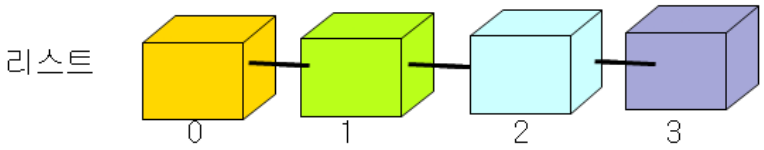
7장 리스트

리스트의 개요. p227

- 여러 개의 값을 저장하여야 하는 경우에 사용
- 리스트는 항목(item)들을 저장하는 컨테이너로서 그 안에 항목들이 순서를 가지고 저장된다. 어떤 타입의 항목라도 저장할 수 있다.



- 예) 1주일 치 기온을 저장하기 위하여 temp1, temp2, temp3, temp4, temp5, temp6, temp7과 같이 변수를 7개 만드는 것은 매우 비효율적이다.

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

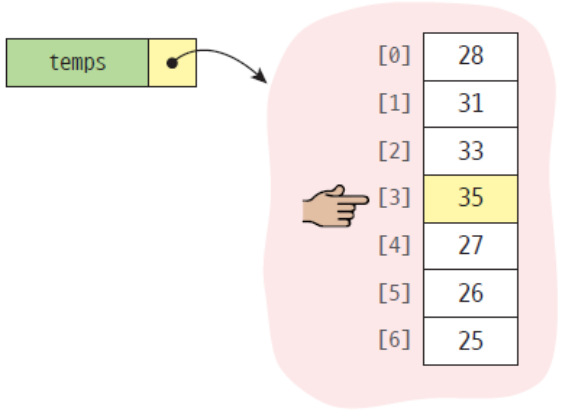
```
temps = [28, 31, 33, 35, 27, 26, 25]
```

리스트 항목

리스트의 항목 접근하기. p228

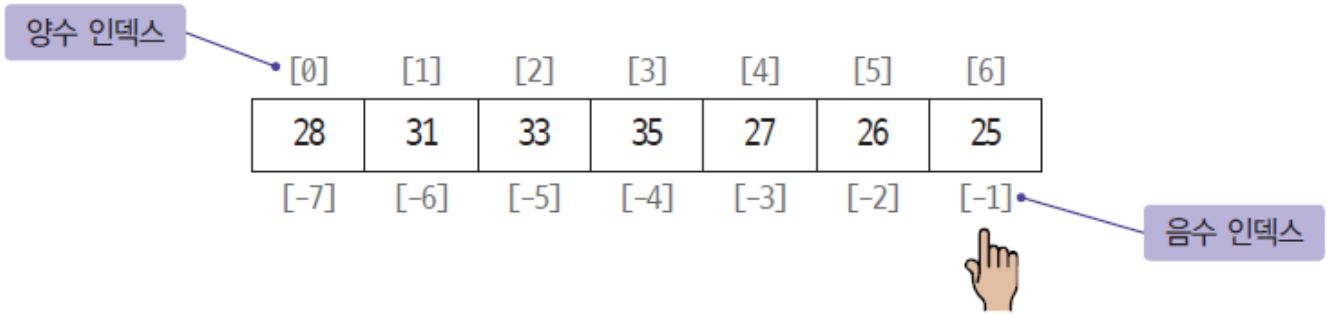
- 항목의 위치(번호)인 인덱스(index)를 사용
- 인덱스는 0부터 시작한다.

```
>>> temps = [ 28, 31, 33, 35, 27, 26, 25]
>>> print(temps[3])              # 인덱스가 3인 항목(네번째 항목)을 출력
35


>>> temps[3] = 36               # 항목의 값 변경
```



음수 인덱스. p228

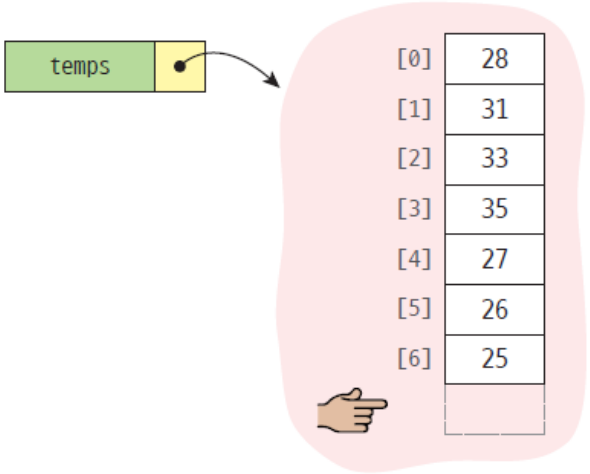
- 음수 인덱스는 리스트의 끝에서부터 매겨진다.
- 리스트에서 가장 마지막 항목의 인덱스는 -1.
- 마지막 요소 접근 : temps[-1] 이 temps[len(temps)-1] 보다 훨씬 간결.



인덱스 오류. p229

- 인덱스를 사용할 때는 인덱스가 적정한 범위에 있는지를 항상 신경써야 한다.

```
>>> temps = [ 28, 31, 33, 35, 27, 26, 25]
>>> temps[7] = 29
IndexError: list index out of range
```



리스트 방문. p229

- 리스트 안에 저장된 요소들을 전부 방문하는 코드를 작성하는 방법
- (1) 인덱스를 이용하는 방법

```
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,
```

- (2) for-in 루프를 사용하는 방법

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

for element in temps:
        print(element, end=', ')
```

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

리스트에 어떤 값이 있는지를 알고 싶다면. p230

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

if 33 in temps:                    # 값이 저장되어 있는지 확인
        print("리스트에 33이 있음!")
```

```
리스트에 33이 있음!
```

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

if 99 not in temps:                # 값이 없다는 것을 확인
        print("리스트에 99가 없음!")
```

```
리스트에 99가 없음!
```

리스트 연산들. p230

- append() : 새로운 요소를 리스트 맨 끝에 추가한다.

```
fruits = [ ]                                # 공백 리스트를 생성한다.
fruits.append("apple")                      # 리스트에 "apple"을 추가한다.
fruits.append("banana")                     # 리스트에 "banana"를 추가한다.
print(fruits)
```

```
['apple', 'banana']
```

- insert() : 지정된 위치에 요소를 추가한다.

```
fruits = ["apple", "banana", "grape"]
fruits.insert(1, "cherry")
print(fruits)
```

```
['apple', 'cherry', 'banana', 'grape']
```

리스트 탐색하기. p231

- index() : 특정 항목이 리스트의 어디에 있는지 알려면 사용

```
fruits = ["apple", "banana", "grape"]
n = fruits.index("banana")                    # n은 1이 된다.
```

- 항목이 리스트에 없다면 오류 발생. 따라서 탐색하기 전에 있는지 부터 확인하는 것이 안전하다.

```
if "banana" in fruits:
    print(fruits.index("banana"))
```

요소 삭제하기. p232

- pop(i) : 리스트에서 i 번째 항목을 삭제하여 반환한다.

```
fruits = ["apple", "banana", "grape"]
item = fruits.pop(0)                    # "apple"이 삭제된다.
print(fruits)
```

["banana", "grape”]

- remove(value) : 리스트에서 저장된 값을 삭제한다.

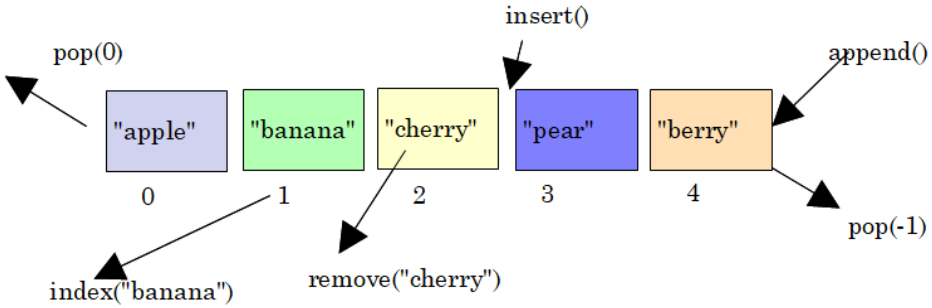
```
fruits = ["apple", "banana", "grape"]
fruits.remove("banana")
print(fruits)
```

[“apple", "grape”]

```
if  " banana" in fruits:
        fruits.remove(“banana")
```

항목이 리스트 안에 있는지 확인후 삭제하는 것이 바람직

리스트 연산 정리. p233



연산의 예	설명
mylist[2]	인덱스 2에 있는 요소
mylist[2] = 3	인덱스 2에 있는 요소를 3으로 설정한다.
mylist.pop(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.

최소값, 최대값, 합계. p233

```
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
```

정렬. p234

- sort() : 제자리(in-place)에서 리스트를 정렬. 따라서 원본 리스트가 변경된다.

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

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

[5, 4, 3, 2, 1]

- sorted() : 새로운 정렬된 리스트를 반환. 원래의 리스트는 수정되지 않는다.

```
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]

리스트에서 랜덤으로 선택하기. p234

- 리스트 안에서 항목을 랜덤하게 선택해야 하는 경우
- random 모듈의 choice() 사용

```
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)          # 문자열 리스트에서 사용할 수 있다.
print ("랜덤하게 선택한 항목=", item)
```

랜덤하게 선택한 항목= Citizen Kane

Example 두번째로 큰 수 계산하기. p235

- 정수들이 저장된 리스트에서 두 번째로 큰 수를 찾아보자.

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

# 리스트를 정렬한다.
list1.sort()

# 뒤에서 두 번째 요소를 출력한다.
print("두 번째로 큰 수=", list1[-2])
```

```
두 번째로 큰 수= 15
```

- (추가) 또 다른 방법

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

# 제일 큰 수는 삭제한다.
list1.remove(max(list1))
 # 그 다음으로 큰 수를 출력한다.
print("두 번째로 큰 수=", max(list1))
```

Example 최대값 최소값 제외하기. p236

- 체조와 같은 콘테스트에서는 심판들의 점수 중에서 최소값과 최대값을 제외하는 경우가 많다. 최대값과 최소값을 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]
```

```
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)
```

- 또 따른 방법은 뭐가 있을까?

Lab 성적 처리 프로그램. p237

- 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
```

리스트 합병과 복제. p243

- 합병은 + 연산자를 사용한다.

```
fruits1 = [ "apple", "cherry" ]
fruits2 = [ "banana", "blueberry" ]
fruits = fruits1 + fruits2
print(fruits)
```

```
['apple', 'cherry', 'banana', 'blueberry']
```

- 복제는 * 연산자를 사용한다.

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

```
numbers = [0] * 12                 # 리스트는 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]이다.
```

리스트 비교. p238

- 리스트의 첫번째 항목부터 순서대로 비교한다.

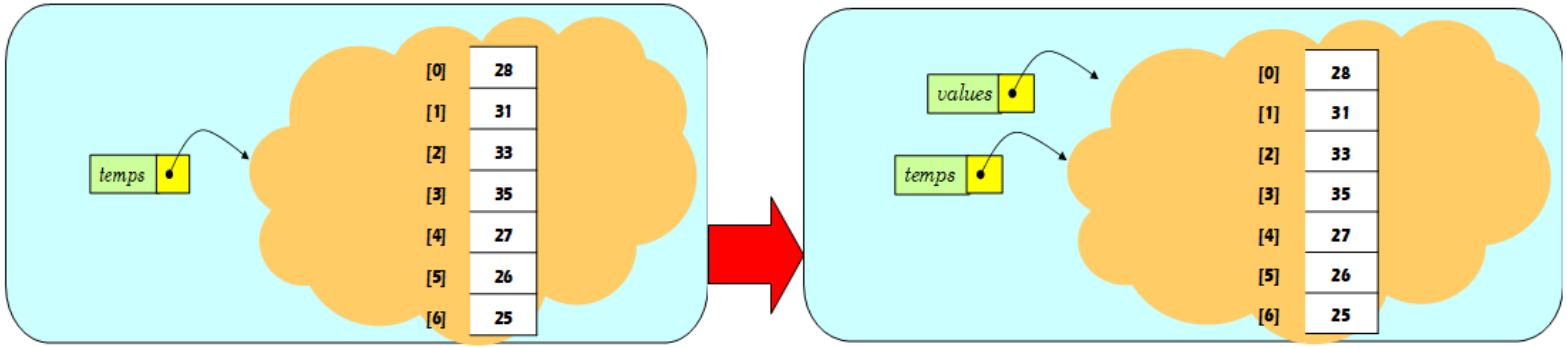
```
list1 = [ 1, 2, 3 ]
list2 = [ 1, 2, 3 ]
print(list1 == list2)                # True
```

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

리스트 복사하기. p239

- 1. 얕은 복사(shallow copy) : 리스트의 참조값(reference)만 복사하고 리스트의 내용은 복사하지 않는다.

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



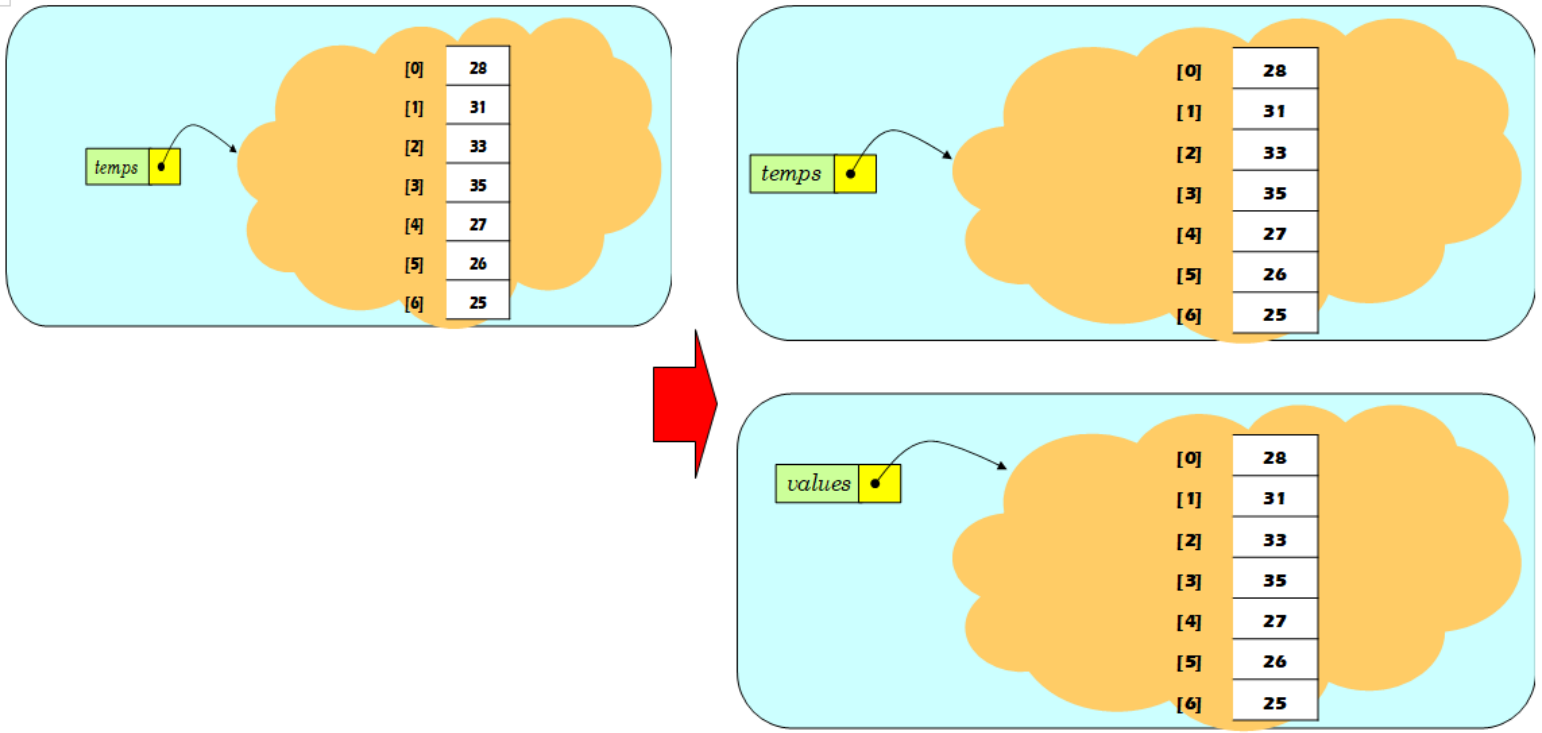

```
temps = [28, 31, 33, 35, 27, 26, 25]
values = temps

print(temps)                # temps 리스트 출력
values[3] = 39              # values 리스트 변경
print(temps)                # temps 리스트가 변경되었다.
```

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

리스트 복사하기. p239

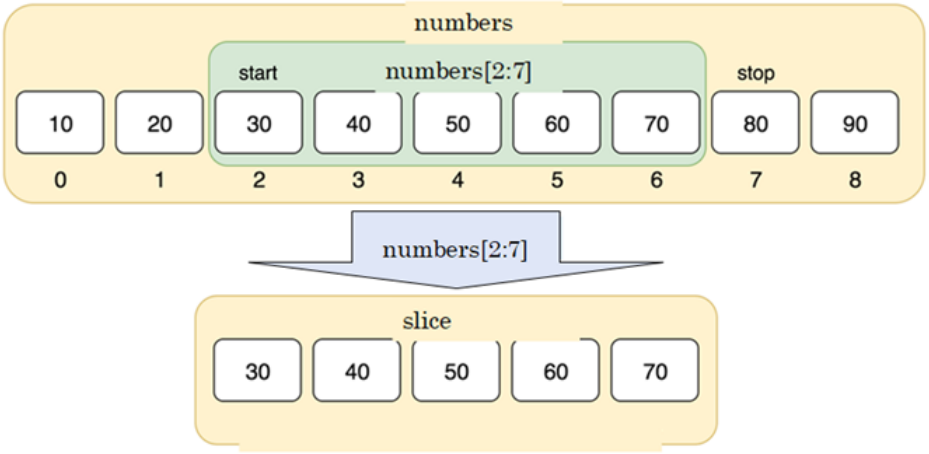
- 2. 깊은 복사(deep copy) : 새로운 리스트를 생성한다. list() 사용

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



슬라이싱. p241

- 슬라이싱(slicing) : 리스트의 일부를 잘라서 추출하는 기법

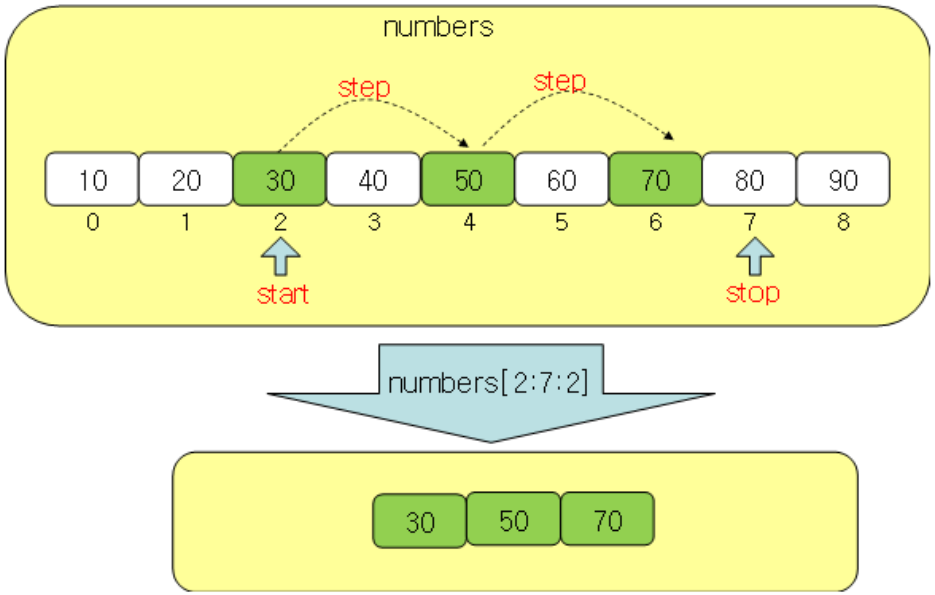



```
numbers[:3]                 # [10, 20, 30]

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

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

```
new_numbers = numbers[:]                # 깊은 복사본을 만든다.
```

고급 슬라이싱. p242

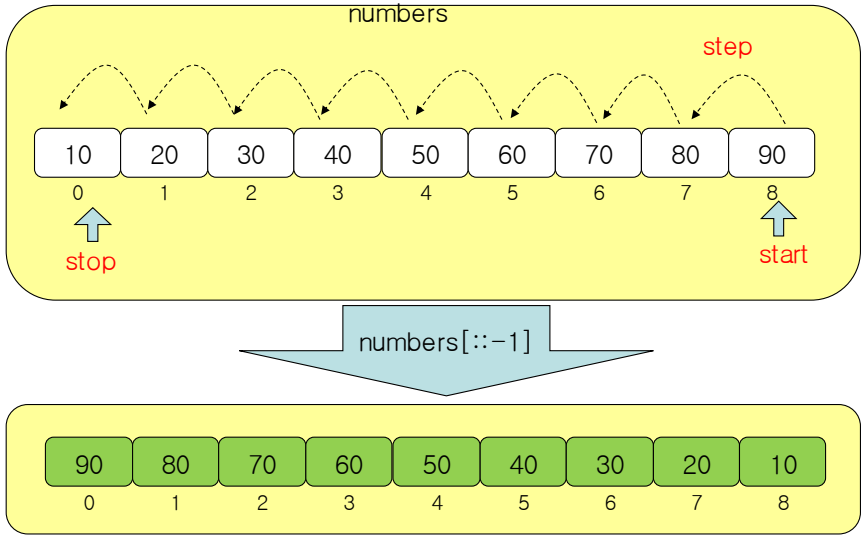
- 슬라이싱을 할 때 단계를 지정할 수 있다.



고급 슬라이싱. p242

- 음수 단계값을 이용하여 역순 리스트를 만들 수 있다.

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



고급 슬라이싱. p243

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

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

```
>>> lst = [1, 2, 3, 4, 5, 6, 7, 8]
>>> lst[:] = [ ]                                 # 리스트의 모든 요소를 삭제
>>> lst
[]
```

고급 슬라이싱. p244

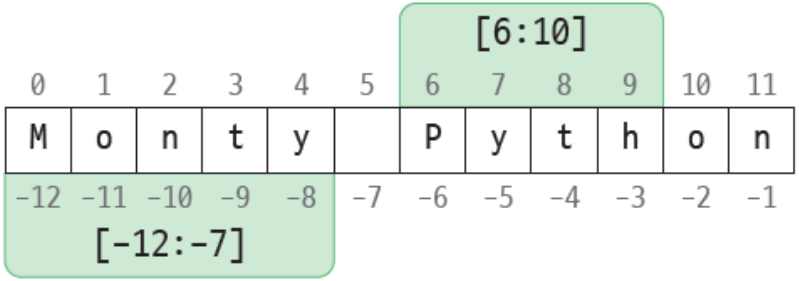
```
>>> numbers = list(range(0, 10))          # 0에서 시작하여 9까지를 저장하는 리스트
>>> numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> del numbers[-1]                       # 마지막 항목을 삭제한다.
>>> numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

```
>>> del numbers[0:2]                      # 인덱스 0부터 인덱스 1까지를 삭제
>>> numbers
[2, 3, 4, 5, 6, 7, 8]
```

```
>>> del numbers[:]                        # 전체를 삭제
>>> numbers
[]
```

문자열과 리스트. p244

- 문자열은 문자들이 모인 리스트라고 생각할 수 있다.




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

Example 리스트 슬라이싱 연습. p245

```
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]
```

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

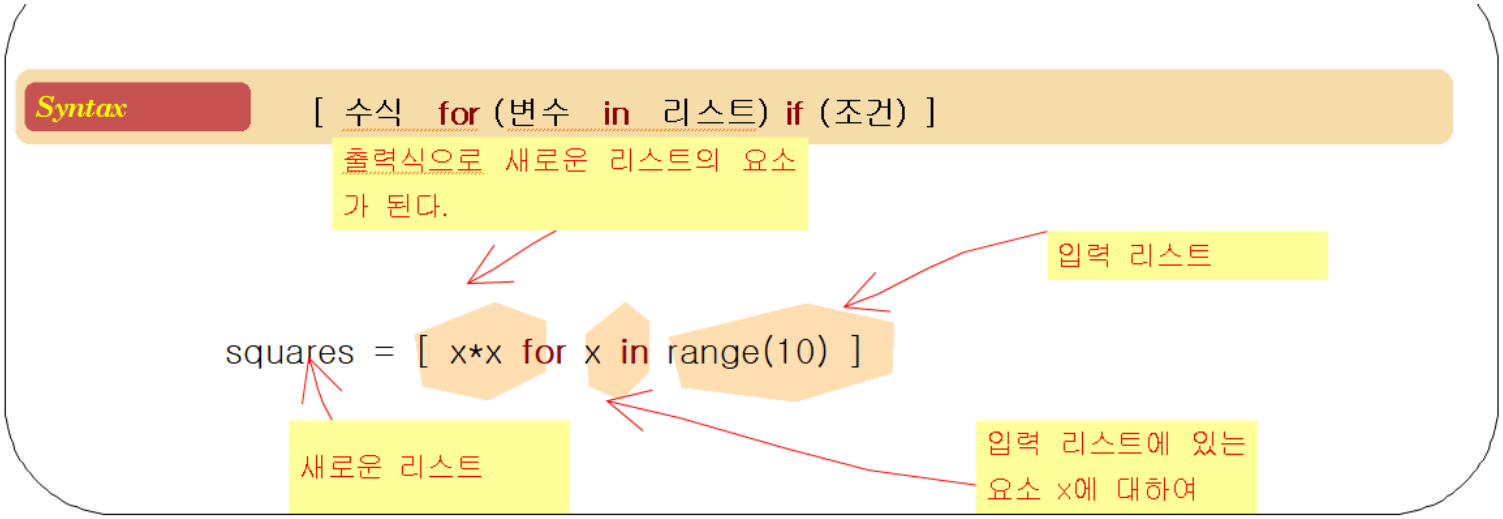
- 리스트 슬라이싱 만을 이용하여 첫 번째 요소만을 남기고 전부 삭제할 수 있는가?

```
[1]
```

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

리스트 함축. p246

- 리스트 함축(list comprehension) : 함축된 표현



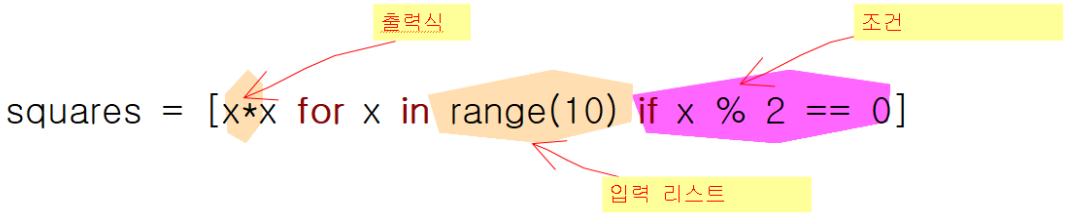

```
squares = [ 0, 1, 4, 9, 16, 25, 36, 49, 64, 81 ]
```

```
squares = []

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

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

리스트 함축

- 리스트 함축에는 if를 사용하여 조건이 추가될 수 있다.



squares = [0, 4, 16, 36, 64]

일반 for 루프

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

리스트 함축

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

다양한 리스트 함축 예제

```
>>> 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']
```

리스트 함축의 단점. p248

Example 2의 배수이면서 3의 배수인 수 찾기. p248

- 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]
```

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

Example 합계 리스트 생성. p249

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

```
원래 리스트: [10, 20, 30, 40, 50]
새로운 리스트: [10, 30, 60, 100, 150]
```

```
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)
```

Example 직각 삼각형 찾기. p249

- 피타고라스의 정리를 만족하는 삼각형들을 리스트 함축으로 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)]
```

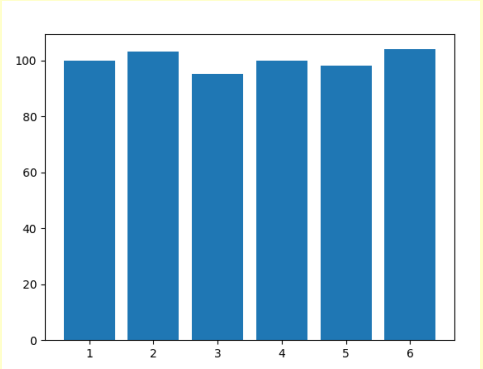
```
[(x,y,z) for x in range(1,30) for y in range(x,30) for z in range(y,30) if x**2 + y**2 == z**2]
```

```
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)
```

Lab 주사위 시뮬레이션. p250

- 주사위를 600번 던졌을 때 각 면이 몇 번이나 나오는지를 데이터로 정리하고 이 데이터를 막대 그래프로 시각화하여 보자. 리스트 함축도 연습해보자.

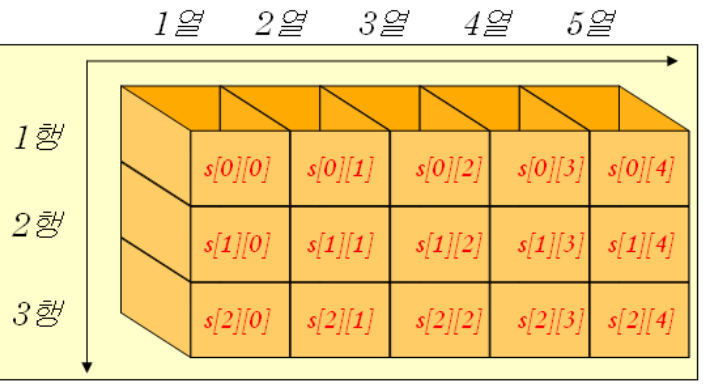
```
import matplotlib.pyplot as plt
import random

values = [random.randint(0, 5) for i in range(600)]

faces = [ 1, 2, 3, 4, 5, 6 ]
rolls = [ 0, 0, 0, 0, 0, 0 ]

for x in values :
        rolls[x] = rolls[x] + 1

plt.bar(faces, rolls)          # 리스트에 저장된 값으로 그래프를 그린다.
plt.show()                     # 그래프를 화면에 출력한다.
```



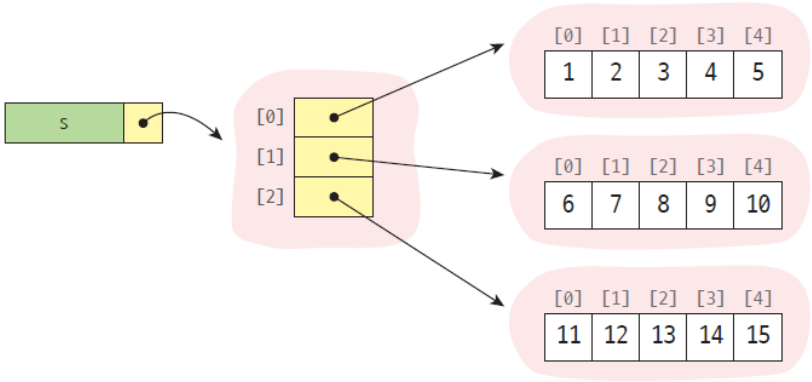
2차원 리스트. p251

- 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)
```




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



2차원 리스트. p251

- 동적으로 2차원 리스트를 생성하는 방법

```
# 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]]
```

- (주의) 다음과 같이 하면 1차원 리스트가 된다.

```
…
for row in range(rows):
        s += [0]*cols                      # 1차원 리스트끼리 합쳐진다.
…
```

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

2차원 리스트. p252

- (추가) 리스트 함축을 이용하여 동적으로 2차원 리스트를 생성하는 코드

```
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]]
```

요소 접근. p252

- 2차원 리스트에서 요소에 접근하려면 2개의 인덱스 번호를 지정하여야 한다.

```
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,
```

Lab 전치 행렬 계산. p254

- 행렬의 전치 연산을 파이썬으로 구현해보자. 인공 지능을 하려면 행렬에 대하여 잘 알아야 한다. 중첩 된 **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]]
```

Mini Project 지뢰 찾기. p255

- 지뢰찾기 게임을 작성해보자. 10×10 크기의 2차원 리스트을 만들고 여기에 지뢰를 숨긴다. 지뢰가 아닌 곳은 .으로 표시하고 지뢰인 곳은 #로 표시하여 보자. 어떤 칸이 지뢰일 확률은 난수를 발생시켜서 결정한다. 전체의 30%를 지뢰로 하고 싶으면 발생된 난수가 0.3보다 적은 경우에 현재 칸에 지뢰를 놓으면 된다.

```
..#...##..
..#.###...
..#....#..
##.##.#.#.
.#.......#
#..#.##...
##.#.##...
..##....#.
........#.
....#..##.
```

Mini Project TIC-TAC-TOC 게임. p256

연습문제. p258

Programming. p261