파이썬 자료 구조

- 자료 구조(data structure)란?
  - 자료를 표현하고 처리하기 위해 사용되는 구조
  - 자료를 효율적으로 처리하고 저장하는 데 사용
  - 파이썬에서 사용되는 기본 자료 구조
    - 문자열(string)
    - 리스트(list)
    - 튜플(tuple)
    - 딕셔너리(dictionary)
    - 집합(set)

리스트(list)

리스트(list)

- 리스트는 여러 개의 데이터를 하나의 이름으로 저장하는 자료 구조

리스트 기본 구조

```
리스트이름 = [항목1, 항목2, ...]
```

```
odd = [1, 3, 5, 7, 9]          ▶[ ]를 이용하여 리스트 생성
squares = [1, 4, 9, 16, 25]
print(squares)
[1, 4, 9, 16, 25]

empty = []                     ▶공백 리스트. empy = list()
print(empty)
[]

a = [1, 'hello', 2, 'world', 1.23]     ▶리스트 요소는 어떤 것이라도 가능
```

리스트(list)

- 리스트의 인덱싱(요소 접근하기)과 슬라이싱(잘라내기)

```
squares = [1, 4, 9, 16, 25]
print(squares[0])          ▶ 리스트 요소 추출하기(인덱싱은 0부터)
1
print(squares[-1])         ▶마지막 요소
25
print(squares[0:3])        ▶0부터(3-1)까지 잘라내기
[1, 4, 9]
print(squares[-3:])        ▶뒤에서 3번째부터 마지막까지 잘라내기
[9, 16, 25]
```

- 리스트 합치기

```
squares + [36, 49, 64, 81, 100]          ▶ 리스트 합치기
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
a = [1, 2, 3]
b = [4, 5, 6]
print(a + b)               ▶ 리스트 합치기
[1, 2, 3, 4, 5, 6]
```

리스트(list)

- 리스트의 요소 수정, 추가하기

```
❶ 수정하기
cubes = [1, 8, 27, 65, 125]     #세제곱 수
print(4 ** 3)     # 4의 세제곱은 64!
64

cubes[3] = 64     # 리스트 요소 수정하기
print(cubes)      #수정된 리스트
[1, 8, 27, 64, 125]

❷ 마지막에 요소 추가하기
cubes.append(216)       # 마지막에 리스트 요소 추가하기
cubes.append(7 ** 3)    # 마지막에 리스트 요소 추가하기
print(cubes)
[1, 8, 27, 64, 125, 216, 343]
```

리스트(list)

- 리스트의 요소 수정

❶ 요소 수정하기

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
['a', 'b', 'c', 'd', 'e', 'f', 'g']

letters[2:5] = ['C', 'D', 'E']          ▶리스트 요소 수정
print(letters)
['a', 'b', 'C', 'D', 'E', 'f', 'g']

letters[2:5] = []          ▶ 리스트 요소 제거
print(letters)

['a', 'b', 'f', 'g']
letters[:] = []          ▶ 리스트 요소를 모두 제거
print(letters)
[]
```

리스트(list)

- 중첩 리스트

```
a = ['a', 'b', 'c']
n = [1, 2, 3]
x = [a, n]
print(x)
[['a', 'b', 'c'], [1, 2, 3]]        ▶리스트 요소가 리스트이다
print(x[0])
['a', 'b', 'c']
print(x[0][1])        ▶중첩 리스트 요소 지정
'b'
```

- 2차원 리스트

```
list2d = [[1,2,3],        ▶배열을 나타낼 때 편리
[4,5,6],
[7,8,9]]
print(list2d[0][0])
1
print(list2d[2][2])
9
```

리스트(list)

- 내장 함수를 이용한 리스트 다루기

```
❶ 리스트 길이 알아내기
letters = ['a', 'b', 'c', 'd']
print(len(letters))
4

❷ 리스트 요소 합, 최소값, 최대값 구하기
num = [1, 2, 3, 4, 5]
print(sum(num))
15
print(min(num))
1
print(max(num))
5
```

리스트(list) 메소드

- 리스트 메소드(List.Method)

함수명	사용 방법	설명
del()	del(List[pos])	List에서 pos 위치의 항목 삭제
len()	len(List)	List 항목의 전체 개수를 센다
sorted()	newList = sorted(List)	List 항목을 정렬해 newList 만듬
append()	List.append(val)	List 맨 뒤에 val 항목 추가
clear()	List.clear()	List의 내용을 지움
copy()	newList = List.copy()	List의 내용을 newList에 복사
count()	List.count(val)	List에서 val 값의 갯수를 센다
extend()	List.extend(newList)	List 뒤에 newList를 추가
index()	List.index(val)	val을 찾아 해당 첨자 반환
insert()	List.insert(pos, val)	List의 pos 위치에 val 삽입
pop()	List.pop()	List 맨 뒤에서 항목 삭제
remove()	List.remove(val)	List에서 val 값 항목 삭제(여러 개면 첫번째만)
reverse()	List.reverse()	List 항목을 뒤집어 역순으로 만듬
sort()	List.sort()	List 항목을 정렬

사용법 확인하기
help(list.sort)

리스트(list) 순서

- 리스트 요소 순서 나열

```
cars = ['taxi', 'sedan', 'bus', 'truck']
cars.sort() #오름차순으로 배열
print(cars)

cars.sort(reverse=True) #역순 배열
print(cars)

print("Here is the original list:")
print(cars)
print("₩nHere is the sorted list:")
print(sorted(cars))       ❷ 오름차순으로 배열한 리스트를 따로 생성
print("₩nHere is the original list again:")
print(cars)
```

리스트(list) 순서

- 리스트 역순으로 출력하기

```
cars = ['taxi', 'sedan', 'bus', 'truck ']
print(cars)
cars.reverse()     #원래 리스트의 요소 순서를 거꾸로 배열
print(cars)
```

- 리스트 관련 에러

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles[3])

Traceback (most recent call last):
File "motorcycles.py", line 3, in <module>
print(motorcycles[3])
IndexError: list index out of range      ▶인덱스 범위 초과
```

리스트 내포(list comprehension)

- 리스트 내포란?
  - 리스트 안에 for문을 포함하여 한 줄로 리스트를 만드는 방법
  - [표현식 for 항목 in 반복객체]
  - [표현식 for 항목 in 반복객체 if 조건문]
  - [표현식 if 조건문 else 문장 for 항목 in 반복객체]

```
#a 원소의 3배수로 구성되는 리스트 생성
a = [1, 2, 3, 4]
result = [num*3 for num in a]

#a의 원소 중 짝수인 원소의 3배수로 구성되는 리스트 생성
result = [num*3 for num in a if num % 2 == 0]

#짝수의 2배수나 홀수의 제곱을 원소로 갖는 리스트 생성
[i*2 if i%2 == 0 else i**2 for i in range(10) ]
```

튜플(tuple)

튜플(tuple)

- 튜플은 리스트와 비슷한 자료구조이나 요소 값을 변경할 수 없다
- 튜플 만들기

```
t1 = (1,2,3)        ▶( ) 속에 요소를 나열하여 만든다
print(t1)
(1, 2, 3)

t2 = 1,2,3          ▶() 없이 요소를 나열해서 만들 수도 있다. 패킹
print(t2)
(1, 2, 3)

t3 = 1,             ▶요소가 한 개인 경우 ,를 붙인다
print(t3)
(1, )

t4 = tuple()        #빈 튜플 생성
```

튜플(tuple)

- 튜플 요소 접근하기(인덱싱과 슬라이싱)

```
t1 = (1,2,3)
print(t1[1])
2
print(t1[1:3])
(2,3)
```

- 튜플 연산

```
t1 = (1,2,3)
t2 = ('a', 'b')
print(t1 + t2)     ▶두 개의 튜플을 하나로 합치기
(1, 2, 3,'a','b')

print(t1*2)        ▶동일한 튜플을 하나 더 생성하여 합치기
(1,2,3,1,2,3)
```

튜플(tuple)

- 튜플과 리스트 상호 변환

```
T1=(1,2,3)
L1=list(T1)        ▶튜플을 리스트로 변환
L1.append(4)
T1=tuple(L1)       ▶리스트를 튜플로 변환
print(T1)
(1, 2, 3, 4)
```

- 튜플 언패킹

```
a, b, c = (1, 2, 3)      ▶튜플의 요소를 하나씩 변수에 할당. 언패킹
print(a)
1
print(b)
2
print(c)
3
```

딕셔너리(dictionary)

딕셔너리(dictionary)

- **키(key):값(value)**로 구성되는 자료 구조
  - {key1: value1, key2: value2}
- 키는 정수, 실수, 문자열, 튜플
- 값은 임의의 데이터형
- 딕셔너리 생성

```
height = {'Jun':174, 'Kim':170, 'Lee':165}      ▶{key:value}
print(height)
{'Jun': 174, 'Kim': 170, 'Lee': 165}

test = dict()      ▶빈 딕셔너리 생성, a = {}
print(test)
{}
```

- 딕셔너리 key 값 조사하기

```
print(height['Kim'])      ▶Dict[key]로 값 조사하기
170
```

딕셔너리(dictionary)

- 딕셔너리 키 조사

```
height = {'Jun':174, 'Kim':170, 'Lee':165}
('Kim' in height)      ▶in 연산자로 key가 있는지 조사
True
```

- 딕셔너리 요소 수정, 추가하기

```
height['Lee'] = 180      ▶요소 수정하기
print(height)
{'Jun': 174, 'Kim': 170, 'Lee': 180}

height['Ihm']=168      ▶요소 추가하기
print(height)
{'Jun': 174, 'Kim': 170, 'Lee': 180, 'Ihm': 168}
```

딕셔너리(dictionary)

- 딕셔너리의 키, 값, 요소를 뽑아내기

```
print(height.keys())        ▶키 뽑아내기
dict_keys(['Jun', 'Kim', 'Lee', 'Ihm'])      ▶iterable(반복객체)이 반환됨

print(height.values())      ▶값 뽑아내기
dict_values([174, 170, 165, 168])

print(height.items())       ▶요소를 튜플로 분리하기
dict_items([('Jun', 174), ('Kim', 170), ('Lee', 165), ('Ihm', 168)])

❶ 뽑아낸 키를 리스트로 변환하기
dict_keys = height.keys()
keys_list = list(dict_keys)
print(keys_list)
['Jun', 'Kim', 'Lee', 'Ihm']
```

딕셔너리(dictionary)

- 딕셔너리 요소 제거하기

```
print(height.pop('Ihm'))        ▶'Ihm' 요소 제거
168

print(height)
{'Jun': 174, 'Kim': 170, 'Lee': 165}
```

- 딕셔너리 지우기

```
height.clear()          ▶요소를 지우기. 빈 딕셔너리로 만든다
print(height)
{}
```

집합(set)

집합(set)

- {} 속에 요소들을 중복되지 않고 순서없이 모아놓은 자료 구조
- 집합의 생성

```
numbers = {1,2,3}
name = set(['Kim', 'Lee', 'Park'])
```

- 빈 집합 생성, 개수 확인, 요소 추가, 요소 제거하기

```
empty = set()       #빈 집합 생성
numbers = {1,2,3}
print(len(numbers))
3

numbers.add(4)     #요소 추가
print(numbers)
{1, 2, 3, 4}

numbers.remove(4)      #요소 제거
print(numbers)
{1, 2, 3}
```

집합(set)

- 집합의 연산

```
setA={1,2,3}
setB={2,3,4}
print(setA & setB)          ▶교집합. setA.intersection(setB)
{2, 3}

print(setA | setB)          ▶합집합 . setA.union(setB)
{1, 2, 3, 4}

print(setA – setB)          ▶차집합. setA에서 setB 요소를 뺀 나머지.
print(setA.difference(setB))
{1}
```

집합(set)

- 부분 집합 연산

```
setA = {1, 2, 3}
setB = {1, 2, 3}
print(setA == setB)        ▶두 개의 집합이 같은가?
True

setA = {1, 2, 3, 4, 5, 6}
setB = {1, 2, 3}
print(setB < setA)         ▶집합 B가 집합 A의 부분집합인가?
True

print(setB.issubset(setA))     ▶집합 B가 집합 A의 부분집합인가?
True
```

집합 예제

- 1부터 45까지의 수 중에서 6개를 선택하여 로또 번호를 만드는 프로그램

```
import random
pick = set()          ▶빈 집합
while len(pick) < 6:
    n = random.randint(1,45)     ▶랜덤 넘버 생성
    if n not in pick:
        pick.add(n)      ▶집합에 요소 추가
print(pick)
print(sorted(pick))    ▶순서대로 나열된 집합 출력
```