

# Ch 6.

## 리스트와 딕셔너리

# 리스트

- 어떤 경우에는 여러 개의 데이터를 하나로 묶어서 저장하는 것이 필요하다.



```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치" ]
```

전체적인 구조



```
리스트 = [ 값1 , 값2 , ... ]
```

```
scores = [ 32, 56, 64, 72, 12, 37, 98, 77, 59, 69 ]
```

# 공백 리스트에서 추가하기

```
>>> heroes = []  
>>> heroes.append("아이언맨")  
['아이언맨']  
>>> heroes.append("닥터 스트레인지")  
>>> print(heroes)  
['아이언맨', '닥터 스트레인지']  
>>>
```



## 점의 의미


- 파이썬에서 모든 것은 객체(object)이다. 객체는 관련되는 변수와 함수를 묶은 것이다.
- 파이썬에서 리스트도 객체이다. 객체 안에 있는 무엇인가를 사용할 때는 객체의 이름을 쓰고 점(.)을 붙인 후에 함수의 이름을 적는다.

```
heroes.append("아이언맨")
```

# 리스트 항목 접근하기

```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']
```

'A'	'B'	'C'	'D'	'E'	'F'
0	1	2	3	4	5



```
>>> print(letters[0])
```

A

```
>>> print(letters[1])
```

B

```
>>> print(letters[2])
```

C

# 슬라이싱

- 슬라이싱(slicing)은 리스트에서 한 번에 여러 개의 항목을 추출하는 기법이다.

```
>>> letters = ['A', 'B', 'C', 'D', 'E', 'F']  
>>> print(letters[0:3])  
['A', 'B', 'C']
```

# 인덱스 생략

---

```
>>> print(letters[:3])  
['A', 'B', 'C']
```

```
>>> print(letters[3:])  
['D', 'E', 'F']
```

```
>>> print(letters[:])  
['A', 'B', 'C', 'D', 'E', 'F']
```

리스트를 복사할 때 사용한다.

# 리스트 항목 변경하기

---

```
>>> heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치" ]  
>>> heroes[1] = "닥터 스트레인지"  
>>> print(heroes)  
['아이언맨', '닥터 스트레인지', '헐크', '스칼렛 위치']  
>>>
```

인덱스를 이용한다.



# 함수를 이용하여 추가하기

```
>>> heroes.append("스파이더맨")
```

```
>>> print(heroes)
```

```
['아이언맨', '닥터 스트레인지', '헐크', '스칼렛 위치', '스파이더맨']
```

```
>>> heroes.insert(1, "배트맨")
```

```
>>> print(heroes)
```

```
['아이언맨', '배트맨', '닥터 스트레인지', '헐크', '스칼렛 위치', '스파이더맨']
```

```
>>>
```

# 항목 삭제하기

---

```
heroes = ["아이언맨", "토르", "헐크", "스칼렛 위치"]  
heroes.remove("스칼렛 위치")  
print(heroes)
```

```
['아이언맨', '토르', '헐크']
```

# 항목이 리스트 안에 있는지 체크

---

```
if "슈퍼맨" in heroes:  
    heroes.remove("슈퍼맨")
```

# del

- del는 인덱스를 사용하여 항목을 삭제한다.

```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치" ]  
del heroes[0]  
print(heroes)
```

```
['토르', '헐크', '스칼렛 위치']
```

# pop()

- pop()은 리스트에서 마지막 항목을 삭제

```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치" ]  
last_hero = heroes.pop()  
print(last_hero)
```

스칼렛 위치

# 리스트 탐색하기

## ■ index() 사용

```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치" ]  
print(heroes.index("헐크"))
```

2

# 리스트의 길이

---

- `len()` 연산은 리스트의 길이를 계산하여 반환한다.

```
>>> letters = ['a', 'b', 'c', 'd']  
>>> len(letters)  
4
```

# 리스트 정렬하기

```
heroes = ["아이언맨", "토르", "헐크", "스칼렛 위치"]  
heroes.sort()  
print(heroes)
```

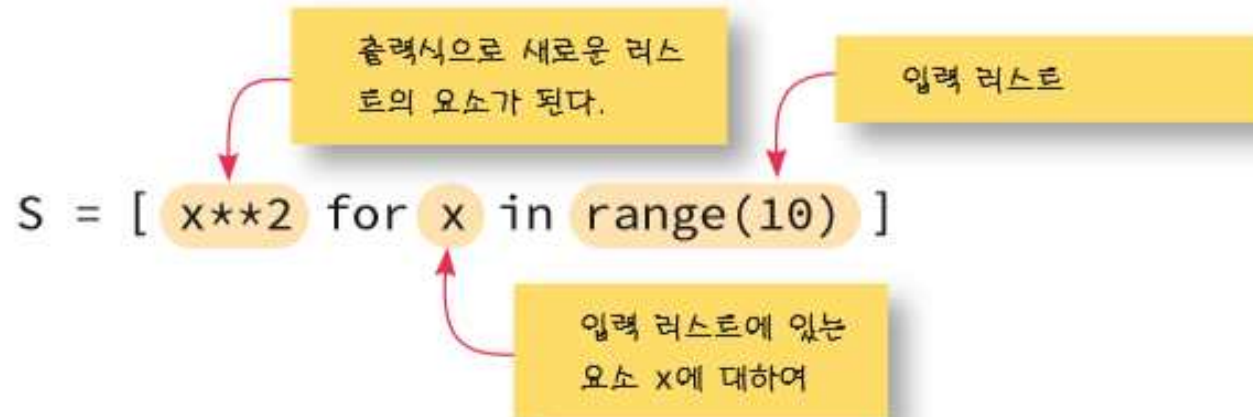
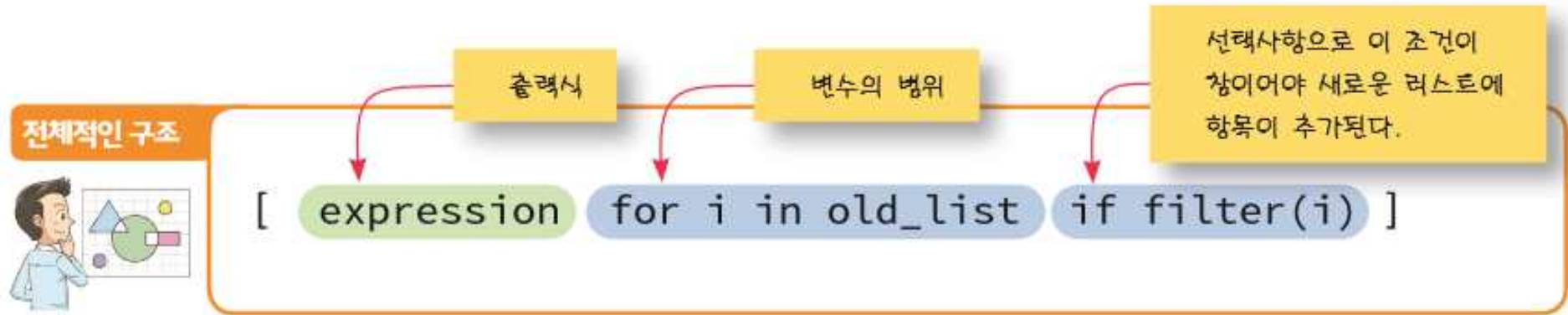
['스칼렛 위치', '아이언맨', '토르', '헐크']





# 리스트 함축

- 리스트를 수학자들이 집합을 정의하는 것과 유사하게 생성하는 것



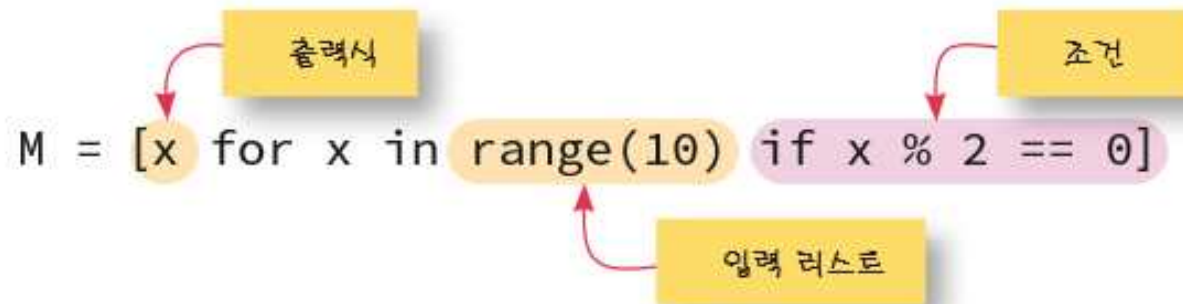
# 예제

---

```
list1 = [3, 4, 5]  
list2 = [x*2 for x in list1]  
print(list2)
```

```
[6, 8, 10]
```

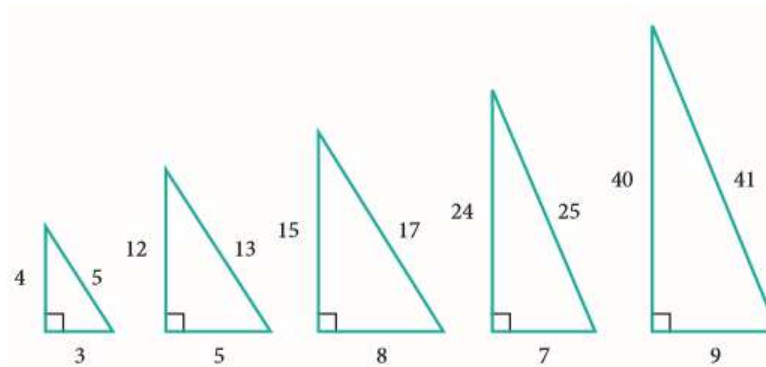
# 조건이 붙는 리스트 함축



[0, 2, 4, 6, 8]

## Lab: 피타고라스 삼각형

- 타고라스의 정리를 만족하는 삼각형들을 모두 찾아보자. 삼각형 한 변의 길이는 1부터 30 이하이다.

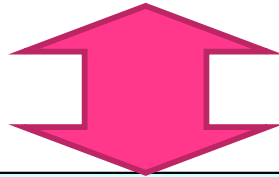


$[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15), (10, 24, 26), (12, 16, 20), (15, 20, 25), (20, 21, 29)]$

# Solution

---

```
new_list = [(x,y,z) for x in range(1,30) for y in range(x,30) for z in range(y,30) if  
x**2 + y**2 == z**2]  
print(new_list)
```



```
new_list = []  
for x in range(1, 30):  
    for y in range(x, 30):  
        for z in range(y, 30):  
            if x**2+y**2==z**2:  
                new_list.append((x, y, z))  
print(new_list)
```

## 2차원 리스트

### ■ 2차원 테이블 표시

학생	국어	영어	수학	과학	사회
김철수	1	2	3	4	5
김영희	6	7	8	9	10
최자영	11	12	13	14	15

# 2차원 리스트를 생성한다.

```
s = [  
    [ 1, 2, 3, 4, 5 ],  
    [ 6, 7, 8, 9, 10 ],  
    [11, 12, 13, 14, 15 ]  
]  
print(s)
```

```
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15]]
```

# 동적으로 2차원 리스트 생성

---

```
# 동적으로 2차원 리스트를 생성한다.
```

```
rows = 3
```

```
cols = 5
```

```
s = []
```

```
for row in range(rows):
```

```
    s += [[0]*cols]
```

```
print("s =", s)
```

```
s = [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

## 2차원 리스트 요소 접근

```
s = [  
    [ 1, 2, 3, 4, 5 ],  
    [ 6, 7, 8, 9, 10 ],  
    [11, 12, 13, 14, 15 ]  
]  
  
# 행과 열의 개수를 구한다.  
rows = len(s)  
cols = len(s[0])  
  
for r in range(rows):  
    for c in range(cols):  
        print(s[r][c], end=",")  
    print()
```

```
1,2,3,4,5,  
6,7,8,9,10,  
11,12,13,14,15,
```



# 튜플

- 튜플(tuple)은 변경될 수 없는 리스트

전체적인 구조



튜플 = ( 항목1 , 항목2 , ... , 항목n )

```
>>> colors = ("red", "green", "blue")
```

```
>>> colors
```

```
('red', 'green', 'blue')
```

```
>>> numbers = (1, 2, 3, 4, 5)
```

```
>>> numbers
```

```
(1, 2, 3, 4, 5)
```

# 튜플은 변경할 수 없다

---

```
>>> t1 = (1, 2, 3, 4, 5);
```

```
>>> t1[0] = 100;
```

*Traceback (most recent call last):*

*File "<pyshell#11>", line 1, in <module>*

*t1[0]=100*

*TypeError: 'tuple' object does not support item assignment*

```
>>> numbers = ( 1, 2, 3, 4, 5 )
```

```
>>> colors = ("red", "green", "blue")
```

```
>>> t = numbers + colors
```

```
>>> t
```

```
(1, 2, 3, 4, 5, 'red', 'green', 'blue')
```

# 기본적인 튜플 연산들

파이썬 수식	결과	설명
<code>len((1, 2, 3))</code>	3	튜플의 길이
<code>(1, 2, 3) + (4, 5, 6)</code>	<code>(1, 2, 3, 4, 5, 6)</code>	접합
<code>('Hi!',) * 4</code>	<code>('Hi!', 'Hi!', 'Hi!', 'Hi!')</code>	반복
<code>3 in (1, 2, 3)</code>	True	멤버십
<code>for x in (1, 2, 3): print x,</code>	1 2 3	반복

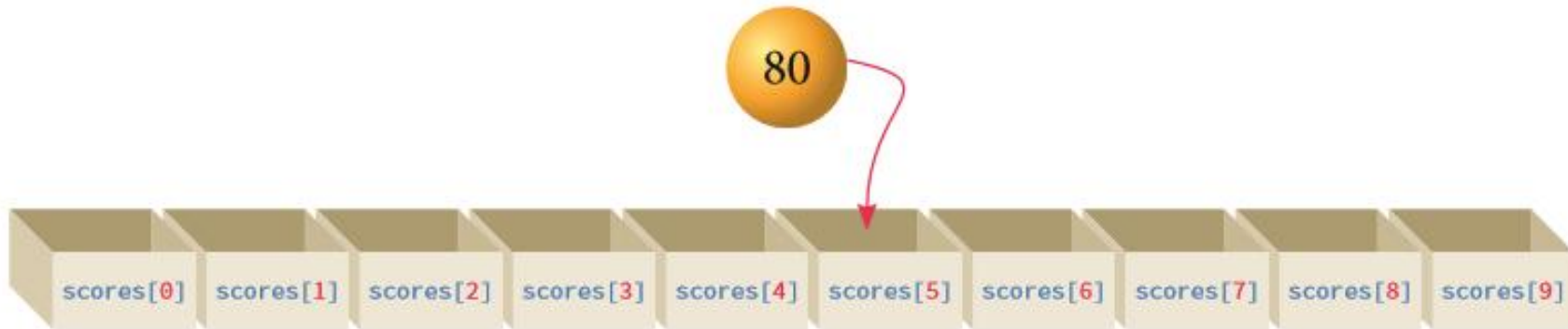
함수	설명
<code>cmp(t1, t2)</code>	2개의 튜플을 비교한다.
<code>len(t)</code>	튜플의 길이를 반환한다.
<code>max(t)</code>	튜플에 저장된 최대값을 반환한다.
<code>min(t)</code>	튜플에 저장된 최소값을 반환한다.
<code>tuple(seq)</code>	리스트를 튜플로 변환한다.

# 튜플 대입 연산

---

```
>>> student1 = ("철수", 19, "CS")
>>> (name, age, major) = student1
>>> name
'철수'
>>> age
19
>>> major
'CS'
```

# 리스트 요소 접근, 순회하기



`scores[5] = 80`

인덱스

리스트 안의 요소들이 차례대로  
변수에 대입되면서 반복된다.

전체적인 구조



for 변수 in 리스트 :

문장1

문장2

```
scores = [ 32, 56, 64, 72, 12, 37, 98, 77, 59, 69]
```

```
for element in scores:  
    print(element)
```

# list 클래스

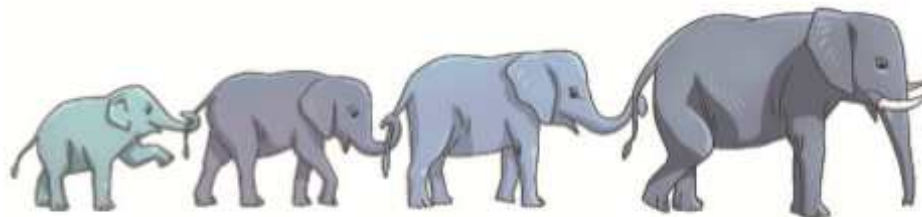
```
list1 = [ ]                # 공백 리스트 생성  
list2 = [ "H", "e", "l", "l", "o" ]    # 문자 H, e, l, l, o를 요소로 가지는 리스트  
list3 = [ 0, 1, 2, 3, 4 ]    # 0, 1, 2, 3, 4를 요소가 가지는 리스트 생성
```

```
list1 = [12, "dog", 180.14]    # 혼합 자료형  
list2 = [ ["Seoul", 10], ["Paris", 12], ["London", 50] ]    # 내장 리스트  
list3 = [ "aaa", [ "bbb", [ "ccc", [ "ddd", "eee", 45] ] ] ]    # 내장 리스트
```

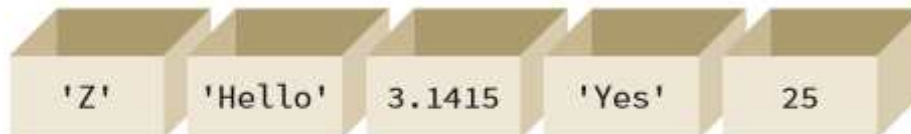
# 시퀀스 자료형

- 시퀀스: 순서를 가진 요소들의 집합
  - 문자열
  - 바이트 시퀀스
  - 바이트 배열
  - 리스트
  - 튜플
  - range 객체

문자열:



리스트:



순서를 가지고 요소들로  
구성된 자료형들을 모두  
시퀀스라고 합니다.

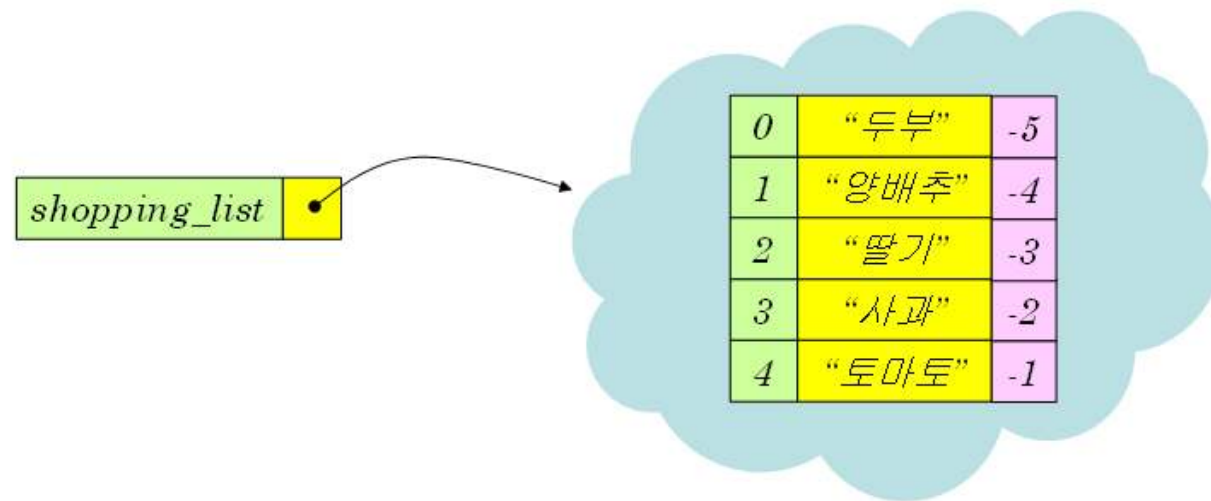


# 시퀀스에서 가능한 연산과 함수

함수나 연산자	설명	예	결과
len()	길이 계산	len([1, 2, 3])	3
+	2개의 시퀀스 연결	[1, 2] + [3, 4, 5]	[1, 2, 3, 4, 5]
*	반복	['Welcome!'] * 3	['Welcome!', 'Welcome!', 'Welcome!']
in	소속	3 in [1, 2, 3]	True
not in	소속하지 않음	5 not in [1, 2, 3]	True
[]	인덱스	myList[1]	myList의 1번째 요소
min()	시퀀스에서 가장 작은 요소	min([1, 2, 3])	1
max()	시퀀스에서 가장 큰 요소	max([1, 2, 3])	3
for 루프	반복	for x in [1, 2, 3]: print (x)	1 2 3



# 음수 인덱스



# 리스트는 변경가능

---

```
>>> squares = [0, 1, 4, 9, 16, 25, 36, 48]    # 잘못된 부분이 있음!
>>> 7 ** 2 # 7의 제곱은 49임!
49
>>> squares[7] = 49                          # 잘못된 값을 변경한다.
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49]
```

# 리스트의 기초 연산

- 두개의 리스트를 합칠 때는 연결 연산자인 + 연산자를 사용할 수 있다.

```
>>> marvel_heroes = [ "스파이더맨", "헐크", "아이언맨" ]  
>>> dc_heroes = [ "슈퍼맨", "배트맨", "원더우먼" ]  
>>> heroes = marvel_heroes + dc_heroes  
>>> heroes  
['스파이더맨', '헐크', '아이언맨', '슈퍼맨', '배트맨', '원더우먼']
```

```
>>> values = [ 1, 2, 3 ] * 3  
>>> values  
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

# 리스트 일치 검사

- 우리는 비교 연산자 `==`, `!=`, `>`, `<`를 사용하여서 2개의 리스트를 비교할 수 있다.

```
>>> list1 = [ 1, 2, 3 ]  
>>> list2 = [ 1, 2, 3 ]  
>>> list1 == list2  
True
```

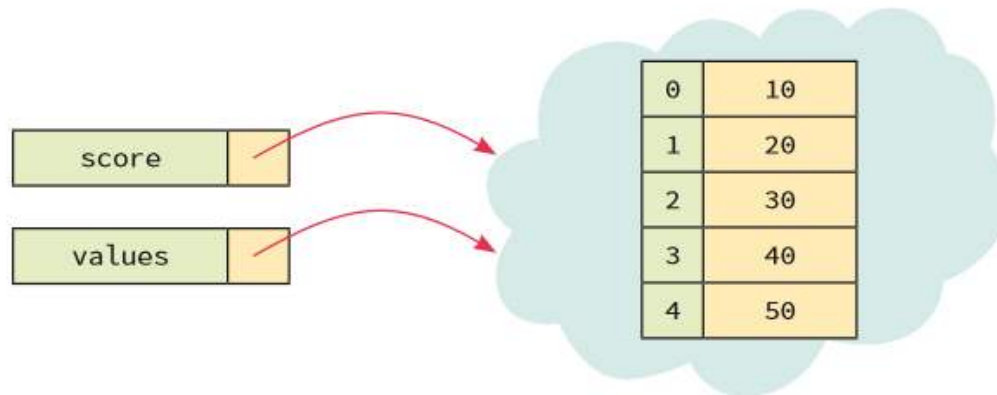
# 리스트 최소값과 최대값 찾기

- 리스트 안에서 최소값과 최대값을 찾으려면 내장 메소드인 `max()`와 `min()`을 사용하면 된다.

```
>>> values = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]  
>>> min(values)  
1  
>>> max(values)  
10
```

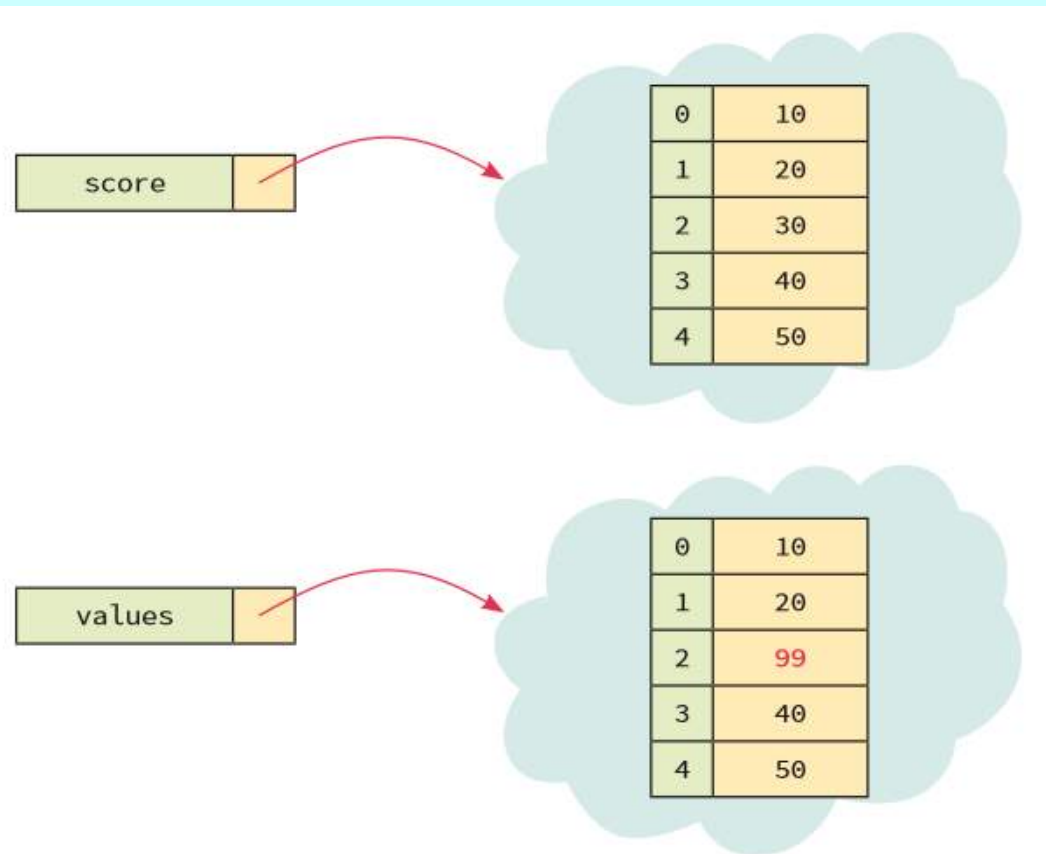
# 리스트 복사하기

```
scores = [ 10, 20, 30, 40, 50 ]  
values = scores
```



# 깊은 복사

```
>>> scores = [ 10, 20, 30, 40, 50 ]  
>>> values = list(scores)  
>>> values[2]=99  
>>> scores  
[10, 20, 30, 40, 50]  
>>> values  
[10, 20, 99, 40, 50]
```



# 리스트와 함수

- 리스트는 “참조로 호출하기”가 적용된다.

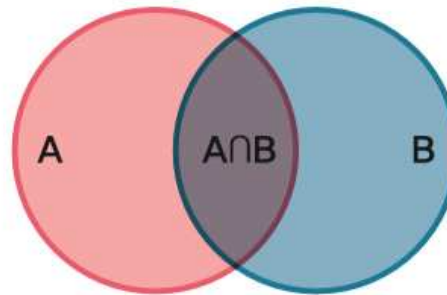
```
def func2(list):  
    list[0] = 99  
  
values = [0, 1, 1, 2, 3, 5, 8]  
print(values)  
func2(values)  
print(values)
```

```
[0, 1, 1, 2, 3, 5, 8]  
[99, 1, 1, 2, 3, 5, 8]
```



# 세트(Set)

- 세트(set)는 우리가 수학에서 배웠던 집합이다.
- 세트는 중복되지 않은 항목들이 모인 것
- 세트의 항목 간에는 순서가 없다.



전체적인 구조



세트 = { 항목1 , 항목2 , ... , 항목n }

# 예제

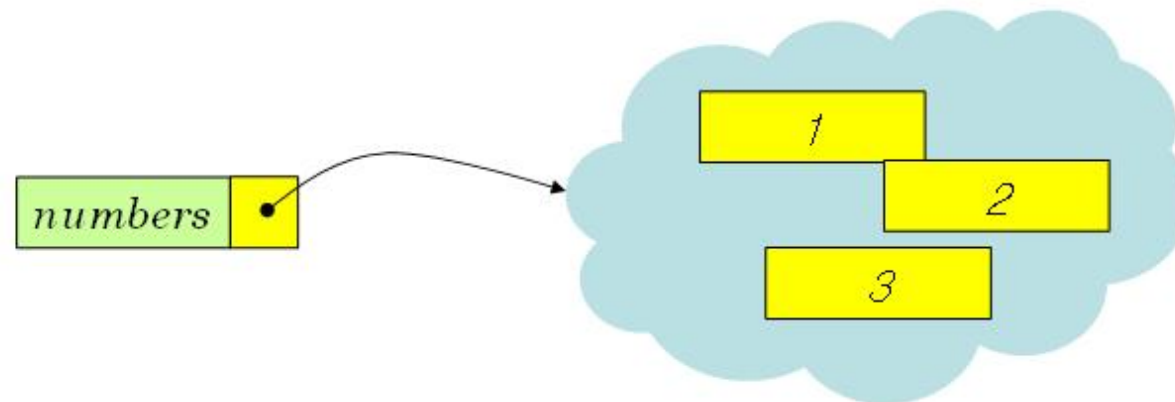
```
>>> numbers = {2, 1, 3}
```

```
>>> numbers  
{1, 2, 3}
```

```
>>> len(numbers)  
3
```

```
>>> fruits = { "Apple", "Banana", "Pineapple" }
```

```
>>> mySet = { 1.0, 2.0, "Hello World", (1, 2, 3) }
```



# in 연산자

---

```
>>> numbers = {2, 1, 3}
>>> if 1 in numbers:
    print("집합 안에 1이 있습니다.")
```

집합 안에 1이 있습니다.

```
>>> numbers = {2, 1, 3}
>>> for x in numbers:
    print(x, end=" ")
```

1 2 3

# 세트에 요소 추가하기

---

```
>>> numbers = { 2, 1, 3 }  
>>> numbers[0]  
...  
TypeError: 'set' object does not support indexing
```

```
>>> numbers.add(4)  
>>> numbers  
{1, 2, 3, 4}
```

```
numbers.update([2,3,4,5])
```

```
numbers.discard(5)
```

```
numbers.clear()
```

# 부분 집합 연산

---

```
>>> A = {1, 2, 3}
```

```
>>> B = {1, 2, 3}
```

```
>>> A == B
```

```
True
```

```
>>> A = {1, 2, 3, 4, 5}
```

```
>>> B = {1, 2, 3}
```

```
>>> B < A
```

```
True
```

```
>>> A = {1, 2, 3, 4, 5}
```

```
>>> B = {1, 2, 3}
```

```
>>> B.issubset(A)
```

```
True
```

# 집합 연산

---

```
>>> A = {1, 2, 3}
```

```
>>> B = {3, 4, 5}
```

```
>>> A | B
```

```
{1, 2, 3, 4, 5}
```

```
>>> A & B
```

```
{3}
```

```
>>> A - B
```

```
{1, 2}
```

## Lab: 파티 동시 참석자 알아내기

- 파티에 참석한 사람들의 명단이 세트 **A**와 **B**에 각각 저장되어 있다. 2개 파티에 모두 참석한 사람들의 명단을 출력하려면 어떻게 해야 할까?

2개의 파티에 모두 참석한 사람은 다음과 같습니다.

```
partyA = set(["Park", "Kim", "Lee"])  
partyB = set(["Park", "Choi"])
```

# Solution

---

```
partyA = set(["Park", "Kim", "Lee"])
partyB = set(["Park", "Choi"])

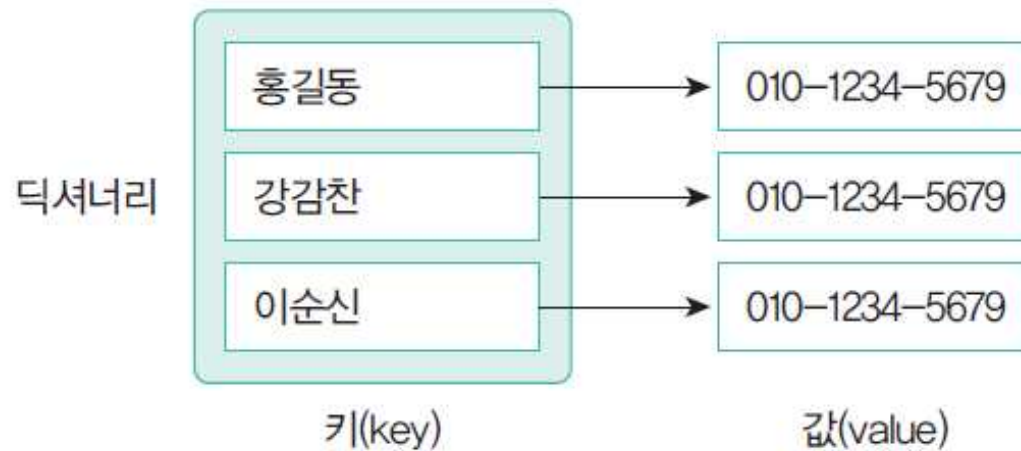
print("2개의 파티에 모두 참석한 사람은 다음과 같습니다. ")
print ( partyA.intersection(partyB))

A.union(B)
A.difference(B)
```



# 딕셔너리

- 딕셔너리(dictionary)도 리스트와 같이 값을 저장하는 방법이다.
  - 딕셔너리에는 값(value)과 관련된 키(key)가 있다.



# 딕셔너리

---

```
>>> phone_book = { }  
>>> phone_book["홍길동"] = "010-1234-5678"
```

```
>>> print(phone_book)  
{'홍길동': '010-1234-5678'}
```

```
>>> phone_book = {"홍길동": "010-1234-5678"}  
>>> phone_book["강감찬"] = "010-1234-5679"  
>>> phone_book["이순신"] = "010-1234-5680"
```

```
>>> print(phone_book)  
{'이순신': '010-1234-5680', '홍길동': '010-1234-5678', '강감찬': '010-1234-5679'}
```

## 딕셔너리에서 탐색

---

- 키를 가지고 값을 찾는다.

```
>>> print(phone_book["강감찬"])  
010-1234-5679
```

# 딕셔너리의 모든 키 출력하기

---

```
>>> phone_book.keys()  
dict_keys(['이순신', '홍길동', '강감찬'])
```

```
>>> phone_book.values()  
dict_values(['010-1234-5680', '010-1234-5678', '010-1234-5679'])
```

## 예제

- 한 학생에 대한 정보를 딕셔너리로 저장하기

```
dict = {'Name': '홍길동', 'Age': 7, 'Class': '초급'}  
print (dict['Name'])  
print (dict['Age'])
```

홍길동  
7

# 딕셔너리 항목 방문

---

```
>>> for key in sorted(phone_book.keys()):  
    print(key, phone_book[key])
```

강감찬 010-1234-5679

이순신 010-1234-5680

홍길동 010-1234-5678

# Lab: 편의점 재고 관리

- 편의점에서 재고 관리를 수행하는 프로그램을 작성해보자.
  - 편의점에서 판매하는 물건의 재고를 딕셔너리에 저장한다.

물건의 이름을 입력하시오: 콜라  
4

## ■ solution

```
items = { "커피음료": 7, "펜": 3, "종이컵": 2, "우유": 1, "콜라": 4, "책": 5 }
```

```
item = input("물건의 이름을 입력하시오: ");  
print (items[item])
```

## Lab: 영한사전

- 영한사전을 구현해보자. 영어 단어를 키로 하고 설명을 값으로 하여 저장하면 될 것이다

단어를 입력하시오: one

하나

단어를 입력하시오: two

둘

```
english_dict = dict()
```

```
english_dict['one'] = '하나'
```

```
english_dict['two'] = '둘'
```

```
english_dict['three'] = '셋'
```

```
word = input("단어를 입력하시오: ");
```

```
print (english_dict[word])
```



# 항목 접근하기

---

```
>>> contacts = {'Kim': '01012345678', 'Park': '01012345679',  
'Lee': '01012345680' }
```

```
>>> contacts['Kim']  
'01012345678'
```

```
>>> contacts.get('Kim')  
'01012345678'
```

```
>>> if "Kim" in contacts:  
    print("키가 딕셔너리에 있음")
```

# 항목 추가 & 삭제하기

---

```
>>> contacts['Choi'] = '01056781234'
>>> contacts
{'Kim': '01012345678', 'Choi': '01056781234', 'Lee': '01012345680',
'Park': '01012345679'}
```

```
>>> contacts = {'Kim': '01012345678', 'Park': '01012345679',
'Lee': '01012345680' }

>>> contacts.pop("Kim")
'01012345678'

>>> contacts
{'Lee': '01012345680', 'Park': '01012345679'}
```

# 도전문제

---

- 숫자 5개를 입력 받아 리스트에 저장하고 저장된 숫자의 평균을 계산하는 프로그램을 작성하시오.

```
-----  
정수를 입력하시오: 1  
정수를 입력하시오: 2  
정수를 입력하시오: 3  
정수를 입력하시오: 4  
정수를 입력하시오: 5  
평균 = 3.0
```

# 도전문제

- 주사위를 1000번 던져서 각 숫자가 나올 확률을 구하는 프로그램을 작성하시오. (random 모듈 사용할 것)

주사위가 1 인 확률은 0.16700000000000012  
주사위가 2 인 확률은 0.16600000000000012  
주사위가 3 인 확률은 0.15700000000000001  
주사위가 4 인 확률은 0.14900000000000001  
주사위가 5 인 확률은 0.16800000000000012  
주사위가 6 인 확률은 0.19300000000000014