

Structured Types 1

By 윤명근 / 박수현

수업목표

- What is a **list**
- 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 list

- List는 아이템(item)들을 모은 **순차 데이터 타입 (Sequential Data Type)**
- 아이템은 integer, string, 다른 list 등 **모든 종류의 데이터 가능**

```
family = ['엄마', '아빠', '형', '동생']
```

```
numbers = [1, 3, 5, 7]
```

```
my_list = [5, 10, 21, 23, 'Hello', x, numbers, family]
```

Creating a list

- “[] ” 사용
 - myList = []

```
# list를 선언합니다.  
subjects = [ " 컴퓨터 네트워크", "캡스톤 디자인", " 소프트웨어적사고 " ]  
  
# 출력하기  
print(subjects)
```

```
[ " 컴퓨터 네트워크", "캡스톤 디자인", " 소프트웨어적사고 " , "Smart IoT " ]
```

Creating a list

4.1example1.py

File Edit Format Run Options Window Help

```
1 family = ['엄마', '아빠', '형', '동생']
2 numbers = [1, 3, 5, 7]
3 my_list = [5, 10, 21, 23, 'Hello', numbers, family]
4
5
6 print ("Wn>> family : String :", family)
7 print (">> numbers : Integer :", numbers)
8 print (">> my_list : Mixings, list in list :", my_list)
9
```

```
>> family : String : ['엄마', '아빠', '형', '동생']
>> numbers : Integer : [1, 3, 5, 7]
>> my_list : Mixings, list in list : [5, 10, 21, 23, 'Hello', [1, 3, 5, 7], ['엄마', '아빠', '형', '동생']]
>>> |
```

Adding things to a list

- append()
 - 사용법: list이름.append(item)
 - list는 객체(object)
 - list 객체는 append()라는 함수(function, method)를 이용해서 item을 추가
 - 객체, 함수 → 향후 설명

```
4.1example2-0.py
File Edit Format Run Options Window Help
myList = []

print ("1) Before adding elements, myList is empty : ", myList)

myList.append('Alice')
print ("2) Add 'Alice' to myList : ", myList)

myList.append(3.14)
print ("3) Add 3.14 to myList : ", myList)

myList.append(10)
print ("4) Add 10 to myList : ", myList)

myList.append([10, 20, "Park"])
print ("5) Add [10, 20, 'Park'] to myList : ", myList)
```

```
1) Before adding elements, myList is empty : []
2) Add "Alice" to myList : ['Alice']
3) Add 3.14 to myList : ['Alice', 3.14]
4) Add 10 to myList : ['Alice', 3.14, 10]
5) Add [10, 20, "Park"] to myList : ['Alice', 3.14, 10, [10, 20, 'Park']]
>>>
```

Adding things to a list

```
4.1example2.py
File Edit Format Run Options Window Help
1 myList = []
2
3 print ("1) element 추가전. myList =", myList)
4 print ("2) elements 추가")
5
6 more_input = "Y"
7
8 while more_input == "Y" or more_input == "y" or more_input == "":
9
10     in_element = input("3) myList에 추가할 값을 입력하세요 : ")
11     print("4) 입력값 : ", in_element)
12
13     if in_element.isnumeric() == True : # 주어진 문자열 in_element가 숫자인지 검사
14         in_element = int(in_element)    # 입력값이 숫자이면 정수로 type casting
15         print("5) {} is type-casted to integer ! ".format(in_element))
16
17     myList.append(in_element)
18     print("6) myList = ", myList)
19
20     more_input = input("7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 : ")
21     print("8) more_input = ", more_input)
22
23
24
25 print ("9) myList에 값 추가를 종료했습니다. \n")
26 print ("10) 최종 myList : ", myList)
27
28
```

```
1) element 추가전. myList = []
2) elements 추가
3) myList에 추가할 값을 입력하세요 : 10
4) 입력값 : 10
5) 10 is type-casted to integer !
6) myList = [10]
7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 :
8) more_input =
9) myList에 추가할 값을 입력하세요 : park
4) 입력값 : park
6) myList = [10, 'park']
7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 :
8) more_input =
9) myList에 추가할 값을 입력하세요 : [10, 20]
4) 입력값 : [10, 20]
6) myList = [10, 'park', '[10, 20]']
7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 :
8) more_input =
9) myList에 추가할 값을 입력하세요 : F9
4) 입력값 : F9
6) myList = [10, 'park', '[10, 20]', 'F9']
7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 : n
8) more_input = n
9) myList에 값 추가를 종료했습니다.
10) 최종 myList : [10, 'park', '[10, 20]', 'F9']
```

Hexadecimal ? 현재로서는 'F9'
라는 string data. 어떻게
Hexadecimal로 인식할 것인가?

- Input() 함수를
통해 입력된
Hexadecimal 인
식 방법

<https://www.geeksforgeeks.org/python-string-hexdigits/>

```
identify_hexa.py
File Edit Format Run Options Window Help
1 import string # importing string library function
2
3 #
4 # Function check(value):
5 #
6 # To the value passed from main (here, the "value" parameter)
7 # If digits other than hexadecimal are included
8 # then return False
9 # else return True
10 #
11 def check(value):
12     for letter in value:
13
14         if letter not in string.hexdigits:
15             return False
16
17     return True
18
19 # Main : driver code
20 input0 = "F9"
21 print(input0, "--> ", check(input0))
22
23 input0 = "FG"
24 print(input0, "--> ", check(input0))
25
26 input1 = "0123456789abcdef"
27 print(input1, "--> ", check(input1))
28
29 input2 = "abcdefABCDEF"
30 print(input2, "--> ", check(input2))
31
32 input3 = "abcdefghGEEK"
33 print(input3, "--> ", check(input3))
34
35 input4 = "aabbccddffg01"
36 print(input4, "--> ", check(input4))
37
```

```
F9 --> True
FG --> False
0123456789abcdef --> True
abcdefABCDEF --> True
abcdefghGEEK --> False
aabbccddffg01 --> False
>>>
```



```

1 myList = []
2
3 print ("1) element 추가전. myList =", myList)
4 print ("2) elements 추가")
5
6 more_input = "Y"
7
8 while more_input == "Y" or more_input == "y" or more_input == "":
9
10     input_value = input("\n3) myList에 추가할 값을 입력하세요 : ")
11     print("4) 입력값 : ", input_value)
12
13     if input_value.isnumeric() == True : # 주어진 문자열 input_value가 integer type numeric 인가 ?
14         input_value_integerType = int(input_value) # input_value(string type)을 integer로 type casting
15         print("5) {} is type-casted to integer ! ".format(input_value_integerType))
16         input_value = input_value_integerType
17
18     else : # 주어진 문자열 input_value가 numeric이 아닌 경우
19         # input_value가 floating point type인지 확인
20         replaced_input_value = input_value.replace('.', '', 1)
21         print("5-1) replaced_input_value =", replaced_input_value)
22
23         is_float = replaced_input_value.isdigit() # is_float가 True이면 input_value가 floating point type
24         print("5-2) is_float =", is_float)
25
26         if is_float == True : # 입력 input_value가 floating point type 인가 ?
27             float_or_str = input("\n6) 입력하신 값 {}을 floating point로 변경하겠습니까 ? : (Y or N) ".format(input_value))
28             if float_or_str == "Y" or more_input == "y" :
29                 input_value = float(input_value) # 주어진 문자열 입력값이 floating point type이면
30                 # float로 type casting
31                 print("`5-3) {} is type-casted to integer ! ".format(input_value))
32             else :
33                 print("\n6) 입력하신 값 {}을 String data type으로 입력합니다.".format(input_value))
34
35     print("7) 최종 입력값 =", input_value)
36
37     myList.append(input_value)
38     print("\n8) myList = ", myList)
39
40     more_input = input("\n7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 : ")
41     print("9) more_input = ", more_input)
42
43
44 print ("\n10) myList에 값 추가를 종료했습니다. \n")
45 print ("11) 최종 myList : ", myList)
46
47

```

```

1) element 추가전. myList = []
2) elements 추가

3) myList에 추가할 값을 입력하세요 : 10
4) 입력값 : 10
5) 10 is type-casted to integer !
7) 최종 입력값 = 10

8) myList = [10]

7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 : Y
9) more_input = Y

3) myList에 추가할 값을 입력하세요 : 10.0
4) 입력값 : 10.0
5-1) replaced_input_value = 100
5-2) is_float = True

6) 입력하신 값 10.0을 floating point로 변경하겠습니까 ? : (Y or N) Y
5-3) 10.0 is type-casted to integer !
7) 최종 입력값 = 10.0

8) myList = [10, 10.0]

7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 :
9) more_input =

3) myList에 추가할 값을 입력하세요 : 10.0
4) 입력값 : 10.0
5-1) replaced_input_value = 100
5-2) is_float = True

6) 입력하신 값 10.0을 floating point로 변경하겠습니까 ? : (Y or N) N

6) 입력하신 값 10.0을 String data type으로 입력합니다.
7) 최종 입력값 = 10.0

8) myList = [10, 10.0, '10.0']

7) 더 추가하고 싶으시면 Y 또는 y 또는 엔터 키를 입력. 아니면 N을 입력하세요 :

```

Adding things to a list

- `dir()`
 - 객체가 어떤 **함수**와 **속성**을 가지고 있는지 출력
- `help(객체명)`
 - 함수나 객체에 대한 정보 제공

데이터 (어트리뷰트) , 오퍼레이션 (메소드), 캡슐

```
1) list 자체의 속성과 함수 :
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
['__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
['__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',
['__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']]
```

```
2) user-defined list의 속성과 함수 :
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
['__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
['__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',
['__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']]
```

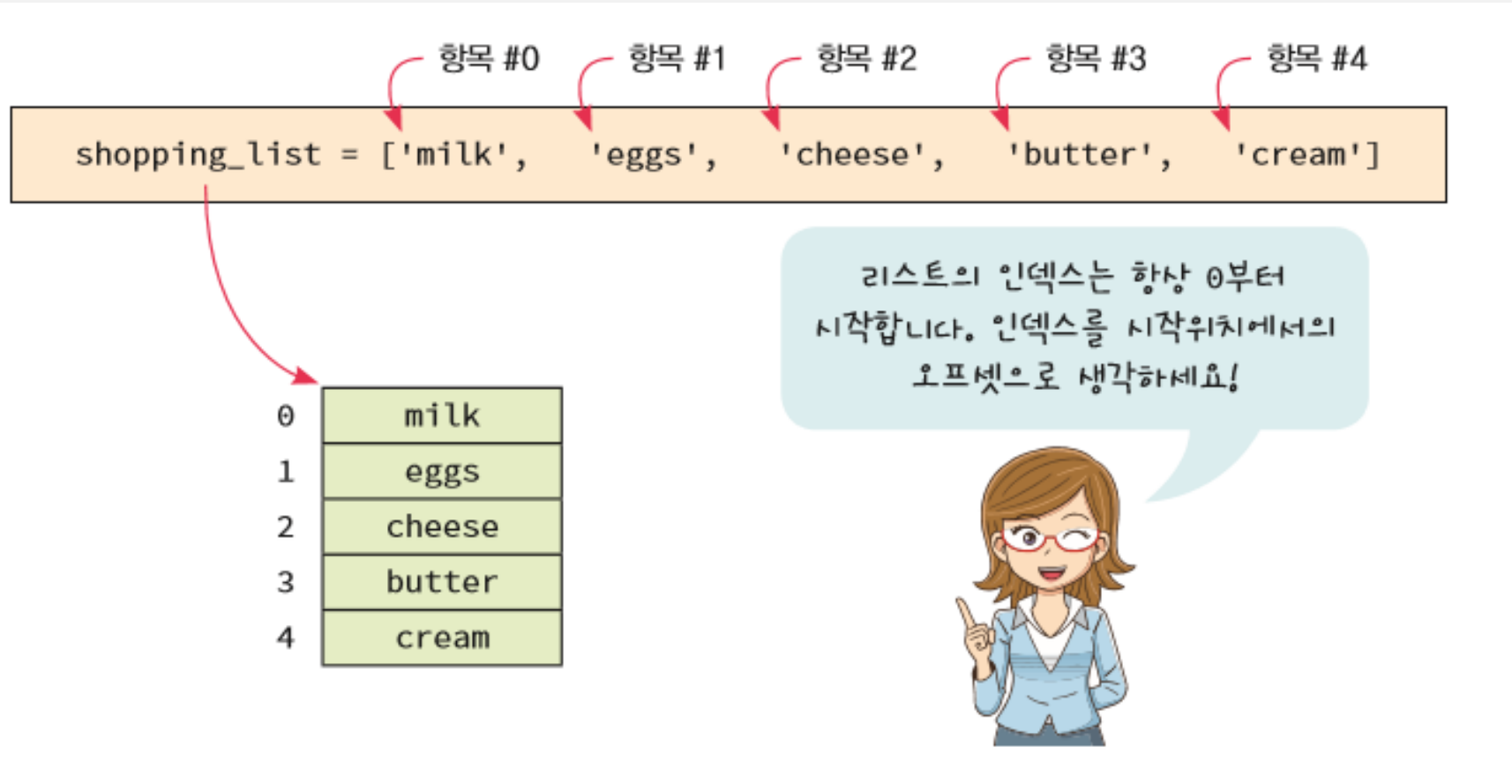
Help on built-in function dir in module builtins:

```
dir(...)
dir([object]) -> list of strings

If called without an argument, return the names in the current scope.
Else, return an alphabetized list of names comprising (some of) the attributes
of the given object, and of attributes reachable from it.
If the object supplies a method named __dir__, it will be used; otherwise
the default dir() logic is used and returns:
for a module object: the module's attributes.
for a class object: its attributes, and recursively the attributes
of its bases.
for any other object: its attributes, its class's attributes, and
recursively the attributes of its class's base classes.
```

Squeezed text (137 lines).

List indexing



Getting items from a list

- list에서 아이템 가져오기
 - 인덱스(index) 사용
 - 인덱스는 0부터 시작 → computer science 특징
 - 인덱스로 “-1” 을 사용함으로써 list의 마지막 값을 쉽게 찾을 수 있음

```

print("1) create myList")
myList = []

print("2) append Alice, Bob, Carole")
myList.append('Alice')
myList.append('Bob')
myList.append('Carole')

print("3) myList : ", myList)
print("\nmyList[0] =", myList[0])
print("myList[1] =", myList[1])
print("myList[2] =", myList[2])

print("\nmyList[-1] =", myList[-1])
print("myList[-2] =", myList[-2])
print("myList[-3] =", myList[-3], "\n")

numbers = [0, 10, 20, 30, 40, 50]

for j in range(len(numbers)):
    print("number[{}] = {}".format(j, numbers[j]))

print("\nnumber[-1] =", numbers[-1])
print("number[-2] =", numbers[-2])
print("number[-3] =", numbers[-3])
print("number[-4] =", numbers[-4])
print("number[-5] =", numbers[-5])
print("number[-6] =", numbers[-6])

# error
print("number[-7]=", numbers[-7])

```

```

1) create myList
2) append Alice, Bob, Carole
3) myList : ['Alice', 'Bob', 'Carole']

```

```

myList[0] = Alice
myList[1] = Bob
myList[2] = Carole

```

```

myList[-1] = Carole
myList[-2] = Bob
myList[-3] = Alice

```

```

number[0] = 0
number[1] = 10
number[2] = 20
number[3] = 30
number[4] = 40
number[5] = 50

```

```

number[-1] = 50
number[-2] = 40
number[-3] = 30
number[-4] = 20
number[-5] = 10
number[-6] = 0

```

Traceback (most recent call last):

File "C:\소사\강의예제\4.1example3.py", line 31, in <module>

print("number[-7]= ", numbers[-7])

IndexError: list index out of range

Slicing a list

- list slicing (자르기)

- “list이름[a:b]” : a번째부터 (b-1)번째까지 아이템으로 구성된 부분 **list** (잘라진 list)가 결과값으로 얻어짐

4.1example4.py

File Edit Format Run Options Window Help

```
myList = ['Aclie', 'Bob', 'Carole', 'SongAh']
print ("myList[0:2] =", myList[0:2])
print ("myList[2:3] =", myList[2:3])
print ("myList[2] =", myList[2])
```

```
numbers = [0,1,2,3,4,5,6]
print ("numbers[0:3] =", numbers[0:3])
print ("numbers[1:4] =", numbers[1:4])
print ("numbers[3:10] =", numbers[3:10])
print ("numbers[5] =", numbers[5])
```

```
float_num = [0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6]
print ("float_num[0:3] =", float_num[0:3])
print ("float_num[1:4] =", float_num[1:4])
print ("float_num[3:10] =", float_num[3:10])
print ("float_num[5] =", float_num[5])
```

#type 출력

```
print ("type(numbers[3:10]) =", type(numbers[3:10]))
print ("type(numbers[5]) =", type(numbers[5]))
```

```
print ("type(float_num[3:10]) =", type(float_num[3:10]))
print ("type(float_num[5]) =", type(float_num[5]))
```

```
print ("type(myList[2:3]) =", type(myList[2:3]))
print ("type(myList[2]) =", type(myList[2]))
```

```
myList[0:2] = ['Aclie', 'Bob']
```

```
myList[2:3] = ['Carole']
```

```
myList[2] = Carole
```

```
numbers[0:3] = [0, 1, 2]
```

```
numbers[1:4] = [1, 2, 3]
```

```
numbers[3:10] = [3, 4, 5, 6]
```

```
numbers[5] = 5
```

```
float_num[0:3] = [0.0, 1.1, 2.2]
```

```
float_num[1:4] = [1.1, 2.2, 3.3]
```

```
float_num[3:10] = [3.3, 4.4, 5.5, 6.6]
```

```
float_num[5] = 5.5
```

```
type(numbers[3:10]) = <class 'list'>
```

```
type(numbers[5]) = <class 'int'>
```

```
type(float_num[3:10]) = <class 'list'>
```

```
type(float_num[5]) = <class 'float'>
```

```
type(myList[2:3]) = <class 'list'>
```

```
type(myList[2]) = <class 'str'>
```

Slicing a list

- list 자르기
 - 잘라진 list type

4.1example5.py

File Edit Format Run Options Window Help

```
1 former = [3, "park", "p"]
2
3 list_data = [0, 1, 2, former]
4
5 print("\n1) former[0] = ", former[0])
6 print("\n2) type(former[0]) = ", type(former[0]))
7
8 print("\n3) former[1] = ", former[1])
9 print("\n4) type(former[1]) = ", type(former[1]))
10
11 # str
12 print("\n5) former[2] = ", former[2])
13 print("\n6) type(former[2]) = ", type(former[2]))
14
15 print("\n7) list_data[1] = ", list_data[1])
16 print("\n8) type(list_data[1]) = ", type(list_data[1]))
17
18 print("\n9) list_data[3] = ", list_data[3])
19 print("\n10) type(list_data[3]) = ", type(list_data[3]))
20
21 print("\n11) list_data[1:4] = ", list_data[1:4])
22 print("\n12) type(list_data[1:4]) = ", type(list_data[1:4]))
23
```

```
1) former[0] = 3
2) type(former[0]) = <class 'int'>
```

```
3) former[1] = park
4) type(former[1]) = <class 'str'>
```

```
5) former[2] = p
6) type(former[2]) = <class 'str'>
```

```
7) list_data[1] = 1
8) type(list_data[1]) = <class 'int'>
```

```
9) list_data[3] = [3, 'park', 'p']
10) type(list_data[3]) = <class 'list'>
```

```
11) list_data[1:4] = [1, 2, [3, 'park', 'p']]
12) type(list_data[1:4]) = <class 'list'>
```

Slicing a list

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

결합

```
concatenate_list.py
File Edit Format Run Options Window Help
1 graduate = [ "Choi", "Yeom", "Hwang", "Jeong", "Jin", "Shurutika" ]
2 undergraduate = [ "Song" ]
3 researcher = [ "Lim", "Hyun", "Ji", "Pradeep" ]
4
5
6 all_lab_members = graduate + undergraduate + researcher
7
8 print("\n1) graduate =", graduate)
9 print("\n2) undergraduate =", undergraduate)
10 print("\n3) researcher =", researcher)
11
12 print("\n4) graduate + undergraduate + researcher =", all_lab_members)
13 print("\n5) graduate * 2 =", graduate * 2)
14
15 print("\n6) graduate 인원수 =", len(graduate))
16 print("\n7) undergraduate 인원수 =", len(undergraduate))
17 print("\n8) researcher 인원수 =", len(researcher))
18
19 new_list = [ "Park", "Kim", "Lee", researcher, graduate, undergraduate ]
20
21 print("\n9) new_list =", new_list)
22 print("\n10) item 수 =", len(new_list))
23
```

- list 연산자

- + : 두 list를 합친다.
- * : list를 숫자만큼 반복 한다.
- len(list) : list의 길이를 반환 한다.

```
1) graduate = ['Choi', 'Yeom', 'Hwang', 'Jeong', 'Jin', 'Shurutika']
2) undergraduate = ['Song']
3) researcher = ['Lim', 'Hyun', 'Ji', 'Pradeep']
4) graduate + undergraduate + researcher = ['Choi', 'Yeom', 'Hwang', 'Jeong', 'Jin', 'Shurutika', 'Song', 'Lim', 'Hyun', 'Ji', 'Pradeep']
5) graduate * 2 = ['Choi', 'Yeom', 'Hwang', 'Jeong', 'Jin', 'Shurutika', 'Choi', 'Yeom', 'Hwang', 'Jeong', 'Jin', 'Shurutika']
6) graduate 인원수 = 6
7) undergraduate 인원수 = 1
8) researcher 인원수 = 4
9) new_list = ['Park', 'Kim', 'Lee', ['Lim', 'Hyun', 'Ji', 'Pradeep'], ['Choi', 'Yeom', 'Hwang', 'Jeong', 'Jin', 'Shurutika'], ['Song']]
10) item 수 = 6
```

Modifying/Deleting/Searching items

- 수정하기 (Modify)
 - 인덱스를 사용해서 대상 아이템 변경

```
4.1example6.py
File Edit Format Run Options Window Help
numbers = [1,2,3,4,5]
print ("\n1) 수정 전 : numbers =", numbers)

numbers[2] = 200
print ("\n2) 수정 후 : numbers =", numbers)

alphabet = ['a', 'b', 'c', 'd']
print ("\n3) 수정 전 : alphabet =", alphabet)

alphabet[3] = 'z'
print ("\n4) 수정 후 : alphabet =", alphabet)

mixed = [100, 'Park', [1, 2, 3]]
print ("\n5) 수정 전 : mixed =", mixed)

mixed[0] = 200
mixed[2][0] = "Soo-Hyun"
print ("\n6) 수정 후 : mixed =", mixed)

mixed[2] = "Kookmin University"
print ("\n7) 수정 후 : mixed =", mixed)
```

- 1) 수정 전 : numbers = [1, 2, 3, 4, 5]
- 2) 수정 후 : numbers = [1, 2, 200, 4, 5]
- 3) 수정 전 : alphabet = ['a', 'b', 'c', 'd']
- 4) 수정 후 : alphabet = ['a', 'b', 'c', 'z']
- 5) 수정 전 : mixed = [100, 'Park', [1, 2, 3]]
- 6) 수정 후 : mixed = [200, 'Park', ['Soo-Hyun', 2, 3]]
- 7) 수정 후 : mixed = [200, 'Park', 'Kookmin University']

Modifying/Deleting/Searching items

- list에 아이터를 넣는 다양한 방법
 - append(): list 뒤에 하나의 아이터 추가
 - Insert(): list 내 원하는 위치에 아이터 추가
 - extend(): 여러 아이터를 list 뒤에 추가

4.1example7.py

File Edit Format Run Options Window Help

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

```
print("2) append e")  
alphabet.append('e')  
print("3) alphabet after append('e') : ", alphabet)
```

```
print("4) insert (1,'z')")  
alphabet.insert(1, 'z') ✓ 1은 여기서 파라미터  
print("5) alphabet after insert (1,'z') : ", alphabet)
```

```
print("6) extend(['f', 'g', 'h'])")  
alphabet.extend(['f', 'g', 'h'])  
print("7) alphabet after extend : ", alphabet)
```

```
1) alphabet : ['a', 'b', 'c', 'd']  
2) append e  
3) alphabet after append("e") : ['a', 'b', 'c', 'd', 'e']  
4) insert (1,"z")  
5) alphabet after insert (1,"z") : ['a', 'z', 'b', 'c', 'd', 'e']  
6) extend(['f', 'g', 'h'])  
7) alphabet after extend : ['a', 'z', 'b', 'c', 'd', 'e', 'f', 'g', 'h']  
~~~
```

Modifying/Deleting/Searching items

- list에서 아이템 삭제하기

- remove()

- <list>.remove(<값>) : list의 **지정한 아이템(값)** 제거
 - 해당 값이 복수 개인 경우 제일 먼저 발견되는 값 제거
 - 동일한 복수 개의 값을 삭제 시 while 등 반복문을 사용하여 제거
 - 예)

```
user_1 = ['Jason' , 'Smith', 'Kevin', 'Smith', 'Smith', 'Smith']
```

```
while 'Smith' in user_1:
```

```
    user_1.remove('Smith') # 'Smith' 삭제
```

- del

- del <list>[<인덱스>] : **인덱스에 해당하는 아이템** 삭제
 - 예)

```
x = [1, 2, 3, 4, 5, 6, 7]
```

```
del x[1] # 결과 x = [1, 3, 4, 5, 6, 7]
```

```
del x[3:] # 결과 x = [1, 3, 4]
```

- pop()

- list의 **마지막 항목 리턴**(return). 값은 **삭제** 됨

- <list>.clear()

- list 내부의 **모든 요소를 제거**

```

1 alphabet = ['a', 'b', 'c', 'd', 'e']
2 print("\n1) alphabet : ", alphabet)
3
4 print("\n2) remove c")
5 alphabet.remove('c')
6 print("\n3) alphabet after remove('c') : ", alphabet)
7
8 print("\n4) del alphabet[1]")
9 del alphabet[1]
10 print("\n5) alphabet after del : ", alphabet)
11
12 print("\n6) pop()한 값을 x 에 저장")
13 x = alphabet.pop()
14
15 print("\n7) alphabet after pop : ", alphabet)
16 print("\n8) x = ", x)
17
18 print("\n9) alphabet.clear()")
19 alphabet.clear()
20 print("\n10) alphabet after clear() : ", alphabet)
21
22 user_1 = ['Jason', 'Smith', 'Kevin', 'Smith', 'Smith', 'Smith']
23 print("\n11) user_1 : ", user_1)
24
25 while 'Smith' in user_1:
26     user_1.remove('Smith') # 'Smith' 삭제
27     print("\n12) user_1 : ", user_1)
28
29 x = [1, 2, 3, 4, 5, 6, 7]
30 print("\n13) x : ", x)
31
32 del x[1] # 결과 x = [1, 3, 4, 5, 6, 7]
33 print("\n14) x : ", x)
34
35 del x[3:] # 결과 x = [1, 3, 4]
36 print("\n15) x : ", x)
37

```

1) alphabet : ['a', 'b', 'c', 'd', 'e']

2) remove c

3) alphabet after remove("c") : ['a', 'b', 'd', 'e']

4) del alphabet[1]

5) alphabet after del : ['a', 'd', 'e']

6) pop()한 값을 x 에 저장

7) alphabet after pop : ['a', 'd']

8) x = e

9) alphabet.clear()

10) alphabet after clear() : []

11) user_1 : ['Jason', 'Smith', 'Kevin', 'Smith', 'Smith', 'Smith']

12) user_1 : ['Jason', 'Kevin', 'Smith', 'Smith', 'Smith']

12) user_1 : ['Jason', 'Kevin', 'Smith', 'Smith']

12) user_1 : ['Jason', 'Kevin', 'Smith']

12) user_1 : ['Jason', 'Kevin']

13) x : [1, 2, 3, 4, 5, 6, 7]

14) x : [1, 3, 4, 5, 6, 7]

15) x : [1, 3, 4]

Modifying/Deleting/Searching items

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

4.1example9.py

File Edit Format Run Options Window Help

```
alphabet = ['a', 'b', 'c', 'd', 'e']
```

```
print("\n1) list search, b ?")  
print("alphabet =", alphabet)  
print('b' in alphabet)
```


```
print("\n2) list search, a ?")  
if 'a' in alphabet:  
    print ("There is 'a'")  
else:  
    print ("There is not 'a'")
```

```
1) list search, b ?  
alphabet = ['a', 'b', 'c', 'd', 'e']  
True
```

```
2) list search, a ?  
There is 'a'
```

Modifying/Deleting/Searching items

- list에서 찾기 (search)
 - `index()` 함수: item의 위치 (index 값) return

 4.1example10.py

File Edit Format Run Options Window Help

```
alphabet = ['a', 'b', 'c', 'd', 'e']
print("\n1) alphabet =", alphabet)

if 'c' in alphabet:
    print ("\nThe index of 'c' is", alphabet.index('c'), "\n")
```

```
1) alphabet = ['a', 'b', 'c', 'd', 'e']
```

```
The index of 'c' is 2
```

Modifying/Deleting/Searching items

- list에서 찾기 (search)
 - `index()` 함수: item의 위치 (index 값) return

```
list_search.py
File Edit Format Run Options Window Help

The_Beatles = [ "John Lennon", "Paul McCartney", "George Harrison", "Ringo Starr" ]
name = input("\n>> The Beatles의 member 이름을 입력하세요 : ")

if name in The_Beatles :
    idx = The_Beatles.index(name)
    print("idx = ", idx)
    print("\n>>> {}는 The Beatles의 {}번째 멤버입니다.".format(name, idx+1))
else :
    print("\n>>> {}는 The Beatles의 멤버가 아닙니다.".format(name))

>> The Beatles의 member 이름을 입력하세요 : Paul McCartney
idx = 1

>>> Paul McCartney는 The Beatles의 2번째 멤버입니다.
>>>
```


Looping through a list

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

```
3.2example3-1.py -
File Edit Format Run Options Window Help
1 for i in range(10):
2     print ("It's number ", i)
3
4 print(">> done")
5
6 print(list(range(10)))
7
```

```
It's number 0
It's number 1
It's number 2
It's number 3
It's number 4
It's number 5
It's number 6
It's number 7
It's number 8
It's number 9
>> done
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Loop in a list

- list 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

- list 안에 loop 문 작성 가능
 - If 조건 추가 가능
 - [표현식 for 항목 in 반복가능객체 if 조건]

```
3.loop_if_in_a_list.py - C:\소사\강의예제\3.loop_if_in_a_list.py (3.11.5)
File Edit Format Run Options Window Help
1 a = [1,2,3,4]
2
3 result = []
4 for num in a:
5     result.append(num*3)
6 print("1) result 1 :", result)
7
8 result = []
9 result = [num * 3 for num in a]
10 print("2) result 2 :", result)
11
12 result = []
13 result = [num * 3 for num in a if num % 2 == 0]
14 print("3) result 3 :", result)
15
```

```
1) result 1 : [3, 6, 9, 12]
2) result 2 : [3, 6, 9, 12]
3) result 3 : [6, 12]
```

Loop in a list

- 실습 (숙제) – 화일명 : list_comprehension_이름_학번_일시.py
 - 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)
 - 내림차순 (descending) 정렬 vs 오름차순 (ascending) 정렬
 - 컴퓨터에게 “크다 작다”를 정해주어야 함 !
 - 디폴트(default): 숫자 크기 순서(오름차순), 사전 등장 순서
 - `sort()` 함수 사용
 - 정렬 순서를 반대로 하려면, `sort(reverse=True)` 사용

```

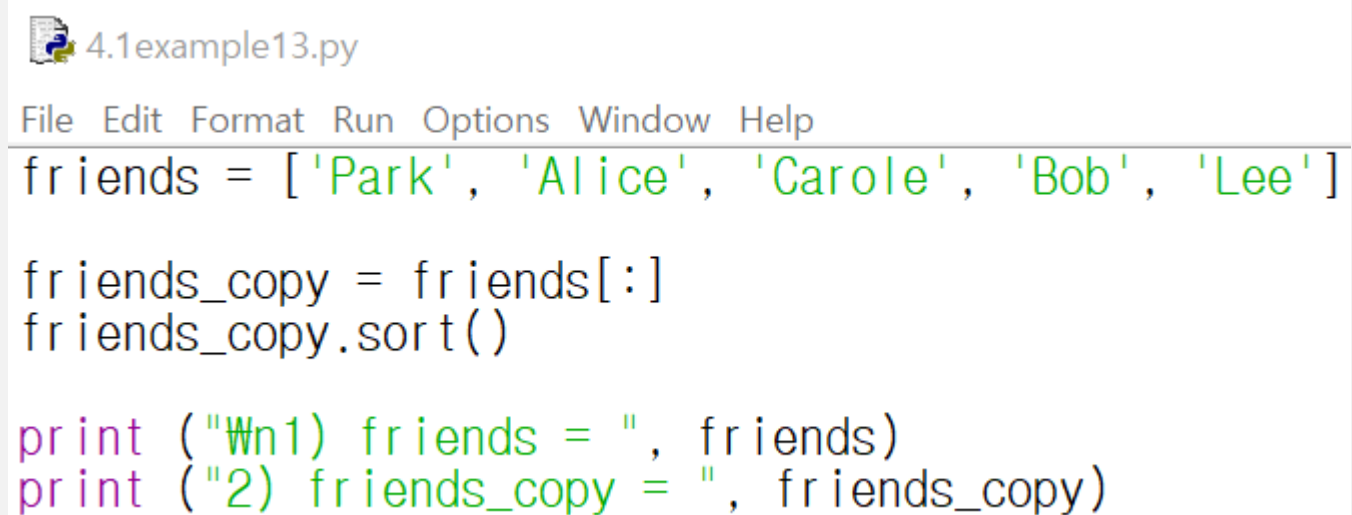
1 # Alphabet / String data type sort
2 alphabet = ['c', 'd', 'a', 'b', 'f', 'e', '0', '3']
3 print("\n1) Before sort : ascending – alphabet : ", alphabet)
4
5 alphabet.sort()
6 print("\n2) After sort : ascending – alphabet : ", alphabet)
7
8 print("\n3) Before reverse sort : descending – alphabet : ", alphabet)
9 alphabet.sort(reverse=True)
10 print("\n4) After reverse sort: descending – alphabet : ", alphabet)
11
12 friends = ['Bob', 'Alice', 'Carole', 'Ann']
13 print("\n5) Before sort – friends : ", friends)
14
15 friends.sort()
16 print("\n6) After sort – friends : ", friends)
17
18 print("\n7) Before reverse sort – friends : ", friends)
19 friends.sort(reverse=True)
20 print("\n8) After reverse sort – friends : ", friends)
21
22
23 # Number data type sort
24 number = [ 50, 30, 10, 20, 40, 50 ]
25 idx1 = number.index(50)
26 idx2 = number.index(50,2)
27 print("idx1 = {}, idx2 = {}".format(idx1, idx2))
28
29 print("\n9) Before sort – number : ", number)
30 number.sort()
31 print("\n10) After sort – number : ", number)
32
33 print("\n11) Before reverse sort – number : ", number)
34 number.sort(reverse=True)
35 print("\n12) After reverse sort – number : ", number)
36

```

1) Before sort : ascending – alphabet : ['c', 'd', 'a', 'b', 'f', 'e', '0', '3']
2) After sort : ascending – alphabet : ['0', '3', 'a', 'b', 'c', 'd', 'e', 'f']
3) Before reverse sort : descending – alphabet : ['0', '3', 'a', 'b', 'c', 'd', 'e', 'f']
4) After reverse sort: descending – alphabet : ['f', 'e', 'd', 'c', 'b', 'a', '3', '0']
5) Before sort – friends : ['Bob', 'Alice', 'Carole', 'Ann']
6) After sort – friends : ['Alice', 'Ann', 'Bob', 'Carole']
7) Before reverse sort – friends : ['Alice', 'Ann', 'Bob', 'Carole']
8) After reverse sort – friends : ['Carole', 'Bob', 'Ann', 'Alice']
idx1 = 0, idx2 = 5
9) Before sort – number : [50, 30, 10, 20, 40, 50]
10) After sort – number : [10, 20, 30, 40, 50, 50]
11) Before reverse sort – number : [10, 20, 30, 40, 50, 50]
12) After reverse sort – number : [50, 50, 40, 30, 20, 10]
>>> .

Sorting a list

- 정렬 (sorting)
 - sort() 적용하면 list **아이템의 순서가 변경**됨에 주의해야 함
 - 원본 유지를 하고 싶으면 **복사본**을 사용해서 정렬해야 함



```
4.1example13.py
File Edit Format Run Options Window Help
friends = ['Park', 'Alice', 'Carole', 'Bob', 'Lee']


friends_copy = friends[:]
friends_copy.sort()

print ("1) friends = ", friends)
print ("2) friends_copy = ", friends_copy)
```

```
1) friends = ['Park', 'Alice', 'Carole', 'Bob', 'Lee']
2) friends_copy = ['Alice', 'Bob', 'Carole', 'Lee', 'Park']
```

Sorting a list

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

 4.1example17.py

File Edit Format Run Options Window Help

```
friends = ['Park', 'Alice', 'Carole', 'Bob', 'Lee']
```

```
friends_copy = sorted(friends)
```

```
#friends_copy.sort()
```

```
print ("\n1) friends = ", friends)
```

```
print ("2) friends_copy = ", friends_copy)
```

```
1) friends = ['Park', 'Alice', 'Carole', 'Bob', 'Lee']
```

```
2) friends_copy = ['Alice', 'Bob', 'Carole', 'Lee', 'Park']
```

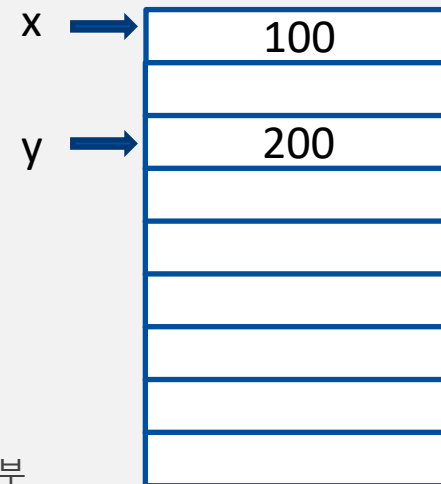
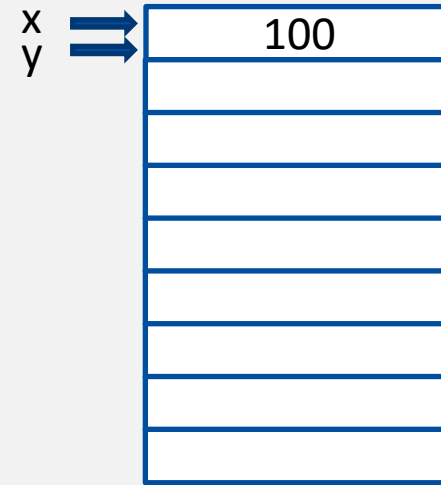
```
^^^
```


Shallow and deep copy

- Python은 객체 지향(기반) 언어
(Object-oriented(based) language)
 - 객체 id → 확인은 `id()` 함수 이용

```
4.1example14.py
File Edit Format Run Options Window Help
1 x=100
2 y=100
3 print("1)", id(x), id(y))
4
5 y=200
6 print("2)", id(x), id(y))
7
```

```
=====
1) 140718615031688 140718615031688
2) 140718615031688 140718615034888
```



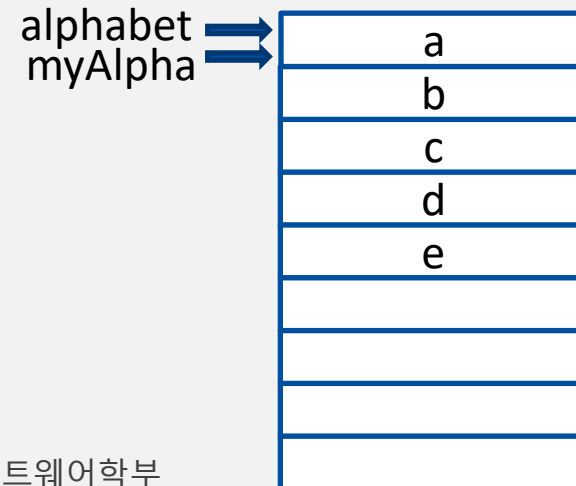
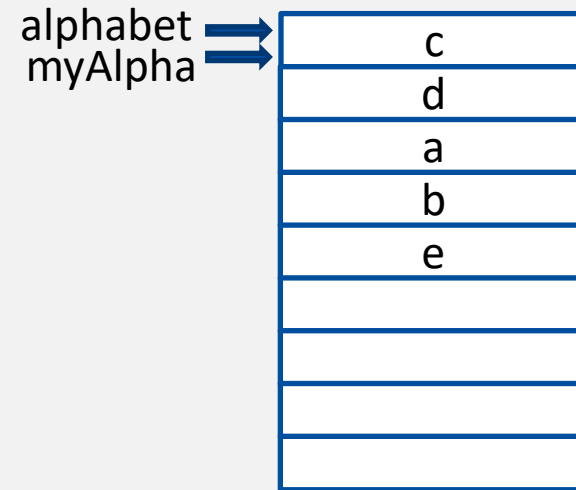
Shallow and deep copy

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

```
4.1example15.py
File Edit Format Run Options Window Help
alphabet = ['c', 'd', 'a', 'b', 'e']
myAlpha = alphabet
print("1)", id(alphabet), id(myAlpha))

alphabet.sort()
print("2)", id(alphabet), id(myAlpha))
```

```
1) 2200875747648 2200875747648
2) 2200875747648 2200875747648
```



Shallow and deep copy

- Python은 객체 지향 언어
 - 깊은 복사 (deep copy)

4.1example15-1.py

File Edit Format Run Options Window Help

```
import copy
```

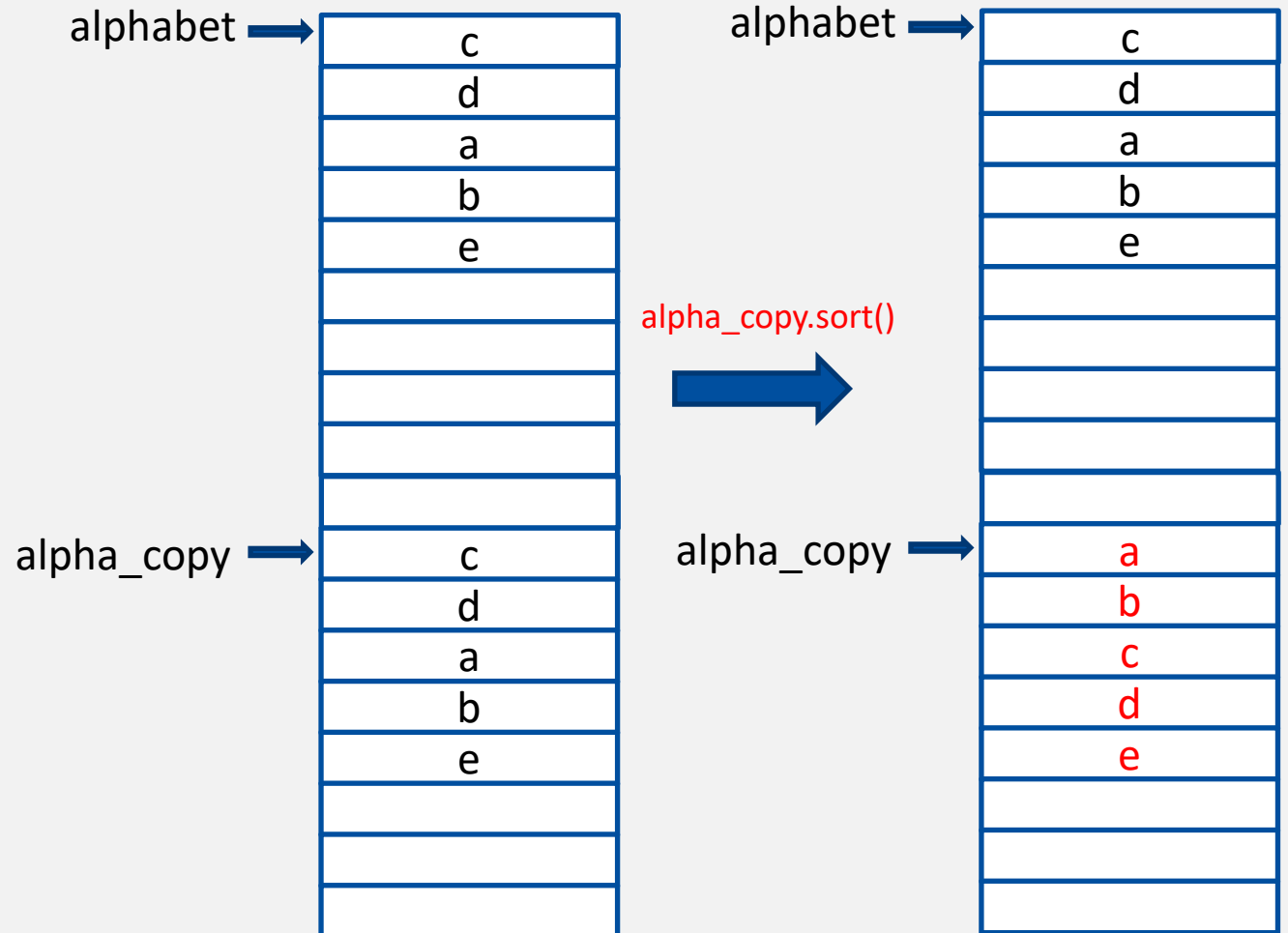
```
alphabet = ['c', 'd', 'a', 'b', 'e']
```

```
alpha_copy = copy.deepcopy(alphabet)
print("1)", id(alphabet), id(alpha_copy))
```

```
alpha_copy.sort()
print("2) alphabet =", alphabet)
print("3) alpha_copy =", alpha_copy)
```

```
print("4)", id(alphabet), id(alpha_copy))
```

```
1) 1711874331200 1711874278848
2) alphabet = ['c', 'd', 'a', 'b', 'e']
3) alpha_copy = ['a', 'b', 'c', 'd', 'e']
4) 1711874331200 1711874278848
```



Lists of lists

- list로 구성된 list
 - 다차원 자료구조
 - 행렬 (行列, matrix)

Column (열, 列) ↓

Row(행, 行) →

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

Lists of lists

- list로 구성된 list
 - 다차원 자료구조
 - 행렬



4.1example19.py

File Edit Format Run Options Window Help

```
BobScore=[80, 70, 90, 60]
```

```
AliceScore=[90, 60, 80, 75]
```

```
CaroleScore=[95, 90, 95, 90]
```

```
AllScore=[BobScore, AliceScore, CaroleScore]
```

```
print ("AllScore = ", AllScore)
```

```
AllScore = [[80, 70, 90, 60], [90, 60, 80, 75], [95, 90, 95, 90]]
```

Lists of lists

- list로 구성된 list
 - Indexing

 4.1example19-1.py

File Edit Format Run Options Window Help

```
BobScore=[80, 70, 90, 60]
AliceScore=[90, 60, 80, 75]
CaroleScore=[95, 90, 95, 90]

AllScore=[BobScore, AliceScore, CaroleScore]

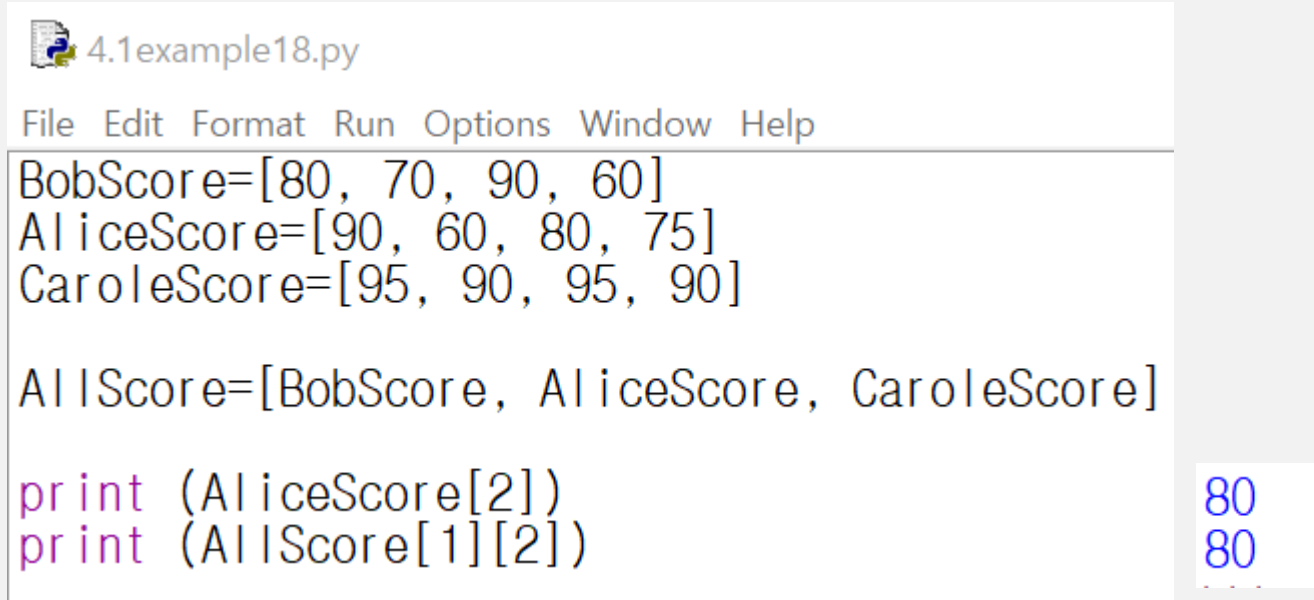
print ("AllScore = ", AllScore)
print ("AllScore[2] = ", AllScore[2])
print ("AllScore[1][2] = ", AllScore[1][2])
```

```
AllScore = [[80, 70, 90, 60], [90, 60, 80, 75], [95, 90, 95, 90]]
AllScore[2] = [95, 90, 95, 90]
AllScore[1][2] = 80
```

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]

Lists of lists

- list로 구성된 list
 - 특정 값 하나에 접근하기: 내부 list의 인덱스 사용



```
4.1example18.py
File Edit Format Run Options Window 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])
```

80
80

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 list에서 가장 작은 수를 구하시오
 - $a = [11, 9, 8, 7, 9, 5, 3]$

실습

- 정수로 구성된 list에서 가장 작은 수를 구하시오

- `a = [11, 9, 8, 7, 9, 5, 3]`
- `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[1]`이 더 작으면 `small = a[3]`

...

4.1example102.py

File Edit Format Run Options Window Help

```
a = [11, 9, 8, 7, 9, 5, 3]
```

```
print("\n1) a = [11, 9, 8, 7, 9, 5, 3] 에서 가장 작은 수를 구하시오.\n")
```

```
small = a[0]
```

```
for i in range(0, len(a)-1):  
    if small >= a[i+1]:  
        small = a[i+1]
```

```
    print(">> step {} : small = {}".format(i+1, small))
```

```
print("\n2) a 에서 가장 작은 수 :", small)
```

1) a = [11, 9, 8, 7, 9, 5, 3] 에서 가장 작은 수를 구하시오.

```
>> step 1 : small = 9  
>> step 2 : small = 8  
>> step 3 : small = 7  
>> step 4 : small = 7  
>> step 5 : small = 5  
>> step 6 : small = 3
```

2) a 에서 가장 작은 수 : 3

실습

- LIST

- LIST 제공 함수 사용해 보기

- sum(list)
 - list.reverse()
 - min(list)
 - max(list)



4.1example100.py

File Edit Format Run Options Window Help

```
a = [5, 2, 5, 7, 9, 10, 1]
```

```
print("\n1) dir(a) :", dir(a))
print("\n2) len(a) :", len(a))
```

```
a.reverse()
print("\n3) a :", a)
print("\n4) sum(a) =", sum(a))
```

```
1) dir(a) :
['_add_', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__',
'__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',
'__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', 'sor
t']
```

```
2) len(a) : 7
```

```
3) a : [1, 10, 9, 7, 5, 2, 5]
```

```
4) sum(a) = 39
```

- sort() 함수 처리속도
 - Time 모듈의 time()함수 사용
 - import time
 - time.time()

4.1example104.py

File Edit Format Run Options Window Help

```
import time
import random
```

```
low = 0
high = 1000000
iter = 100
```

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

```
print("\n1) 생성된 list a =", a)
```

```
start = time.time()
a.sort()
end = time.time()
```

```
print("\n2) Running sort(%d) takes %s" %(iter,end - start))
```

```
1) 생성된 list a = [512861, 595731, 469299, 297022, 945345, 46170, 258071, 44714
6, 696384, 518809, 568326, 632142, 177925, 71758, 558780, 477144, 515637, 593972
, 251587, 500963, 42059, 132635, 472398, 331576, 641770, 26958, 102554, 590729,
663801, 1124, 335459, 879061, 327154, 729620, 881248, 441488, 909067, 606848, 47
8622, 287441, 905774, 373271, 741088, 943745, 499170, 47522, 714695, 918272, 646
07, 845258, 946453, 425347, 747864, 103995, 686564, 667743, 719151, 146807, 5744
00, 791182, 884081, 227342, 943378, 673593, 102461, 267413, 997466, 677185, 6640
17, 353453, 649859, 333283, 404162, 920163, 974118, 256750, 21320, 655764, 10994
5, 965182, 883770, 220146, 354937, 702863, 301768, 587762, 197628, 116791, 88627
2, 544889, 194843, 549536, 543807, 922095, 961378, 180891, 445273, 159922, 13391
6, 608327]
```

```
2) Running sort(100) takes 0.0
```

숙제 – 파일명 : 에라토스테네스-이름-학번-일시.py

- 에라토스테네스(Ερατοσθένης)의 체 (Sieve of Eratosthenes)
 - n 이 주어졌을 때, n 까지의 모든 소수를 출력하시오. list를 사용하시오.

https://ko.wikipedia.org/wiki/%EC%97%90%EB%9D%BC%ED%86%A0%EC%8A%A4%ED%85%8C%EB%84%A4%EC%8A%A4%EC%9D%98_%EC%B2%B4

- 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 – 파일명 : 버블-이름-학번-일시.py

1. 버블정렬(bubble sort)을 구현하시오

<https://www.youtube.com/watch?v=JP5KkzdUEYI>

2. 구현한 버블정렬과 sort()함수와의 처리 시간을 비교하시오

- sort() 함수는 Python이 제공하는 정렬 함수

숙제

- 버블정렬 구현

- n개의 정수로 구성된 list 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 list 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

Homework 1 - 4.1 Structured Type - 에라토스테네스의 체

- 제출방법

파일명 :

에라토스테네스-이름-학번.py

예) 에라토스테네스 -김국민-20221234.py

- 파일이 여러 개일 경우 zip으로 묶어서 ecampus 숙제제출 link에 upload
- 제출마감
 - 2023.10.19(목) 13:00
 - 제출 마감 일시까지만 제출 가능. 마감일시 이후 ecampus 숙제제출 링크 자동 close

Homework 2 - 4.1 Structured Type – Bubble Sort

- 제출방법

파일명 :

버블-이름-학번.py

예) 버블-김국민-20221234.py

- 파일이 여러 개일 경우 zip으로 묶어서 ecampus 숙제제출 link에 upload
- 제출마감
 - 2023.10.19(목) 13:00
 - 제출 마감 일시까지만 제출 가능. 마감일시 이후 ecampus 숙제제출 링크 자동 close