서 교환 a=[4,8,5,9]

- [0..2] 중 가장 큰 수를 찾아 a[2]로 이동시킴

- » 4과 8비교 → 교환 없음 a=[4,8,5,9]

- » 8과 5비교 → 순서 교환 a=[4,5,8,9]

- [0..1] 중 가장 큰 수를 찾아 a[1]로 이동시킴

- » 4과 5비교 → 교환 없음 a=[4,5,8,9]

숙제

- 버블정렬 구현

- n 개의 정수로 구성된 리스트 a 가 주어졌을 때 작은 값부터 순서대로 정렬하는 프로그램을 작성하시오(sort()와 동일 기능)

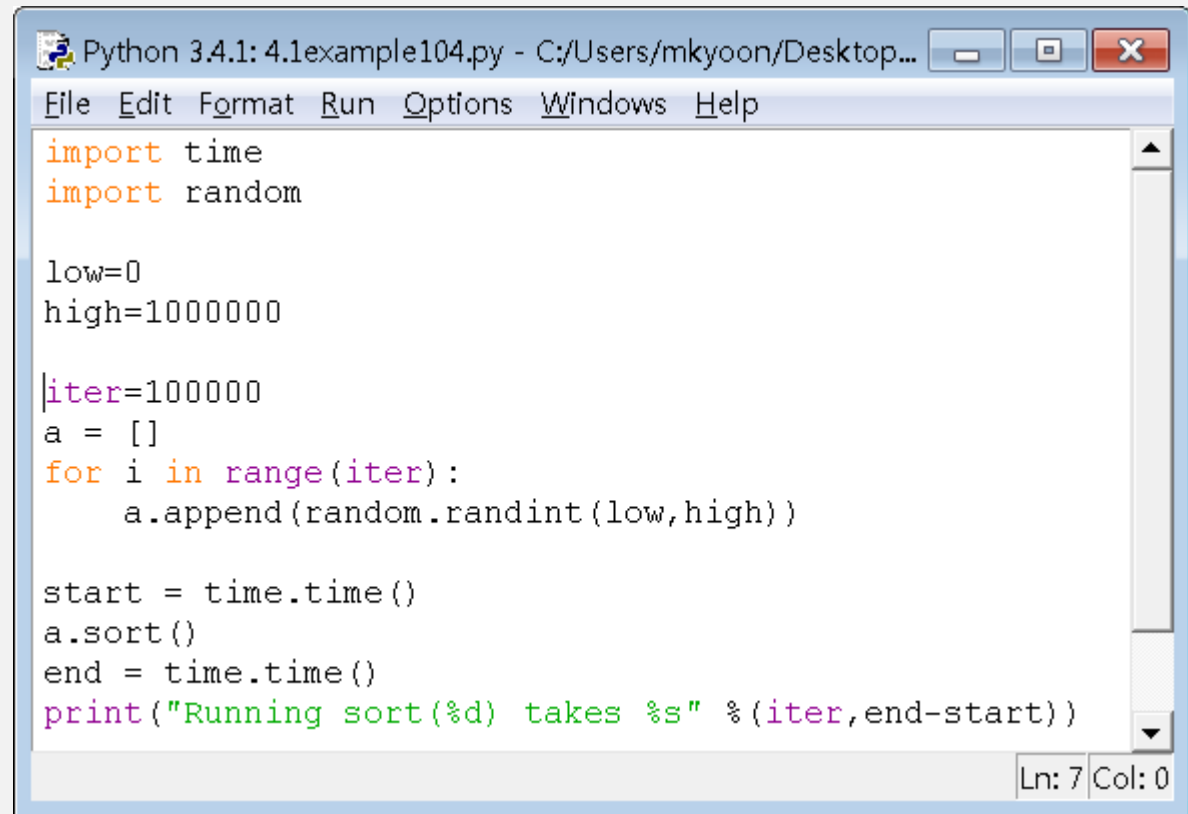
- 일반화 시키면,

- $[0..n-1]$ 중 가장 큰 수를 찾아 $a[n-1]$ 로 이동시킴 $\rightarrow n-1$ 번 비교 발생
 - $[0..n-2]$ 중 가장 큰 수를 찾아 $a[n-2]$ 로 이동시킴 $\rightarrow n-2$ 번 비교 발생
 - ...
 - $[0..1]$ 중 가장 큰 수를 찾아 $a[1]$ 로 이동시킴 $\rightarrow 1$ 번 비교 발생

- $(n-1) + (n-2) + \dots + 1$ 번의 비교가 발생 $\rightarrow (n-1)*n/2$ 번의 비교 발생

속제

- 처리 속도 구하는 방법
 - Time 모듈의 time()함수 사용
 - import time
 - time.time()



```
Python 3.4.1: 4.1example104.py - C:/Users/mkyoon/Desktop...
File Edit Format Run Options Windows Help

import time
import random

low=0
high=1000000

iter=100000
a = []
for i in range(iter):
    a.append(random.randint(low,high))

start = time.time()
a.sort()
end = time.time()
print("Running sort(%d) takes %s" %(iter,end-start))

Ln: 7 Col: 0
```

Running sort(100000) takes 0.045001983642578125