여러 개의 자료를 한 변수에 담기

지금까지는 하나의 변수에 하나의 자료를 저장했다. 그러나 파이썬에서는 하나의 변수에 여러 개의 자료를 한꺼번에 저장하고 필요한 때에 꺼내 쓸 수도 있다.

- 리스트
- 딕셔너리
- 튜플
- 세트

- 리스트 내포, 사전 내포, 세트 내포

List

여러개의 순서를 가지는 자료형

```
리스트변수 = [자료1, 자료2, 자료3]
```

In [1]:
```
a_list = [2, 3, 7, 'string', [1,2,3]]

print(a_list)
print('length:',len(a_list))
```

```
[2, 3, 7, 'string', [1, 2, 3]]
length: 5
```


인덱싱(Indexing)

- 리스트 자료형 변수에서 특정한 자료를 꺼내려면 인덱싱(indexing)이라는 연산을 사용

```
리스트변수[index]
```

> 가장 첫번째 자료를 가리키는 인덱스는 1이 아니라 0 이다

In [4]:
```
a_list = [2, 3, 7, 7, 'string']
print('first component:', a_list[0])
print('(a_list[0] + a_list[1]) / 2 = ', (a_list[0] + a_list[1]) / 2)

######## find index ###########
print('where is 7:', a_list.index(7))

####### change component ##########
a_list[2] = 'change'
print(a_list)

###### reverse ######
a_list.reverse()
print(a_list)
```

```
first component: 2
(a_list[0] + a_list[1]) / 2 =  2.5
where is 7: 2
[2, 3, 'change', 7, 'string']
['string', 7, 'change', 3, 2]
```

In [9]:
```
a_string = 'string'
string_list = list(a_string)

print(string_list)

a_string_b = ''.join(string_list)

print(a_string_b)
```

```
['s', 't', 'r', 'i', 'n', 'g']
string
```

리스트에 원소 추가 삭제 이어붙이기

In [16]:
```
b_list = ['foo','bar']
b_list.append('dwarf')
print('append:',b_list)

######## insert #########
b_list.insert(1,'red')
print('insert:', b_list)

######### pop ##########
c = b_list.pop(1)
print('pop:', b_list, c)

######### remove #########
b_list.remove('foo')
print('remove:', b_list)

######### extend ##########
#[4, None, 'foo'] + [7, 8, (2, 3)]
c_list = [4, None, 'foo']
c_list.extend([7, 8, (2, 3)])
print('extend:', c_list)
```

```
append: ['foo', 'bar', 'dwarf']
insert: ['foo', 'red', 'bar', 'dwarf']
pop: ['foo', 'bar', 'dwarf'] red
remove: ['bar', 'dwarf']
extend: [4, None, 'foo', 7, 8, (2, 3)]
```

정렬(sort)

In [23]:
```
a = [7, 2, 5, 1, 3]
sorted_a = sorted(a)
print(sorted_a)
a.sort() # reverse=False
print(a)

b = ['saw', 'small', 'He', 'foxes', 'six']
# b.sort()
b.sort(key=len)
print(b)
```

```
[1, 2, 3, 5, 7]
[1, 2, 3, 5, 7]
['He', 'saw', 'six', 'small', 'foxes']
```

In [24]:
```
# bisect 메서드는 값이 추가될 때 리스트가 정렬된 상태를
# 유지 할 수 있는 위치를 반환한다.

from bisect import bisect, insort

c = [1, 2, 2, 2, 3, 4, 7]
print('where 2 go:', bisect(c, 2))
print('where 5 go:', bisect(c, 5))

# insort 메서드는 실제로 정렬된 상태를 유지한 채 값을 추가한다.
insort(c, 6)
print(c)
```

```
where 2 go: 4
where 5 go: 6
[1, 2, 2, 2, 3, 4, 6, 7]
```

슬라이싱(Slicing)

In [26]:
```
seq = [7, 2, 3, 7, 5, 6, 0, 1]
print(seq[1:5])

print(seq[:5])
print(seq[3:])
print(seq[::-1]) # 역순
```

```
[2, 3, 7, 5]
[7, 2, 3, 7, 5]
[7, 5, 6, 0, 1]
[1, 0, 6, 5, 7, 3, 2, 7]
```

In [8]:
```
# 리스트로 matrix 표현하기
matrix =[[1, 10],
         [2, 20]]
print(matrix[0])
print(matrix[1])
print(matrix[0][0])
print(matrix[1][1])
```

```
[1, 10]
[2, 20]
1
20
```

In [32]:
```
# count
a = [1,2,3,1,1]
a.count(1)
```

Out[32]: 3

문제!!!

- start_list 의 값을 제곱하여 값이 작은 순서대로 정렬한 새로운 리스트를 만들어보세요.
- start_list = [-5, 3, 1, -2, 4]

In [48]:
```
start_list = [-5, 3, 1, -2, 4]

square_list = []
for num in start_list:
    square_list.append(num ** 2)

square_list2 = []
for i in range(len(start_list)):
    square_list2.append(start_list[i]**2)

square_list3 = []
for i, num in enumerate(start_list):
    square_list3.append(num**2)


square_list.sort()
square_list2.sort()
square_list3.sort()
print (square_list, square_list2, square_list3)
```

```
[1, 4, 9, 16, 25] [1, 4, 9, 16, 25] [1, 4, 9, 16, 25]
```

문제!!!

- 10과목 평균 성적 구하기
  - 90, 85, 95, 80, 90, 100, 85, 75, 85, 80
- 평균 평점(GPA) 구하기
  - value = 4,3,2,3,4
  - credit = 3,3,1,1,2

In [11]:
```
## 1. 평균 성적
Soo = [90, 85, 95, 80, 90, 100, 85, 75, 85, 80]

# soln 1
sum_1 = 0
for i in range(len(Soo)):
    sum_1 += Soo[i]

# soln 2
sum_2 = 0
for num in Soo:
    sum_2 += num

# soln 3
sum_3 = sum(Soo)

average = sum_1 / len(Soo)
print('average:', average)
```

```
average: 86.5
```

In [12]:
```
## 2. GPA 구하기
value = [4,3,2,3,4]
credit = [3,3,1,1,2]

sum_4 = 0
overall_credit = 0

for i in range(len(value)):
    sum_4 += value[i]*credit[i]
    overall_credit += credit[i]

GPA = sum_4 / overall_credit
print('GPA:', GPA)
```

```
GPA: 3.4
```

리스트 안에 for문 포함하기

리스트 안에 for문을 포함하는 것을 리스트 내포(List comprehension)라고 함.

```
[표현식 for 항목 in 반복가능객체 if 조건]
```

```
start_list = [-5, 3, 1, -2, 4]
square_list = []

for num in start_list:
    square_list.append(num ** 2)
start_list.sort()

>>> print(square_list)
```

를 리스트 내포 기능을 써서 줄이면

In [13]:
```
start_list = [-5, 3, 1, -2, 4]
result = [num ** 2 for num in start_list]
result.sort()

print(result)
```

```
[1, 4, 9, 16, 25]
```

> 만약 a리스트에서 짝수에만 10을 곱하여 담고 싶다면 다음과 같이 "if 조건"을 사용할 수 있다.

In [51]:
```
start_list = [-5, 3, 1, -2, 4]

result = [num ** 2 for num in start_list if num > 0]

# for num in start_list:
#     if num > 0:
#         result.append(num**2)

result.sort()
print(result)
```

```
[1, 9, 16]
```

List method 정리

Method Name	Use	Explanation
append	alist.append(item)	Adds a new item to the end of a list
insert	alist.insert(i,item)	Inserts an item at the ith position in a list
pop	alist.pop()	Removes and returns the last item in a list
pop	alist.pop(i)	Removes and returns the ith item in a list
sort	alist.sort()	Modifies a list to be sorted
reverse	alist.reverse()	Modifies a list to be in reverse order
del	del alist[i]	Deletes the item in the ith position
index	alist.index(item)	Returns the index of the first occurrence of item
count	alist.count(item)	Returns the number of occurrences of item
remove	alist.remove(item)	Removes the first occurrence of item

Dictionary

- `dict`(사전)은 유연한 크기를 가지는 키-밸류 쌍이다.
- 일반적으로 해시맵 또는 연관 배열이라고 알려져 있다.
- 중괄호 `{}`를 사용해, 콜론:로 구분되는 키와 값을 둘러싸서 생성한다.
- 순서가 없다. 출력을 할 때 마다 내부 데이터들의 순서가 달라짐

```
딕셔너리변수 = {key1:value1, key2:value2, ...}
```

선언, 참조

In [64]:
```
###### 빈 딕셔너리 생성 #######
empty_dict = {}

d1 = {'a':'some value'}

d1['b'] = [1,2,3,4]
d1['c']='wow'
print(d1)
print('-----------------------------------')

######### 참조 ##########
d1['b'] = 'an integer'  # 7이란 키가 없으면 생성해서 넣는다.
print(d1)
print('-----------------------------------')

######### 참조 ##########
print("'b'라는 키가 d1딕셔너리에 있나요?:",'b' in d1)
print("'b' 안에는 어떤 값이 있나요?:", d1['b'])
print("'b' 안에는 어떤 값이 있나요?:", d1.get('e'))
print("'c' 안에는 어떤 값이 있나요?:",d1.get('d', 'No Key'))
```

```
{'a': 'some value', 'c': 'wow', 'b': [1, 2, 3, 4]}
-----------------------------------
{'a': 'some value', 'c': 'wow', 'b': 'an integer'}
-----------------------------------
'b'라는 키가 d1딕셔너리에 있나요?: True
'b' 안에는 어떤 값이 있나요?: an integer
'b' 안에는 어떤 값이 있나요?: None
'c' 안에는 어떤 값이 있나요?: No Key
```

Update, del

In [70]:
```
print('original:',d1)
d1.update({'a' : 'foo', 'd': 12})
print('updated:',d1)
print('----------------')

# del d1['c']

print('del:', d1)
```

```
original: {'b': 'an integer', 'd': 12, 'a': 'foo'}
updated: {'b': 'an integer', 'd': 12, 'a': 'foo'}
----------------
del: {'b': 'an integer', 'd': 12, 'a': 'foo'}
```

In [17]:
```
popped = d1.pop('b')
print('popped:',popped)
print(d1)

d1.clear()
print(d1)
```

```
popped: an integer
{'a': 'foo'}
{}
```

keys, values, items

In [71]:
```
d1 = {'a':'some value', 'b':55, 'c':[1,2,3]}
######### KEY, VALUE ##########
print('d1.keys():', d1.keys())
print('d1.values():', d1.values())
print('list of d1 key:', list(d1.keys()))

print(d1.items())
```

```
d1.keys(): dict_keys(['a', 'c', 'b'])
d1.values(): dict_values(['some value', [1, 2, 3], 55])
list of d1 key: ['a', 'c', 'b']
dict_items([('a', 'some value'), ('c', [1, 2, 3]), ('b', 55)])
```

In [75]:
```
# Dictionary를 for문에서 key와 value 표현
# .items() 붙는거 참고

interest_stocks = {"Naver":10, "Samsung":50, "SK Hynix":30}
for company, stock_num in interest_stocks.items():
    print("%s: Buy %s" %(company, stock_num))
```

```
SK Hynix: Buy 30
Samsung: Buy 50
Naver: Buy 10
```

Coding convention of Dictionary

- 일반적 로직

```
if(for) key in some_dict :
    value = some_dict[key]
else :
    value = default_value
```

- 만약 `some_dict` 에 `key` 가 있으면 그 `key` 를 가지는 `value` 를 사용함
- 그 `key` 가 `some_dict` 에 존재하지 않는다면 `default_value`

In [20]:
```
interest_stocks = {"Naver":10, "Samsung":50, "SK Hynix":30}

for key in interest_stocks:
    print (key, interest_stocks[key])
```

```
SK Hynix 30
Samsung 50
Naver 10
```

문제!!!

- 네이버를 2주, 삼성을 3주 구매했을 때, interest_stocks을 참고해서 내가 구매한 주식 가격의 총합을 구해보세요.

In [76]:
```
buying_stocks = {'Naver':2, 'Samsung':3}

purchasing_cost = 0

for key in buying_stocks:
    purchasing_cost += interest_stocks[key]*buying_stocks[key]

print(purchasing_cost)
```

```
170
```

보통 사전의 `값(value)`으로 `list` 같은 다른 컬렉션 값을 많이 이용한다.

In [22]:
```
words = ['apple', 'bat', 'bar', 'atom', 'book']
by_letter = {}

for word in words :
    letter = word[0]
    if letter not in by_letter :
        by_letter[letter] = [word]
    else :
        by_letter[letter].append(word)

print(by_letter)
```

```
{'b': ['bat', 'bar', 'book'], 'a': ['apple', 'atom']}
```

`setdefault` 메서드를 사용하면 `value`의 기본자료형을 설정하므로, 위의 코드를 더 줄일 수 있다.

In [23]:
```
words = ['apple', 'bat', 'bar', 'atom', 'book']
by_letter = {}

for word in words :
    letter = word[0]
    by_letter.setdefault(letter, []).append(word)

by_letter
```

Out[23]:
```
{'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

내장 collections 모듈의 `defaultdict` 클래스를 사용하면 더 쉽다.

In [24]:
```
from collections import defaultdict

words = ['apple', 'bat', 'bar', 'atom', 'book']
by_letter = defaultdict(list)
for word in words :
    by_letter[word[0]].append(word)

by_letter
```

Out[24]:
```
defaultdict(list, {'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']})
```

Dictionary method 정리

Method Name	Use	Explanation
keys	adict.keys()	Returns the keys of the dictionary in a dict_keys object
values	adict.values()	Returns the values of the dictionary in a dict_values object
items	adict.items()	Returns the key-value pairs in a dict_items object
get	adict.get(k)	Returns the value associated with k, None otherwise
get	adict.get(k,alt)	Returns the value associated with k, alt otherwise

Tuple

- 1차원, 고정된 크기를 가지는 변경 불가능한 순차 자료형
- , 로 구분되는 개별 원소의 열 혹은 괄호 ()나 tuple 함수로 생성.

```
튜플변수 = (value1, value2, value3, ...)
```

선언, 참조

In [59]:
```
tup = (4, 5, 6)  # 혹은 tup = 4, 5, 6
print(tup)

tup = tuple('str')
print(tup)
print(tup[0])
a, b, c = tup
print(a,b)

# count 메서드
a = (1, 2, 2, 2, 3, 4, 2)
a.count(2)
```

```
(4, 5, 6)
('s', 't', 'r')
s
s t
```

Out[59]: 4

Set

- `세트(집합)`는 유일한 원소만 담는`(중복을 허용하지 않는다)` 정렬되지 않은`(순서가 없다)` 자료형이다. `set` 함수나 중괄호 `{}`를 사용해 생성한다.
- 필터 역할로 종종 사용되기도 한다

In [26]:
```
s1 = set([1, 2, 3]) # s1 = {1,2,3}
print(s1)

s1.add(4)
print(s1)

s1.update([3, 4, 5])
print(s1)

s1.remove(3)
print(s1)

string = 'I am a boy you are a boy'
lists = list(set(string.split()))
print(lists)
```

```
{1, 2, 3}
{1, 2, 3, 4}
{1, 2, 3, 4, 5}
{1, 2, 4, 5}
['boy', 'am', 'a', 'you', 'are', 'I']
```

In [27]:
```
a = {1, 2, 2, 3, 4, 5}
b = {3, 4, 5, 6, 6, 6, 7, 8}

print('a:',a)
print('b:',b)

## 합집합
print ('합집합:', a | b)

##교집합
print ('교집합:',a & b)

##차집합
print ('차집합:',a - b)

## 여집합

## 부분집합 검사
print({1, 2, 3}.issubset(a))
```

```
a: {1, 2, 3, 4, 5}
b: {3, 4, 5, 6, 7, 8}
합집합: {1, 2, 3, 4, 5, 6, 7, 8}
교집합: {3, 4, 5}
차집합: {1, 2}
True
```


Set method 정리

Method Name	Use	Explanation
union	aset.union(otherset)	Returns a new set with all elements from both sets
intersection	aset.intersection(otherset)	Returns a new set with only those elements common to both sets
difference	aset.difference(otherset)	Returns a new set with all items from first set not in second
issubset	aset.issubset(otherset)	Asks whether all elements of one set are in the other
add	aset.add(item)	Adds item to the set
remove	aset.remove(item)	Removes item from the set
pop	aset.pop()	Removes an arbitrary element from the set
clear	aset.clear()	Removes all elements from the set