Structured Types 1

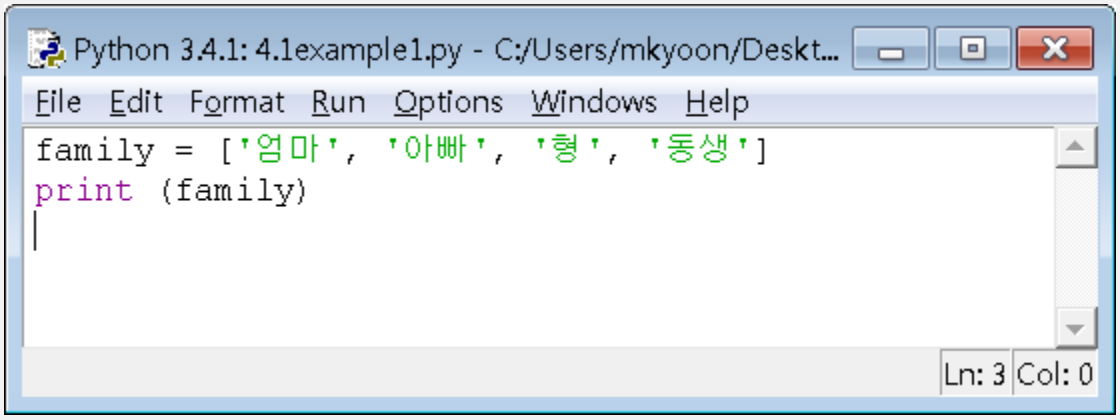
국민대 소프트웨어학부

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]



Creating a list

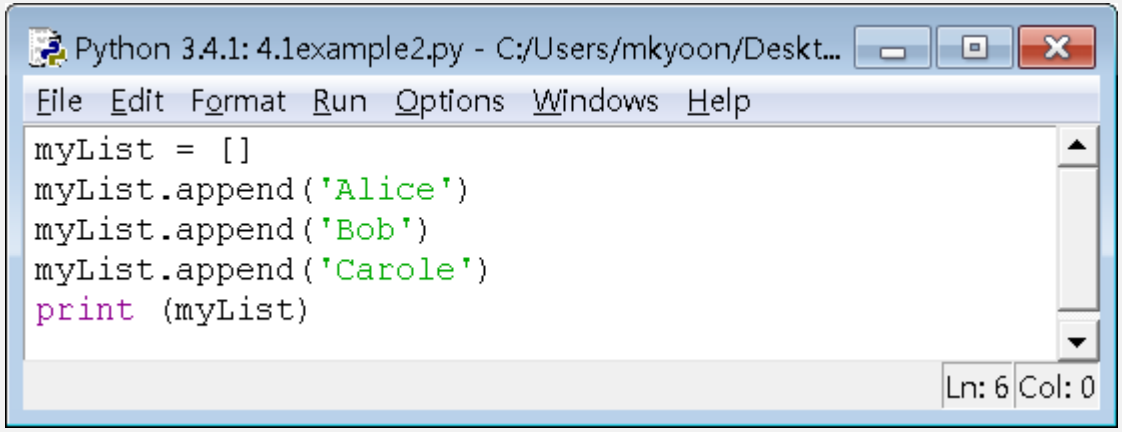
- “[]” 사용
  - myList = []

```
# 리스트를 선언합니다.
infinite_members = [ "김성규", "장동우", "남우현", "이성열", "엘", "이성종" ]

# 출력하기
print(infinite_members)
```

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

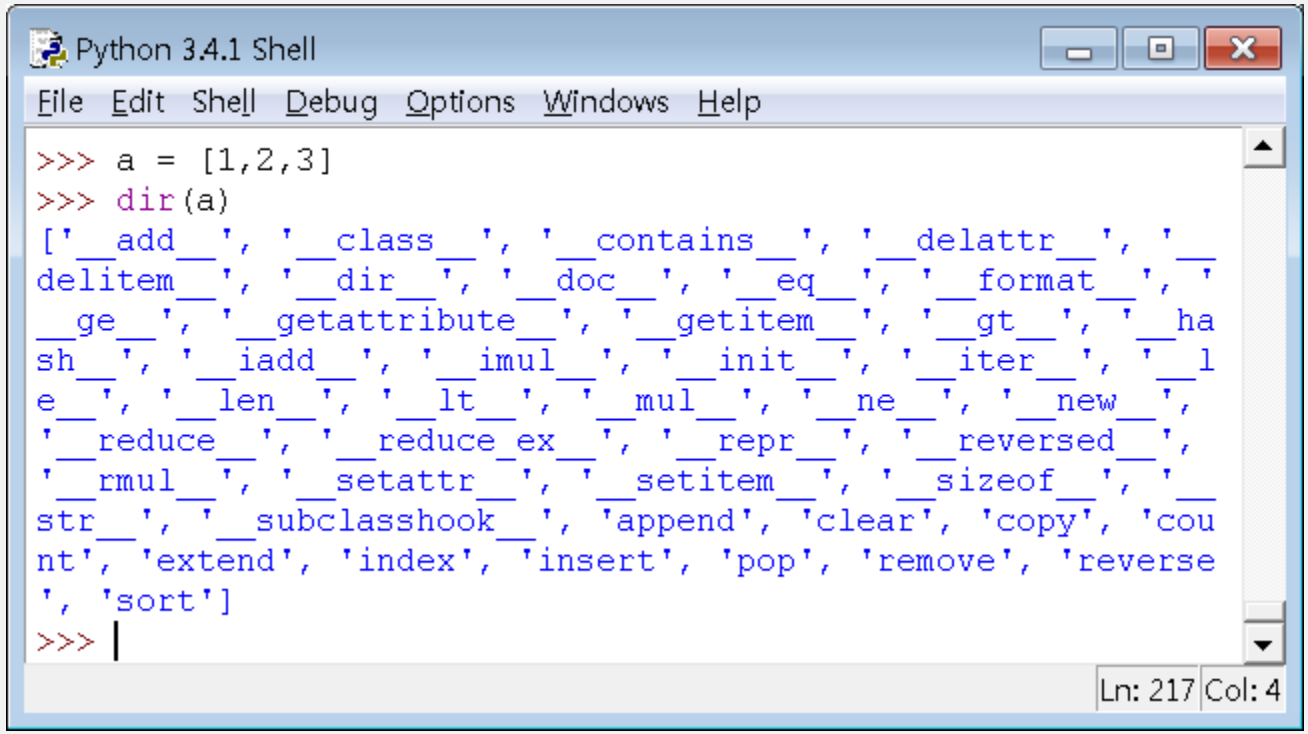
Adding things to a list

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



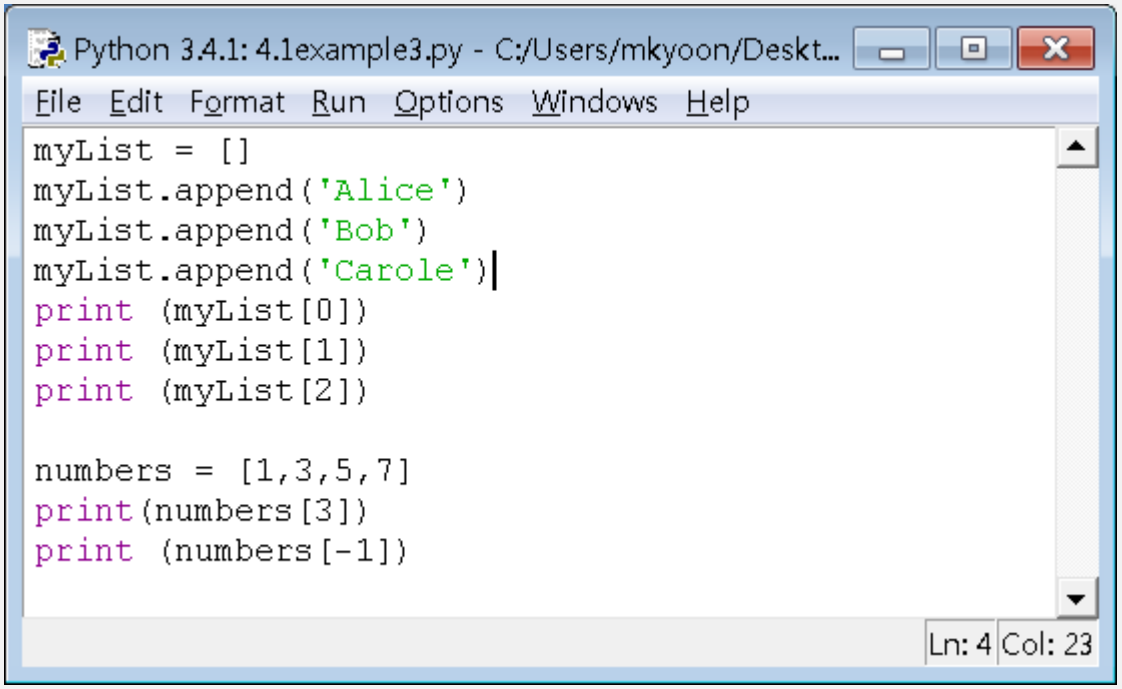
Adding things to a list

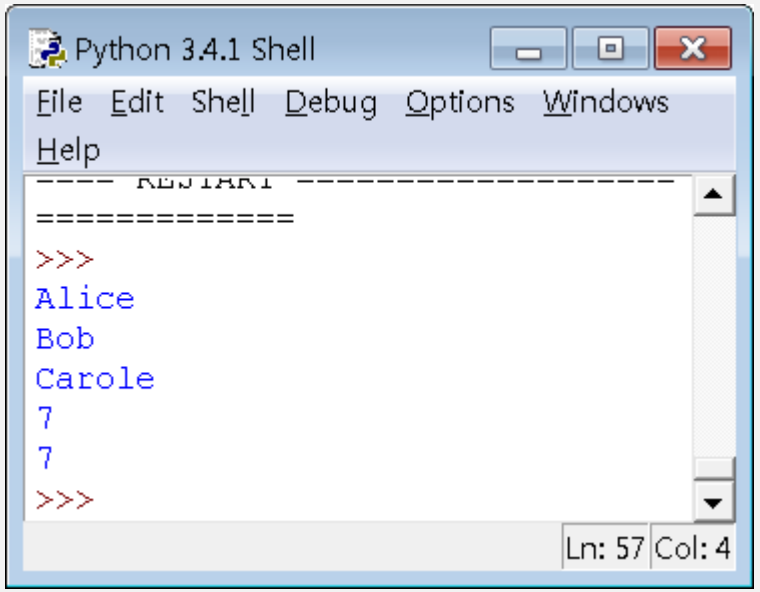
- append() 사용
  - 객체, 함수 → 향후 설명
  - 객체가 어떤 함수와 속성을 가지고 있는지 확인하려면 dir()함수 이용



Getting items from a list

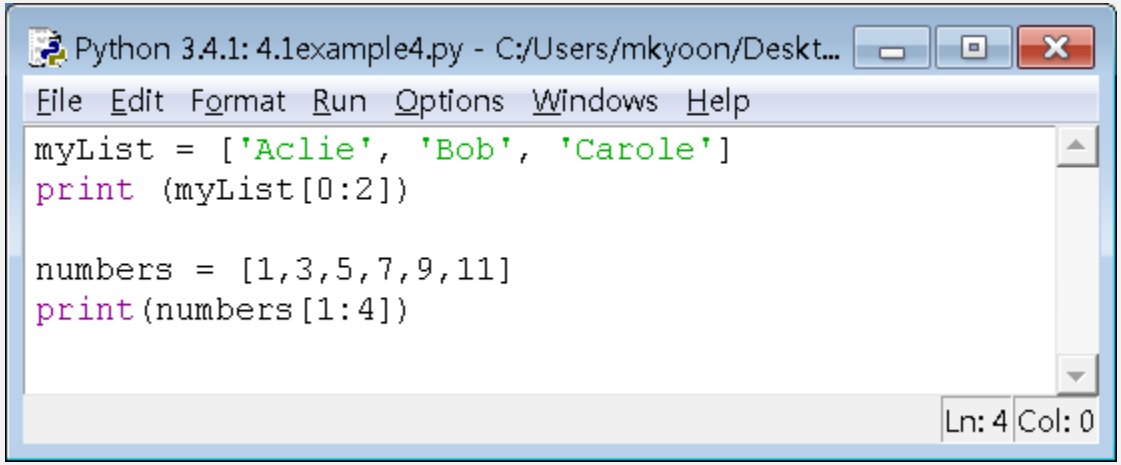
- 리스트에서 아이템 가져오기
  - 인덱스(index) 사용
  - 인덱스는 0부터 시작 → computer science 특징

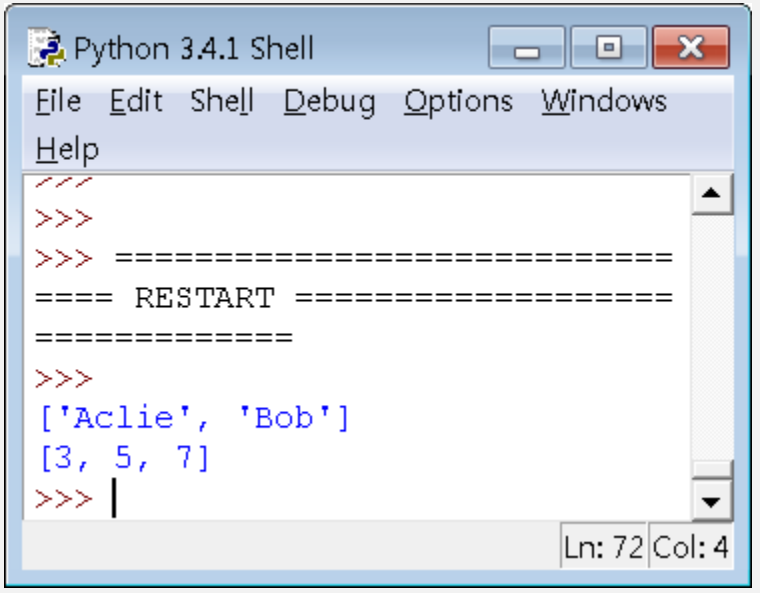




Slicing a list

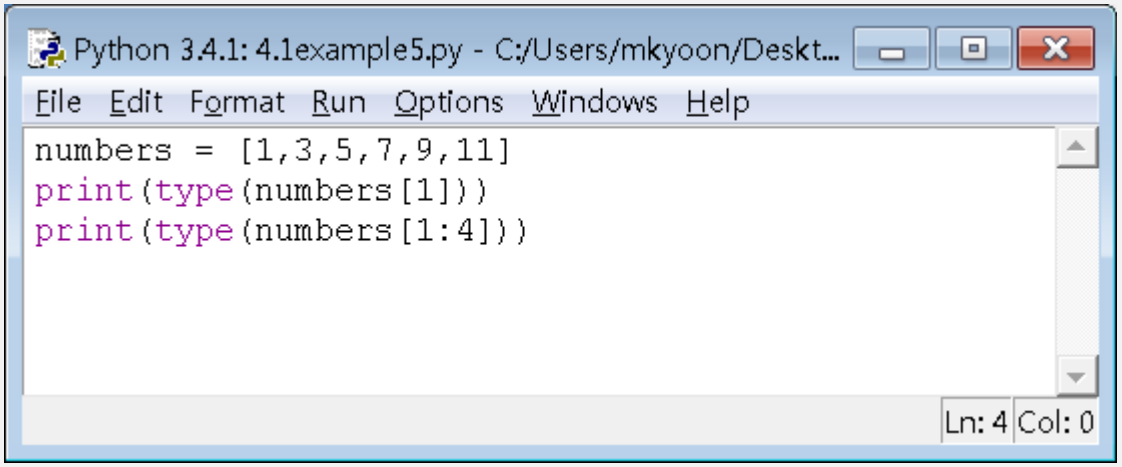
- 리스트 자르기
  - “리스트이름[a:b]” : a번째부터 (b-1)번째까지 아이템으로 구성된 부분 ***리스트***(잘라진 리스트)가 결과값으로 얻어짐
    - 앞에서는 ***아이템***이 결과값으로 얻어짐

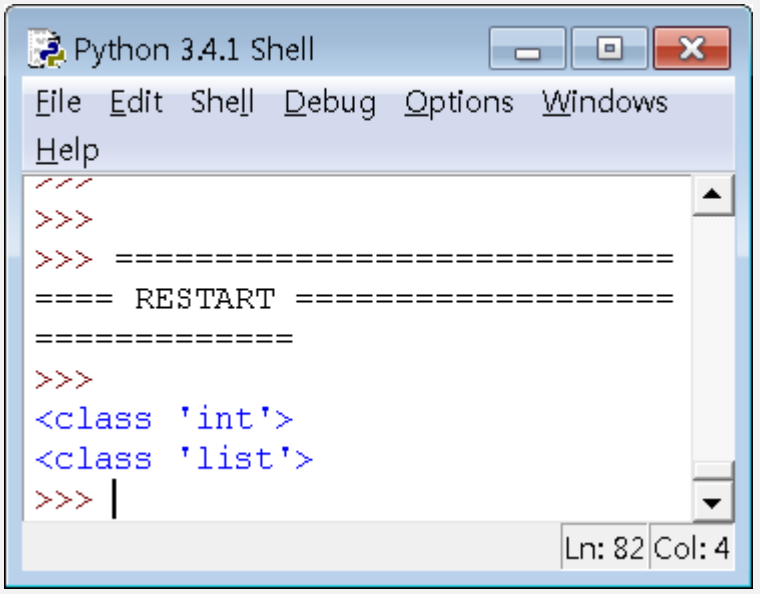




Slicing a list

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





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)
```

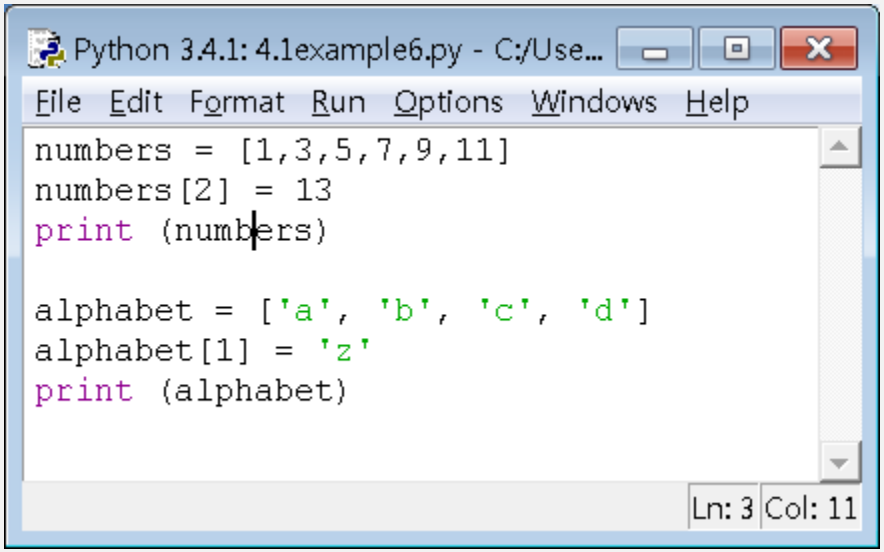
리스트 크기

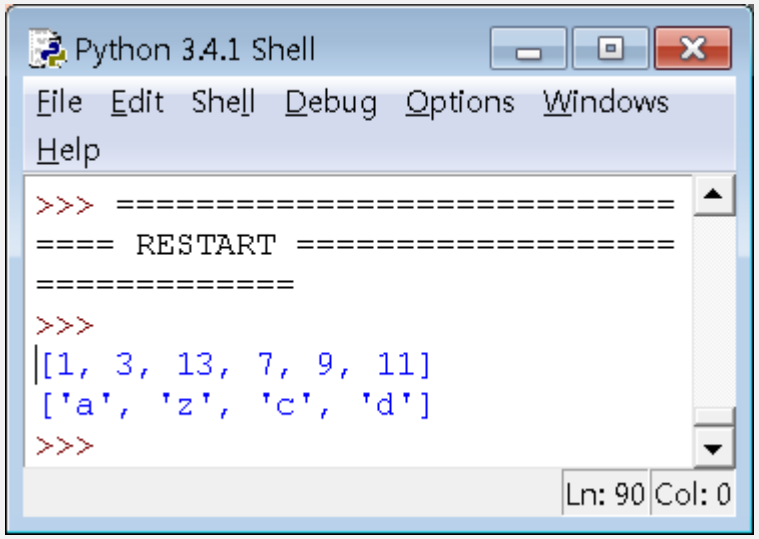
- len(리스트) : 리스트의 길이를 반환 한다.

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

print("인원 수: ", len(graduate))
```

Modifying/Deleting/Searching items

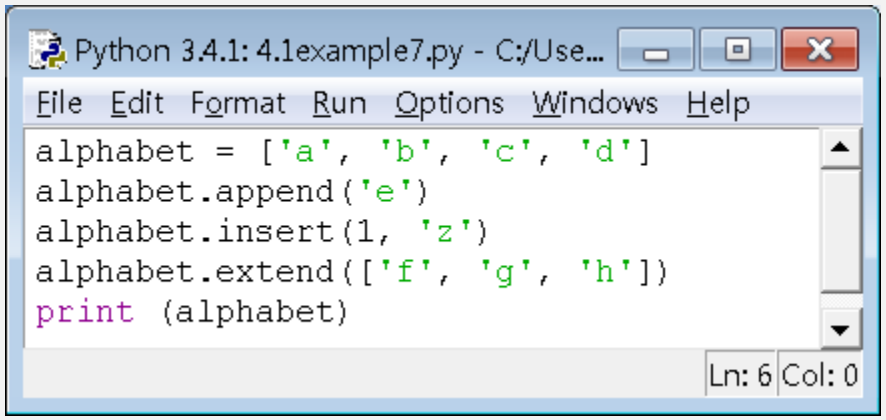
- 수정하기 (Modify)
  - 인덱스를 사용해서 대상 아이템 변경

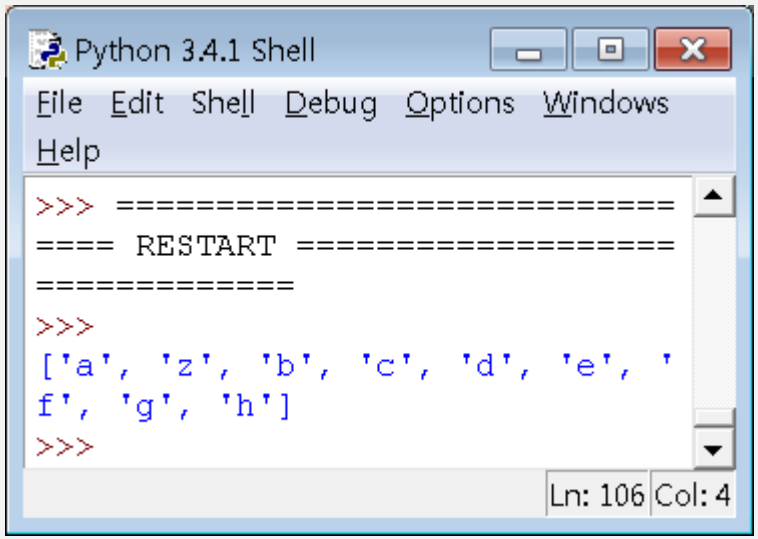




Modifying/Deleting/Searching items

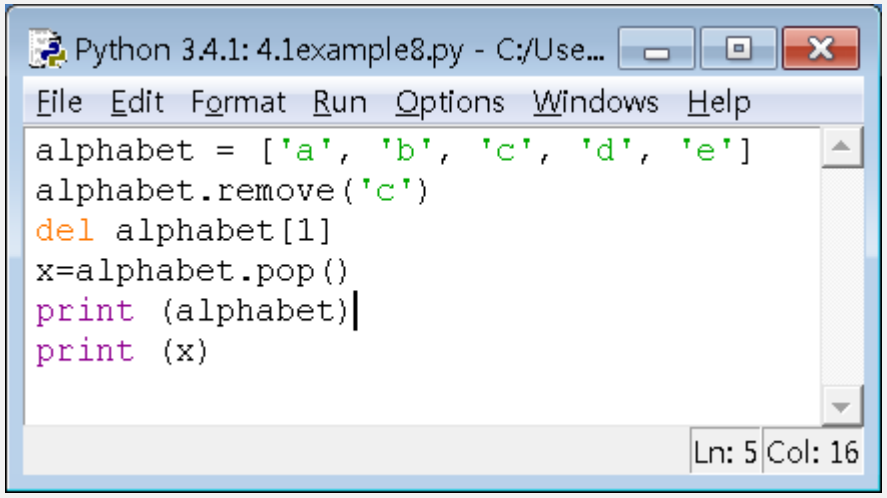
- 리스트에 아이템을 넣는 다양한 방법
  - append(): 리스트 뒤에 아이템 추가
  - insert(): 리스트 내 원하는 위치에 아이템 추가
  - extend(): 여러 아이템을 리스트 뒤에 추가

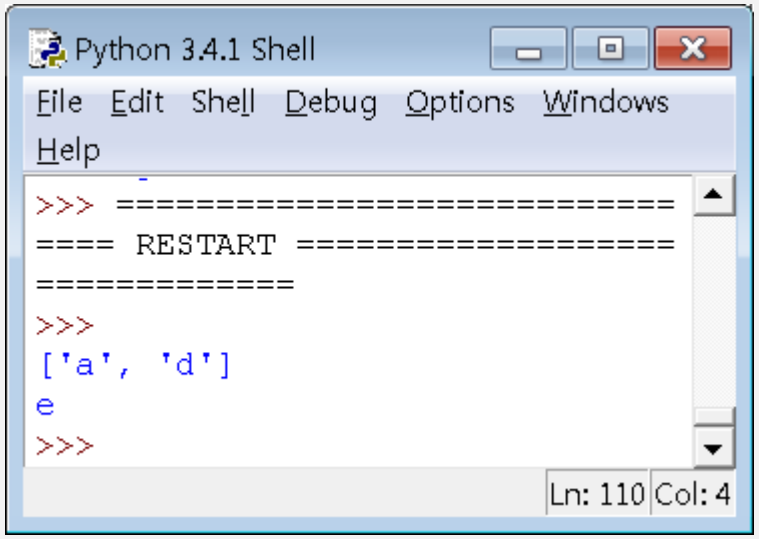




Modifying/Deleting/Searching items

- 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





Modifying/Deleting/Searching items

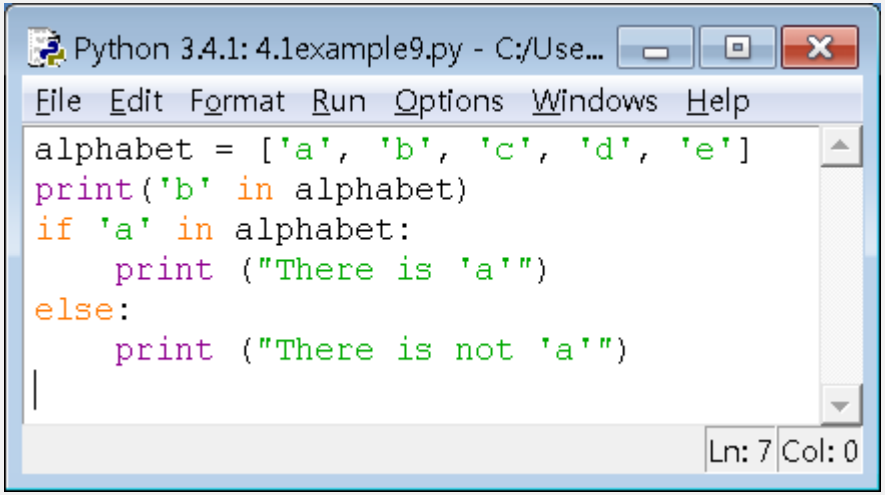
- 리스트에서 아이템 삭제하기

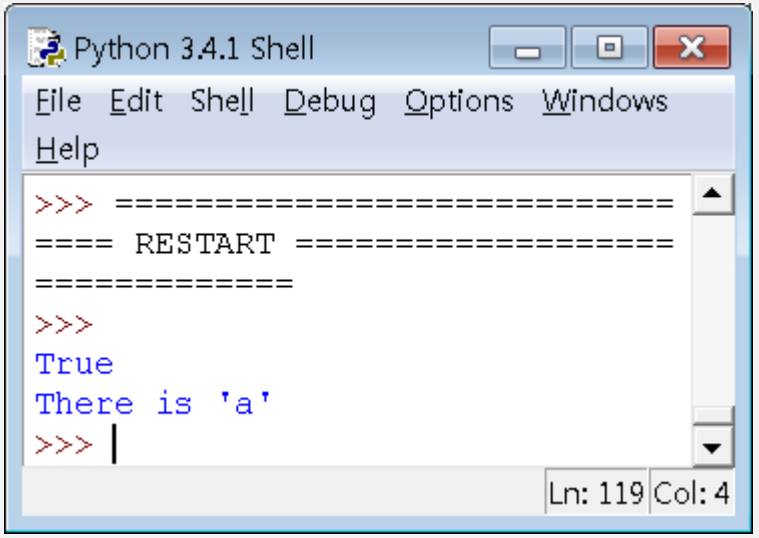
```
lab_members = [ "Jeong", "Jeong", "Lee", "Kim",
                "Yeongjae", "Myeong", "Han"]

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

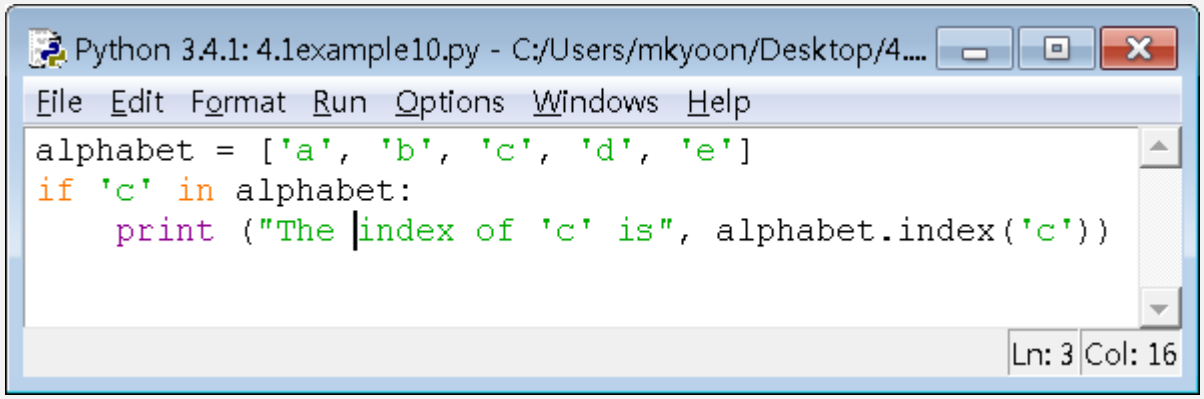
- 리스트에서 찾기 (search)
  - 'in' 키워드: Boolean 값(True/False) 리턴

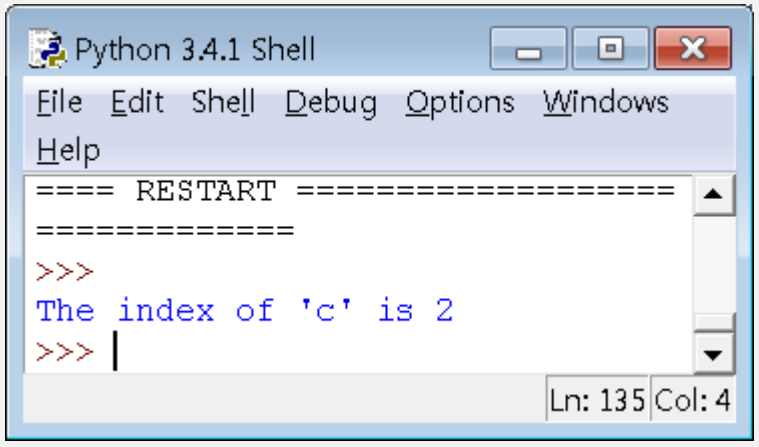




Modifying/Deleting/Searching items

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





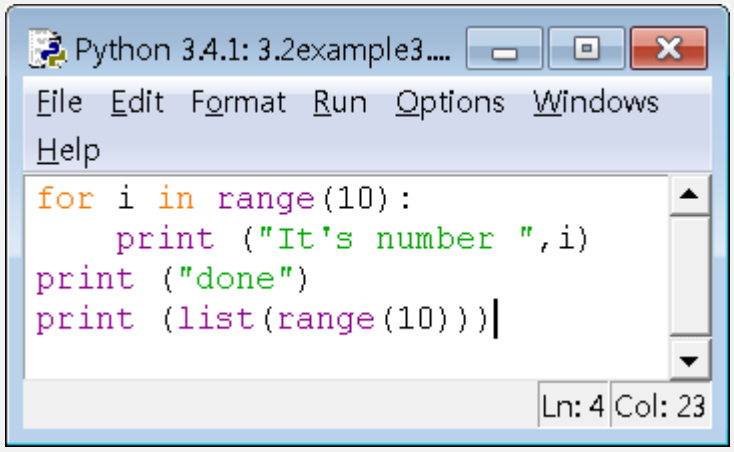
Modifying/Deleting/Searching items

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

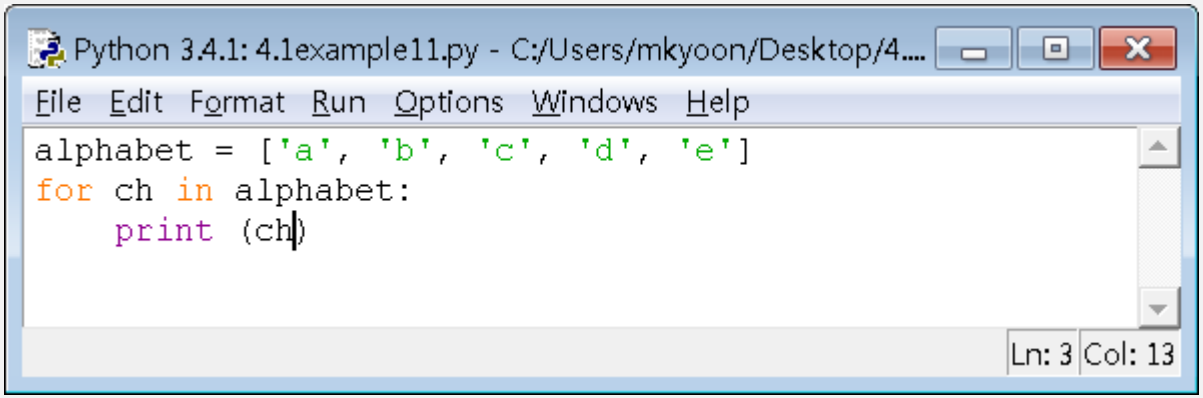
```
infinite_members = [ "김성규", "장동우", "남우현", "이성열", "엘", "이성종" ]
name = input("이름 >")
if name in infinite_members :
    idx = infinite_members.index(name)
    print("{}는 인피니트의 {}번째 멤버입니다.".format(name, idx))
else :
    print("{}는 인피니트의 멤버가 아닙니다.".format(name))
```

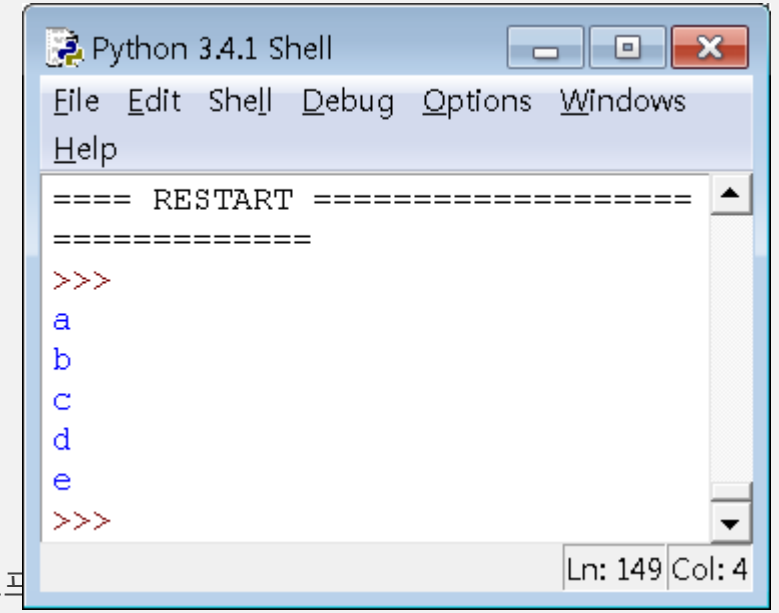
Looping through a list

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



Looping through a list





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]
```


참고: https://wikidocs.net/22

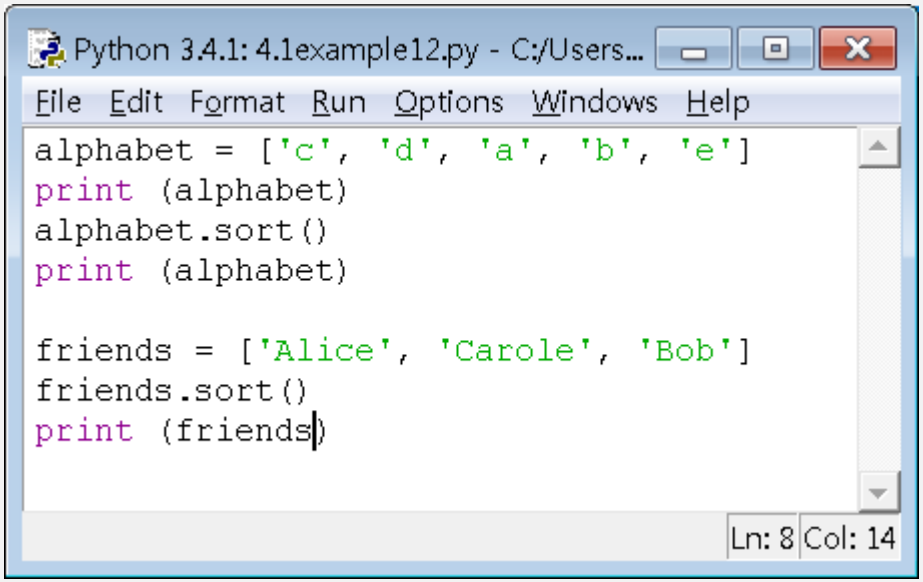
Loop in a list

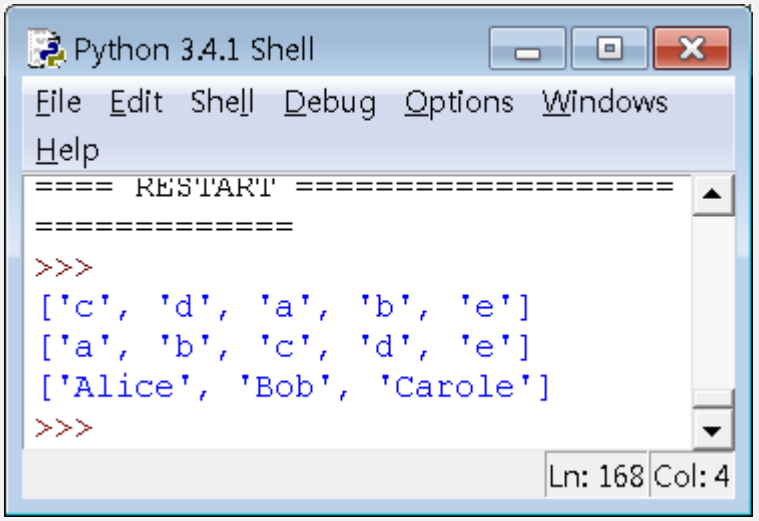
- 실습
  - List comprehension을 사용해서 X를 생성하시오.
  - $X=\{x^2 \mid x$는 1 이상 10 이하의 자연수$\}$

참고: https://wikidocs.net/22

Sorting a list

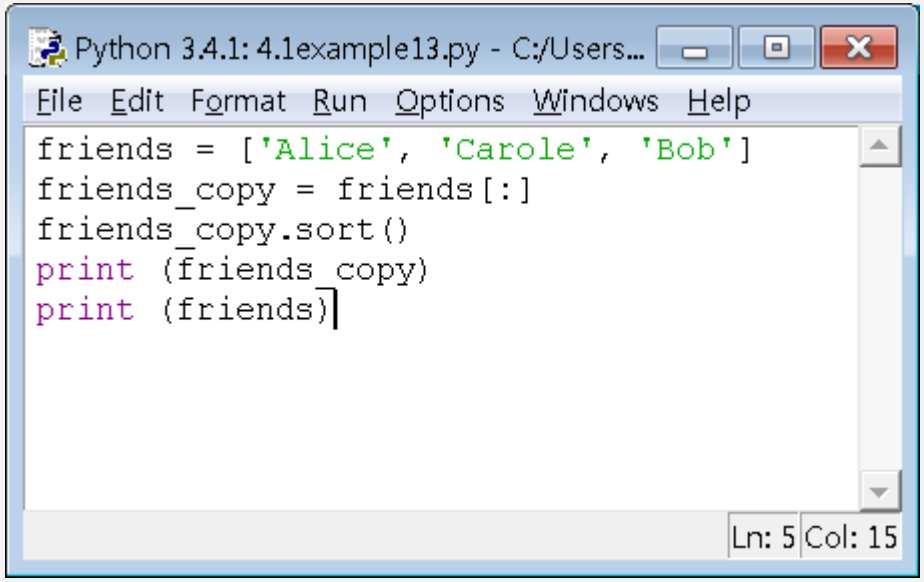
- 정렬 (sorting)
  - 작은 순서 (descending) 정렬 vs 큰 순서 (ascending) 정렬
  - 컴퓨터에게 "크다 작다"를 정해주어야 함!
    - 디폴트(default): 숫자 크기 순서, 사전 등장 순서
  - sort() 함수 사용
    - 정렬 순서를 반대로하려면, sort(reverse=True) 사용

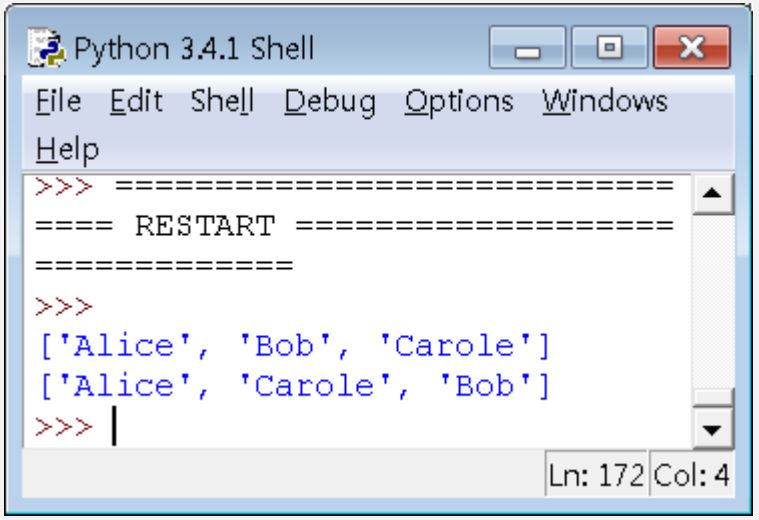




Sorting a list

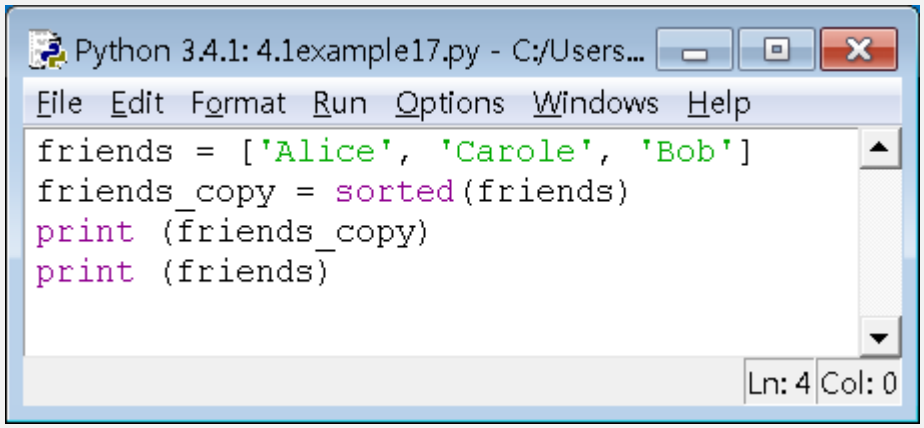
- 정렬 (sorting)
  - sort() 적용하면 리스트 아이템의 순서가 변경됨에 주의해야 함
  - 원본 유지를 하고 싶으면 복사본을 사용해서 정렬해야 함

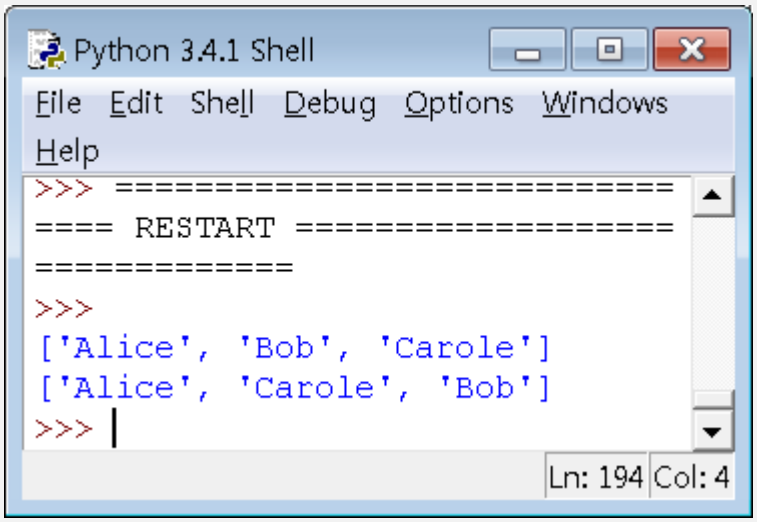




Sorting a list

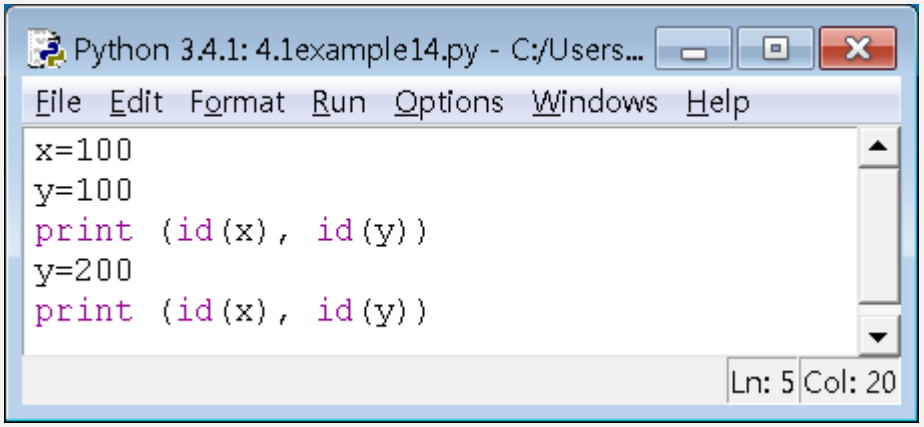
- 정렬 (sorting)
  - sorted() : 정렬된 복제 리스트 리턴
    - 원본 유지

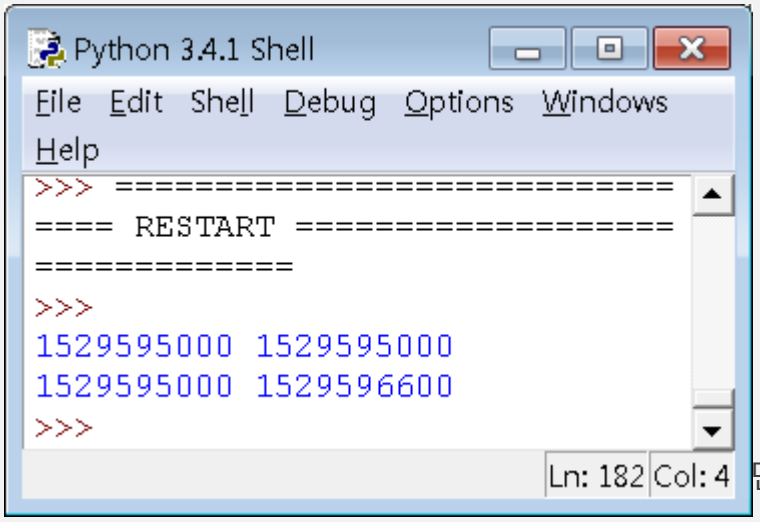




Shallow and deep copy

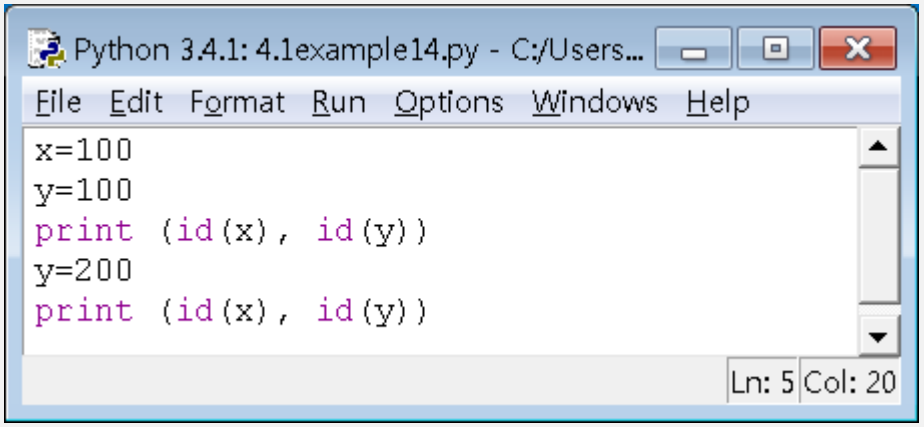
- 파이썬은 객체 지향 언어
  - 객체 id → 확인은 id() 함수 이용

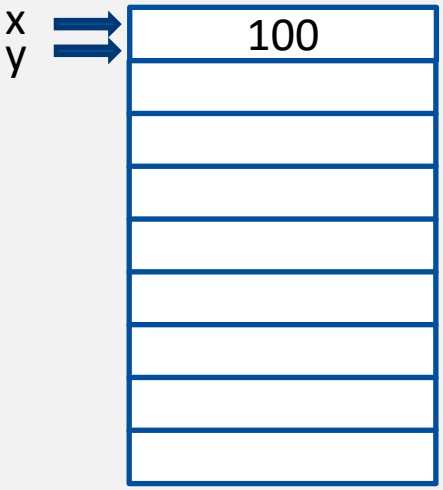


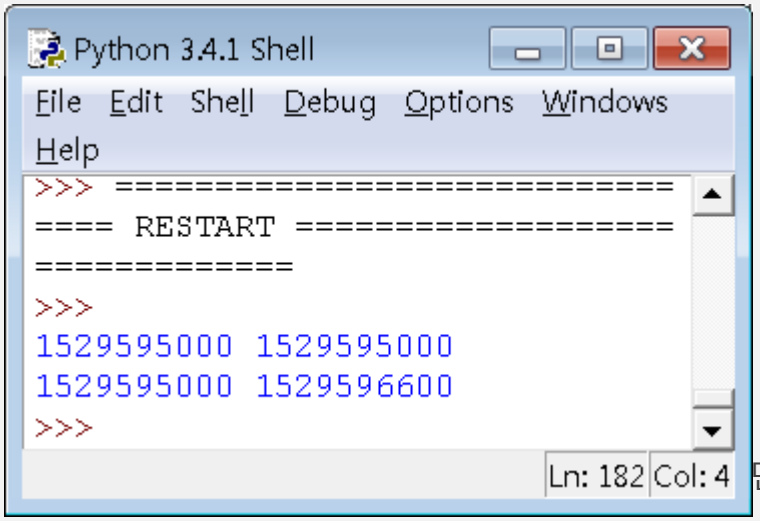


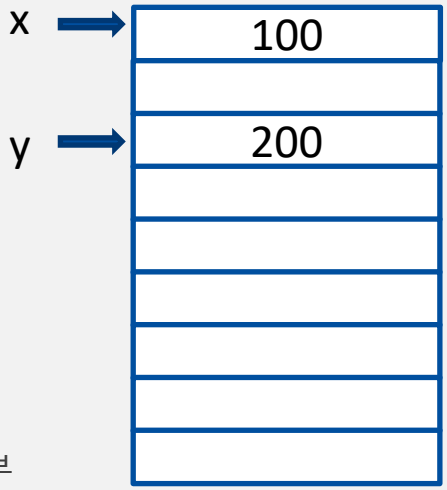
Shallow and deep copy

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



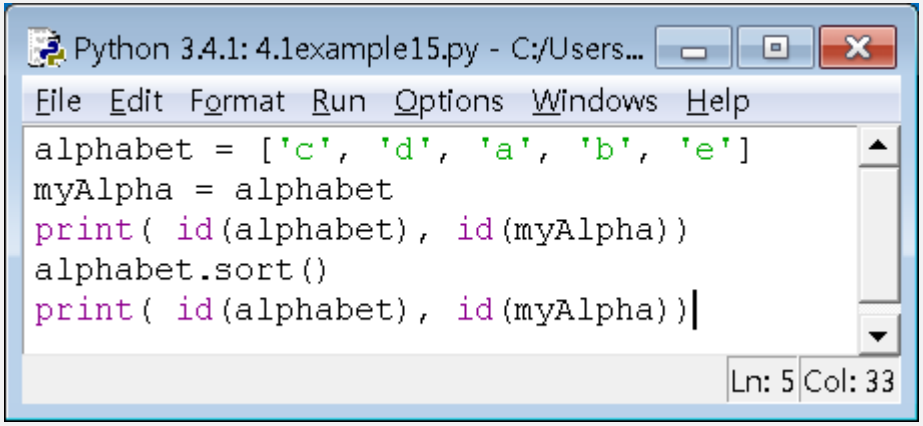


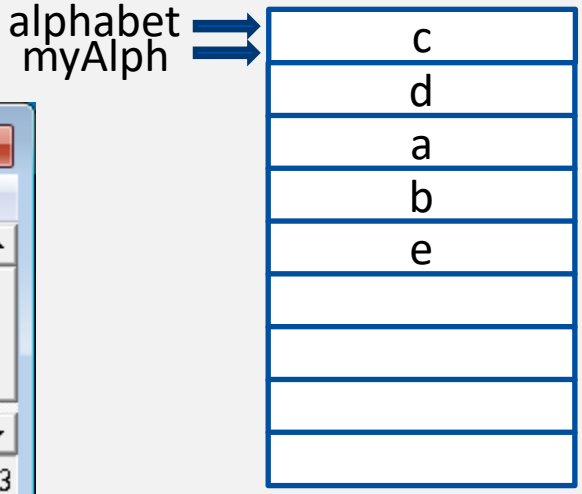


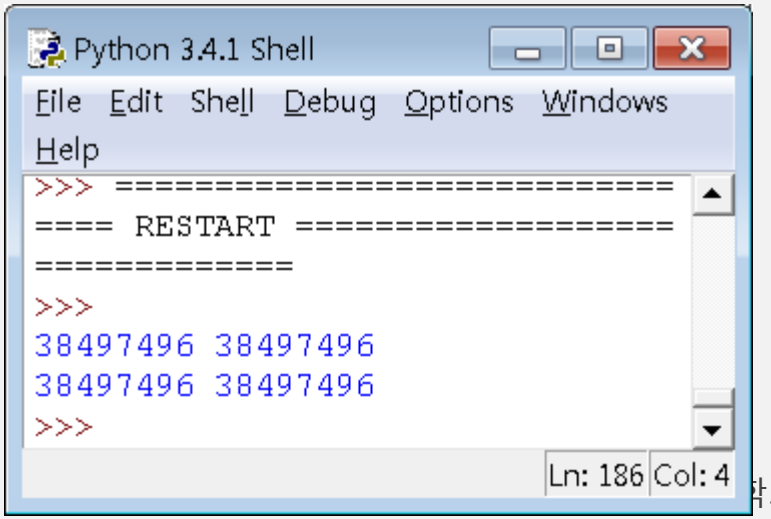


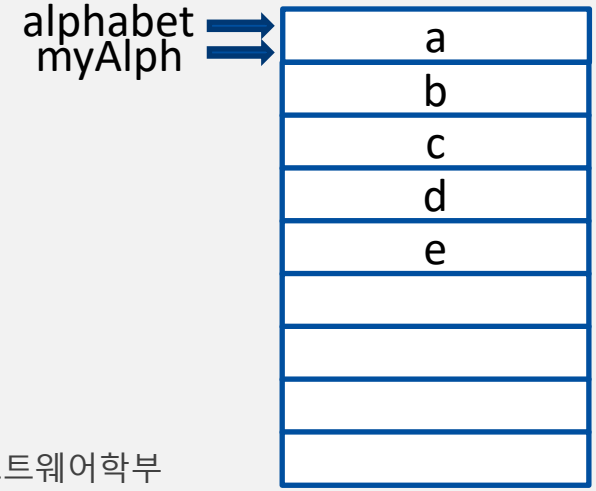
Shallow and deep copy

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



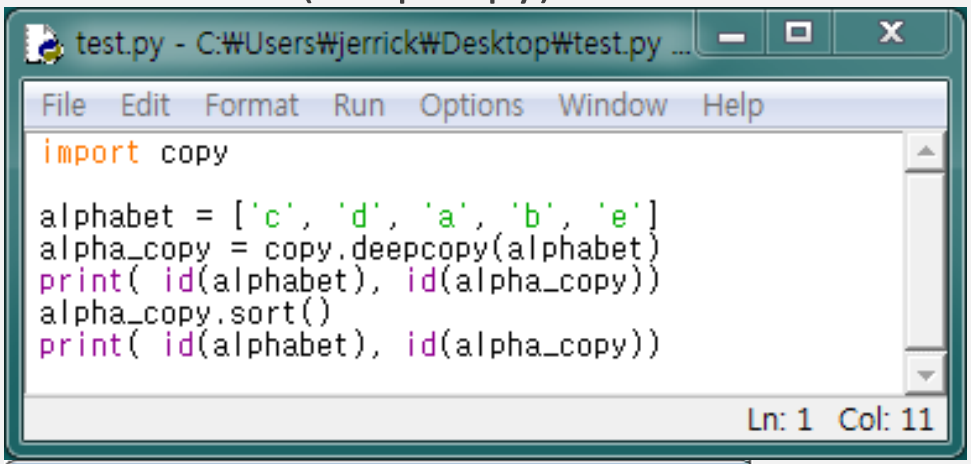


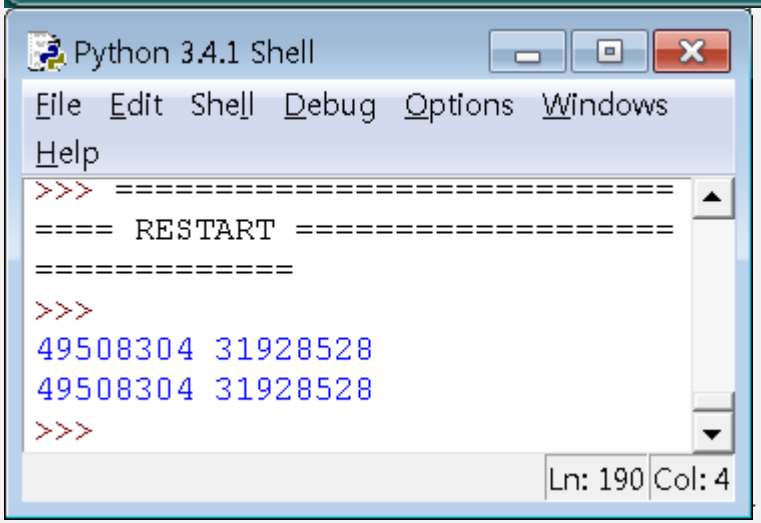


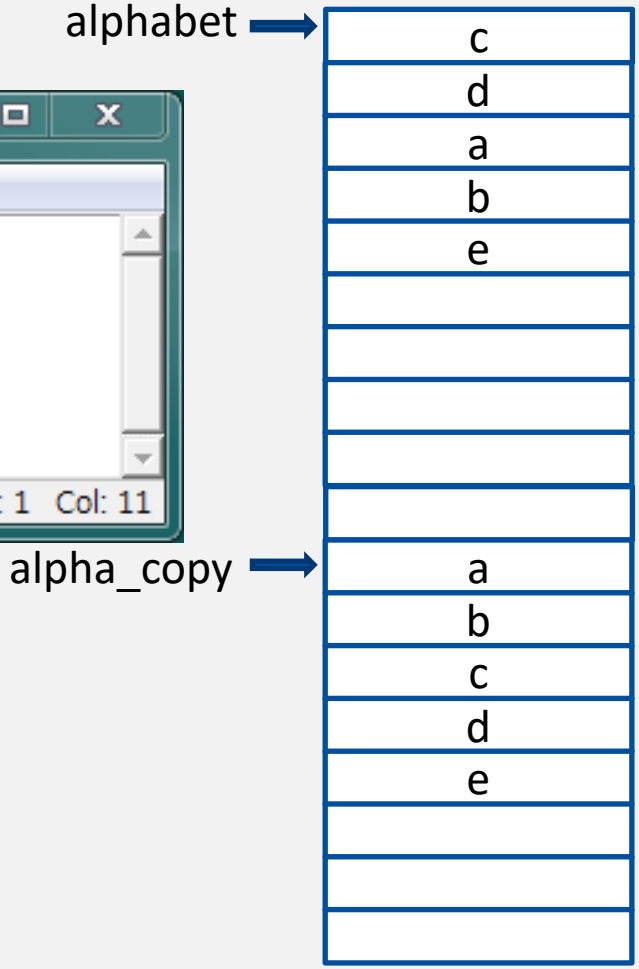


Shallow and deep copy

- 파이썬은 객체 지향 언어
  - 깊은 복사 (deep copy)







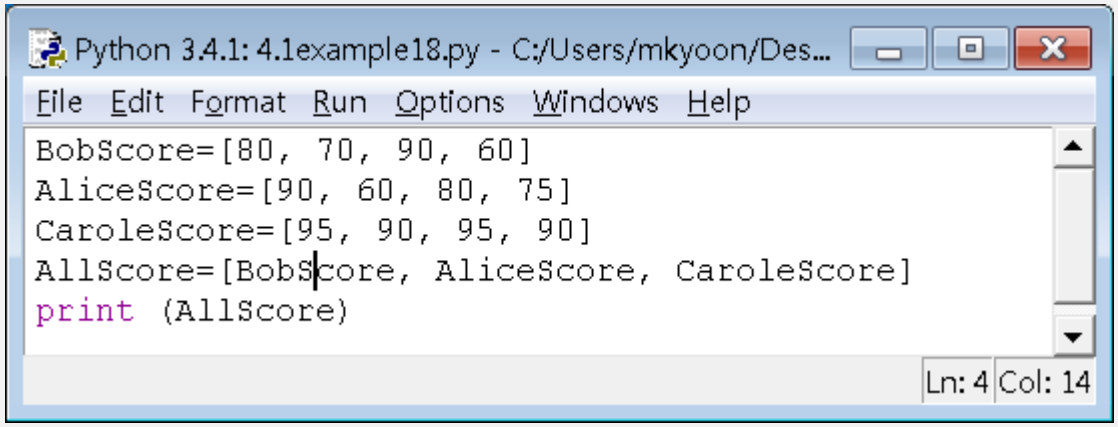
Lists of lists

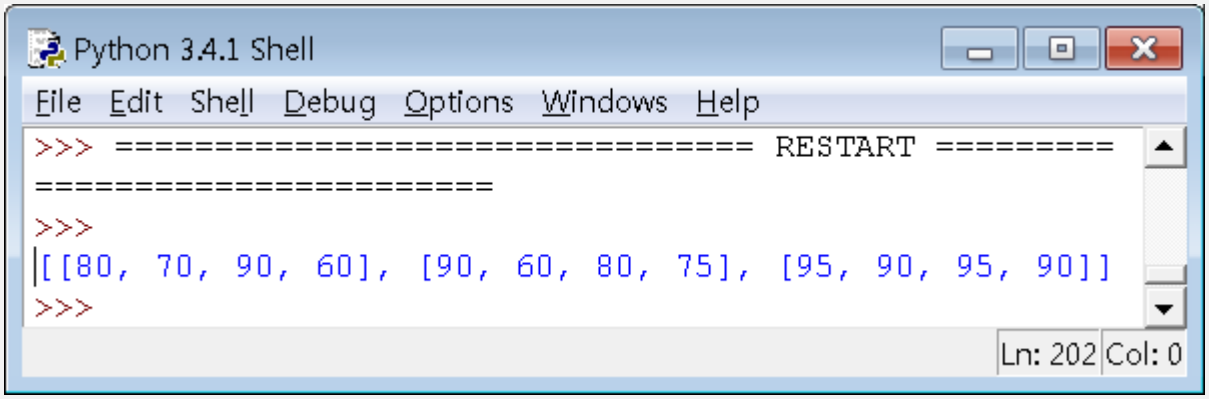
- 리스트로 구성된 리스트
  - 다차원 자료구조
  - 행렬

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

Lists of lists

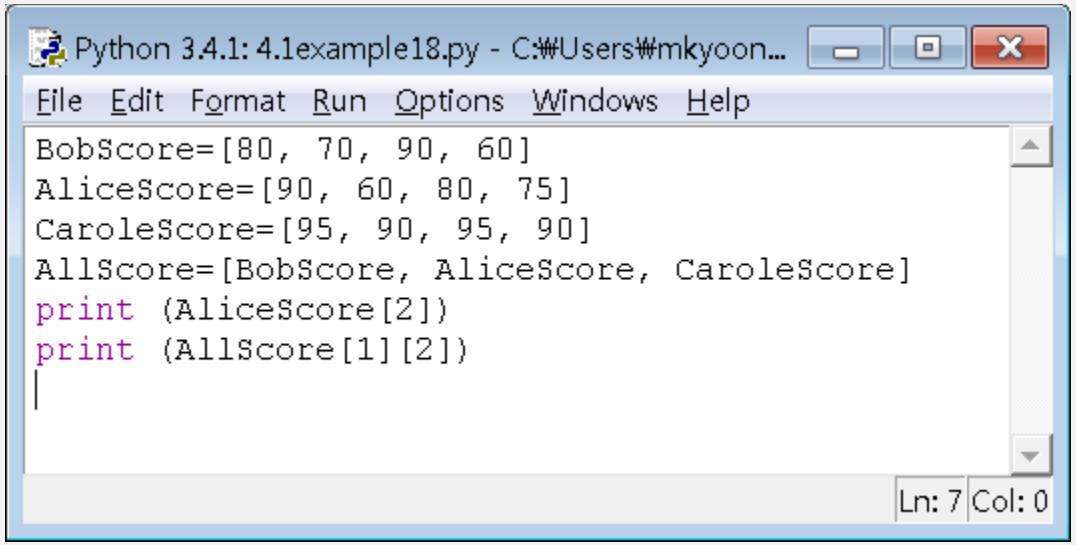
- 리스트로 구성된 리스트
  - 다차원 자료구조
  - 행렬

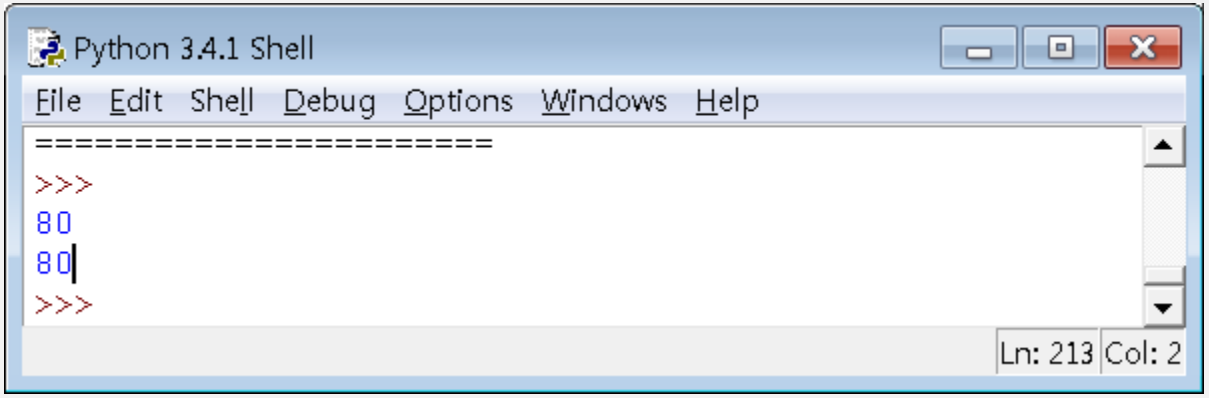




Lists of lists

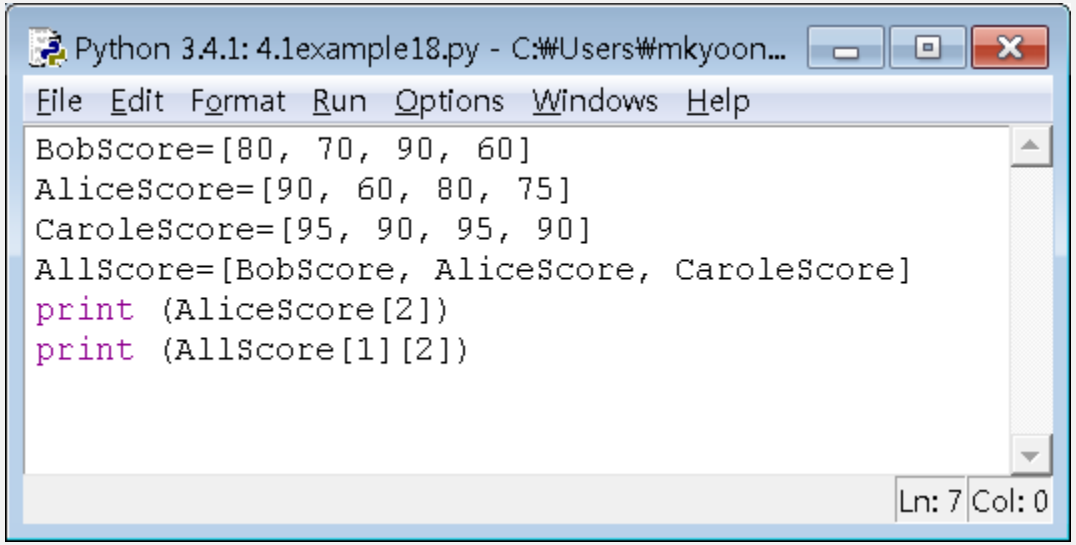
- 리스트로 구성된 리스트
  - 특정 값 하나에 접근하기: 내부 리스트의 인덱스 사용





Lists of lists

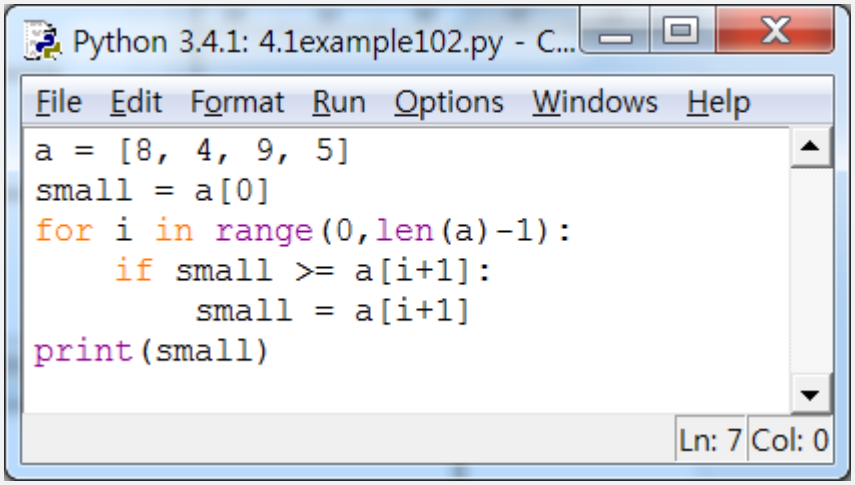
- 리스트로 구성된 리스트
  - 인덱싱



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]
  - small과 a[1]비교 → a[1]이 더 작으면 small = a[1]
  - small과 a[2]비교 → a[2]이 더 작으면 small = a[2]
  - small과 a[3]비교 → a[1]이 더 작으면 small = a[3]

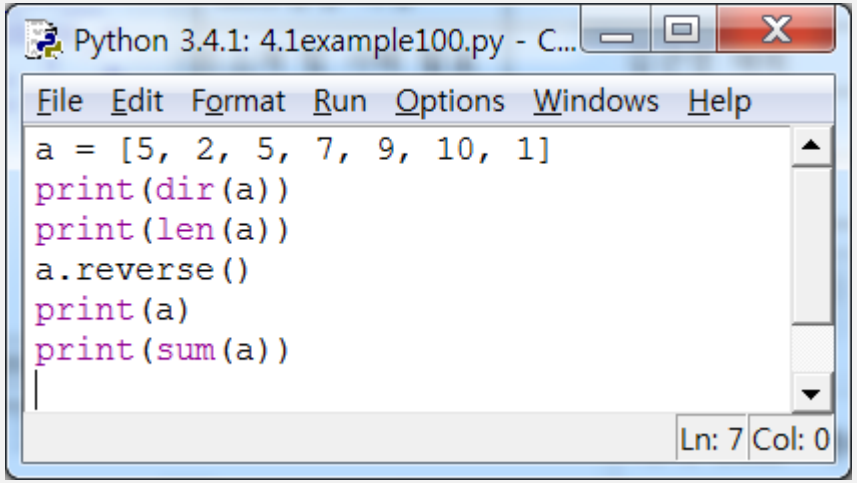


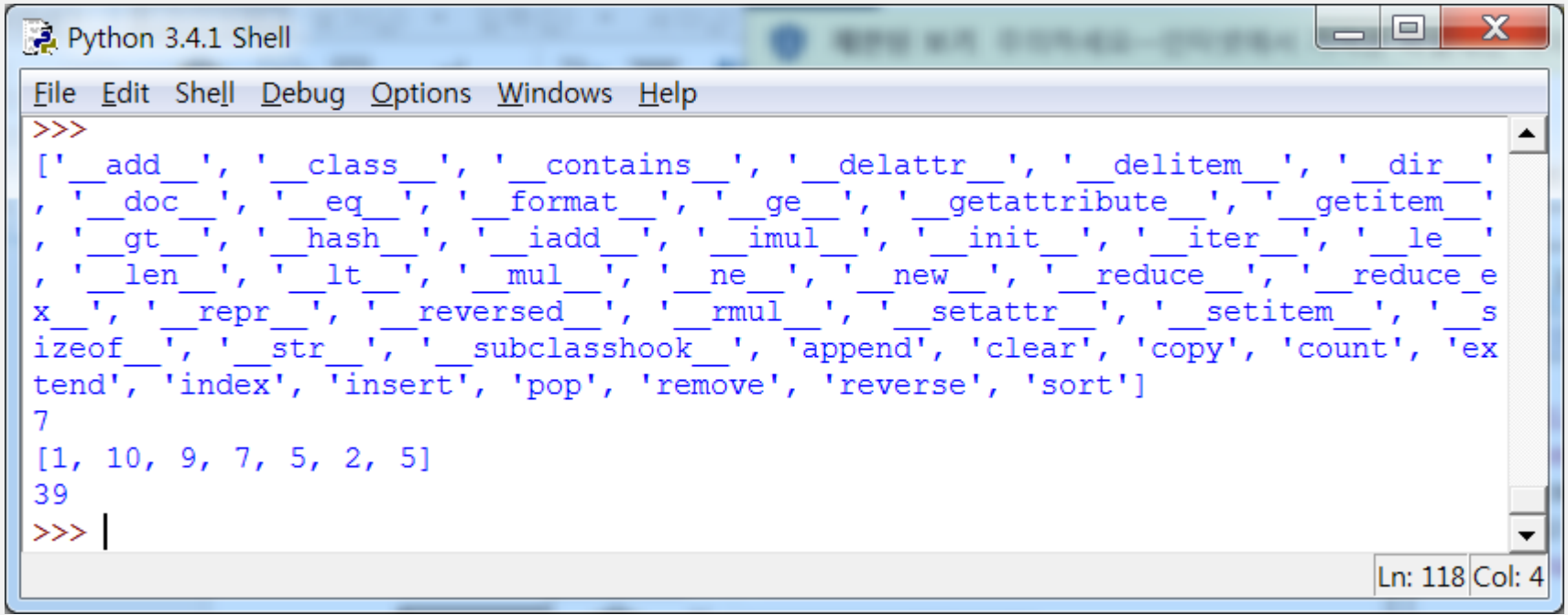
실습

- 정수로 구성된 리스트에서 가장 큰 수를 구하시오

실습

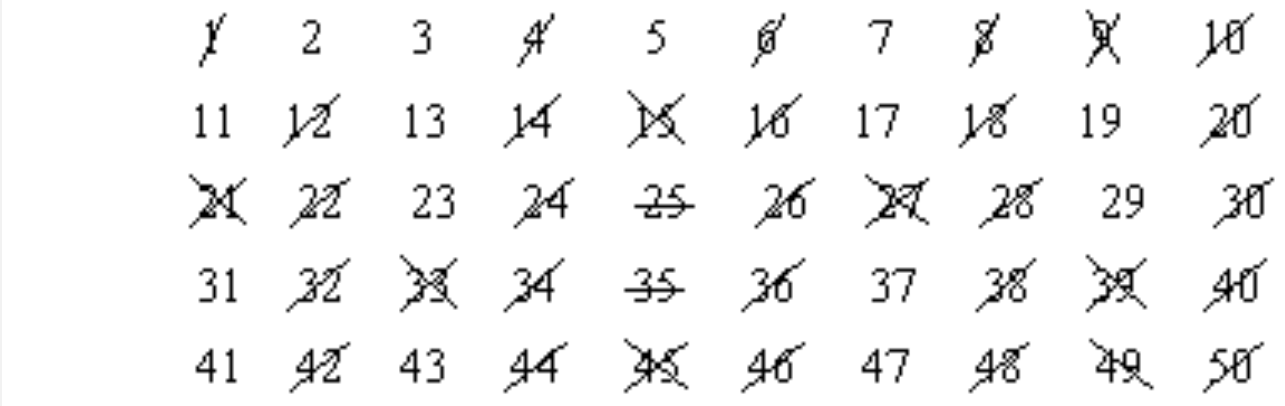
- LIST
  - LIST 제공 함수 사용해 보기
    - sum(리스트)
    - 리스트.reverse()
    - min(리스트)
    - max(리스트)





실습

- 에라토스테네스의 체
  - n이 주어졌을 때, n까지의 모든 소수를 출력하시오. 리스트를 사용하시오.
    - N까지 모든 소수를 찾는 방법 → 에라토스테네스의 체
      - sqrt(n) 이하의 소수를 찾는다
        - » sqrt(n) 사용 법
          - import math
          - math.sqrt(n)
      - 찾아진 소수들의 배수를 제거한다



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
  - n개의 정수로 구성된 리스트 a가 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 → n-1번 비교 발생
      - [0..n-2] 중 가장 큰 수를 찾아 a[n-2]로 이동시킴 → n-2번 비교 발생
      - …
      - [0..1] 중 가장 큰 수를 찾아 a[1]로 이동시킴 → 1번 비교 발생
    - (n-1) + (n-2) + … +1 번의 비교가 발생→(n-1)*n/2번의 비교 발생

숙제

- 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 0.04500198364257812