

새내기 파이썬



기자
리스트





하스 검 목 표

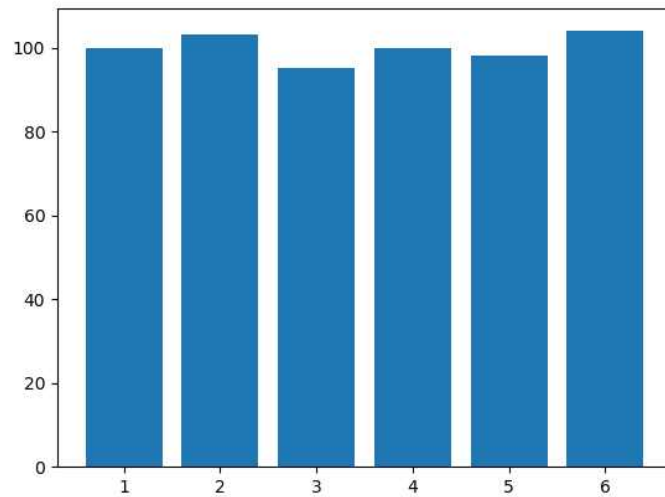
- 리스트를 사용하여 항목들을 저장할 수 있나요?
- 리스트에서 일부분을 추출할 수 있나요?
- 리스트를 함수로 전달하거나 반환받을 수 있나요?
- 리스트 함축을 사용할 수 있나요?
- 리스트를 2차원으로 만들 수 있나요?





이번장에서 만든 프로그램

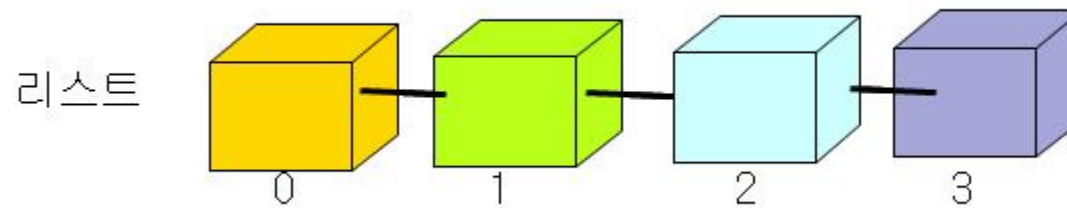
성적을 입력하시요: 10
성적을 입력하시요: 20
성적을 입력하시요: 60
성적을 입력하시요: 70
성적을 입력하시요: 80
성적 평균= 48.0
최대점수= 80
최소점수= 10
80점 이상= 1





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.



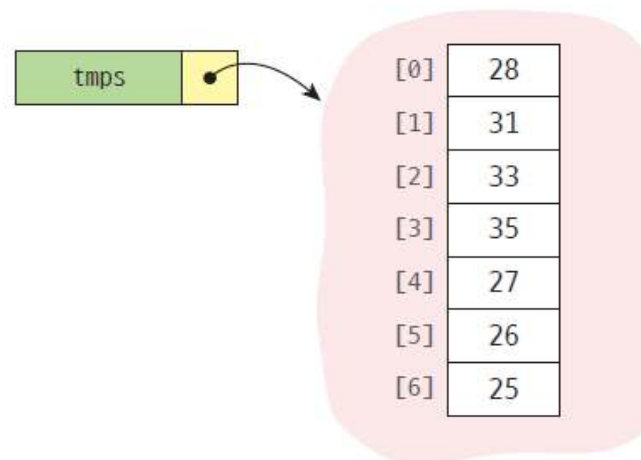


리스트를 사용하는 예

- 지난 1주일 중에서 가장 더운 날을 찾으려고 한다고 하자. 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 만들기

- 파이썬에서 리스트는 대괄호 [...]을 이용하여 만들어진다. 리스트 안에 저장된 각각의 데이터를 항목(item) 또는 요소(element)라고 한다. 항목들은逗를 이용하여 분리하여야 한다.

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

리스트

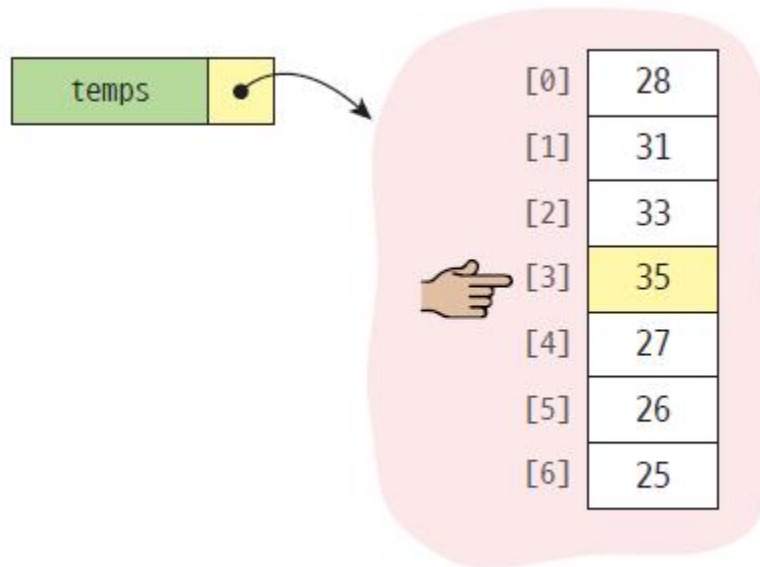
항목



리스트 인덱스

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

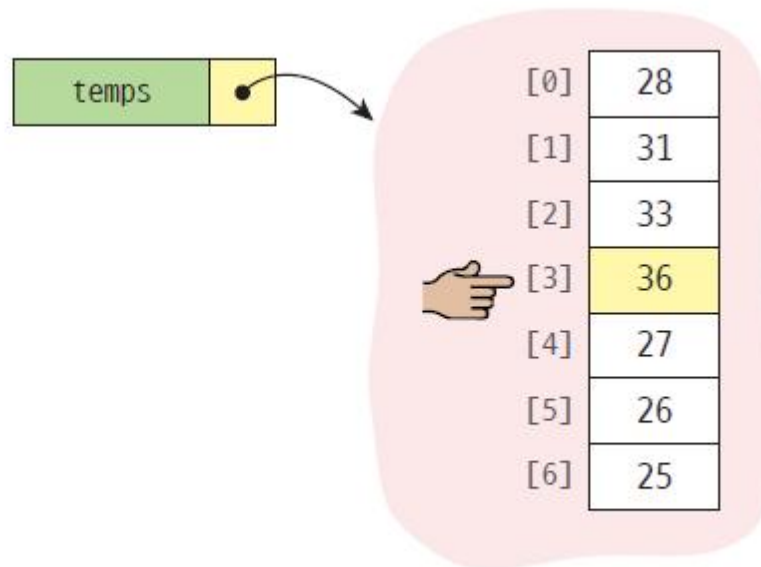
```
>>> temps = [ 28, 31, 33, 35, 27, 26, 25]
>>> print(temps[3])
35
```





리스트 요소 변경

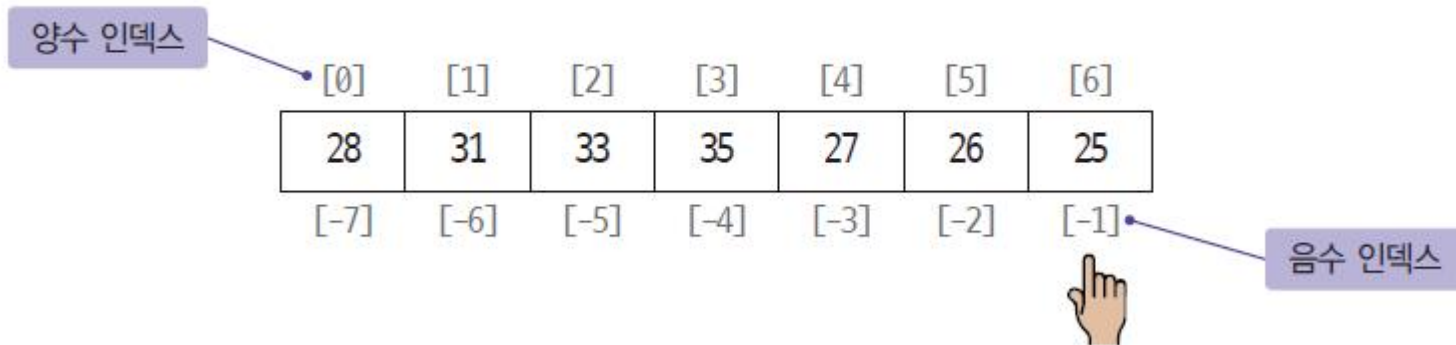
```
>>> temps = [ 28, 31, 33, 35, 27, 26, 25]  
>>> temps[3]=36
```





음수 인덱스

- 음수 인덱스는 리스트의 끝에서부터 매겨진다.
- 리스트의 마지막 요소에 접근하려면 무조건 `temps[-1]`이라고 하면 된다.

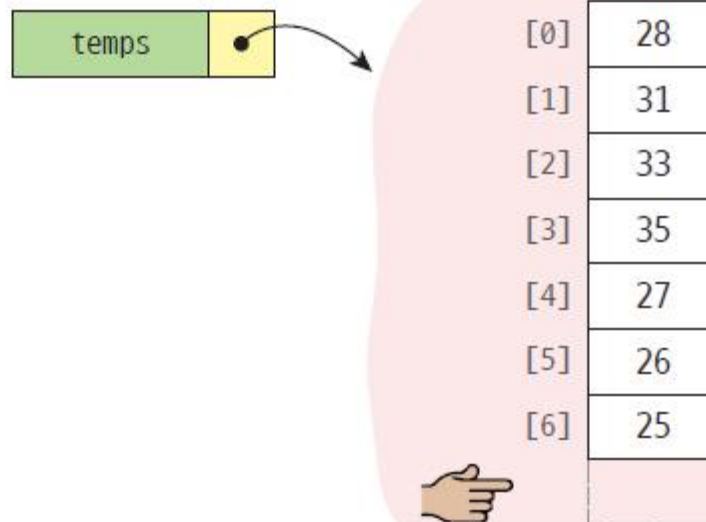




인덱스 오류

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

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





리스트 방문

- 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,
```

- for-each 루프를 이용하는 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 있는지 알고 싶다면?

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

```
if 33 in temps:
```

```
    print("리스트에 33이 있음!")
```

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

```
if 99 not in temps:
```

```
    print("리스트에 99가 없음!")
```



append()

```
fruits = []  
fruits.append("apple")  
fruits.append("banana")  
print(fruits)
```

```
# 공백 리스트를 생성한다.  
# 리스트에 "apple"을 추가한다.  
# 리스트에 "banana"를 추가한다.
```

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



insert()

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

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



리스트 탐색하기

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

탐색할 때, 오류를 발생시키지
않으려면



요소 삭제하기

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

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

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




remove()

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

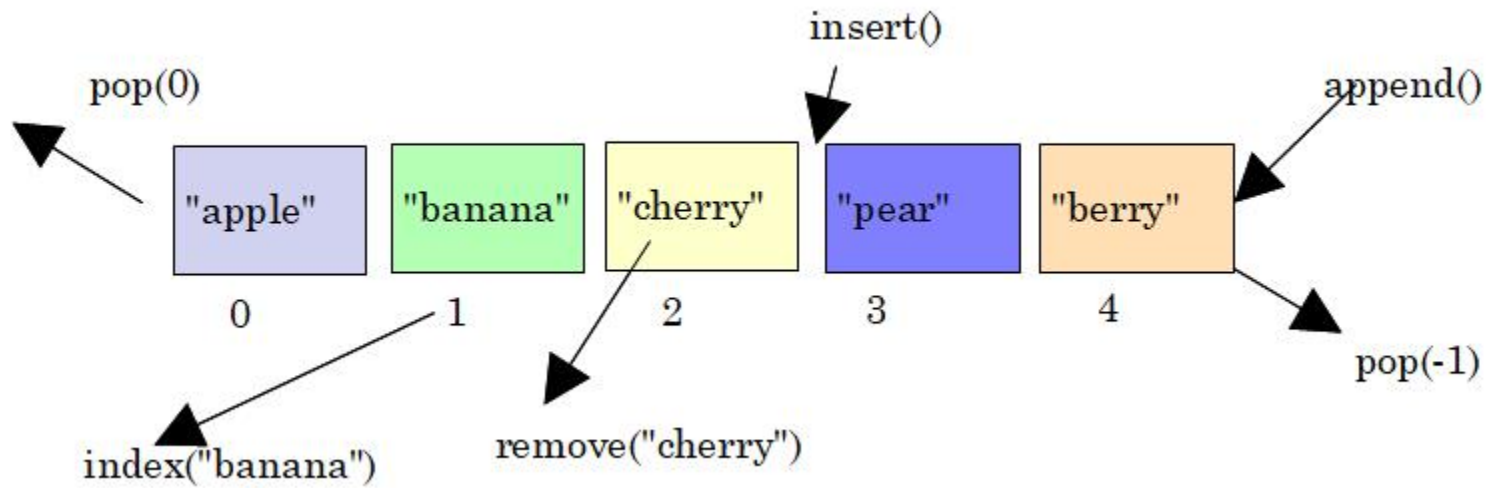
만약 삭제하고자 하는 항목이 없다면 오류(예외)가 발생된다.

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

확인후 삭제한다.



리스트 연산 정리





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 True
<code>"value" not in mylist</code>	<code>"value"</code> 가 <code>mylist</code> 에 없으면 True
<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

```
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
```

max()와 min()은 내장 함수로서 거의 모든 객체에 사용 가능



리스트 정렬 #1: sort()

```
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]
```

역순으로 정렬할때



리스트 정렬 #2: sorted()

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

```
ascending_numbers =sorted(numbers)
```

```
print( ascending_numbers )
```

내장 함수

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



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)
```

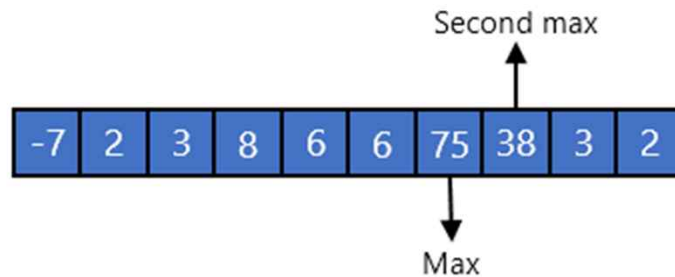
랜덤하게 선택한 항목= Citizen Kane



Example: 리스트에서 두 번째로 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 수는 삭제한다.
```

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

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

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



Example: 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: 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)
```



리스트 합병과 복제

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

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

```
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

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

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

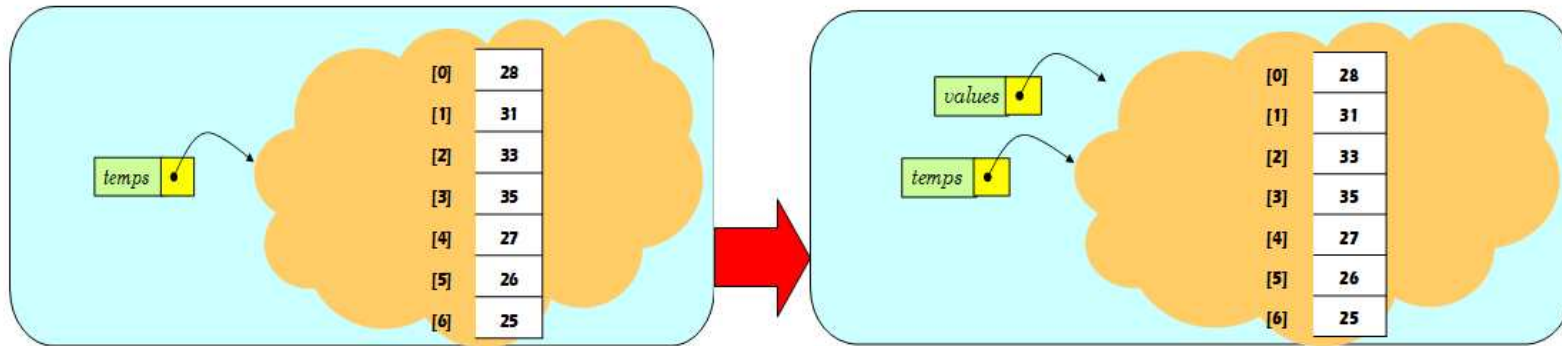
참고:

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]
```

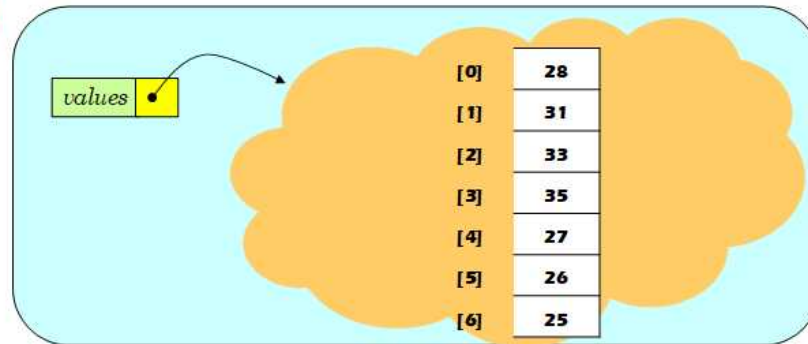
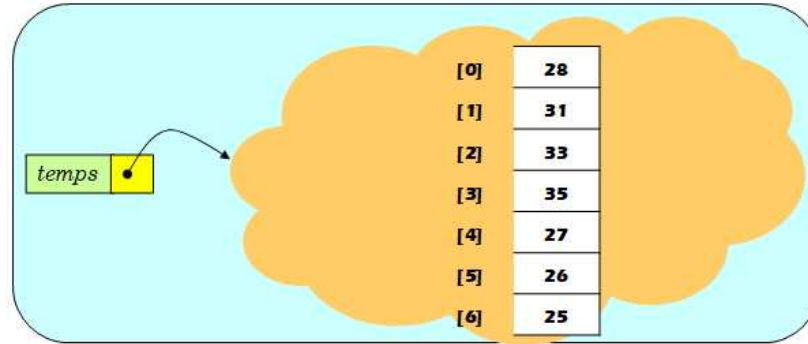
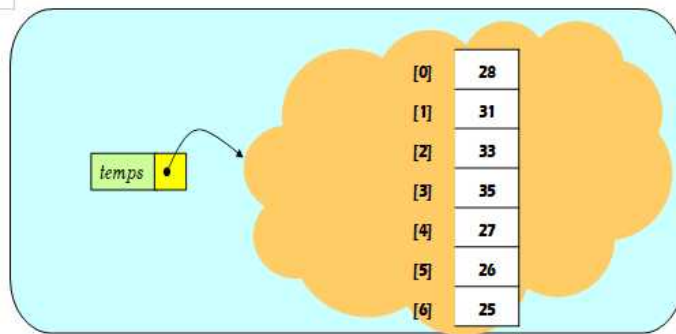
이것을 얇은 복사라고 함.



리스트 기본 복사

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

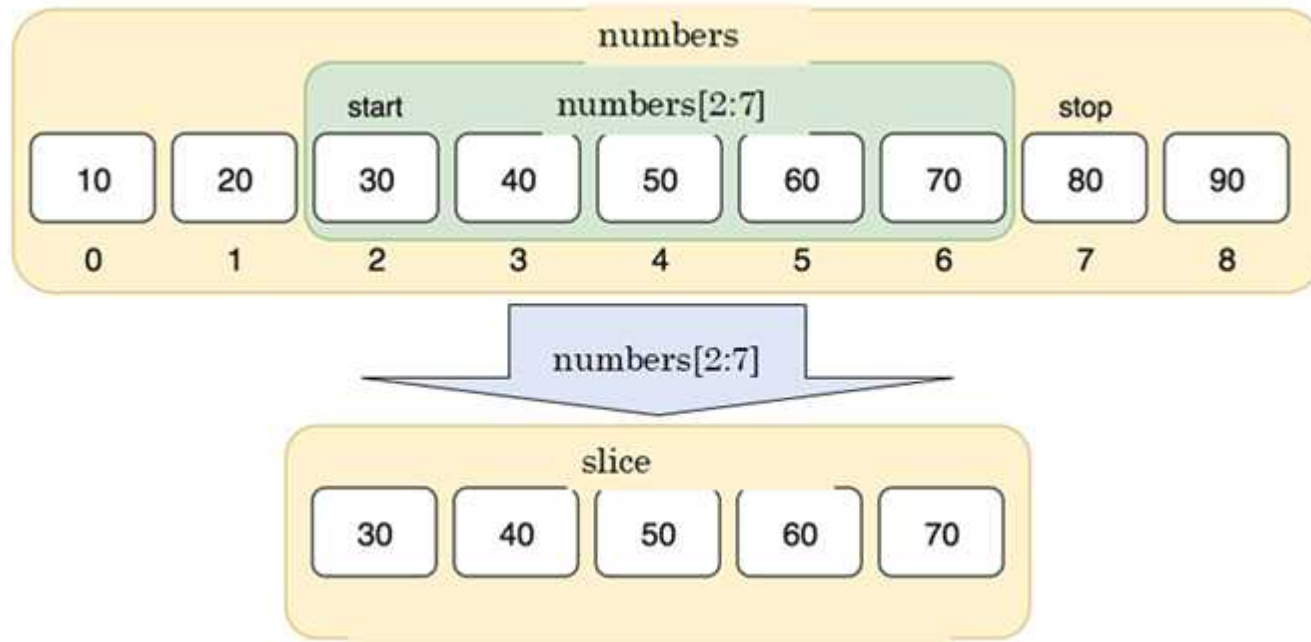
```
temps = [28, 31, 33, 35, 27, 26, 25]  
values = list(temps)
```





슬라이싱

- 슬라이싱(slicing)은 리스트의 일부를 잘라서 추출하는 기법이다. 리스트에서 `numbers[2:7]`와 같이 적으면 인덱스 2부터 시작하여서 항목들을 추출하다가 인덱스 7이 나오기 전에 중지한다.





시작과 끝 인덱스 생략이 가능하다.

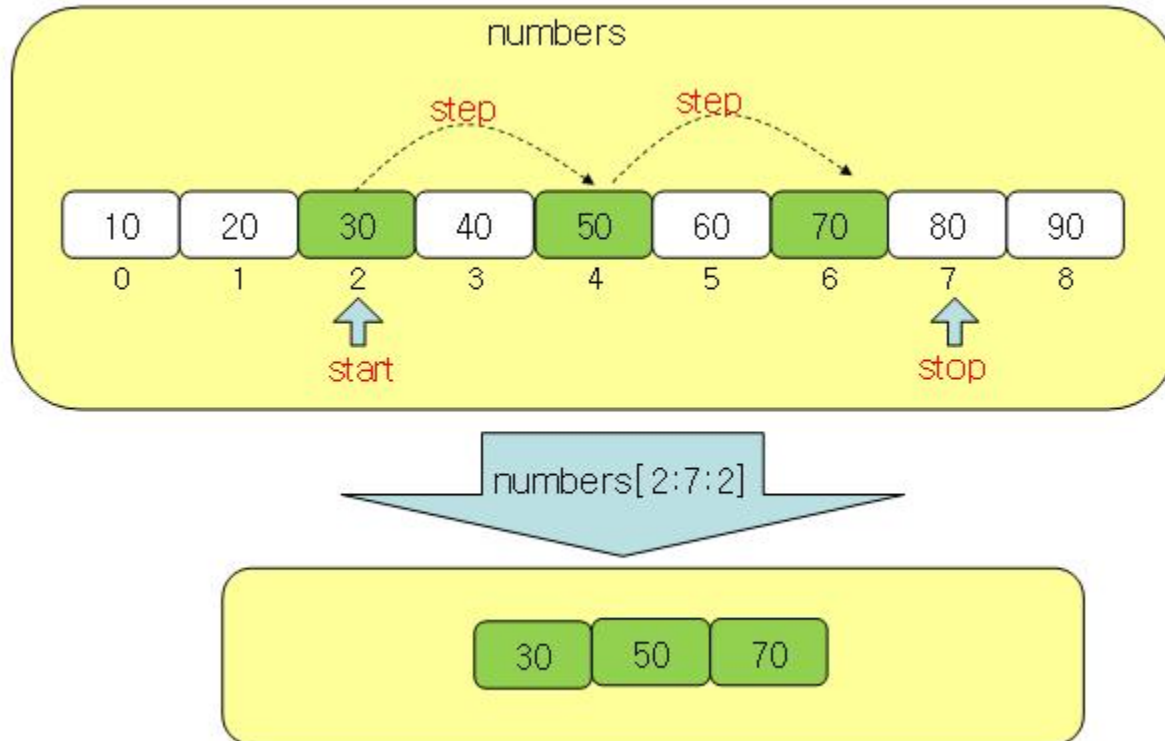
```
numbers[:3]           # [10, 20, 30]
numbers[3:]          # [40, 50, 60, 70, 80, 90]
numbers[:]           # [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

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

깊은 복사가 된다.



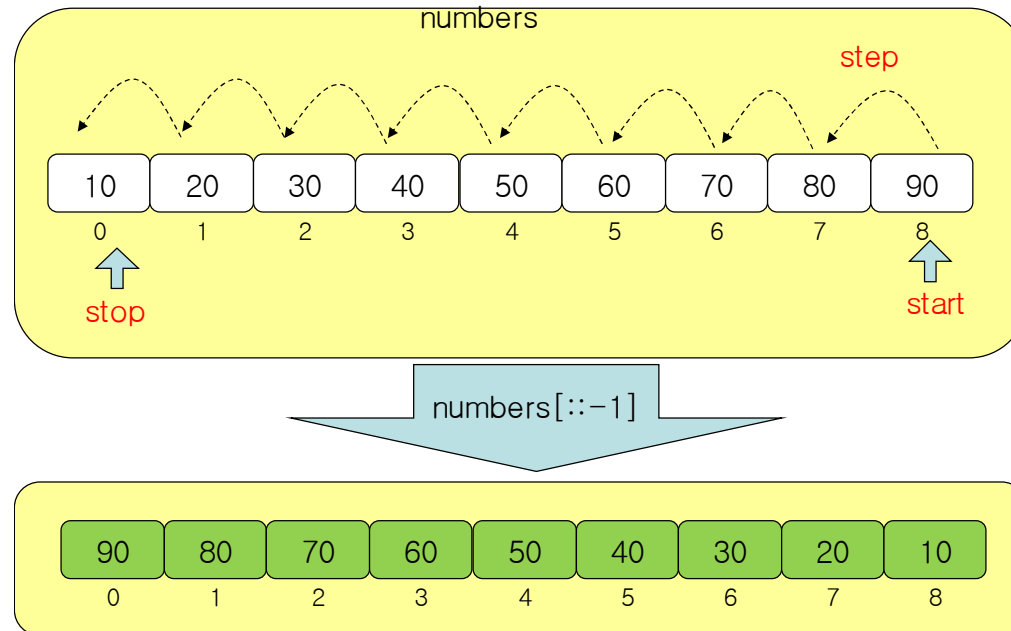
리스트 슬라이싱





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)) # 0에서 시작하여 9까지를 저장하는 리스트  
print(numbers)
```

```
del numbers[-1] # 마지막 항목을 삭제한다.  
print(numbers)
```

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	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
[-12:-7]											

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



Example: 리스트 슬라이싱 연습

- 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)
```



리스트 함축

Syntax: 리스트 함축

Syntax

[수식 for (변수 in 리스트) if (조건)]

출력식으로 새로운 리스트의 요소가 된다.

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

새로운 리스트

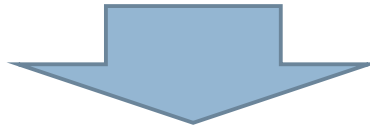
입력 리스트

입력 리스트에 있는 요소 x에 대하여



리스트 함수

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



```
squares = []
```

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

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



리스트 함축

- 리스트 함축에는 if를 사용하여 조건이 추가될 수 있다.

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



리스트 함축

- 리스트 함축에는 if를 사용하여 조건이 추가될 수 있다.

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 합성

```
>>> 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']
```




Example: 리스트 합집 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)
```



Example: 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 함축을 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)
```

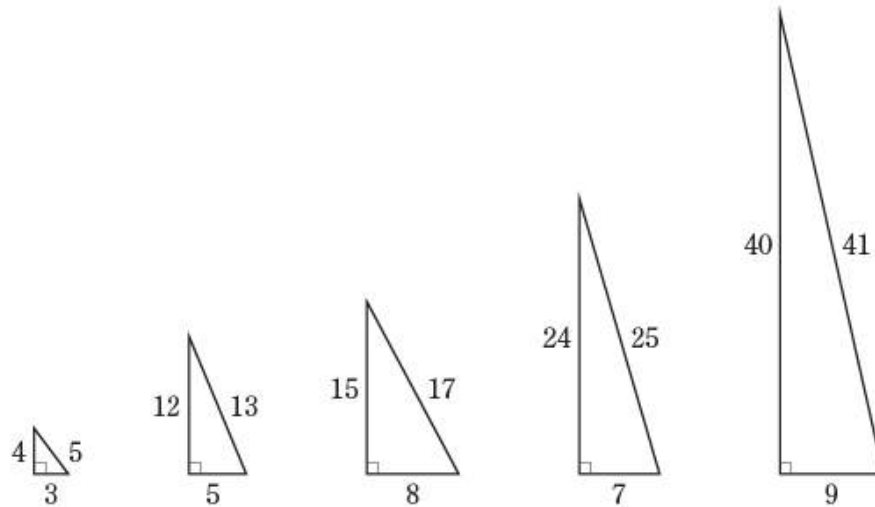
```
print("새로운 리스트: ",list2)
```



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:

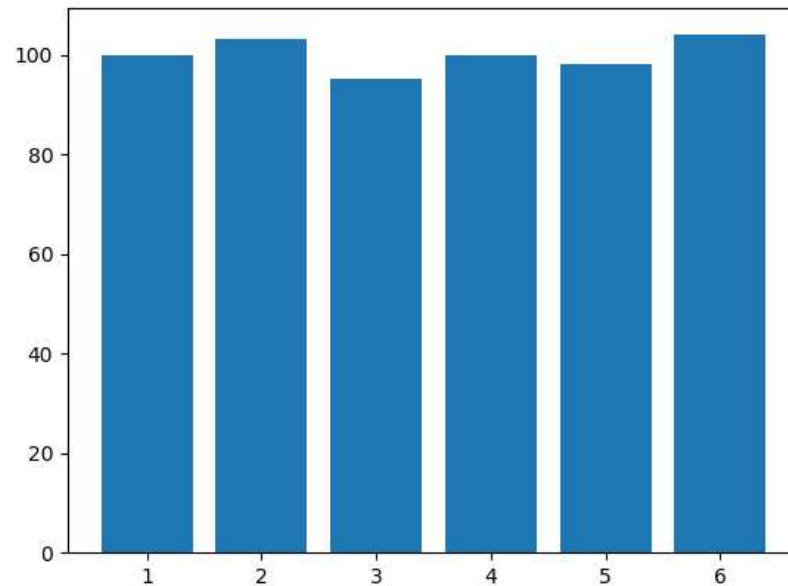
```
[(x,y,z) for x in range(1,30) for y in range(x,30) for z in range(y,30) if  
x**2 + y**2 == z**2]
```

```
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)
```



Lab: 주사위 시뮬레이션

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





Solution:

```
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)
plt.show()
```

리스트에 저장된 값으로 그래프를 그린다.
그래프를 화면에 출력한다.



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)
```

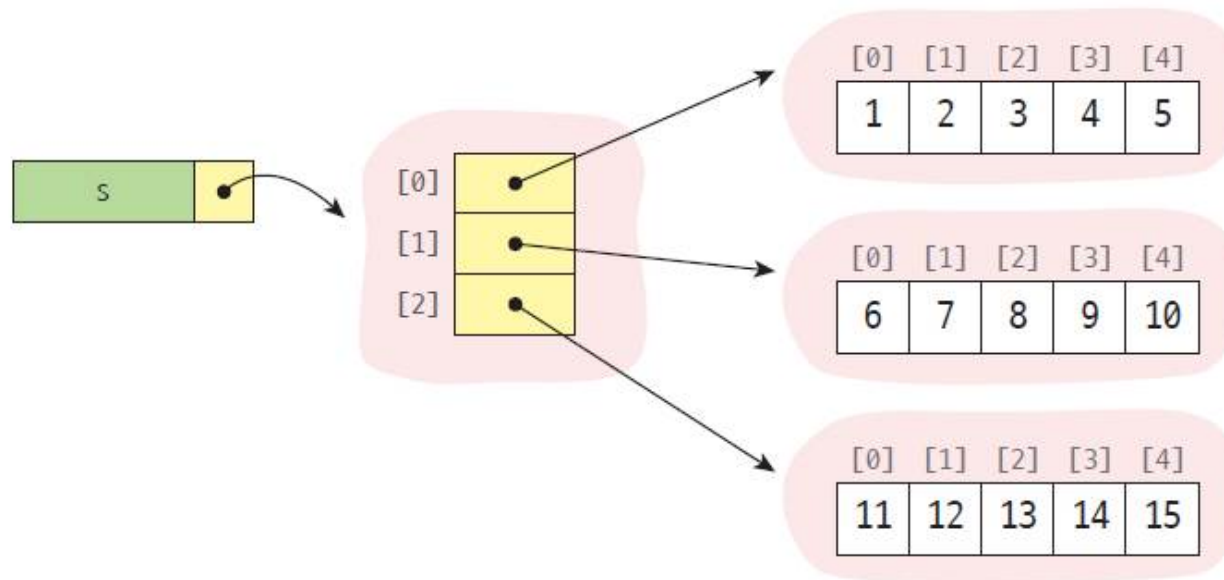
	1열	2열	3열	4열	5열
1행	s[0][0]	s[0][1]	s[0][2]	s[0][3]	s[0][4]
2행	s[1][0]	s[1][1]	s[1][2]	s[1][3]	s[1][4]
3행	s[2][0]	s[2][1]	s[2][2]	s[2][3]	s[2][4]

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



근차원 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

- 다음과 같이 하면 1차원 리스트가 된다.

```
# 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 함축을 이용한 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]]
```



근차원 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,
```



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)
```




Mini Project: 지뢰 찾기

- 2차원의 게임판 안에 지뢰가 숨겨져 있고 이 지뢰를 모두 찾아내는 게임이다. 지뢰가 없는 곳을 클릭했을 때 숫자가 나오면 주변칸에 지뢰가 숨겨져 있다는 것을 의미한다. 예를 들어서 숫자가 2이면 주변 칸에 지뢰가 두개 있다는 의미가 된다.





Mini Project: 지뢰 찾기

- 지뢰찾기 게임을 작성해보자. 10×10 크기의 2차원 리스트를 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.

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



Q & A

