산업 인공지능 - 실습 3

Python 프로그래밍

2021 Spring

1. 리스트

❖ **리스트(list)**

- 여러 개의 데이터가 저장되어 있는 장소

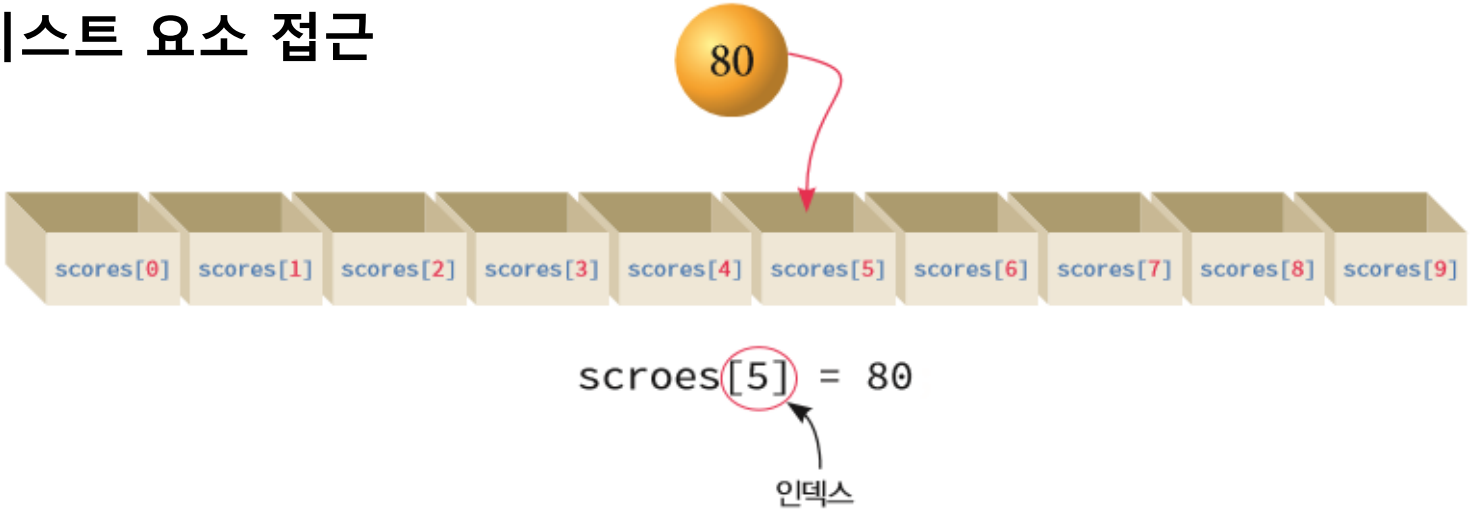
```
리스트 = [ 값1, 값2, ...]
```

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

```
scores = [ ]
for i in range(10):
    scores.append(int(input("성적을 입력하시오:")))
print(scores)
```

리스트

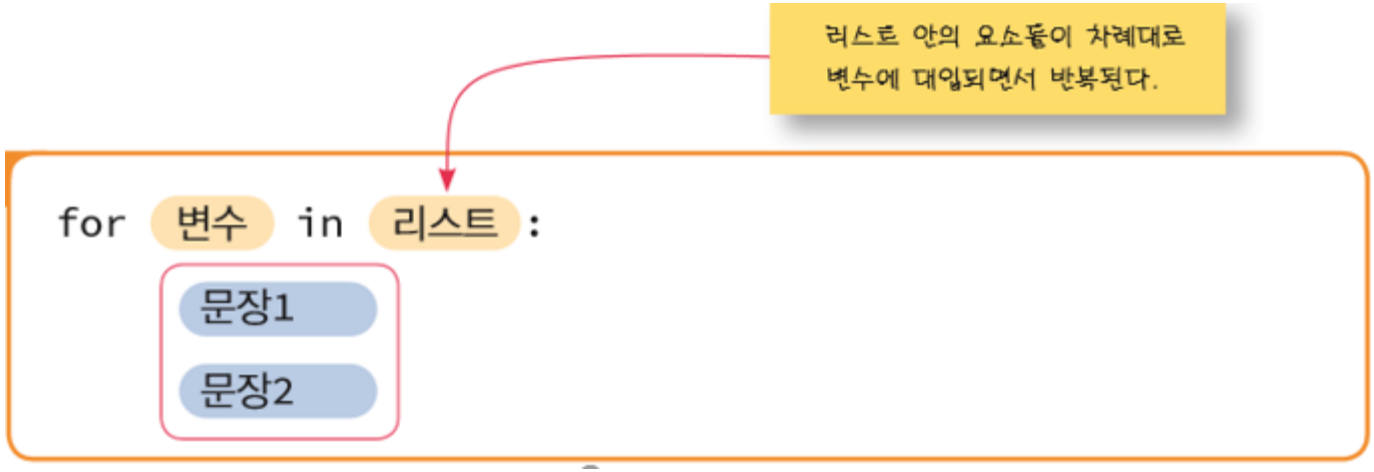
❖ 리스트 요소 접근




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

scores[0] = 80;
scores[1] = scores[0];

scores[i] = 10;                 // i는 정수 변수
scores[i+2] = 20;               // 수식이 인덱스가 된다.

if i >= 0 and i < len(scores) :
    scores[i] = number
```

리스트

❖ 리스트 순회하기




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

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

리스트

❖ list 클래스

```
list1 = list()                  # 공백 리스트 생성
list2 = list("Hello")           # 문자 H, e, l, l, o를 요소로 가지는 리스트 생성
list3 = list(range(0, 5))       # 0, 1, 2, 3, 4를 요소가 가지는 리스트 생성
```

```
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
```

예: 성적 처리

- **사용자로부터 5명의 성적을 입력받아서 리스트에 저장 성적의 평균을 구하고 80점 이상 성적을 받은 학생의 숫자를 계산하여 출력**

```
성적을 입력하시요: 10
성적을 입력하시요: 20
성적을 입력하시요: 60
성적을 입력하시요: 70
성적을 입력하시요: 80
성적 평균은 48.0 입니다.
80점 이상 성적을 받은 학생은 1 명입니다.
```

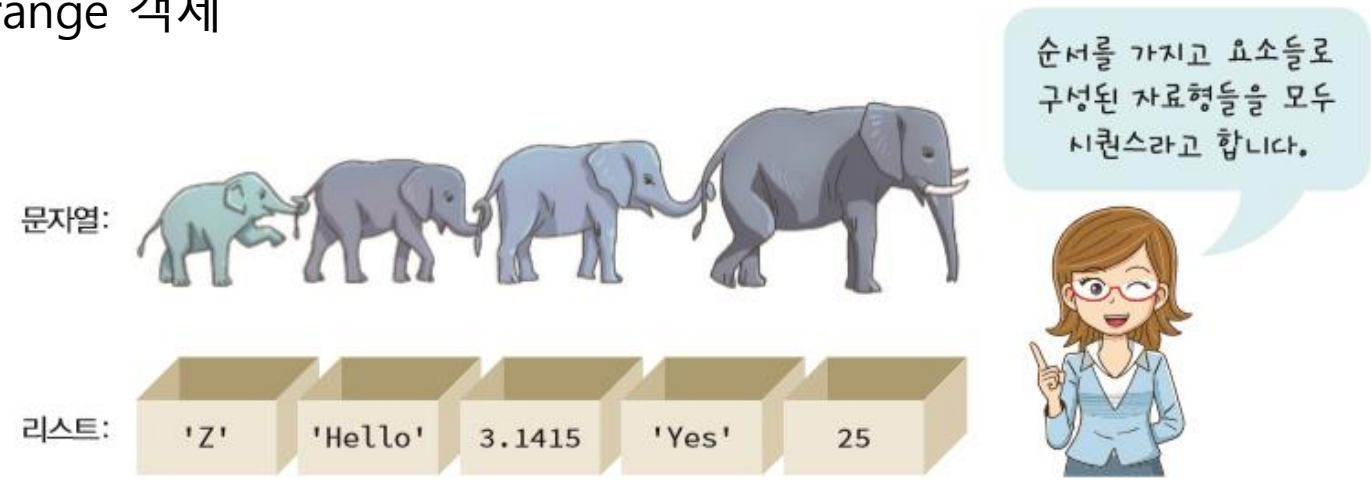
예: 성적 처리

```
STUDENTS = 5

scores = []
scoreSum = 0

for i in range(STUDENTS):
    value = int(input("성적을 입력하시요: "))
    scores.append(value)
    scoreSum += value

scoreAvg = scoreSum / len(scores)

highScoreStudents = 0
for i in range(len(scores)):
    if scores[i] >= 80:
        highScoreStudents += 1

print("성적 평균은", scoreAvg, "입니다.")
print("80점 이상 성적을 받은 학생은 ", highScoreStudents, " 명입니다.")
```

2. 시퀀스 자료형

❖ **시퀀스 (Sequence)**

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



시퀀스 자료형

```
text = "Will is power."
print(text[0], text[3], text[-1])

flist = ["apple", "banana", "tomato", "peach", "pear" ]
print(flist[0], flist[3], flist[-1])
```

```
W l .
apple peach pear
```

시퀀스 자료형

❖ 시퀀스에서 가능한 연산과 함수

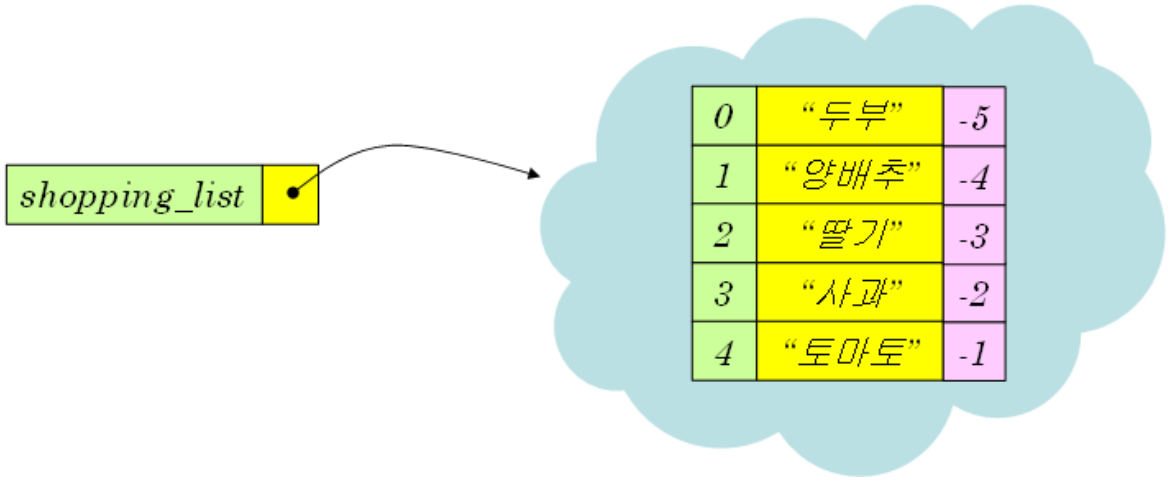
함수나 연산자	설명	예	결과
len()	길이 계산	len([1, 2, 3])	3
+	2개의 시퀀스 연결	[1, 2] + [3, 4, 5]	[1, 2, 3, 4, 5]
*	반복	['Welcome!'] * 3	['Welcome!', 'Welcome!', 'Welcome!']
in	소속	3 in [1, 2, 3]	True
not in	소속하지 않음	5 not in [1, 2, 3]	True
[]	인덱스	myList[1]	myList의 1번째 요소
min()	시퀀스에서 가장 작은 요소	min([1, 2, 3])	1
max()	시퀀스에서 가장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

3. 인덱싱과 슬라이싱

❖ **인덱싱(indexing)**

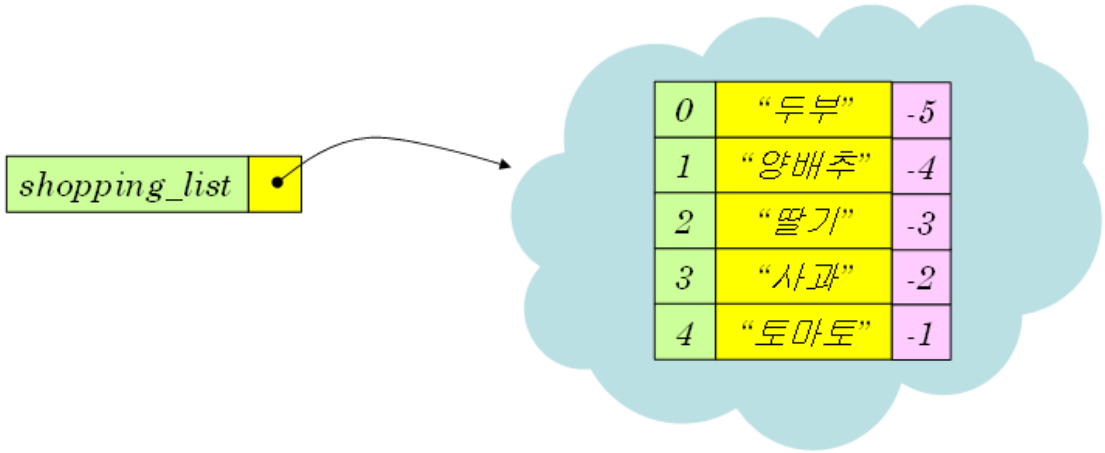
- 리스트에서 하나의 요소를 인덱스 연산자를 통하여 참조(접근)하는 것

```
>>> shopping_list = ["두부", "양배추", "딸기", "사과", "토마토"]
>>> shopping_list[0]
'두부'
```



인덱싱과 슬라이싱

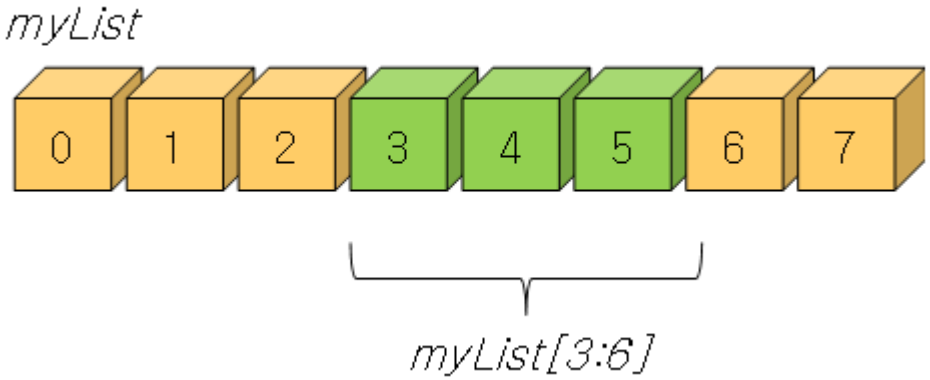
❖ 음수 인덱스



인덱싱과 슬라이싱

❖ **슬라이싱(slicing)**

- 리스트 안에서 범위를 정하여서 원하는 요소들을 선택하는 연산
  - 리스트[start:end]
    - 인덱스 start에 있는 원소부터 end-1의 원소까지 선택
    - start가 생략되면 0으로 간주
    - end가 생략되면 (리스트 길이) - 1로 간주
- 요구된 요소를 포함하는 부분 리스트 반환




```
>>> squares = [0, 1, 4, 9, 16, 25, 36, 49]
>>> squares[3:6]    # 슬라이싱은 새로운 리스트를 반환
[9, 16, 25]
```

인덱싱과 슬라이싱

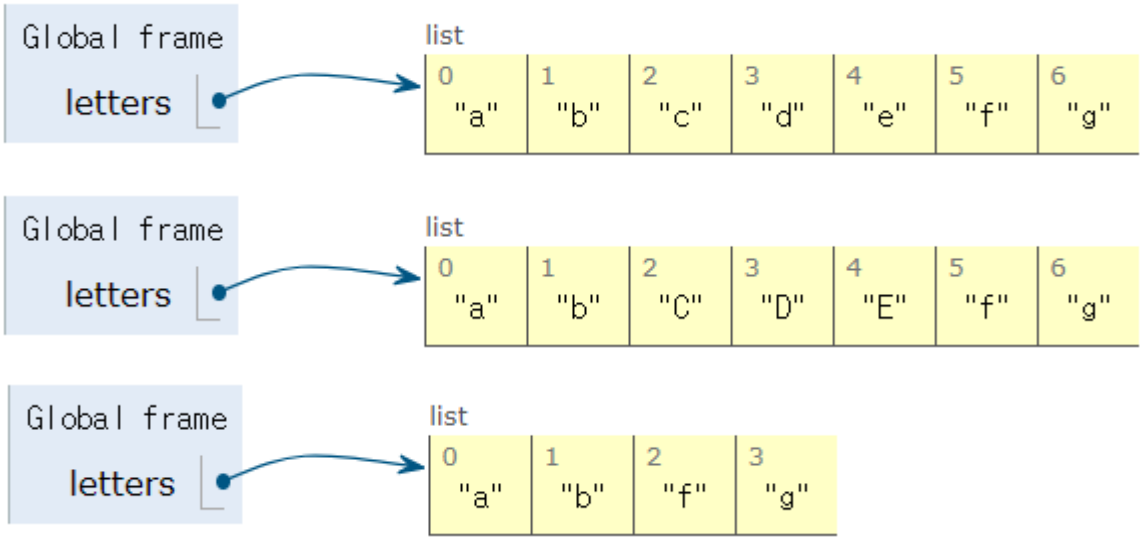
❖ 슬라이싱 범위

```
>>> squares = [0, 1, 4, 9, 16, 25, 36, 49]
>>> squares[:3]
[0, 1, 4]

>>> squares[4:]
[16, 25, 36, 49]

>>> squares[:]
[0, 1, 4, 9, 16, 25, 36, 49]
```

인덱싱과 슬라이싱

❖ 리스트 원소 변경

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
letters[2:5] = ['C', 'D', 'E']
print(letters)
letters[2:5] = []
print(letters)
```



4. 리스트의 기초 연산

❖ **리스트의 기초 연산**

- **+ 연산자** : 두개의 리스트를 합치는 연산자

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

- *** 연산자** : 반복

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

리스트의 기초 연산

❖ len() 함수

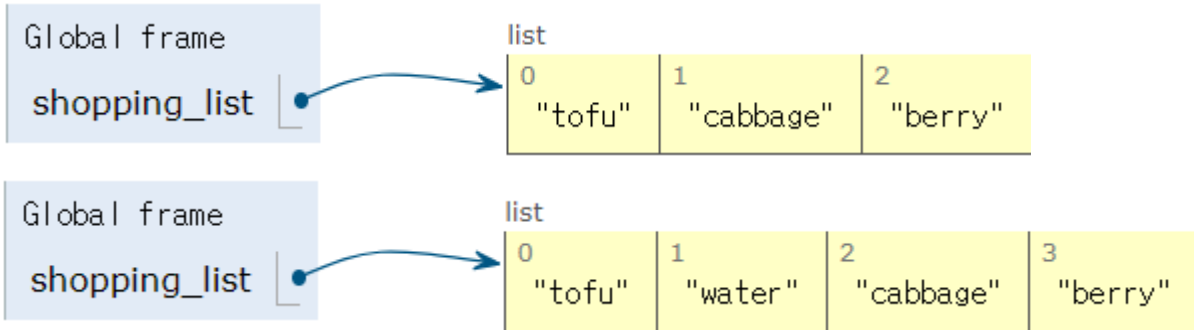
- 리스트의 길이 계산

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

❖ insert() 함수

- 리스트의 특정 위치에 요소 삽입

```
>>> shopping_list = ["tofu", "cabbage", "berry"]
>>> shopping_list.insert(1, "water")
```

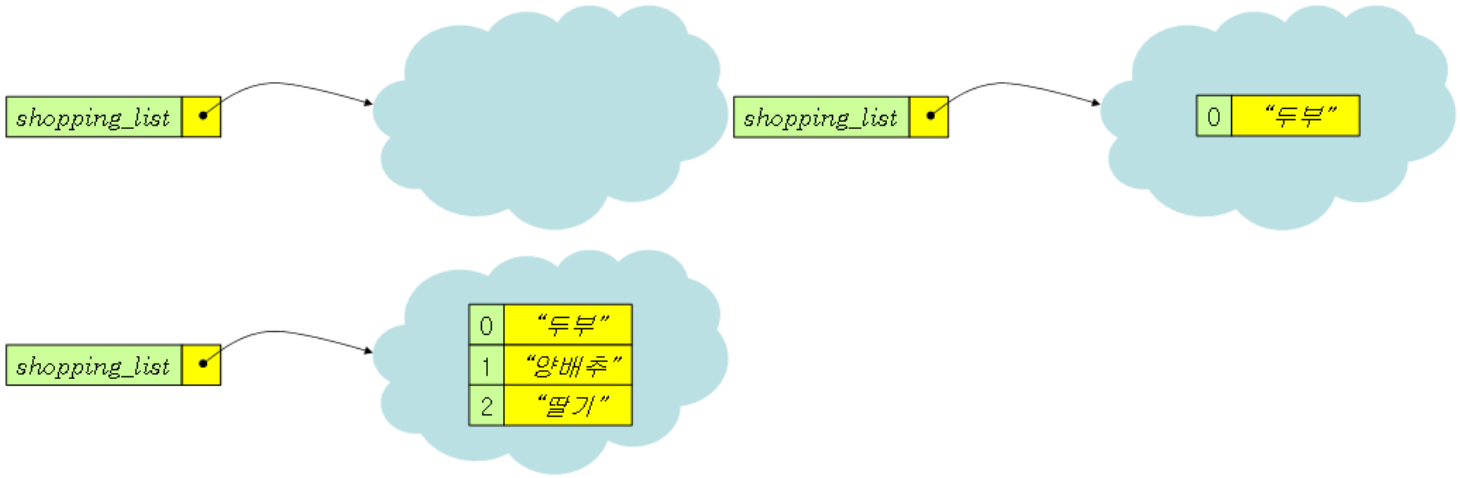


리스트의 기초 연산

❖ **append() 함수**

- 리스트의 끝에 새로운 항목 추가

```
>>> shopping_list = [ ]
>>> shopping_list.append("두부")
>>> shopping_list.append("양배추")
>>> shopping_list.append("딸기")

>>> shopping_list
['두부', '양배추', '딸기']
```



리스트의 기초 연산

❖ **in 연산자**

- 어떤 요소가 리스트에 있는지 확인

```
heroes = [ "스파이더맨", "슈퍼맨", "헐크", "아이언맨", "배트맨" ]
if "배트맨" in heroes :
    print("배트맨은 영웅입니다. ")
```

- **not in : 어떤 요소가 리스트에 없는지 확인**

❖ **index() 함수**

- 어떤 **요소**의 리스트 안에서의 **위치 반환**

```
heroes  = ["스파이더맨", "슈퍼맨", "헐크", "아이언맨", "배트맨" ]
index = heroes.index("슈퍼맨")      # index는 1이 된다.
```

리스트의 기초 연산

❖ **pop() 메소드**

- 특정한 위치에 있는 항목 삭제

```
>>> heroes  = [ "스파이더맨", "슈퍼맨", "헐크", "아이언맨", "배트맨" ]
>>> heroes.pop(1)
'슈퍼맨'
>>> heroes
['스파이더맨', '헐크', '아이언맨', '배트맨']
```

❖ **remove()** 메소드

- 주어진 항목 삭제

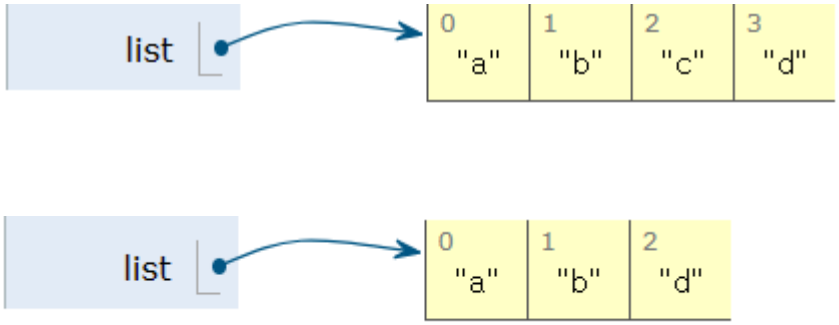
```
>>> heroes  = [ "스파이더맨", "슈퍼맨", "헐크", "아이언맨", "배트맨", "조커" ]
>>> heroes.remove("조커")
>>> heroes
['스파이더맨', '슈퍼맨', '헐크', '아이언맨', '배트맨']
```

리스트의 기초 연산

❖ **del**

- 리스트에서 지정한 요소 제거

```
>>> list = ['a', 'b', 'c', 'd']
>>> del list[2]
```



리스트의 기초 연산

❖ **리스트 일치 검사**

- 비교 연산자 ==, !=, >, < 사용

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

>>> list1 = [ 1, 2, 3 ]
>>> list2 = [ 1, 2 ]
>>> list1 == list2
False

>>> list1 = [ 3, 4, 5 ]
>>> list2 = [ 1, 2, 3 ]
>>> list1 > list2
True

>>> list1 = [ 3, 4, 5 ]
>>> list2 = [ 1 ]
>>> list1 > list2
True
```

리스트의 기초 연산

❖ **리스트 최소값과 최대값 찾기**

- max() 함수와 min() 함수 사용

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

리스트의 기초 연산

❖ **리스트 정렬하기**

1. 리스트 객체의 **sort() 메소드** 사용

```
>>> a = [ 3, 2, 1, 5, 4 ]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

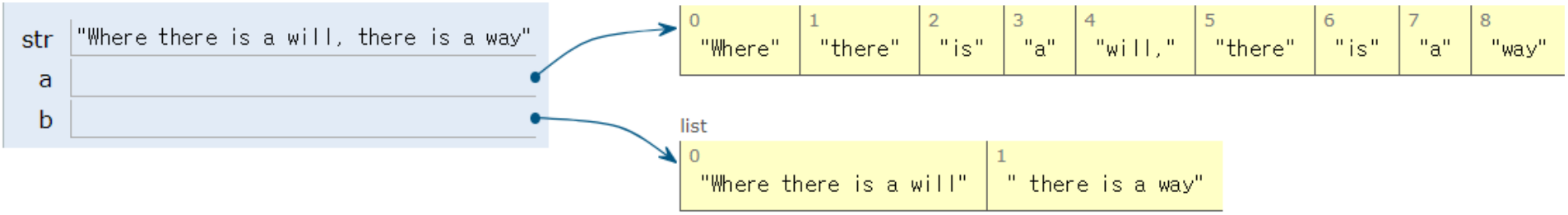
2. **sorted() 내장 함수** 사용

```
>>> a = [ 3, 2, 1, 5, 4 ]
>>> b = sorted(a)
>>> b
[1, 2, 3, 4, 5]
```

리스트의 기초 연산

❖ 문자열에서 리스트 만들기

```
>>>str = "Where there is a will, there is a way“
a = str.split()
b = str.split(",")
```



리스트의 기초 연산

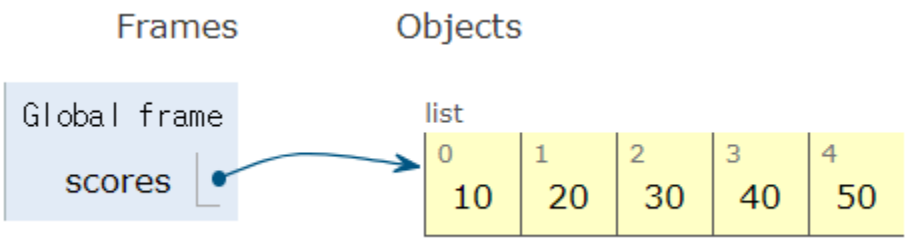
❖ 리스트 연산

연산의 예	설명
`mylist[2]`	인덱스 2에 있는 요소
`mylist[2] = 3`	인덱스 2에 있는 요소를 3으로 설정한다.
`del mylist[2]`	인덱스 2에 있는 요소를 삭제한다.
`len(mylist)`	mylist의 길이를 반환한다.
`"value" in mylist`	"value"가 mylist에 있으면 True
`"value" not in mylist`	"value"가 mylist에 없으면 True
`mylist.sort()`	mylist를 정렬한다.
`mylist.index("value")`	"value"가 발견된 위치를 반환한다.
`mylist.append("value")`	리스트의 끝에 "value"요소를 추가한다.
`mylist.remove("value")`	mylist에서 "value"가 나타나는 위치를 찾아서 삭제한다.

5. 리스트 복사하기

❖ **리스트**

- **리스트 변수**
  - 리스트의 참조값(reference)만 저장
  - **참조값** : 메모리에서 리스트 객체의 위치
- **리스트 객체**
  - 별도의 장소에 저장

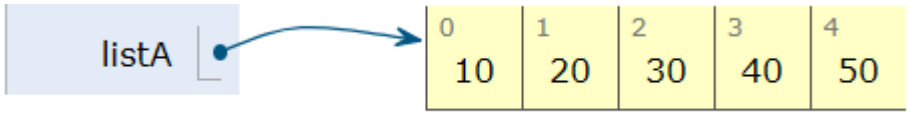


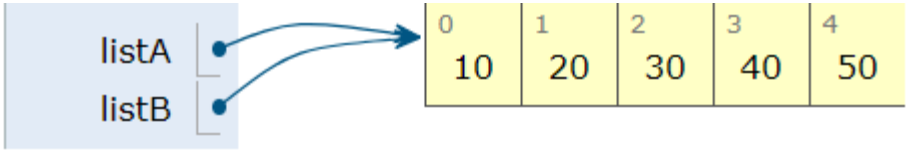
리스트 복사하기

❖ **얕은 복사 (shallow copy)**

- 리스트 객체가 복사되는 것이 아니라 참조값만 복사

```
>>> listA = [ 10, 20, 30, 40, 50 ]
>>> listB = listA
```



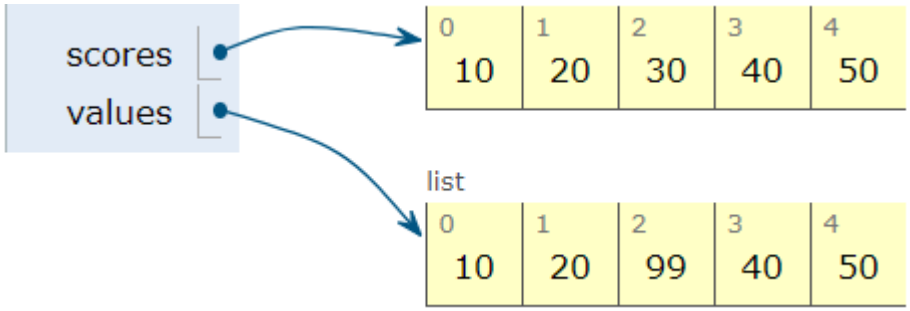


리스트 복사하기

❖ **깊은 복사(deep copy)**

- list() 메소드 사용 : 복사본 생성

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

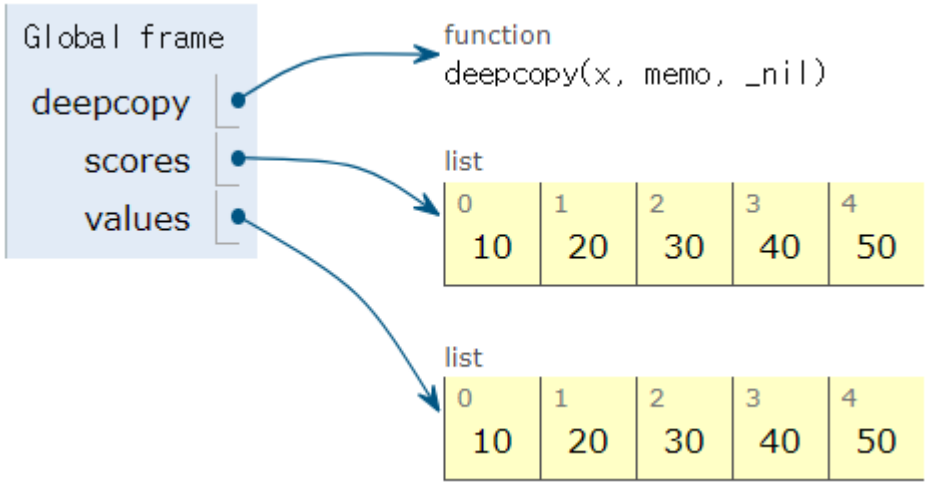


리스트 복사하기

❖ **deepcopy() 메소드 사용**

- 리스트 복사본 생성

```
>>> from copy import deepcopy
>>> scores = [10, 20, 30, 40, 50]
>>> values = deepcopy(scores)
```



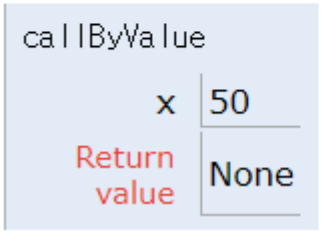
6. 리스트와 함수

- **함수에 대한 인수 전달 방식**
  - **값으로 호출하기 (call-by-value)**
    - 변수의 값이 복사
    - 원본 변수 변경 불가
  - **참조로 호출하기 (call-by-reference)**
    - 변수의 참조 값 전달
    - 원본 변수 변경 가능

리스트와 함수

❖ 값으로 호출하기 (call-by-value)

```
def callByValue(x):
    print("x = ", x)
    x = 50
    print("x = ", x)

y = 30
print("y = ", y)
callByValue(y)
print("y = ", y)
```

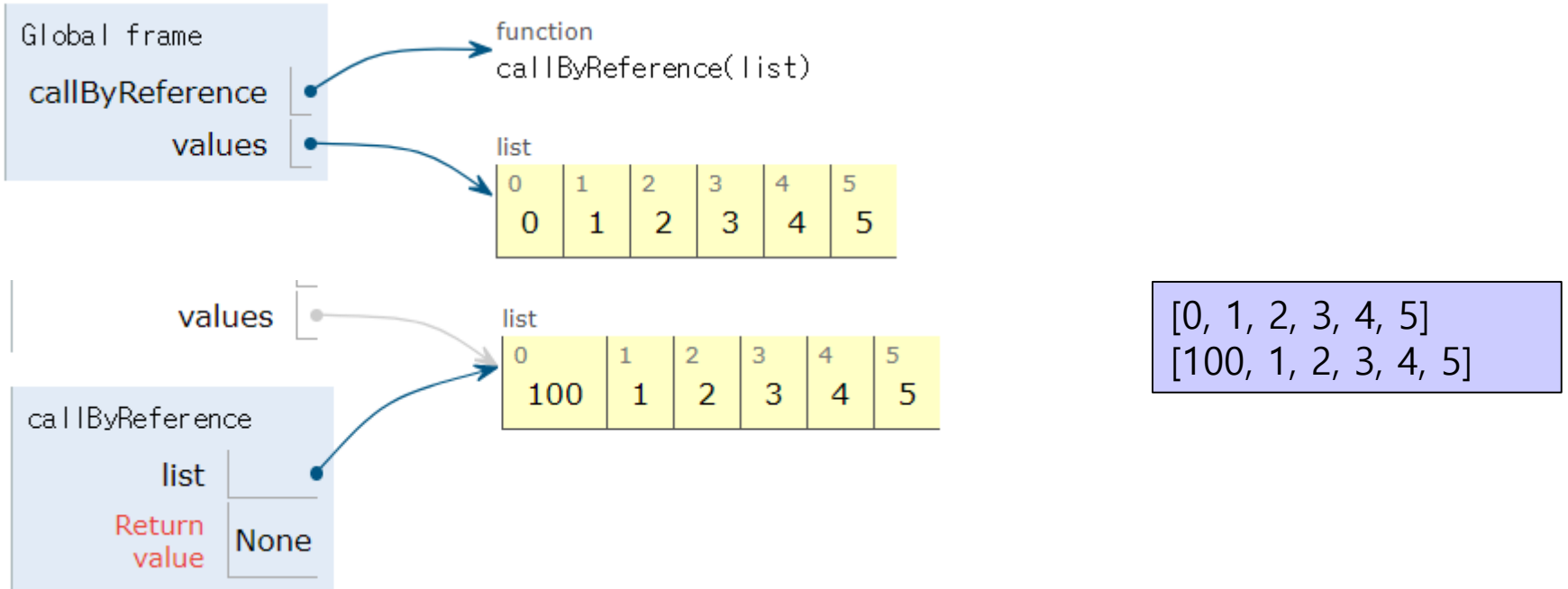



```
y =  30
x =  30
x =  50
y =  30
```

리스트와 함수

❖ 참조로 호출하기 (call-by-reference)

```
def callByReference(list):
    list[0] = 100

values = [0, 1, 2, 3, 4, 5]
print(values)
callByReference(values)
print(values)
```



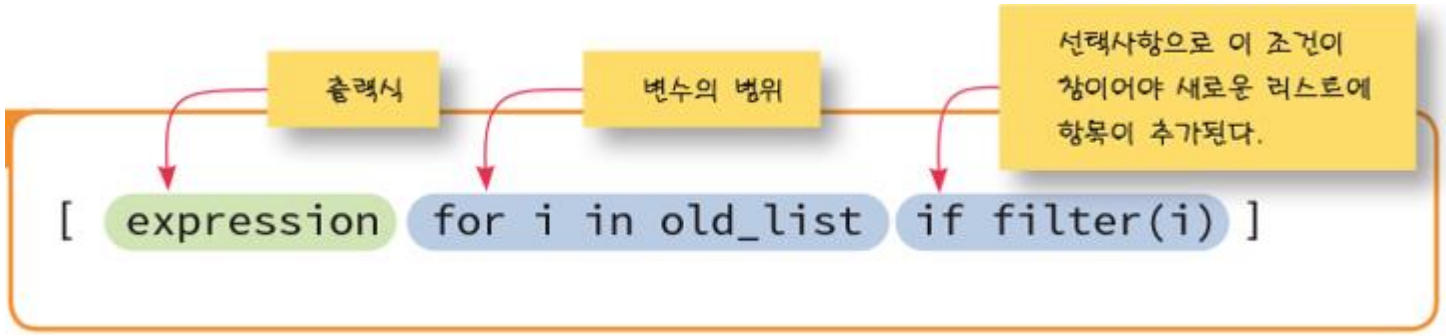

```
[0, 1, 2, 3, 4, 5]
[100, 1, 2, 3, 4, 5]
```

7. 리스트 함축

❖ **리스트 함축(list comprehension)**

- comprehension : 함축, 포함, 내포
- 집합의 정의와 유사
  - 예. 제곱인 정수의 집합 $\{x^2 \mid x \in N\}$

```
>>> [ x**2 for x in range(1,10) ]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```



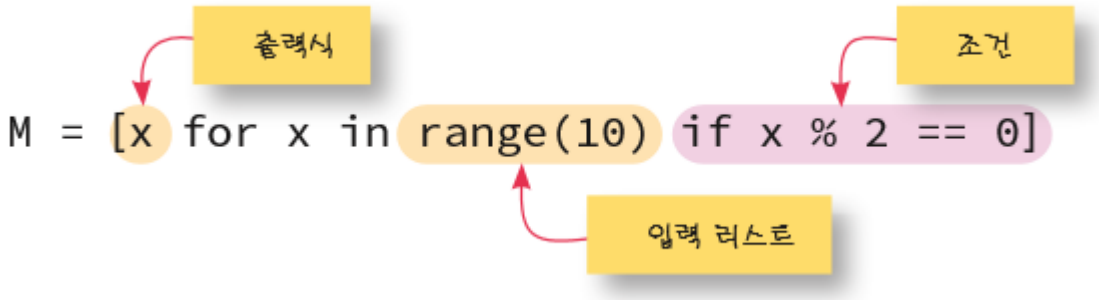
리스트 함축

❖ 리스트 함축의 예

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

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

❖ 조건이 붙는 리스트 함축

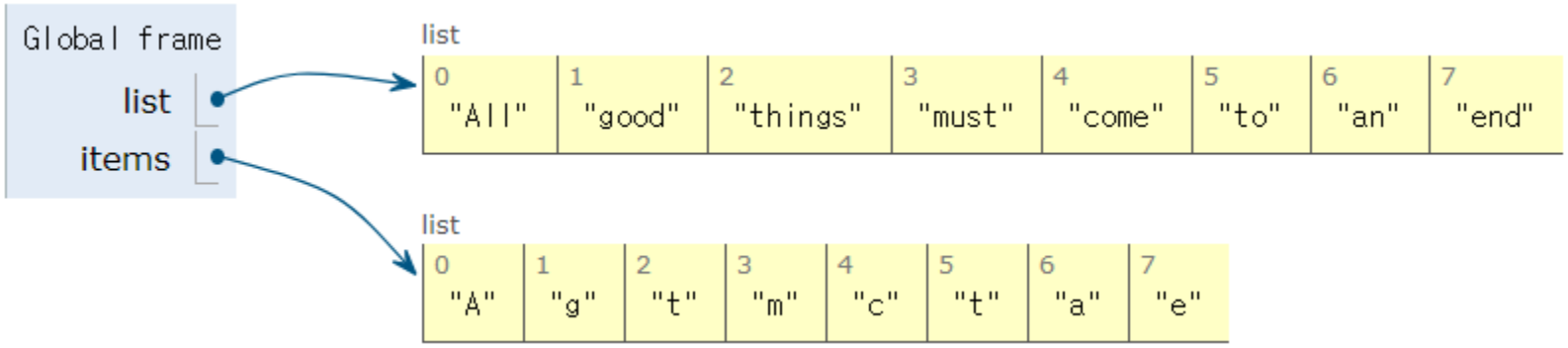



```
[0, 2, 4, 6, 8]
```

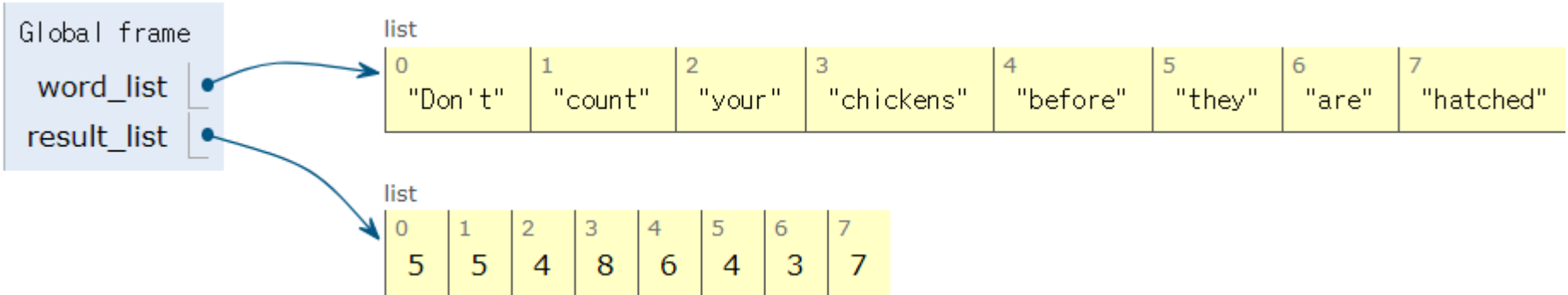
리스트 함축

❖ 문자열 리스트에 대한 리스트 함축

```
list = ["All", "good", "things", "must", "come", "to", "an", "end"]
items = [ word[0] for word in list ]
```



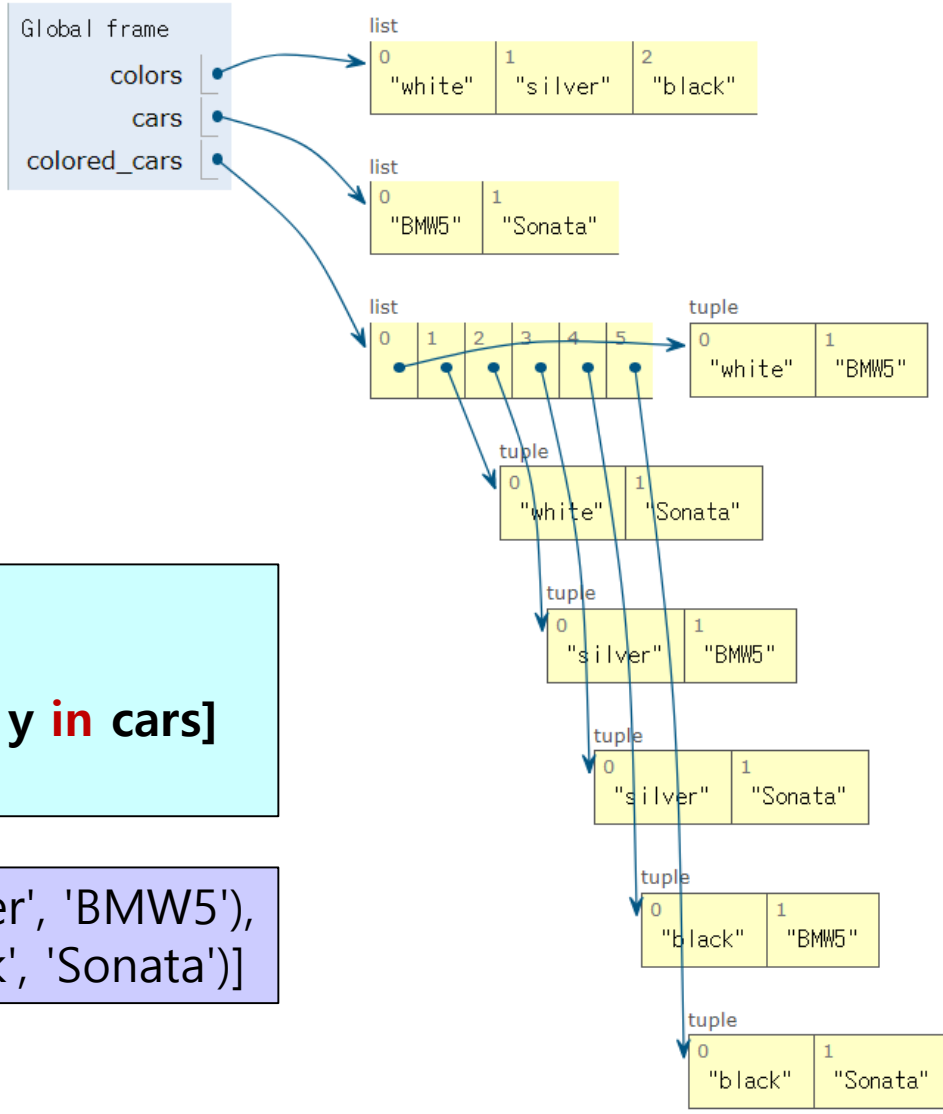

```
word_list = "Don't count your chickens before they are hatched".split()
result_list = [len(w) for w in word_list]
```



리스트 함축

❖ **상호곱(cross product) 형태의 집합**

- 2개 이상의 집합의 상호곱 생성




```
colors = ["white", "silver", "black"]
cars = ["BMW5", "Sonata"]
colored_cars = [(x,y) for x in colors for y in cars]
print(colored_cars)
```

```
[('white', 'BMW5'), ('white', 'Sonata'), ('silver', 'BMW5'),
('silver', 'Sonata'), ('black', 'BMW5'), ('black', 'Sonata')]
```