

6장. 리스트 (List)

리스트란?

- 리스트(list)는 여러 개의 데이터가 저장되어 있는 장소

전체적인 구조

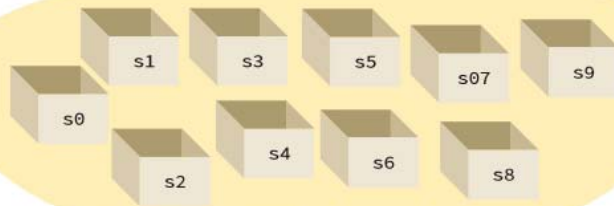


리스트 = [값1 , 값2 , ...]

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

리스트가 필요한 이유

리스트는 하나의 이름을
공유해 자료의 조작이
편리해요^^



별도의 이름을 가지니
조작하기가 어려워!



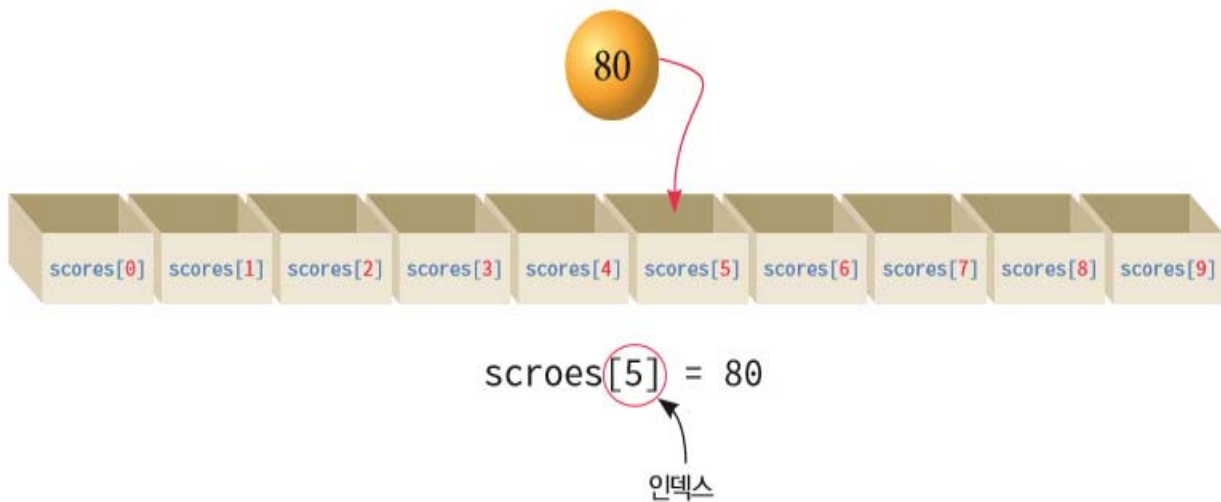
리스트는 하나의 이름을 공유한다.



예제

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

리스트의 요소 접근



예제

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

```
scores[0] = 80;
```

```
scores[1] = scores[0];
```

```
...
```

```
i = 2
```

```
scores[i] = 10
```

i는 정수 변수

```
scores[i+2] = 20
```

수식이 인덱스가 된다.

```
...
```

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

```
    scores[i] = number
```

리스트 순회하기

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 = 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
```

복잡한 리스트

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

Lab: 문자열 처리 프로그램

- 리스트는 문자열도 저장할 수 있다. 강아지를 많이 키우는 사람을 가정하자. 강아지들의 이름을 저장하였다가 출력하는 프로그램을 작성해보자.

강아지의 이름을 입력하시오(종료시에는 엔터키) 미나
강아지의 이름을 입력하시오(종료시에는 엔터키) 초롱이
강아지의 이름을 입력하시오(종료시에는 엔터키) 써니
강아지의 이름을 입력하시오(종료시에는 엔터키) 팅커벨
강아지의 이름을 입력하시오(종료시에는 엔터키)
강아지들의 이름:
미나, 초롱이, 써니, 팅커벨,

Solution

```
dogNames = [ ]  
while True:  
    name = input('강아지의 이름을 입력하시오(종료시에는 엔터키) ')  
    if name == "":  
        break  
    dogNames.append(name)  
  
print('강아지들의 이름:')  
for name in dogNames:  
    print(name, end=" ", )
```

Lab: 성적 처리 프로그램

- 학생들의 성적을 처리하는 프로그램을 완성시켜보자. 사용자로부터 성적을 입력 받아서 리스트에 저장한다. 성적의 평균을 구하고 80점 이상 성적을 받은 학생의 숫자를 계산하여 출력해보자.

```
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.
```

Solution

```
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.")
```

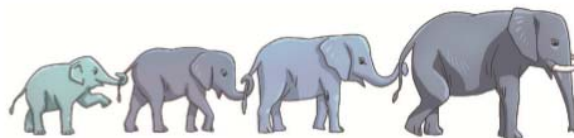
```
print("80점 이상 성적을 받은 학생은", highScoreStudents, "명입니다.")
```

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.



예제

```
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 I .  
apple peach pear
```

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

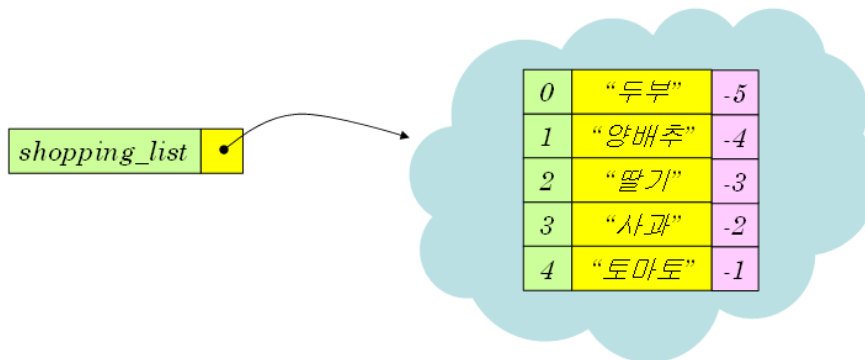
인덱싱과 슬라이싱

- 인덱싱(indexing)이란 리스트에서 하나의 요소를 인덱스 연산자를 통하여 참조(접근)하는 것을 의미한다.

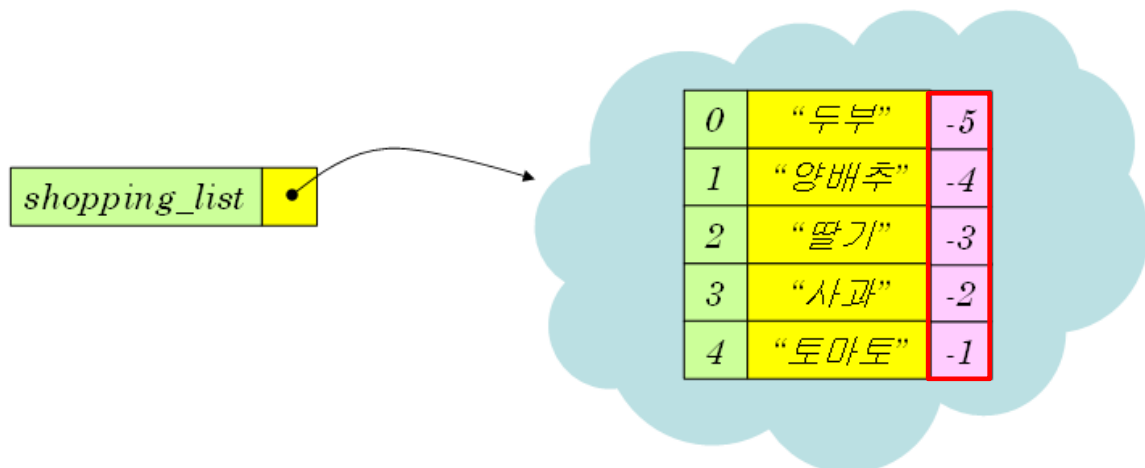
```
>>> shopping_list = [ "두부", "양배추", "딸기", "사과", "토마토" ]
```

```
>>> shopping_list[0]
```

```
'두부'
```

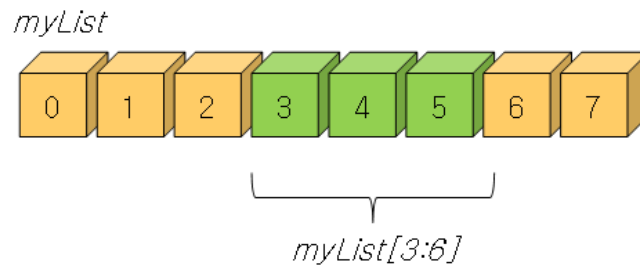


음수 인덱스



슬라이싱

- 슬라이싱(slicing)은 리스트 안에서 범위를 지정하여서 원하는 요소들을 선택하는 연산이다.



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

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]
```

리스트의 길이

- len() 연산은 리스트의 길이를 계산하여 반환한다.

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

요소 추가하기

- `append()`를 사용하여 리스트의 끝에 새로운 항목을 추가할 수 있다.

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

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



요소 찾기

- 어떤 요소가 리스트에 있는지 없는지만 알려면 `in` 연산자를 사용하면 된다.

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



요소 찾기

- 어떤 요소의 리스트 안에서의 위치를 알려면 `index()`을 사용한다.

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



요소 삭제하기

- `pop()` 메소드는 특정한 위치에 있는 항목을 삭제한다.

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

```
>>> heroes = [ "스파이더맨", "슈퍼맨", "헐크", "아이언맨", "배트맨", "조커" ]  
>>> heroes.remove("조커")    # 일치하는 항목의 값을 제거  
>>> heroes  
['스파이더맨', '슈퍼맨', '헐크', '아이언맨', '배트맨']
```

리스트 일치 검사

- 우리는 비교 연산자 `==`, `!=`, `>`, `<`를 사용하여 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 정렬하기

1. 리스트 객체의 `sort()` 메소드를 사용하는 방법
2. `sorted()` 내장 함수를 사용하는 방법

```
>>> a = [ 3, 2, 1, 5, 4 ]
>>> a.sort()           # 원본 리스트가 변경되어 정렬됨
>>> a
[1, 2, 3, 4, 5]
```

```
>>> a = [ 3, 2, 1, 5, 4 ]
>>> b = sorted(a)      # 원본은 변경하지 않고, 새로운 리스트를 만들어 정렬
>>> b
[1, 2, 3, 4, 5]
```

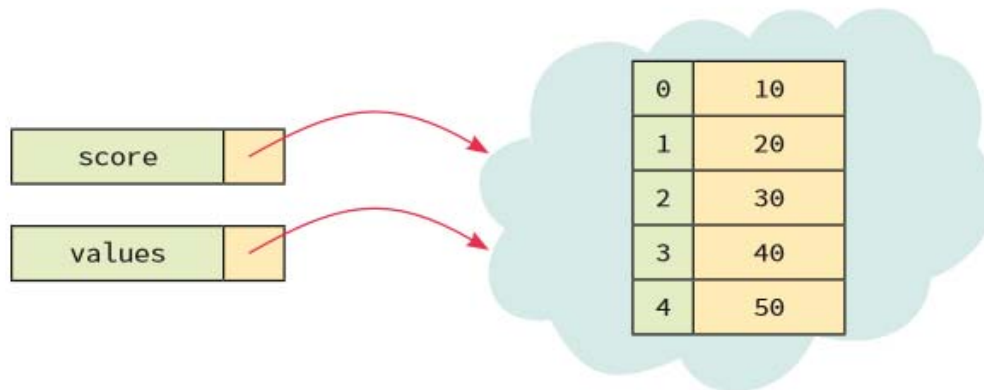
리스트 연산 정리

연산의 예	설명
<code>mylist[2]</code>	인덱스 2에 있는 요소
<code>mylist[2] = 3</code>	인덱스 2에 있는 요소를 3으로 설정한다.
<code>del mylist[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 복사하기

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

```
values = scores
```



얕은 복사(shallow copy)

깊은 복사

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

```
>>> values = list(scores)
```

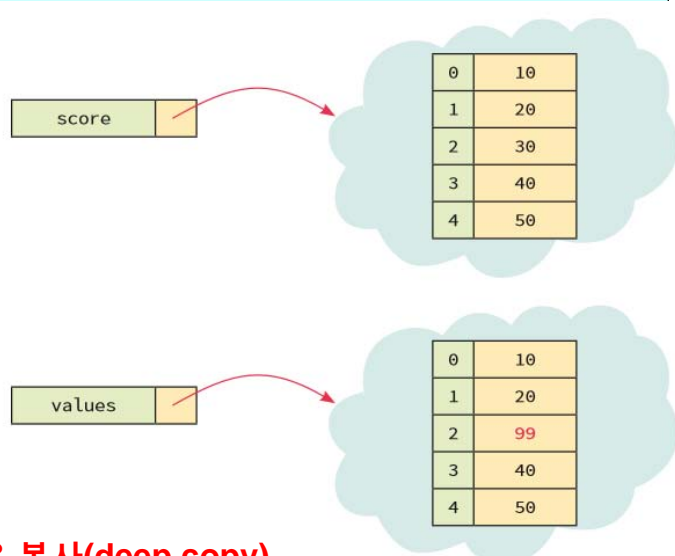
```
>>> values[2]=99
```

```
>>> scores
```

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

```
>>> values
```

```
[10, 20, 99, 40, 50]
```



깊은 복사(deep copy)

리스트와 함수

- 리스트는 “참조로 호출하기”가 적용된다.

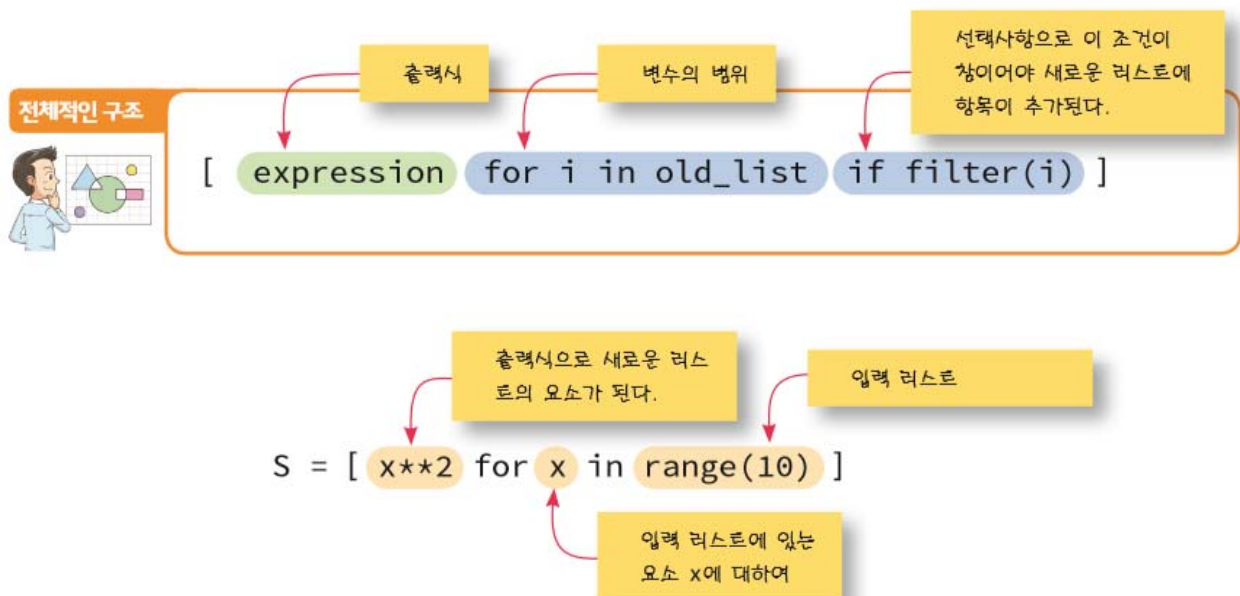
```
def func2(list):  
    list[0] = 99  
  
values = [0, 1, 1, 2, 3, 5, 8]  
print(values)  
func2(values)  
print(values)
```

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

← 리스트는 변경 가능한 객체이므로

리스트 함축 (List Comprehension)

- 리스트를 수학자들이 집합을 정의하는 것과 유사하게 생성



예 제

```
list1 = [3, 4, 5]
```

```
list2 = [x*2 for x in list1]
```

```
print(list2)
```

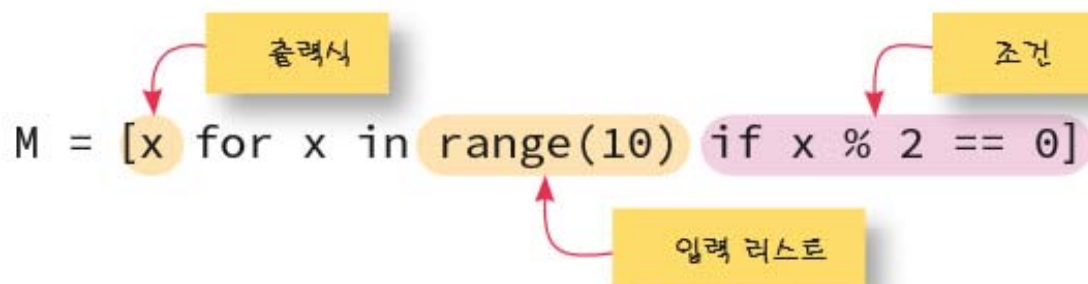
```
[6, 8, 10]
```

조건이 붙는 리스트 함축

출력식 조건

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

입력 리스트



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

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(a)  
cols = len(a[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,
```

핵심 정리

- 리스트는 많은 데이터를 저장할 수 있는 자료 구조이다.
리스트에 요소를 추가할 때는 `append()`를 사용한다.
- 리스트의 길이는 `len()`로 계산할 수 있고 리스트의 요소들을 정렬하려면 `sort()`를 호출한다.
- 리스트도 2차원으로 만들 수 있다. 2차원 리스트는 1차원 리스트의 모임으로 구성된다.