

# 자료구조

리스트

---

# 리스트 (List)

- 리스트

- 말 그대로, 여러 자료들을 목록 형태로 관리하기 위한 자료 구조이다.
- 순서가 있는 여러 개의 값을 저장하고 관리한다.

이름	국어	수학	영어
홍길동	91	79	56
성춘향	80	92	62
김철수	73	66	81
이순신	100	85	43

홍길동의 점수들에 대한 목록 :  
91, 79, 56

학생들의 국어 점수에 대한 목록 : 91, 80, 73, 100

# 리스트 (List)

- 리스트 만들기

**리스트의 이름** = [ 항목1, 항목2, ... ]

- 리스트는 여러 값을 쉼표 ‘,’ 기호로 구분하여 기재하고 그 값들의 양 쪽 끝을 대괄호 ‘[ ]’ 기호로 씌워서 만들 수 있다.
- 또는 명령어 list를 이용해서 만들 수 있다.

$a = [1, 2, 3, 4, 5, 6, 7]$

a	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

※ 1개의 자료 구조 안에 들어 있는 각각의 값들을 항목(item) 또는 원소/요소(element)라고 한다.

# 리스트 (List)

- 리스트 만들기

리스트의 이름 = [ 항목1, 항목2, ... ]

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2
3 b = ["Hi", "bye"]
4
5 c = [1, 3, "Hi", "bye", True]
6
7 d = [1, 3, ["Hi", "bye"], True]
8
9 e = []
10
11 f = list()
```

```
1 print(a)
2 print(b)
3 print(c)
4 print(d)
5 print(e)
6 print(f)
```

```
[1, 2, 3, 4, 5, 6, 7]
['Hi', 'bye']
[1, 3, 'Hi', 'bye', True]
[1, 3, ['Hi', 'bye'], True]
[]
[]
```

# 리스트 (List)

- 리스트 만들기

리스트의 이름 = [ 항목1, 항목2, ... ]

b = ["Hi", "bye"]

b	"Hi"	"bye"
---	------	-------

c = [1, 3, "Hi", "bye", True]

c	1	3	"Hi"	"bye"	True
---	---	---	------	-------	------

d = [1, 3, ["Hi", "bye"], True]

d	1	3	"Hi"	"bye"	True
---	---	---	------	-------	------

# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기
  - 리스트에 있는 여러 항목들은 모두 각각 그 위치가 숫자로 매겨져 있다.
  - 이 번호는 0부터 시작하는 정수이다.

a	1	2	3	4	5	6	7
번호	0	1	2	3	4	5	6

리스트 a에 있는 정수 값 1은

- 리스트 a의 첫 번째 항목(원소)이다.
- 리스트 a의 0번 항목(원소)이다.
- 인덱스가 0인 항목(원소)이다.

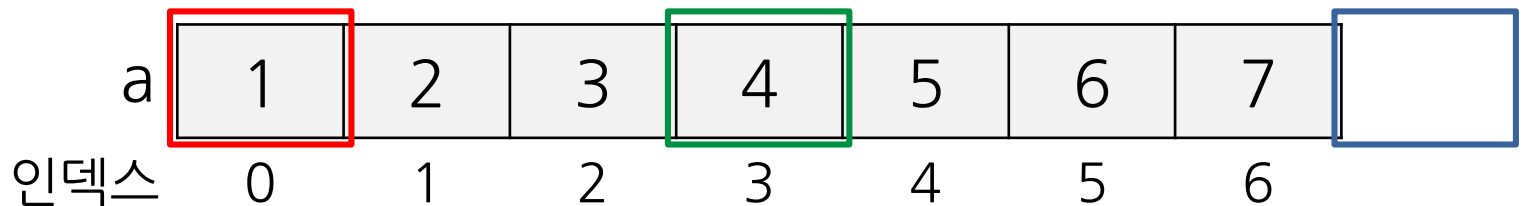
※ 자료 구조에서 각 항목의 위치에 해당하는 숫자를 인덱스 (index)라고 한다.

# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기

**리스트의 이름**[**항목의 인덱스**]

- 리스트에 있는 항목 1개를 특정(specify)하려면 리스트 이름 뒤에 그 항목의 인덱스를 기재하고 양쪽 끝을 대괄호 '[' ]' 기호로 씌워서 접근할 수 있다.



$a[0]$  : 정수 값 1

$a[3]$  : 정수 값 4

$a[7]$  : 오류

※ 가져온 항목은 1개의 자료이다.

- ※ 이렇게 자료 구조 내의 1개의 항목에 접근하는 것을 인덱싱(indexing)이라고 한다.

# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기

리스트의 이름[항목의 인덱스]

c = [1, 3, "Hi", "bye", True]

c	1	3	"Hi"	"bye"	True
인덱스	0	1	2	3	4

```
1 c = [1, 3, "Hi", "bye", True]
2
3 print(c[0])
4 print(c[2])
5 print(c[4])
```

```
1
Hi
True
```



# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기

리스트의 이름[항목의 인덱스]

d = [1, 3, ["Hi", "bye"], True]

d	1	3	"Hi"	"bye"	True
인덱스	0	1	2	3	

```
1 d = [1, 3, ["Hi", "bye"], True]
2
3 print(d[0])
4 print(d[2])
5 print(d[2][0])
6 print(d[2][1])
```

```
1
['Hi', 'bye']
Hi
bye
```

# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기

**리스트의 이름**[**항목의 인덱스**]

- 인덱스를 음의 정수로 표현할 수도 있다.
- 음수로 표현하면 자료 구조의 뒷부분부터 접근한다.

a	1	2	3	4	5	6	7
인덱스	0	1	2	3	4	5	6
	-7	-6	-5	-4	-3	-2	-1
	a[0]			a[3]			a[6]
	a[-7]			a[-4]			a[-1]

# 리스트 (List)

- 리스트 내의 1개의 항목 접근하기

리스트의 이름[항목의 인덱스]

d = [1, 3, ["Hi", "bye"], True]

d	1	3	"Hi"	"bye"	True
인덱스	0	1	2	3	
	-4	-3	-2	-1	

```
1 d = [1, 3, ["Hi", "bye"], True]
2
3 print(d[-1])
4 print(d[-2])
5 print(d[-3] - d[0])
6 print(d[-2][0] + d[-2][-1])
```

```
True
['Hi', 'bye']
2
Hibye
```

# 리스트 (List)

- 리스트 내의 여러 개의 항목 접근하기

**리스트의 이름**[시작 인덱스:끝 인덱스:간격]

- 리스트에서 1개의 인덱스 번호 대신 시작 지점, 종료 지점, 간격을 지정하여 여러 개의 항목을 접근할 수 있다.

a	1	2	3	4	5	6	7
인덱스	0	1	2	3	4	5	6

$a[0:3:1]$  : 리스트 a에서  
0번 항목부터  
3번 항목 **전까지**  
1개 간격으로 가져온다.

※ 가져온 항목들은  
1개의 리스트이다.

- ※ 이렇게 자료 구조 내의 여러 개의 항목에 접근하는 것을 슬라이싱(slicing)이라고 한다.

# 리스트 (List)

- 리스트 내의 여러 개의 항목 접근하기

**리스트의 이름**[시작 인덱스:끝 인덱스:간격]

- 슬라이싱할 때 수치들은 상황에 따라 생략할 수 있다.
  - 시작 인덱스 : 생략할 경우 기본 값은 0이다.
  - 끝 인덱스 : 생략할 경우 마지막 위치까지 포함한다.
  - 간격 : 생략할 경우 기본 값은 1이다.

a	1	2	3	4	5	6	7
인덱스	0	1	2	3	4	5	6

$a[0:3:1]$ 과  $a[:3:]$ 과  $a[:3]$ 은 결과가 모두 동일하다.

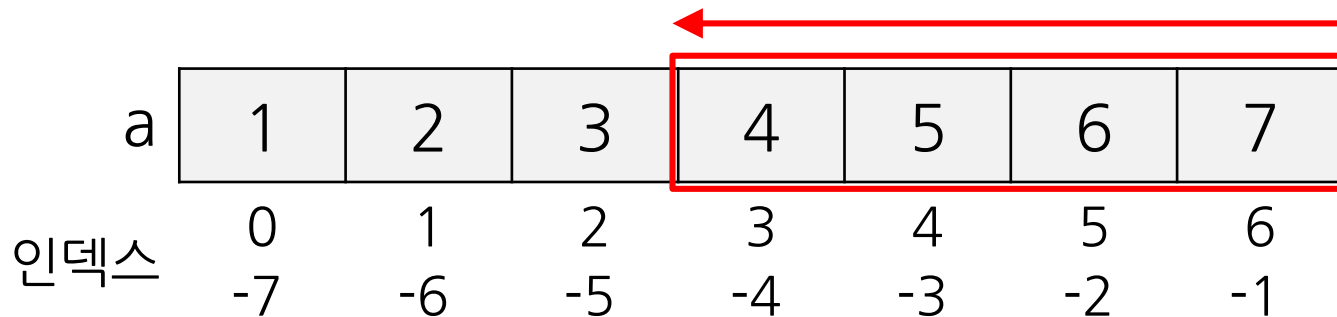
- ※ 간격을 생략할 때는 간격 앞에 있는 콜론 “:” 기호를 함께 생략할 수 있다.

# 리스트 (List)

- 리스트 내의 여러 개의 항목 접근하기

**리스트의 이름**[**시작 인덱스**:**끝 인덱스**:**간격**]

- 슬라이싱할 때 간격을 음의 정수로 표현할 수도 있다.
- 음수로 표현하면 뒤 쪽부터 앞 쪽 방향으로 진행한다.



a	1	2	3	4	5	6	7
인덱스	0	1	2	3	4	5	6
	-7	-6	-5	-4	-3	-2	-1

$a[-1:2:-1]$  : 리스트 a에서  
-1번 (즉, 마지막) 항목부터  
2번 항목 **전까지**  
1개씩 반대 방향으로 가져온다.

# 리스트 (List)

- 리스트 내의 여러 개의 항목 접근하기

**리스트의 이름**[**시작 인덱스**:**끝 인덱스**:**간격**]

c = [1, 3, "Hi", "bye", True]

c	1	3	"Hi"	"bye"	True
인덱스	0	1	2	3	4
	-5	-4	-3	-2	-1

```
1 c = [1, 3, "Hi", "bye", True]
2
3 print(c[1:])
4 print(c[:-1])
5 print(c[-3:2])
6 print(c[-3::-1])
```

[3, 'Hi', 'bye', True]

[1, 3, 'Hi', 'bye']

['Hi', True]

['Hi', 3, 1]

# 리스트 (List)

- 리스트 내의 여러 개의 항목 접근하기
  - 슬라이싱 정리

리스트의 이름[시작 인덱스 : 끝 인덱스 : 간격]

	시작 인덱스	끝 인덱스	간격
의미	시작 위치부터	끝 위치가 되기 전까지	이 간격만큼 씩 가져온다.
0 또는 양수인 경우	왼쪽부터 위치를 판단한다.	왼쪽부터 위치를 판단한다.	왼쪽부터 오른쪽으로 진행한다.
음수인 경우	오른쪽부터 위치를 판단한다.	오른쪽부터 위치를 판단한다.	오른쪽에서 왼쪽으로 진행한다.
생략할 경우	시작 위치가 0, 즉 처음 항목이다.	끝 위치가 마지막 항목을 포함한다.	간격이 1이다.



# 리스트 (List)

- 리스트에 대해 덧셈 및 곱셈 연산하기

연산자	참고사항
+	리스트끼리의 덧셈이 가능하다.
*	리스트와 정수의 곱셈이 가능하다.

```
1 a = [1, 2]
2
3 b = [1, 3, 5]
4
5 print(a + b)
6
7 print(2 * a)
8
9 print(b + [7, 9])
```

```
[1, 2, 1, 3, 5]
[1, 2, 1, 2]
[1, 3, 5, 7, 9]
```

```
1 a = [1, 2]
2 b = [1, 3, 5]
3
4 print(a + b[0:1])
```

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

```
1 a = [1, 2]
2 b = [1, 3, 5]
3
4 print(a + b[0])
```

TypeError

# 리스트 (List)

- 리스트에 대해 크기 비교하기

연산자	참고사항
< , > , <= , >=	리스트끼리 비교 가능하다.
== , !=	다른 자료형과 리스트 간의 비교가 가능하다.

- 2개의 리스트에서 같은 위치의 항목, 즉 동일한 인덱스의 항목 값을 비교하여 리스트의 대소를 판단한다.

```
1 [1, 2] < [1, 2, 3]
```

True

```
1 [2, 4, 6] != "2, 4, 6"
```

True

```
1 [1, 3, 5, 7, 9] >= [2.5, 3.1415926]
```

False

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

True

# 리스트 (List)

- 리스트에 특정 값이 존재하는지 확인하기

**찾으려는 값** in **리스트의 이름**

- 리스트에 특정 값이 있는지 검사하여, 값이 그 리스트에 들어 있다면 결과로 참(True)을 되돌려 준다.

```
1 a = [1, 2, 3]
2
3 1 in a
```

True

```
1 b = ["1", "2", "3"]
2
3 3 in b
```

False

**찾으려는 값** not in **리스트의 이름**

- 리스트에 특정 값이 있는지 검사하여, 값이 그 리스트에 들어 있다면 결과로 거짓(False)을 되돌려 준다.

```
1 a = [1, 2, 3]
2
3 1 not in a
```

False

```
1 b = ["1", "2", "3"]
2
3 3 not in b
```

True

# 리스트 (List)

- 리스트의 특정 항목 값 수정하기

**리스트의 이름[항목의 인덱스] = 수정할 값 또는 연산**

– 항목을 인덱싱하여 원하는 값을 할당한다.

```
1 a = [1, 2, 3]
2 print(a)
3
4 a[1] = 0
5 a[2] = -1
6 print(a)
```

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

```
1 a = [1, 2, 3]
2 print(a)
3
4 a[1:] = [0, -1]
5 print(a)
```

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

※ 슬라이싱하여 한 번에 여러 개의 항목 값을 수정할 수도 있다.

# 리스트 (List)

- 리스트의 끝에 새로운 항목 1개 추가하기  
**리스트의 이름.append(추가할 값 또는 연산)**
  - 대상 리스트의 마지막 위치에 항목 1개를 추가한다.

```
1 a = [1, 2, 3]
2
3 a.append(4)
4 print(a)
5
6 a.append("python")
7 print(a)
8
9 a.append(True)
10 print(a)
11
12 a.append([-1, -2])
13 print(a)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4, 'python']
[1, 2, 3, 4, 'python', True]
[1, 2, 3, 4, 'python', True, [-1, -2]]
```

# 명령어의 형태

- 명령어의 두 가지 형태

- 어떤 명령어들은 특정 자료 구조에서만 실행될 수 있다. 이러한 경우, 명령어의 앞에 수행 대상을 지정한다.

수행할 작업 대상. **명령어**(필요한 값)

`a.append(4)`

리스트 `a` 에 정수 값 `4` 를 `append` 하라

- 반면에, 어떤 명령어들은 특정 대상이 지정되지 않고 범용으로 사용된다.

**명령어**(필요한 값)

`print("Hello")`

문자열 `"Hello"` 를 화면에 `print` 하라

# 리스트 (List)

- 리스트의 특정 위치에 새로운 항목 1개 추가하기  
**리스트의 이름.insert(추가할 위치, 추가할 값)**
  - 대상 리스트의 특정 위치에 항목 1개를 추가하고, 원래 그 위치에 있던 항목들을 한 칸 씩 뒤로 밀어낸다.

```
1 a = [1, 2, 3]
2
3 a.insert(0, 4)
4 print(a)
5
6 a.insert(2, "python")
7 print(a)
8
9 a.insert(-1, True)
10 print(a)
```

[4, 1, 2, 3]

[4, 1, 'python', 2, 3]

[4, 1, 'python', 2, True, 3]

# 리스트 (List)

- 리스트에서 항목 삭제하기

- ① **del** 리스트의 이름[삭제하려는 항목의 인덱스]

- 명령어 del을 이용하여 인덱싱한 항목을 삭제할 수 있다.

1	a = [1, 2, 3, 4, 5]
2	
3	del a[2]
4	print(a)
5	
6	del a[3]
7	print(a)

[1, 2, 4, 5]

[1, 2, 4]

1	a = [1, 2, 3, 4, 5]
2	
3	del a[2:4]
4	print(a)

[1, 2, 5]

※ 슬라이싱하여 한 번에 여러 개의 항목 값을 삭제할 수도 있다.



# 리스트 (List)

---

- 리스트에서 항목 삭제하기

- ② **리스트의 이름.remove(삭제하려는 항목 값)**

- 대상 리스트에서 첫 번째로 나오는 특정 값을 삭제한다.

```
1 a = [1, 2, 3, 2, 1]
2
3 a.remove(2)
4 print(a)
5
6 a.remove(2)
7 print(a)
```

[1, 3, 2, 1]

[1, 3, 1]

# 리스트 (List)

- 리스트에서 항목 삭제하기

- ③ 리스트의 이름.pop(삭제하려는 항목의 인덱스)

- 대상 리스트에서 인덱싱한 항목을 꺼내서 가져온다.
    - 인덱스를 생략할 수 있으며, 이 경우에는 마지막 항목을 꺼내서 가져온다.
    - 항목 값을 단순히 지우는 것이 아니라 꺼내서 되돌려 주기 때문에, 그 값을 변수에 할당하고 사용할 수 있다.

```
1 a = [1, 2, 3, 4, 5]
2
3 a.pop(2)
4 print(a)
5
6 a.pop()
7 print(a)
```

[1, 2, 4, 5]  
[1, 2, 4]

```
1 a = [1, 2, 3, 4, 5]
2
3 b = a.pop(2)
4 print(a)
5 print(b)
```

[1, 2, 4, 5]  
3

# 리스트 (List)

---

- 리스트의 항목들 정렬하기

## 리스트의 이름.sort()

- 대상 리스트의 항목들의 순서를 오름차순으로 정렬한다.
- 리스트 내의 항목들은 모두 서로 크기 비교가 가능한 자료형들만으로 구성되어 있어야 동작한다.

```
1 a = [4, 2, 5, 3, 1]
2
3 a.sort()
4 print(a)
5
6 b = ["python", "kim", "Python", "apple"]
7
8 b.sort()
9 print(b)
```

[1, 2, 3, 4, 5]

['Python', 'apple', 'kim', 'python']

# 리스트 (List)

- 리스트의 항목들의 순서 뒤집기

## 리스트의 이름.reverse()

- 대상 리스트의 항목들을 역순으로 배치한다.
- 정렬을 하는 것이 아니라, 원래의 순서를 거꾸로 뒤집는 기능만 수행하는 것이다.

```
1 a = [4, 2, 5, 3, 1]
2
3 a.reverse()
4 print(a)
5
6 b = ["python", "kim", "Python", "apple"]
7
8 b.reverse()
9 print(b)
```

[1, 3, 5, 2, 4]

['apple', 'Python', 'kim', 'python']

# 리스트 (List)

- 리스트에서 특정 항목의 개수 세기

**리스트의 이름.count(개수를 세려는 항목 값)**

- 대상 리스트에서 특정 항목이 몇 개 존재하는지 확인하여 그 개수를 되돌려 준다.
- 개수를 세서 출력하는 것이 아니라 되돌려 주기 때문에, 그 값을 변수에 할당하고 사용할 수 있다.

```
1 a = [1, 2, 3, 2, 1]
2
3 b = a.count(1)
4 print(b)
5
6 c = a.count(7)
7 print(c)
```

2  
0

# 리스트 (List)

- 리스트의 길이 구하기

**len(리스트의 이름)**

- 대상 리스트에서 항목들이 모두 몇 개 존재하는지 확인하여 그 개수를 되돌려 준다.
- 개수를 세서 출력하는 것이 아니라 되돌려 주기 때문에, 그 값을 변수에 할당하고 사용할 수 있다.

```
1 t1 = ["Hi", "bye"]
2 a = len(t1)
3 print(a)
4
5 t2 = [1, 3, "Hi", "bye", True]
6 b = len(t2)
7 print(b)
```

2  
5

```
1 t3 = [1, 3, ["Hi", "bye"], True]
2 c = len(t3)
3 print(c)
4
5 t4 = []
6 d = len(t4)
7 print(d)
```

4  
0

# 리스트 (List)

- 리스트에 있는 항목들의 합계 구하기

`sum(리스트의 이름)`

– 대상 리스트에 들어 있는 항목들의 총 합을 구한다.

```
1 t1 = [1, 2, 3]
2 a = sum(t1)
3 print(a)
4
5 t2 = [3.14, -0.5, 123, 99]
6 b = sum(t2)
7 print(b)
```

6  
224.64

```
1 t3 = ["a", "b", "c"]
2 c = sum(t3)
3 print(c)
```

-----  
**TypeError**

※ 이 때, 리스트는 산술적인 덧셈 연산이 가능한 항목들로만 구성되어 있어야 한다.

# 리스트 (List)

- 리스트에 있는 항목들 중 최대값 구하기

`max(리스트의 이름)`

– 대상 리스트에 들어 있는 항목들 중 최대값을 구한다.

```
1 t1 = [1, 2, 3]
2 a = max(t1)
3 print(a)
4
5 t2 = [3.14, -0.5, 123, 99]
6 b = max(t2)
7 print(b)
```

3  
123

```
1 t3 = ["a", "b", "c"]
2 c = max(t3)
3 print(c)
4
5 t4 = [1, 3, "python", True]
6 d = max(t4)
7 print(d)
```

c

**TypeError**

※ 이 때, 리스트는 서로 크기 비교가 가능한 항목들로만 구성 되어 있어야 한다.



# 리스트 (List)

- 리스트에 있는 항목들 중 최소값 구하기

**min(리스트의 이름)**

– 대상 리스트에 들어 있는 항목들 중 최소값을 구한다.

```
1 t1 = [1, 2, 3]
2 a = min(t1)
3 print(a)
4
5 t2 = [3.14, -0.5, 123, 99]
6 b = min(t2)
7 print(b)
```

1  
-0.5

```
1 t3 = ["a", "b", "c"]
2 c = min(t3)
3 print(c)
4
5 t4 = [1, 3, "python", True]
6 d = min(t4)
7 print(d)
```

a

**TypeError**

※ 이 때, 리스트는 서로 크기 비교가 가능한 항목들로만 구성 되어 있어야 한다.

# 리스트 (List)

- 리스트 자료형 확인하기

**type(자료형을 확인하고 싶은 1개의 자료)**

- 명령어 type을 이용하여 값 또는 변수가 어떤 자료형인지 확인할 수 있다.

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 print(type(a))
3 print(type(a[2]))
4
5 b = [1, 3, "Hi", "bye", True]
6 print(type(b))
7 print(type(b[-2]))
8
9 c = []
10 print(type(c))
```

```
<class 'list'>
<class 'int'>
<class 'list'>
<class 'str'>
<class 'list'>
```

# 리스트 (List)

- 리스트에 다른 리스트 결합하기

**리스트의 이름.extend(결합할 1개의 리스트)**

- 대상 리스트의 끝에 다른 리스트를 결합하여 확장한다.
- 2개의 리스트를 결합하는 것으로, 리스트에 대한 덧셈 연산과 동일한 기능이다.

```
1 a = [1, 2]
2 b = [1, 3, 5]
3
4 a.extend(b)
5 print(a)
6
7 a.extend(a)
8 print(a)
```

[1, 2, 1, 3, 5]

[1, 2, 1, 3, 5, 1, 2, 1, 3, 5]

```
1 a = [1, 2]
2
3 a.extend(["a", "b"])
4 print(a)
```

[1, 2, 'a', 'b']

```
1 a = [1, 2]
2
3 a.extend("a", "b")
4 print(a)
```

**TypeError**