

# Ask Django

## 기본 자료구조 (list, tuple, set, dict)

# Container #ref

- 여러 원소들을 가지고 있는 자료 구조
  - **list**, deque
  - **set**, frozensets
  - **dict**, defaultdict, OrderedDict, Counter
  - **tuple**, namedtuple
  - **\*\*str**
- in 연산자로 멤버십 테스트를 지원

```
>>> 'hello' in 'hello world'
True
```

# **list** #doc

- 생성문법 : `[]`, `list()`, `list(iterable)`
- 여러 값을 순차적으로 저장, 순서를 보장
- 리스트를 한 줄로 쓸 때에는 대개 끝에 쉼표를 쓰지 않으며,
- 여러 줄로 나눠쓸 때에는 끝에 쉼표를 씁니다. 항목 추가/삭제가 용이하기 때문입니다.

```
numbers = [1, 3, 5, 7, 9]
names = [
    'Tom',
    'Steve',
    'Min',
]
```

- 색인 (index) 지원 : 0 부터 시작하여 1 씩 증가
  - 음수 색인 지원 : 끝에서부터 역으로 -1 부터 1 씩 감소
- 다른 타입의 값들로도 구성 가능
- 한 List 에 서로 다른 데이터타입의 값을 넣을 수 있지만, 가급적 같은 타입으로 맞춰 주는 것이 보다 알기 쉬운 코드가 됩니다.

```
numbers = [1, 3, 5, 7]           # 리스트 선언
print(7 in numbers)             # 멤버십 체크
print(numbers[0], numbers[-3])  # 색인 0과 -3의 값 출력
print(len(numbers))             # 리스트의 길이 출력
for i in numbers:               # for 루프를 통해 List 내 모든 값을 순회
    print(i)

bad_values = [10, 'Tom', (1, 2, 3)] # BAD
```

## 범위 밖의 색인 (Index) 을 참조하는 경우, IndexError 예외가 발생

```
>>> numbers = [1, 3, 5, 7]
>>> print(numbers[10])
Traceback (most recent call last):
  File "t.py", line 4, in <module>
    print(numbers[10])
IndexError: list index out of range
```

## 데이터 변경

```
numbers = [1, 3, 5, 7, 5]
numbers[0] = 10      # 지정 색인의 값을 변경
numbers.append(9)     # 끝에 값을 추가
numbers.pop(3)        # 특정 색인값을 가져옴과 동시에 제거
numbers.remove(5)     # 특정 값을 1회 제거
numbers.insert(1, 11) # 특정 위치에 값을 추가
```

## 값을 잘라내기 (Slice)

```
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(numbers[1:])      # 유형1) 리스트[시작인덱스:]
[2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(numbers[1:7])    # 유형2) 리스트[시작인덱스:끝인덱스] - 시작인덱스 이상, 끝인덱스 미만
[2, 3, 4, 5, 6, 7]
>>> print(numbers[1:7:2])  # 유형3) 리스트[시작인덱스:끝인덱스:인덱스증가량]
[2, 4, 6]
>>> print(numbers[::-1])   # 유형4) Reverse
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

## 리스트 합치기

```
>>> numbers1 = [1, 3, 5, 7]
>>> numbers2 = [2, 4, 6, 8]
>>> print(numbers1 + numbers2)  # 합쳐진 새로운 리스트
[1, 3, 5, 7, 2, 4, 6, 8]
```

## List Comprehension

```
>>> numbers1 = [1, 3, 5, 7]
>>> numbers2 = [2, 4, 6, 8]
>>> print([i + j for (i, j) in zip(numbers1, numbers2)])
[3, 7, 11, 15]
```

# tuple #doc

- 생성문법 : `()`, `tuple()`, `tuple(iterable)`
- **list**와 유사하지만 변경 불가능(Immutable)한 특성

```
>>> numbers = (1, 3, 5, 7, 5)
```

```
>>> numbers[0] = 10
```

# 지정 색인의 값 변경 시도 => TypeError 예외 발생

```
Traceback (most recent call last):
```

```
File "t.py", line 2, in <module>
```

```
TypeError: 'tuple' object does not support item assignment
```

```
>>> numbers.append(9)
```

# 끝에 값을 추가 시도 => AttributeError 예외 발생

```
Traceback (most recent call last):
```

```
File "t.py", line 3, in <module>
```

```
AttributeError: 'tuple' object has no attribute 'append'
```



- 소괄호는 때에 따라, 우선순위 연산자 혹은 튜플로 쓰인다.
- 설정값으로서 튜플을 쓰실 때, 헷갈리시면 리스트를 쓰세요. 성능 차이 거의 없습니다. 코드 오류가 나지 않는 것이 중요.

```
>>> tuple1 = (1 + 3)      # 우선순위 (갯수가 1개일 때, 콤마를 생략하면 NOT 튜플)
>>> tuple2 = (1 + 3,)     # 튜플
>>> tuple3 = (3)          # 우선순위
>>> tuple4 = (3,)         # 튜플
```

```
>>> list1 = [1 + 3]       # 리스트 (갯수가 1개일 때, 콤마를 생략해도 리스트)
>>> list2 = [1 + 3,]      # 리스트
>>> list3 = [3]           # 리스트
>>> list4 = [3,]          # 리스트
```

- Packing / Unpacking : list/tuple에 동일하게 적용
- unpacking 시에 갯수가 맞지 않으면 ValueError

```
>>> numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)      # packing - 다수 값을 하나의 변수에 담음.
>>> v1, v2, v3, v4, v5, v6, v7, v8, v9, v10 = numbers  # unpacking - 하나의 값을 다수의 변수에 나눠 담음.

>>> v1, v2, v3, v4 = numbers[:4]
>>> v1, v2, v3, v4, *others = numbers

>>> *others, v8, v9, v10 = numbers
>>> v1, v2, *others, v9, v10 = numbers
>>> v1, v2, v3, *others = numbers

>>> v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14 = (*numbers, 11, 12, 13, 14)
```

- swap : list/tuple에 동일하게 적용

```
>>> x, y = 1, 2      # x에는 1이 대입, y에는 2가 대입
>>> x, y = y, x      # x에는 y값이, y에는 x값이 동시에 대입
```

# set (집합형) #doc

- 중복을 허용하지 않는 데이터의 집합
  - list/tuple에서 중복을 제거코자 할 때, set을 활용하면 유용
- List/Tuple 과 다르게 추가된 순서를 유지하지 않음.

```
>>> set_numbers = { 1, 3, 4, 5, 1, 4, 3, 1 }
>>> set_numbers
{1, 3, 4, 5}
```

```
>>> list_numbers = [1, 1, 2, 1, 1, 2, 2, 3, 1, 2, 3]
>>> list_numbers = list(set(list_numbers))
>>> list_numbers
[1, 2, 3]
```

## 합집합, 교집합, 차집합, 여집합 연산 지원

```
>>> set_numbers1 = { 1, 3, 4, 5, 1, 4, 3, 1 }
>>> set_numbers2 = { 1, 3, 4, 11, 13, 14, 15, 11, 14, 13, 11 }
>>> print(set_numbers1 - set_numbers2) # 차
{5}
>>> print(set_numbers1 | set_numbers2) # 합
{1, 3, 4, 5, 11, 13, 14, 15}
>>> print(set_numbers1 & set_numbers2) # 교
{1, 3, 4}
>>> print(set_numbers1 ^ set_numbers2) # 여
{5, 11, 13, 14, 15}
```

# dict (사전형) doc

- Key와 Value의 쌍으로 구성된 집합
- Key중복을 허용하지 않음.
- 중괄호 내에 콜론 (:) 으로 Key/Value를 구분

```
dict_values = { 'blue': 10, 'yellow': 3, 'blue': 9, 'red': 7 }
```

- in 연산자로 멤버십 체크 지원 (Key의 등록 여부)
- 순회 시에는 key 목록만 순회
- 멤버함수
  - .keys() : key 목록
  - .values() : value 목록
  - .items() : (key, value) 목록

```
>>> dict_values = { 'blue': 10, 'yellow': 3, 'red': 7 }
>>> print(dict_values['blue'])      # 지정 Key 의 Value 를 조회
10

>>> print(dict_values['magenta'])   # 등록되지 않은 Key 에 접근할 경우, KeyError 발생
KeyError: 'magenta'

>>> dict_values['blue'] = 20        # 지정 Key 의 값을 변경

>>> dict_values['black'] = 30       # 새로운 Key:Value 를 등록

>>> del dict_values['black']        # 지정 Key:Value 를 제거

>>> del dict_values['magenta']     # 등록되지 않은 Key 에 접근하여, KeyError 발생
KeyError: 'magenta'
```

in 연산자로 지정 Key 의 등록여부를 확인

```
>>> dict_values = { 'blue': 10, 'yellow': 3, 'red': 7 }
```

```
>>> print('magenta' in dict_values)
False
```

```
>>> print('blue' in dict_values)
True
```

for 루프 순회

```
>>> dict_values = { 'blue': 10, 'yellow': 3, 'red': 7 }
```


```
>>> for key in dict_values:
    print(key)
```

```
red
yellow
blue
```

```
>>> for (key, value) in dict_values.items():
    print(key, value)
```

```
red 7
yellow 3
blue 10
```



A person in a red shirt and black shorts is running on a rocky path towards a sunset over mountains. The scene is captured in a soft, hazy light, with the sun low on the horizon, casting a warm glow over the landscape. The person is in the foreground, slightly out of focus, running towards the right. The background features rugged mountains and a body of water reflecting the sunset colors.

*Life is short,  
use Python3/Django.*