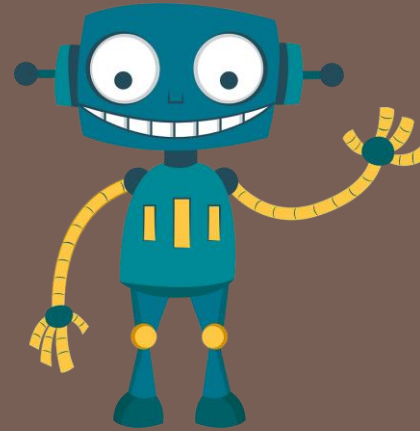
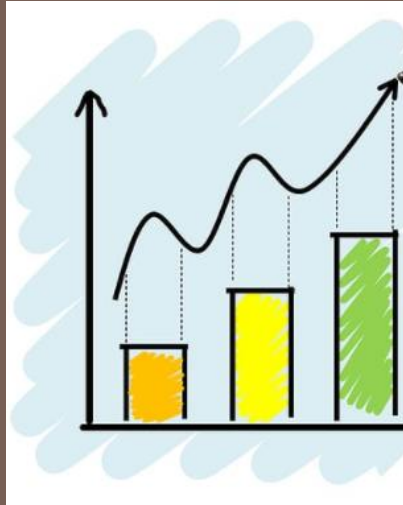


파이썬 익스프레스



6장 파이썬 자료구조 | (리스트)

학습 목표

- 리스트를 생성하고 초기화할 수 있다.
- 리스트의 요소를 참조하는 방법을 학습한다.
- 리스트를 함수에 전달하는 방법을 학습한다.
- 리스트에서 항목 검색, 정렬, 항목 삽입 및 항목 제거와 같은 연산을 할 수 있다.
- 리스트에서 슬라이싱의 개념을 이해하고 활용할 수 있다.
- 리스트 함축을 이해하고 사용할 수 있다.
- 2차원 리스트를 사용할 수 있다.



이번장에서 만들 프로그램

성적을 입력하시요: 10

성적을 입력하시요: 20

성적을 입력하시요: 60

성적을 입력하시요: 70

성적을 입력하시요: 80

성적 평균 = 48.0

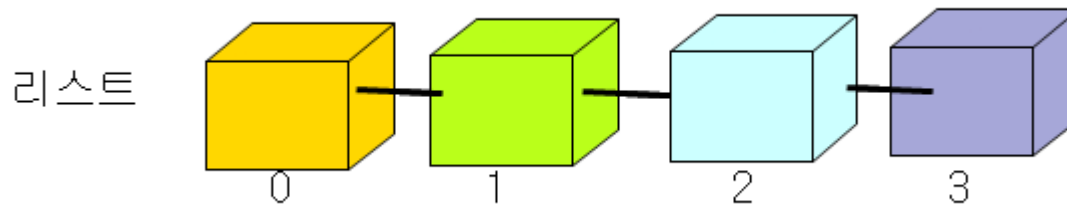
최대점수 = 80

최소점수 = 10

80점 이상 = 1








리스트

- 리스트는 항목(item)들을 저장하는 컨테이너로서 그 안에 항목들이 순서를 가지고 저장된다.
- 리스트는 어떤 타입의 항목라도 저장할 수 있다. 파이썬에서 리스트는 정말 유용하고 많이 사용된다.



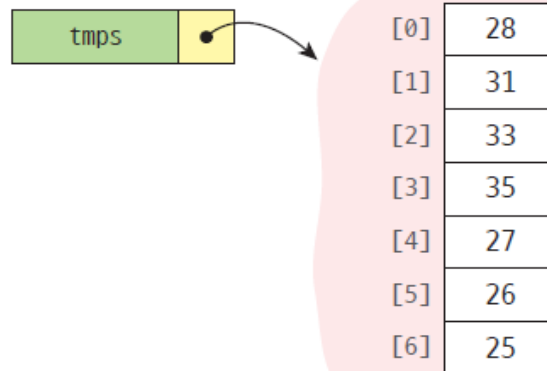
리스트를 사용하는 예

- 지난 1주일 중에서 가장 더운 날을 찾으려고 한다고 하자.

| MON | TUE | WED | THU | FRI | SAT | SUN |
|---|---|---|---|--|---|---|
|  |  |  |  |  |  |  |
| 28 | 31 | 33 | 35 | 27 | 26 | 25 |



```
temps = [28, 31, 33, 35, 27, 26, 25]
```



리스트 만들기

Syntax: 리스트

형식 리스트_이름 = [요소1, 요소2, ...]

예 temps = [28, 31, 33, 35, 27, 26, 25]
e = temps[3]

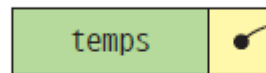
초기값을 가진 리스트를 생성한다.

리스트의 이름

대괄호를 사용하여 요소에 접근한다.

리스트의 항목 저장하기

```
temps = [28, 31, 33, 35, 27, 26, 25]
```

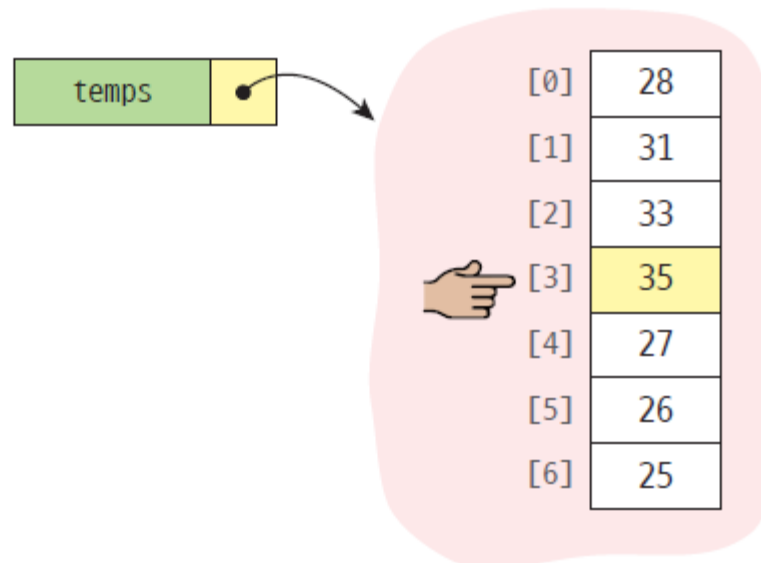


| | |
|-----|----|
| [0] | 28 |
| [1] | 31 |
| [2] | 33 |
| [3] | 35 |
| [4] | 27 |
| [5] | 26 |
| [6] | 25 |

리스트 인덱스

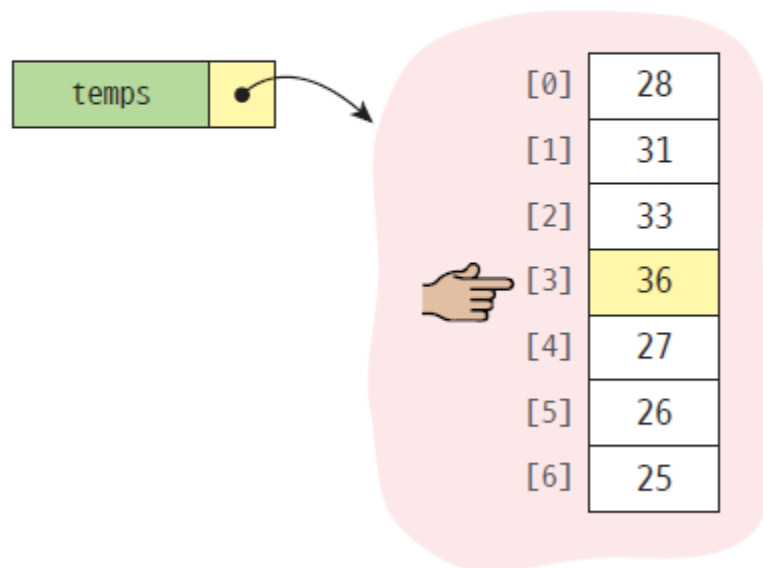
- 인덱스(index)란 리스트에서의 항목의 위치(번호)이다.
- 0부터 시작한다.

```
temps = [28, 31, 33, 35, 27, 26, 25]
```



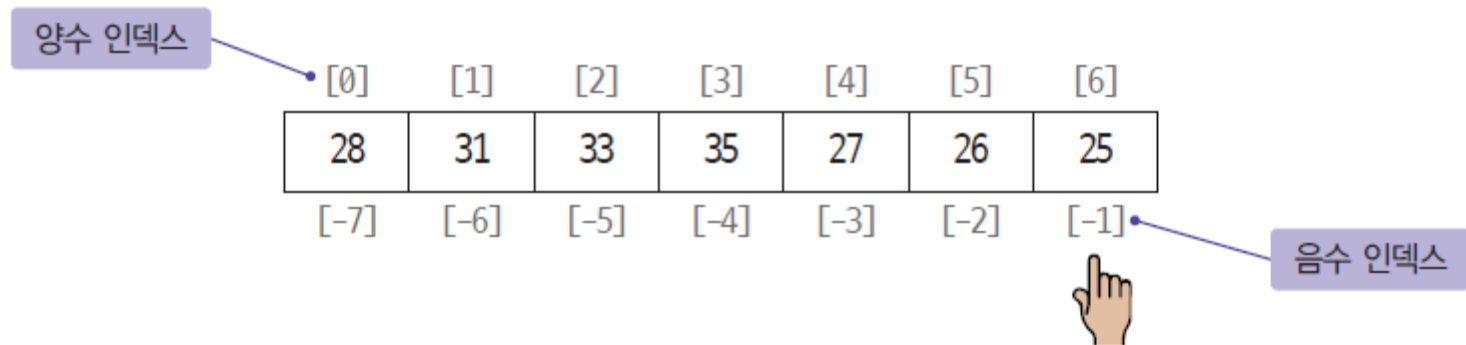
리스트 요소 변경

`temps[3] = 36`



음수 인덱스

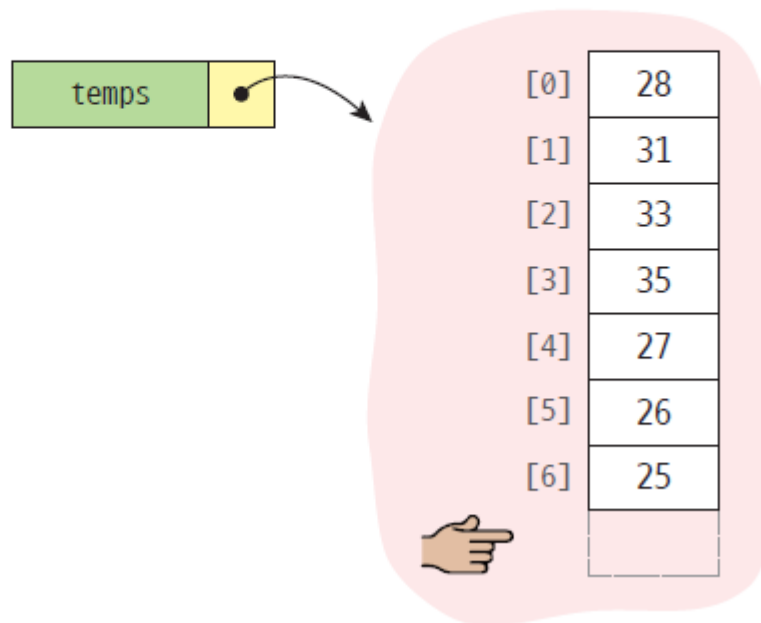
- 음수 인덱스는 리스트의 끝에서부터 매겨진다.



인덱스 오류

- 인덱스를 사용할 때는 인덱스가 적정한 범위에 있는지를 항상 신경 써야 한다.

```
temps[7] = 29      # 오류!
```



리스트 방문

```
temps =[28,31,33,35,27,26,25]
for i in range(len(temps)):
    print(temps[i], end=', ')
```

28, 31, 33, 35, 27, 26, 25,

```
temps =[28,31,33,35,27,26,25]
for element in temps:
    print(element, end=', ')
```

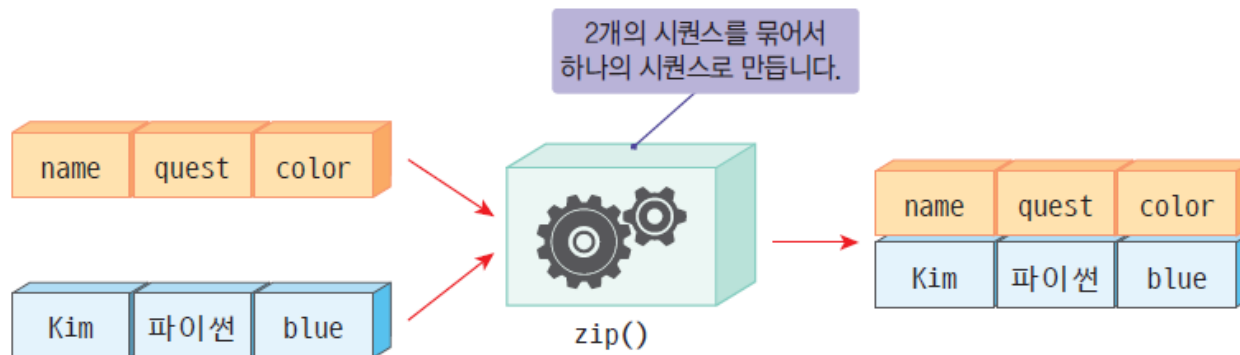
28, 31, 33, 35, 27, 26, 25,

zip() 함수

- zip() 함수는 2개의 리스트를 받아서 항목 2개를 묶어서 제공한다.

```
questions = ['name', 'quest', 'color']  
answers = ['Kim', '파이썬', 'blue']  
for q, a in zip(questions, answers):  
    print(f"What is your {q}? It is {a}")
```

What is your name? It is Kim
What is your quest? It is 파이썬
What is your color? It is blue



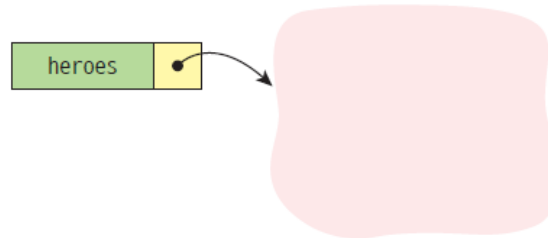
append()

```
heroes = []  
heroes.append("아이언맨")  
heroes.append("토르")  
print(heroes)
```

공백 리스트를 생성한다.
리스트에 "아이언맨"을 추가한다.
리스트에 "토르"를 추가한다.

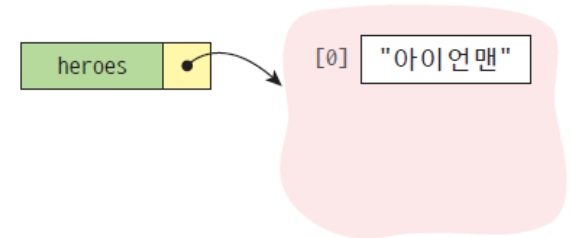
['아이언맨', '토르']

① heroes=[]



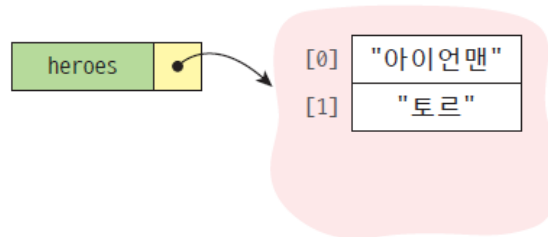
새로운 공백 리스트를 생성한다.

② heroes.append("아이언맨")



리스트에 하나의 항목을 추가한다.

③ heroes.append("토르")

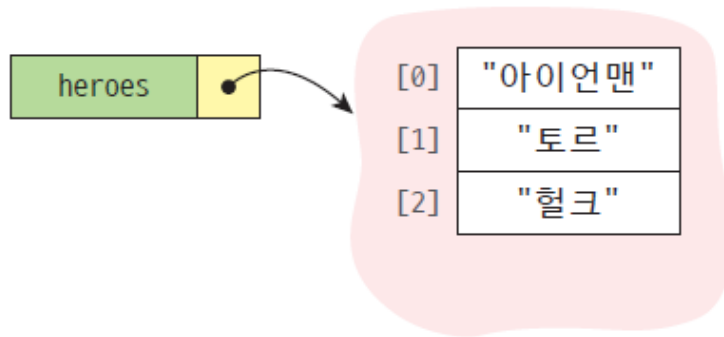


리스트에 하나의 항목을 추가한다.

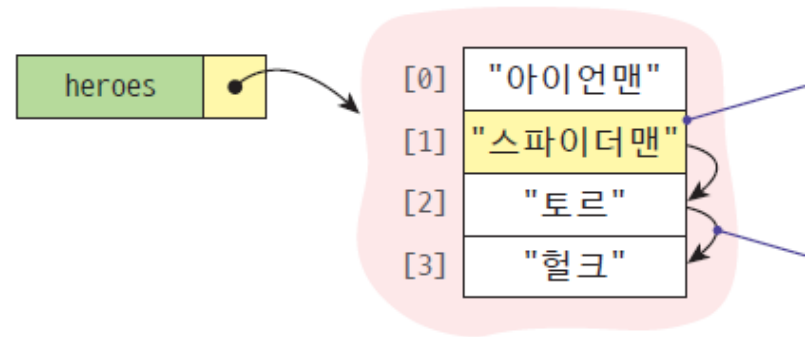
insert()

고재 오타!

heroes=["아이언맨", "토르", 헐크"]

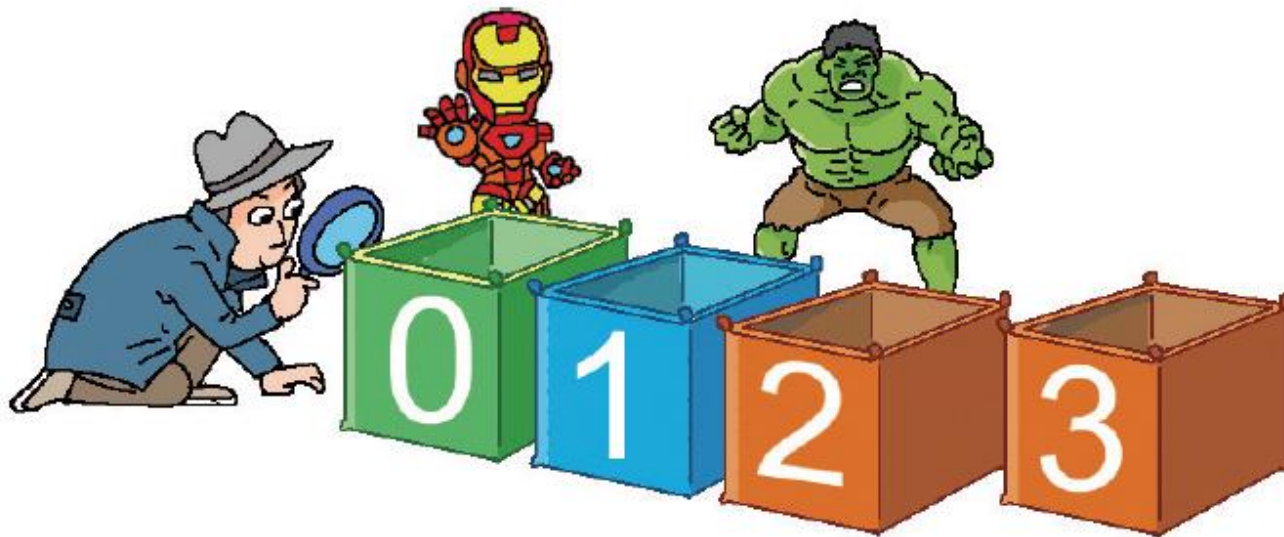


heroes.insert(1, "스파이더맨")



리스트 탐색하기

```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치", "헐크" ]  
n = heroes.index("헐크")           # n은 2가 된다.
```



탐색할 때 오류를 발생시키지 않으려면

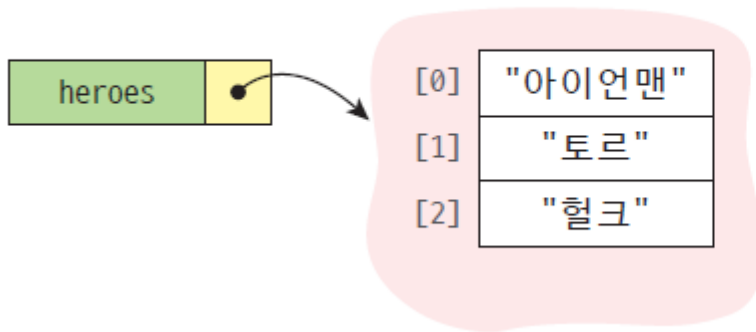
```
if "헐크" in heroes:  
    print(heroes.index("헐크"))
```

```
heroes = [ "아이언맨", "토르", "헐크", "스칼렛 위치", "헐크" ]  
n = heroes.index("헐크", 3)          # n은 4가 된다.
```

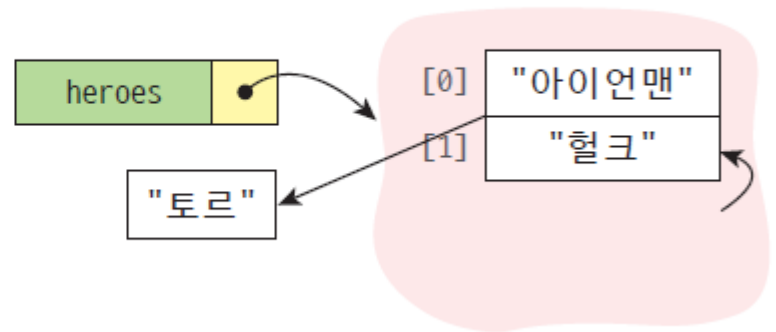
요소 삭제하기

- 항목이 저장된 위치를 알고 있다면 `pop(i)`을 사용한다.
- 항목의 값만 알고 있다면 `remove(value)`를 사용한다.

`heroes=["아이언맨", "토르", "헐크"]`



`heroes.pop(1)`



remove()

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

만약 삭제하고자 하는 항목이 없다면 오류(예외)가 발생된다.

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

확인후 삭제한다.

리스트 연산 정리

| 연산의 예 | 설명 |
|-------------------------------------|--|
| <code>mylist[2]</code> | 인덱스 2에 있는 요소 |
| <code>mylist[2] = 3</code> | 인덱스 2에 있는 요소를 3으로 설정한다. |
| <code>mylist.pop(2)</code> | 인덱스 2에 있는 요소를 삭제한다. |
| <code>len(mylist)</code> | <code>mylist</code> 의 길이를 반환한다. |
| <code>"value" in mylist</code> | <code>"value"</code> 가 <code>mylist</code> 에 있으면 <code>True</code> |
| <code>"value" not in mylist</code> | <code>"value"</code> 가 <code>mylist</code> 에 없으면 <code>True</code> |
| <code>mylist.sort()</code> | <code>mylist</code> 를 정렬한다. |
| <code>mylist.index("value")</code> | <code>"value"</code> 가 발견된 위치를 반환한다. |
| <code>mylist.append("value")</code> | 리스트의 끝에 <code>"value"</code> 요소를 추가한다. |
| <code>mylist.remove("value")</code> | <code>mylist</code> 에서 <code>"value"</code> 가 나타나는 위치를 찾아서 삭제한다. |

리스트 항목의 최대값과 최소값

```
values = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
```

```
min(values) # 1
```

```
max(values) # 10
```

max()와 min()은 내장 함수로서 거의 모든 객체에
사용 가능

리스트 정렬 #1

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

```
heroes = ['아이언맨', '헐크', '토르']  
heroes.sort()           # ['아이언맨', '토르', '헐크']
```

```
a = [3, 2, 1, 5, 4]  
a.sort(reverse=True)    # [5, 4, 3, 2, 1]
```

역순으로 정렬할때
reverse=True

리스트 정렬 #2

```
numbers =[10,3,7,1,9,4,2,8,5,6]  
ascending_numbers =sorted(numbers)  
print( ascending_numbers )
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

내장 함수

리스트 메소드 정리

| 메소드 | 설명 |
|-----------|---------------------------|
| append() | 요소를 리스트의 끝에 추가한다. |
| extend() | 리스트의 모든 요소를 다른 리스트에 추가한다. |
| insert() | 지정된 위치에 항목을 삽입한다. |
| remove() | 리스트에서 항목을 삭제한다. |
| pop() | 지정된 위치에서 요소를 삭제하여 반환한다. |
| clear() | 리스트로부터 모든 항목을 삭제한다. |
| index() | 일치되는 항목의 인덱스를 반환한다. |
| count() | 인수로 전달된 항목의 개수를 반환한다. |
| sort() | 오름차순으로 리스트 안의 항목을 정렬한다. |
| reverse() | 리스트 안의 항목의 순서를 반대로 한다. |
| copy() | 리스트의 복사본을 반환한다. |

리스트에서 사용할 수 있는 내장 함수

| 함수 | 설명 |
|--------------|---|
| round() | 주어진 자리수대로 반올림한 값을 반환한다. |
| reduce() | 특정한 함수를 리스트 안의 모든 요소에 적용하여 결과값을 저장하고 최종 합계값만을 반환한다. |
| sum() | 리스트 안의 숫자들을 모두 더한다. |
| ord() | 유니코드 문자의 코드값을 반환한다. |
| cmp() | 첫 번째 리스트가 두 번째 보다 크면 1을 반환한다. |
| max() | 리스트의 최대값을 반환한다. |
| min() | 리스트의 최소값을 반환한다. |
| all() | 리스트의 모든 요소가 참이면 참을 반환한다. |
| any() | 리스트 안의 한 요소라도 참이면 참을 반환한다. |
| len() | 리스트의 길이를 반환한다. |
| enumerate() | 리스트의 요소들을 하나씩 반환하는 객체를 생성한다. |
| accumulate() | 특정한 함수를 리스트의 요소에 적용한 결과를 저장하는 리스트를 반환한다. |
| filter() | 리스트의 각 요소가 참인지 아닌지를 검사한다. |
| map() | 특정한 함수를 리스트의 각 요소에 적용하고 결과를 담은 리스트를 반환한다. |

내장 함수 예

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

```
print("합=",sum(numbers))
```

항목의 합계를 계산한다.

```
print("최대값=",max(numbers))
```

가장 큰 항목을 반환한다.

```
print("최소값=",min(numbers))
```

가장 작은 항목을 반환한다

```
합= 150
```

```
최대값= 50
```

```
최소값= 10
```

내장 함수

리스트에서 랜덤으로 선택하기

```
import random

numberList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("랜덤하게 선택한 항목=", random.choice(numberList))
```

랜덤하게 선택한 항목= 6

```
import random
movie_list = ["Citizen Kane", "Singin' in the Rain", "Modern Times",
              "Casablanca", "City Lights"]

item = random.choice(movie_list)
print("랜덤하게 선택한 항목=", item)
```

랜덤하게 선택한 항목= Citizen Kane

Lab: 성적 처리 프로그램

- 학생들의 성적을 사용자로부터 입력받아서 리스트에 저장한다. 성적의 평균을 구하고 최대점수, 최소점수, 80점 이상 성적을 받은 학생의 숫자를 계산하여 출력해보자.

성적을 입력하시요: 10
성적을 입력하시요: 20
성적을 입력하시요: 60
성적을 입력하시요: 70
성적을 입력하시요: 80
성적 평균 = 48.0
최대점수 = 80
최소점수 = 10
80점 이상 = 1



Solution:

```
STUDENTS = 5
lst = []
count=0

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

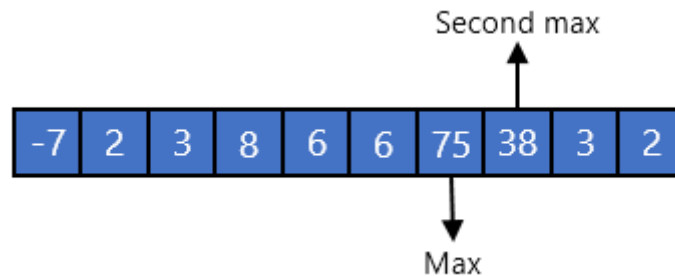
print("\n성적 평균=", sum(lst) / len(lst))
print("최대점수=", max(lst))
print("최소점수=", min(lst))

for score in lst:
    if score >= 80:
        count += 1
print("80점 이상=", count)
```

Lab: 리스트에서 2번째로 큰 수 찾기

- 정수들이 저장된 리스트에서 두 번째로 큰 수를 찾아보자.

두 번째로 큰 수 = 15



Solution:

```
list1 = [1, 2, 3, 4, 15, 99]
```

```
# 리스트를 정렬한다.
```

```
list1.sort()
```

```
# 뒤에서 두 번째 요소를 출력한다.
```

```
print("두 번째로 큰 수=", list1[-2])
```

```
list1 = [1, 2, 3, 4, 15, 99]
```

```
# 제일 큰 수를 삭제한다.
```

```
list1.remove(max(list1))
```

```
# 그 다음으로 큰 수를 출력한다. 리스트는 변경되었다.
```

```
print("두 번째로 큰 수=", max(list1))
```

Lab: 콘테스트 평가

- 심판들의 점수가 리스트에 저장되어 있다고 가정하고 최소값과 최대값을 리스트에서 제거하는 프로그램을 작성해보자.

제거전 [10.0, 9.0, 8.3, 7.1, 3.0, 9.0]

제거후 [9.0, 8.3, 7.1, 9.0]



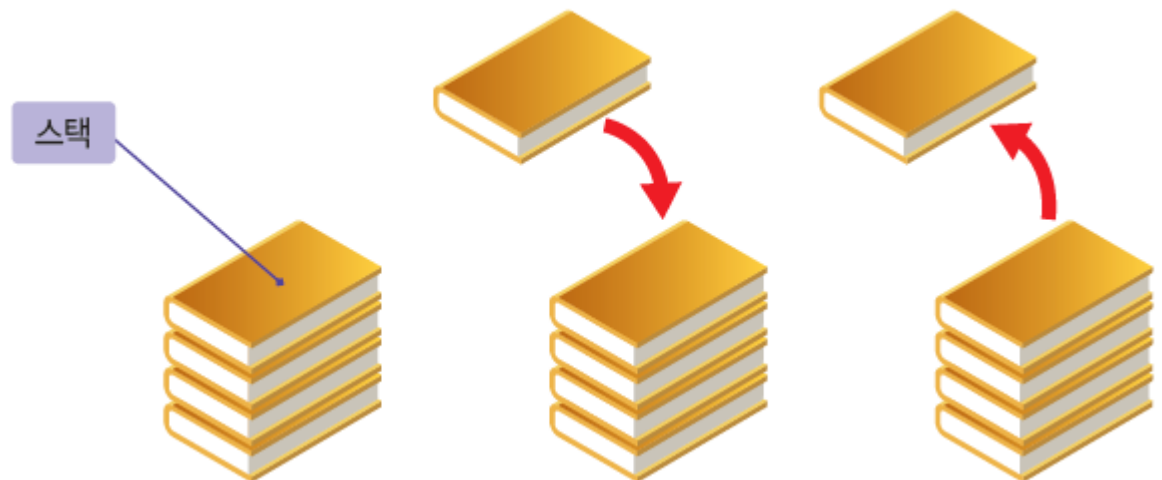
Solution:

```
scores = [10.0, 9.0, 8.3, 7.1, 3.0, 9.0]
print("제거전", scores)
scores.remove(max(scores))
scores.remove(min(scores))
print("제거후", scores)
```

Lab: 리스트로 스택 흉내내기

- 스택은 데이터 구조 중의 하나로서 데이터를 추가한 순서와 삭제하는 순서가 완전히 반대인 데이터 구조이다. 파이썬에는 내장 스택 유형이 없지만, 스택을 리스트로 구현할 수 있다.

```
과일을 입력하시오: apple  
과일을 입력하시오: orange  
과일을 입력하시오: grape  
grape  
orange  
apple
```



Solution:

```
stack = []

for i in range(3) :
    f = input("과일을 입력하시오: ")
    stack.append(f)

for i in range(3) :
    print( stack.pop() )
```

Lab: 친구 관리 프로그램

- 파이썬을 이용하여 친구들을 관리하는 프로그램을 작성하여 보자.
친구 관리 프로그램은 다음과 같은 메뉴를 가져야 한다.

```
-----  
1. 친구 리스트 출력  
2. 친구추가  
3. 친구삭제  
4. 이름변경  
9. 종료  
메뉴를 선택하시오: 2  
이름을 입력하시오: 홍길동
```

```
-----  
1. 친구 리스트 출력  
2. 친구추가  
3. 친구삭제  
4. 이름변경  
9. 종료  
메뉴를 선택하시오: 1  
['홍길동']
```

Solution.

```
menu = 0
friends = []
while menu != 9:
    print("-----")
    print("1. 친구 리스트 출력")
    print("2. 친구추가")
    print("3. 친구삭제")
    print("4. 이름변경")
    print("9. 종료")
    menu = int(input("메뉴를 선택하시오: "))
    if menu == 1:
        print(friends)
    elif menu == 2:
        name = input("이름을 입력하시오: ")
        friends.append(name)
    elif menu == 3:
        del_name = input("삭제하고 싶은 이름을 입력하시오: ")
        if del_name in friends:
            friends.remove(del_name)
        else:
            print("이름이 발견되지 않음")
```

Solution:

```
elif menu == 4:
    old_name = input("변경하고 싶은 이름을 입력하시오: ")
    if old_name in friends:
        index = friend.index(old_name)
        new_name = input("새로운 이름을 입력하시오: ")
        friends[index] = new_name
    else:
        print("이름이 발견되지 않음")
```

리스트 합병과 복제

```
heroes1 = [ "아이언맨", "토르" ]  
heroes2 = [ "헐크", "스칼렛 위치" ]  
avengers = heroes1 + heroes2
```

avengers는 ['아이언맨', '토르', '헐크', '스칼렛 위치']가 된다.

```
numbers = [ 1, 2, 3, 4 ] * 3           # 리스트는 [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]이다.
```

리스트 비교

```
list1 = [ 1, 2, 3 ]  
list2 = [ 1, 2, 3 ]  
print(list1 == list2)      # True
```

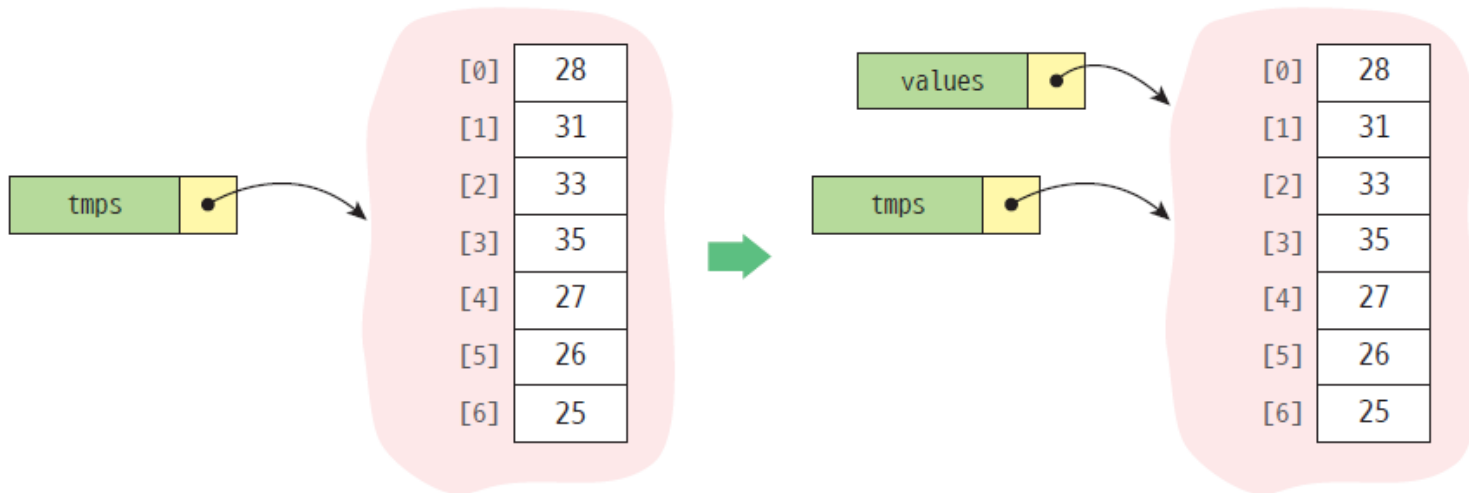
```
list1 = [ 3, 4, 5 ]  
list2 = [ 1, 2, 3 ]  
print(list1 > list2)      # True
```

참고:

리스트와 리스트를 비교할 때,
요소들이 기초 자료형(정수, 실수,
문자열)이 아니고 사용자가 정의한
객체인 경우에는, 사용자가 객체 안에
== 연산과 != 연산을 정의하여야
한다.

리스트 복사하기

```
temps = [28, 31, 33, 35, 27, 26, 25]  
values = temps
```



리스트 복사하기

```
temps = [28, 31, 33, 35, 27, 26, 25]
values = temps
```

```
print(temps)
values[3] = 39
print(temps)
```

temps 리스트 출력
values 리스트 변경
temps 리스트가 변경되었다.

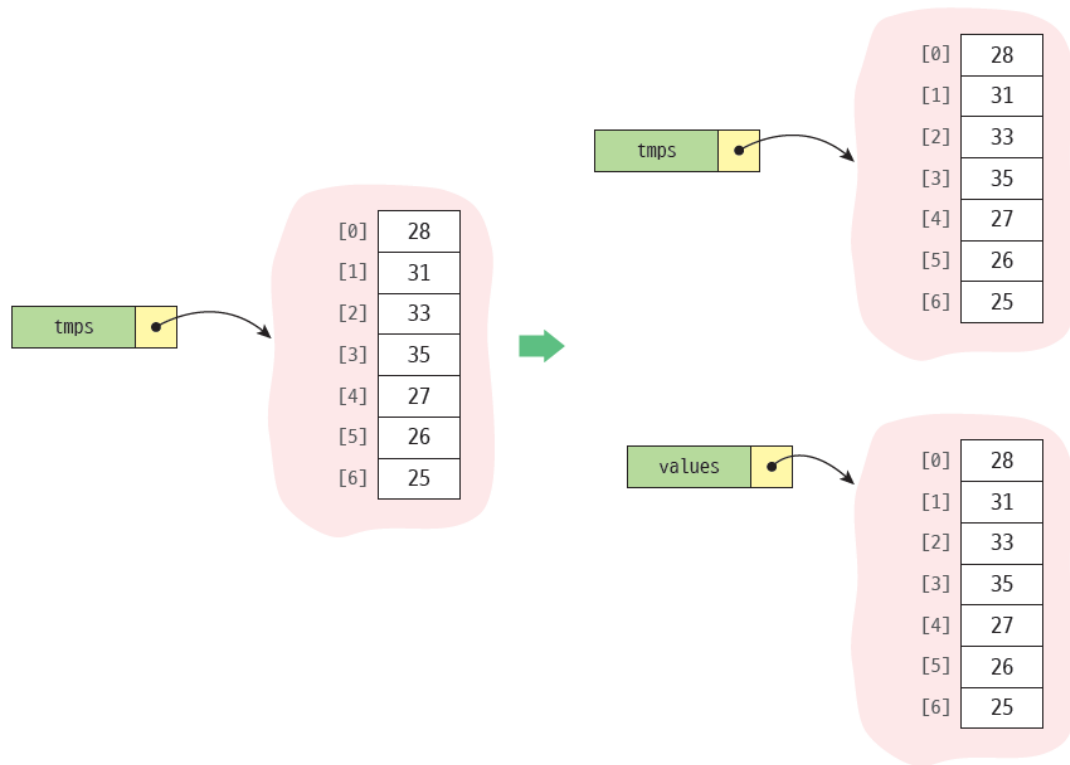
```
[28, 31, 33, 35, 27, 26, 25]
[28, 31, 33, 39, 27, 26, 25]
```

이것을 얕은 복사라고 함.

리스트 기본 보사

list()는 리스트 객체의 생성자임,
객체를 생성하고 초기화하는 함수임
다른 객체들을 받아서 리스트로
변환함

```
temps = [28, 31, 33, 35, 27, 26, 25]  
values = list(temps)
```

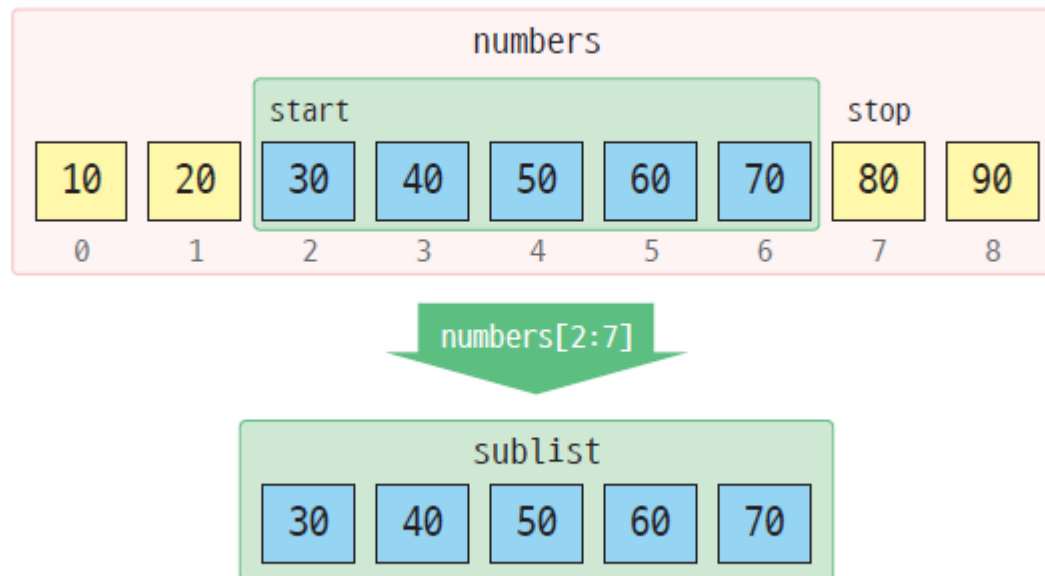


슬라이싱

Syntax: 슬라이싱 #1

형식 리스트[start : stop]

예 numbers = [10, 20, 30, 40, 50, 60, 70, 80, 90]
sublist = numbers[2:7]



시작과 끝 인덱스 생략이 가능하다.

```
numbers[:3] # [10, 20, 30]
```

```
numbers[3:] # [40, 50, 60, 70, 80, 90]
```

```
numbers[:] # [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

```
new_numbers = numbers[:]
```

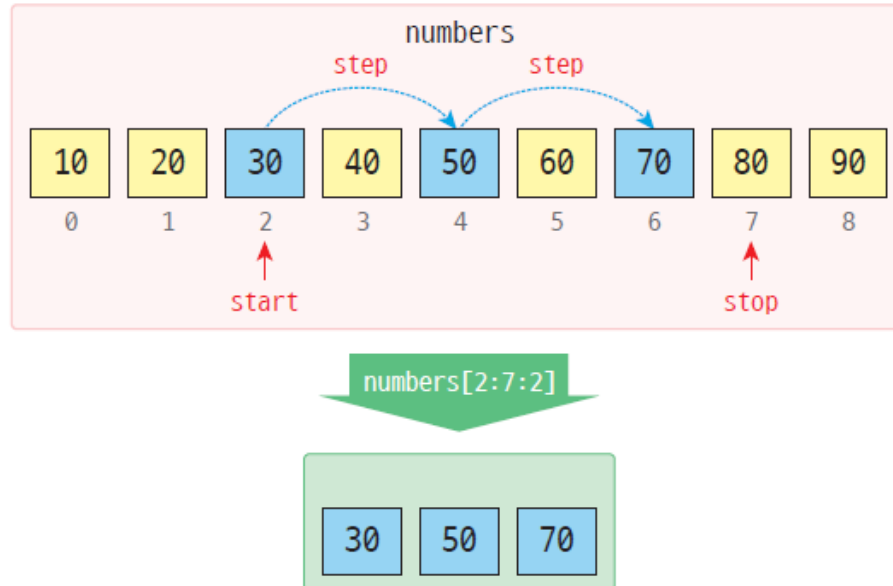
깊은 복사가 된다.

고급 슬라이싱

Syntax: 슬라이싱 #2

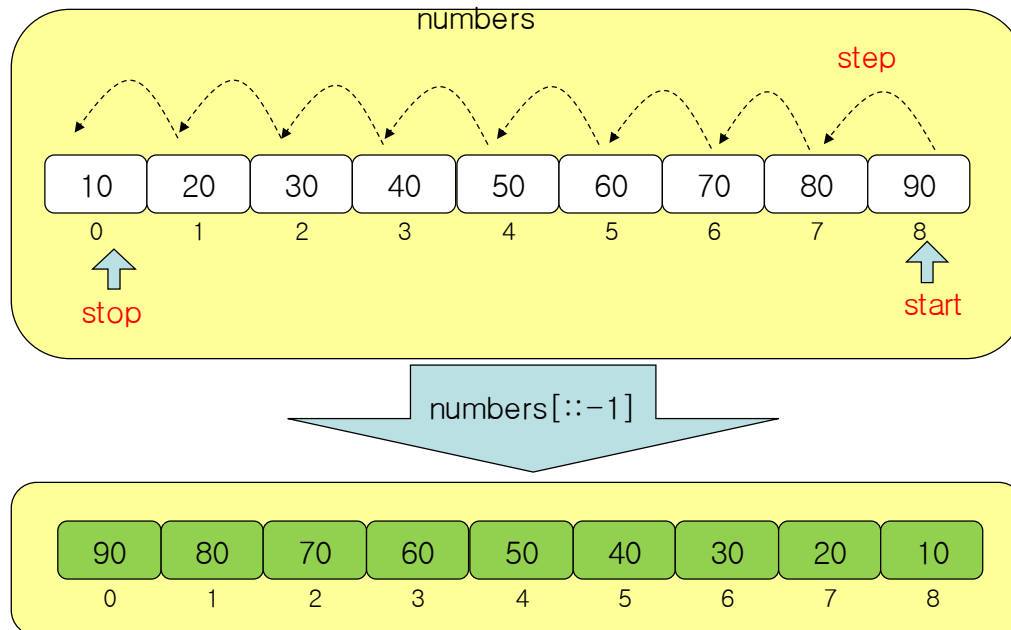
형식 리스트[start : stop : step]

예 numbers = [10, 20, 30, 40, 50, 60, 70, 80, 90]
sublist = numbers[2:7:2]



리스트를 역순으로 만드는 방법

```
>>> numbers = [ 10, 20, 30, 40, 50, 60, 70, 80, 90 ]  
>>> numbers[::-1]  
[90, 80, 70, 60, 50, 40, 30, 20, 10]
```



리스트 변경

```
>>> lst = [1, 2, 3, 4, 5, 6, 7, 8]
>>> lst[0:3] = ['white', 'blue', 'red']
>>> lst
['white', 'blue', 'red', 4, 5, 6, 7, 8]
```

리스트 일부 변경

```
>>> lst = [1, 2, 3, 4, 5, 6, 7, 8]
>>> lst[::2] = [99, 99, 99, 99]
>>> lst
[99, 2, 99, 4, 99, 6, 99, 8]
```

99를 중간에 추가한다.

```
>>> lst = [1, 2, 3, 4, 5, 6, 7, 8]
>>> lst[:] = [ ]
>>> lst
[]
```

리스트의 모든 요소를 삭제한다.

리스트 변경

```
numbers = list(range(0, 10))  
print(numbers)  
del numbers[-1]  
print(numbers)
```

0에서 시작하여 9까지를 저장하는 리스트

마지막 항목을 삭제한다.

리스트의 특정 요소 삭제

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

문자열과 리스트

- 문자열은 문자들이 모인 리스트라고 생각할 수 있다.

| | | | | | | | | | | | |
|----------|---|---|---|---|---|--------|----|----|----|----|----|
| | | | | | | [6:10] | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| M | o | n | t | y | | P | y | t | h | o | n |
| [-12:-7] | | | | | | -6 | -5 | -4 | -3 | -2 | -1 |

```
s = "Monty Python"
print(s[0])           # M
print(s[6:10])        # Pyth
print(s[-12:-7])      # Monty
```

정가저거 종간검

1. 함수란 무엇인가?
2. 함수를 사용하는 이유는 무엇인가?



예제

```
def get_area(radius):  
    area = 3.14*radius**2  
    return area  
  
result = get_area(3)  
print("반지름이 3인 원의 면적=", result)
```

반지름이 3인 원의 면적= 28.26

Lab: 리스트 슬라이싱

- 리스트 슬라이싱을 응용해보자.

```
numbers = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
```

- 리스트 슬라이싱 만을 이용하여 리스트의 요소들의 순서를 거꾸로 하면서 하나씩 건너뛰어 보자.

```
[10, 8, 6, 4, 2]
```

- 리스트 슬라이싱 만을 이용하여 첫 번째 요소만을 남기고 전부 삭제할 수 있는가?

```
[1]
```

Solution:

```
numbers = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]  
reversed = numbers[::-2]  
print(reversed)
```

```
numbers = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]  
numbers[1:] = [ ]  
print(numbers)
```

불변 객체가 함수로 전달되면

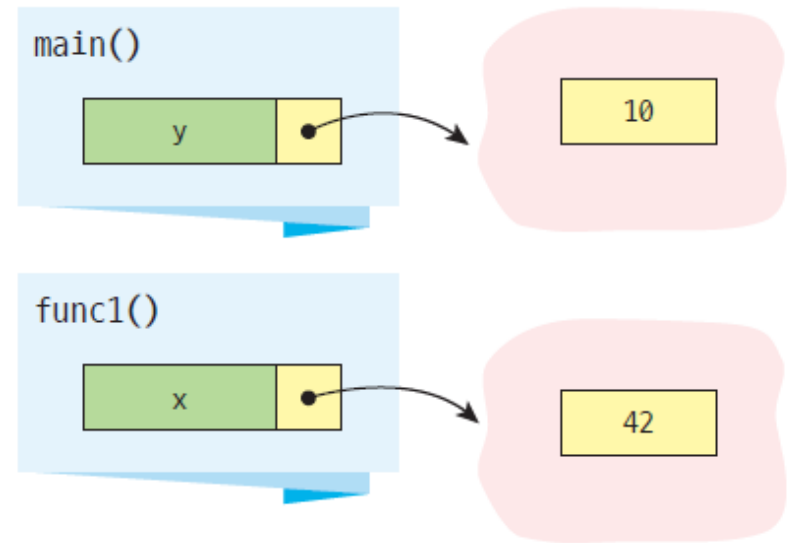
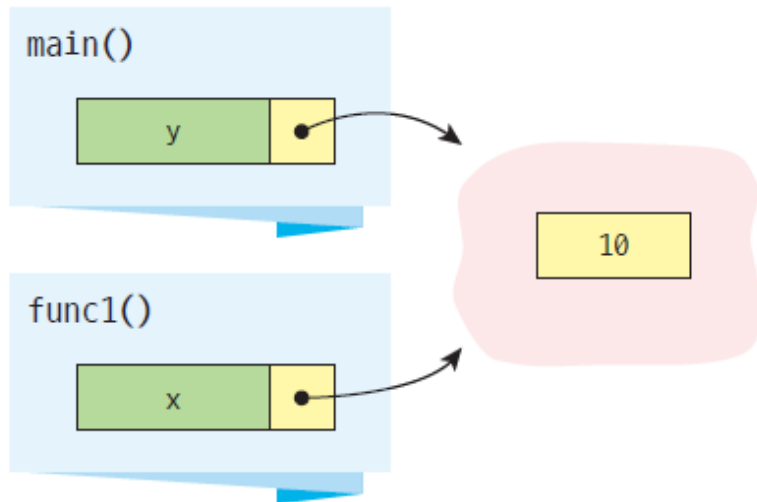
- 변경이 불가능한 객체는 값으로 전달된다고 볼 수 있다. 객체의 참조 값이 함수의 매개 변수로 전달되지만 함수 안에서 객체의 값을 변경하면 새로운 객체가 생성되기 때문이다.

```
def func1(x):  
    x = 42  
    print( "x=",x," id=",id(x))
```

```
y = 10  
func1(y)  
print( "y=",y," id=",id(y))
```

```
x= 42 id= 1640249760  
y= 10 id= 1640249248
```

보통 객체가 함수로 전달되면

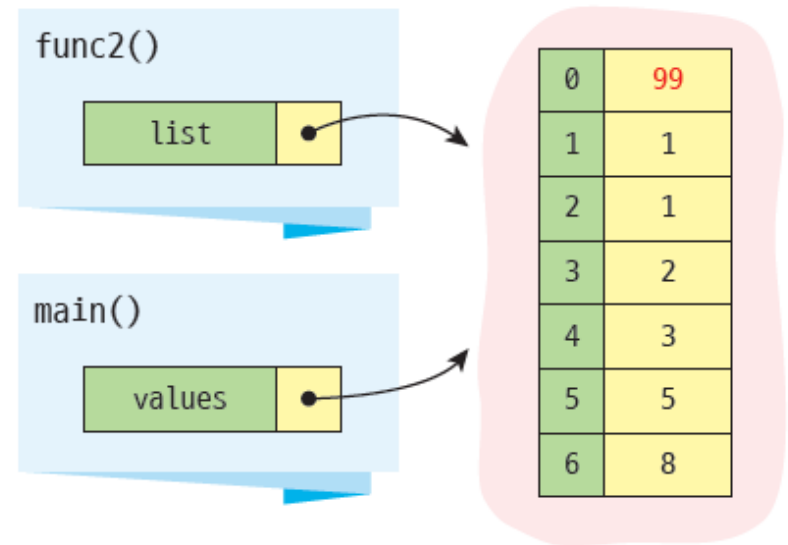
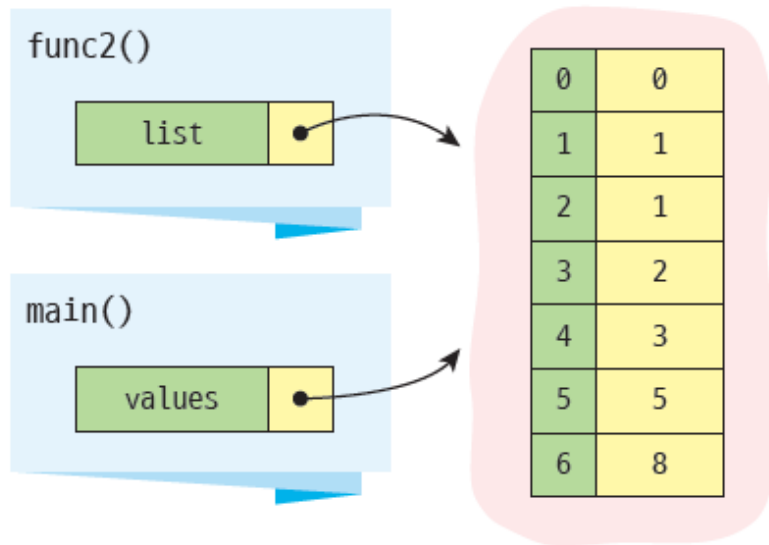


리스트가 함수로 전달되면

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

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

리스트가 함수로 전달되면



Lab: 리스트 변경 함수

- 어떤 회사에서 리스트에 직원들의 월급을 저장하고 있다. 회사에서 일괄적으로 30%의 월급 인상을 하기로 하였다. 리스트의 모든 요소들을 30% 증가시키는 함수 `modify()`를 작성하고 테스트 해보자.

인상전 [200, 250, 300, 280, 500]

인상후 [260.0, 325.0, 390.0, 364.0, 650.0]



Solution:

```
salaries = [200, 250, 300, 280, 500]
```

```
def modify(values, factor) :  
    for i in range(len(values)) :  
        values[i] = values[i] * factor
```

```
print("인상전", salaries)  
modify(salaries, 1.3)  
print("인상후", salaries)
```

리스트 합축

Syntax: 리스트 합축

형식 [수식 for (변수 in 리스트) if (조건)]

예 `squares = [x*x for x in range(10)]`

새로운 리스트

출력식으로 새로운
리스트의 요소가 된다.

입력 리스트에 있는
요소 x에 대하여

입력 리스트

이것과 같다.

```
squares = []
```

```
for x in range(10):  
    squares.append(x*x)
```

리스트 합축

- 리스트 합축에는 if를 사용하여 조건이 추가될 수 있다.

```
squares = [ x*x for x in range(10) if x % 2 == 0 ]
```

출력식

입력 리스트

조건

일반 for 루프

```
squares = [ ]  
for x in range(10):  
    if x%2 == 0 :  
        squares.append(x*x)
```

리스트 합축

```
squares = [ x*x for x in range(10) if x%2 == 0 ]
```

다양한 리스트 합치기

```
>>> prices = [135, -545, 922, 356, -992, 217]
>>> mprices = [i if i > 0 else 0 for i in prices]
>>> mprices
[135, 0, 922, 356, 0, 217]
```

```
>>> words = ["All", "good", "things", "must", "come", "to", "an", "end."]
>>> letters = [ w[0] for w in words ]
>>> letters
['A', 'g', 't', 'm', 'c', 't', 'a', 'e']
```

```
>>> numbers = [x+y for x in ['a','b','c'] for y in ['x','y','z']]
>>> numbers
['ax', 'ay', 'az', 'bx', 'by', 'bz', 'cx', 'cy', 'cz']
```

Lab: 리스트 합속 사용하기

- 0부터 99까지의 정수 중에서 2의 배수이고 동시에 3의 배수인 수들을 모아서 리스트로 만들어보자.

```
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96]
```


Solution:

```
numbers = [x for x in range(100) if x % 2 == 0 and x % 3 == 0]  
print(numbers)
```

0부터 9까지의 정수 중에서 짝수이면 “짝수”를 리스트에 추가하고 홀수이면 “홀수”를 리스트에 추가하는 리스트 함축도 만들어보자.

🔗 실행결과

```
['짝수', '홀수', '짝수', '홀수', '짝수', '홀수', '짝수', '홀수', '짝수', '홀수']
```



Lab: 누적합 리스트 만들기

- i 번째 요소가 원래 리스트의 0부터 i 번째 요소까지의 합계인 리스트를 생성하는 프로그램을 작성하라.

원래 리스트: [10, 20, 30, 40, 50]

새로운 리스트: [10, 30, 60, 100, 150]

1. 빈 리스트를 선언하고 초기화한다.
2. 리스트 합축을 사용하여 리스트에 있는 요소 0부터 $x+1$ 까지의 누적 합계를 계산하여 새로운 리스트로 만든다.
3. 원래 리스트와 새로운 리스트를 출력한다.

Solution:

```
list1=[10, 20, 30, 40, 50]
```

```
list2=[sum(list1[0:x+1]) for x in range(0, len(list1))]
```

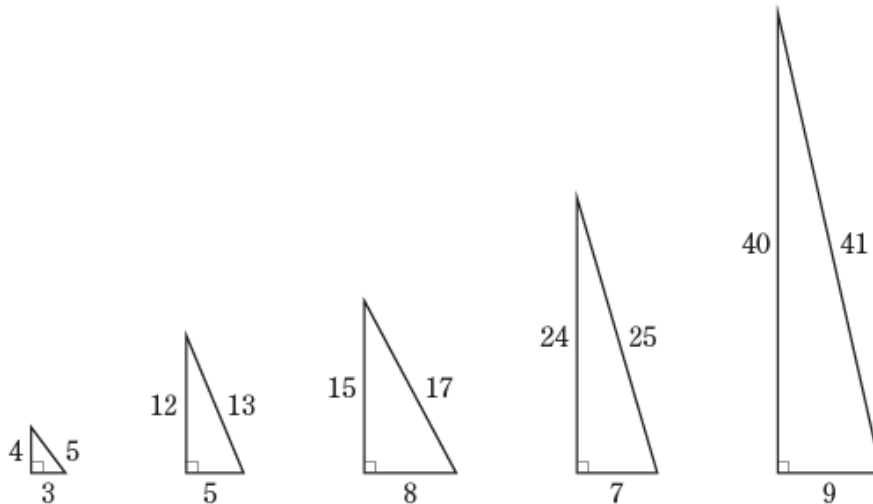
```
print("원래 리스트: ",list1)
```

```
print("새로운 리스트: ",list2)
```

Lab: 피타고라스 삼각형

- 피타고라스의 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:

```
[(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]
```

```
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 리스트 == 2차원 테이블

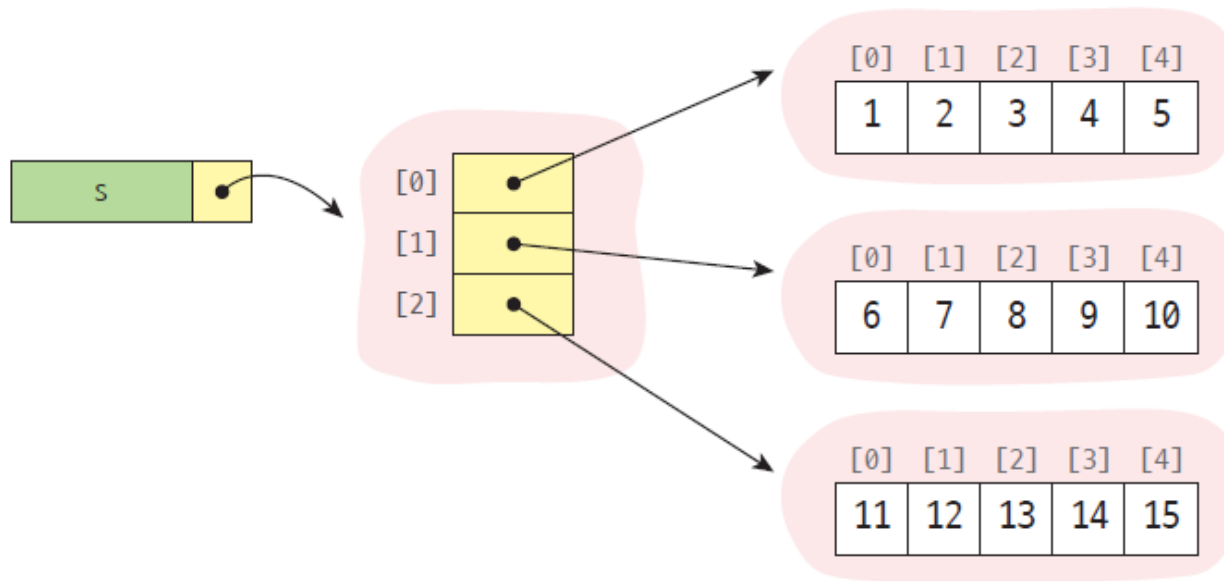


2차원 리스트를 생성한다.

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

2차원 리스트의 구현

- 리스트의 리스트로 구현된다.



2차원 리스트의 동적 생성

- 실제로는 동적으로 2차원 리스트를 생성하는 경우가 더 많다.

```
# 동적으로 2차원 리스트를 생성한다.  
rows = 3  
cols = 5  
  
s = []  
for row in range(rows):  
    s += [[0]*cols]           # 2차원 리스트끼리 합쳐진다.  
  
print("s =", s)
```

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


주의할 점

- 실제로는 동적으로 2차원 리스트를 생성하는 경우가 더 많다.

```
# 동적으로 2차원 리스트를 생성한다.  
rows = 3  
cols = 5  
  
s = []  
for row in range(rows):  
    s += [0]*cols          # 1차원 리스트끼리 합쳐진다.  
  
print("s =", s)
```

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

리스트 합치기 이용하는 방법

- 실제로는 동적으로 2차원 리스트를 생성하는 경우가 더 많다.

```
rows = 3  
cols = 5  
  
s = [ ([0] * cols) for row in range(rows) ]  
  
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,
```

2차원 리스트를 함수로 넘기기

- 2차원 리스트도 함수로 전달할 수 있다. 함수 안에서는 2차원 리스트의 차원을 추출할 수 있다. 2차원 리스트를 `s`라고 하면 `len(s)`는 행의 개수이고 `len(s[0])`는 열의 개수이다.

```
def sum(numbers) :  
    total = 0  
    for i in range(len(numbers)) :  
        for j in range(len(numbers[0])) :  
            total = total + numbers[i][j]  
  
    return total
```

체커 보드 패턴 만들기

- 1과 0이 반복되는 체커보드 형태의 10×10 크기의 2차원 리스트를 초기화하는 함수 `init()`를 작성하고 테스트하자.

```
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
```

```
table = [ ]
```

```
# 2차원 리스트를 화면에 출력한다.
```

```
def printList(mylist):  
    for row in range(len(mylist)):  
        for col in range(len(mylist[0])):  
            print(mylist[row][col], end=" ")  
        print()
```

```
# 2차원 리스트를 체커보드 형태로 초기화한다.
```

```
def init(mylist):  
    for row in range(len(mylist)):  
        for col in range(len(mylist[0])):  
            if (row+col)%2 == 0:  
                table[row][col] = 1
```

```
# 2차원 리스트를 생성한다.
```

```
for row in range(10):  
    table += [[0]*10]
```

```
init(table)  
printList(table)
```

2차원 리스트와 리스트 합치기

```
matrix = [[i for i in range(5)] for _ in range(6)]  
print(matrix)
```

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

```
matrix = [ [0, 0, 0], [1, 1, 1], [2, 2, 2] ]  
result = [num for row in matrix for num in row]  
print(result)
```

```
[0, 0, 0, 1, 1, 1, 2, 2, 2]
```

Lab: 전치 행렬 계산

- 행렬의 전치 연산을 파이썬으로 구현해보자. 인공지능을 하려면 행렬에 대하여 잘 알아야 한다. 중첩된 **for** 루프를 이용하면 행렬의 전치를 계산할 수 있다.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

원래 행렬 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

전치 행렬 = [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

Solution:

```
transposed = []
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

print("원래 행렬=", matrix)
# 열의 개수만큼 반복한다.
for i in range(len(matrix[0])):
    transposed_row = []
    for row in matrix:                # 행렬의 각 행에 대하여 반복
        transposed_row.append(row[i]) # i번째 요소를 row에 추가한다.
    transposed.append(transposed_row)

print("전치 행렬=", transposed)
```

Lab: Tic-Tac-Toe

- 다음과 같이 텍스트 모드에서 컴퓨터와 사람이 Tic-Tac-Toe 게임을 할 수 있는 프로그램을 작성하여 보자.

```
| | |  
---|---|---  
| | |  
---|---|---  
| | |
```

다음 수의 x좌표를 입력하시오: 0

다음 수의 y좌표를 입력하시오: 0

```
 x|  0|  
---|---|---  
| | |  
---|---|---
```

Solution:

```
board= [[' ' for x in range (3)] for y in range(3)]
while True:
    # 게임 보드를 그린다.
    for r in range(3):
        print("  " + board[r][0] + "|" + board[r][1] + "|" + board[r][2])

        if (r != 2):
            print("---|---|---")

    # 사용자로부터 좌표를 입력받는다.
    x = int(input("다음 수의 x좌표를 입력하시오: "))
    y = int(input("다음 수의 y좌표를 입력하시오: "))
```

Solution:

```
# 사용자가 입력한 좌표를 검사한다.
if board[x][y] != ' ':
    print("잘못된 위치입니다. ")
    continue
else:
    board[x][y] = 'X'

# 컴퓨터가 놓을 위치를 결정한다. 첫 번째로 발견하는 비어있는 칸에 놓는다.
done = False
for i in range(3):
    for j in range(3):
        if board[i][j] == ' ' and not done:
            board[i][j] = 'O';
            done=True
            break;
```

이번 장에서 배운 것

- 리스트는 일련의 값을 저장하는 컨테이너이다.
- 리스트의 각 요소는 인덱스라는 정수로 접근된다. 예를 들어서 i번째 요소는 `mylist[i]`가 된다.
- `insert()` 메소드를 사용하여 리스트의 임의 위치에 새로운 요소를 삽입할 수 있다.
- `pop()` 메소드를 사용하여 리스트의 임의 위치에서 요소를 제거할 수 있다.
- `remove()` 메소드를 사용하여 리스트에서 원하는 요소를 제거할 수 있다.
- `+` 연산자를 사용하여 두 리스트를 연결할 수 있다.
- 슬라이스 연산자 (`:`)를 사용하여 부분 리스트를 만들 수 있다.
- 리스트 함축은 기존 리스트를 기반으로 새로운 리스트를 작성하는 우아한 방법이다.



Q & A

