# 11.Python Tuple 활용

# 1. Python 튜플 활용법

1] 튜플 개념

- 순서있는 임의의 객체 모음 (시퀀스형)
- 튜플은 **변경 불가능(Immutable)하므로 , 메소드를 가지지 않음**
- 시퀀스형이 가지는 다음 연산 모두 지원(인덱싱, 슬라이싱, 연결, 반복, 멤버쉽 테스트 )
- 자료가 한 개일 때는 반드시 콤마가 있어야 함(괄호가 없어도 콤마 있으면 튜플로 인식 )
- 튜플 내부 원소로 다른 튜플(리스트/사전)을 가질 수 있음
- 튜플을 사용하는 경우 1: 함수가 하나 이상의 값을 리턴하는 경우

예시 1

```
t = (12345, 54321, 'hello!')
u = t, (1, 2, 3, 4, 5) # 튜플 내부 원소로 다른 튜플을 가질 수 있음
print ("t, (1, 2, 3, 4, 5) -->", u )

t2 = [1, 2, 3] # 튜플 내부 원소로 리스트 가질 수 있음
u2 = t2, (1, 2, 4)
print ("t2, (1, 2, 4) -->", u2 )
t3 = {1:"abc", 2:"def"} # 튜플 내부 원소로 사전 가질 수 있음
u3 = t3, (1, 2, 3)
print ("t3, (1, 2, 3) -->", u3)
```

결과

```
---------------------------
t, (1, 2, 3, 4, 5) --> ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
t2, (1, 2, 4) --> ([1, 2, 3], (1, 2, 4))
t3, (1, 2, 3) --> ({1: 'abc', 2: 'def'}, (1, 2, 3))
```

# 1. Python 튜플 활용법

2] **패킹과 언패킹**
- 패킹 (Packing): 하나의 튜플 안에 여러 개의 데이터를 넣는 작업
- 언패킹 (Unpacking): 하나의 튜플에서 여러 개의 데이터를 한꺼번에 꺼내와 각각 변수에 할당하는 작업

예시2 *# 패킹 = 3개의 객체를 하나의 변수에 묶는 것*

```
t = (1, 2, 'hello')
print (t)

T = (1,2,3,4,5)
L = list(T)
L[0] = 100      # 언패킹 (Unpacking):
print ("list(T), T=(1,2,3,4,5) ", L )
```

결과

```
---------------------------
(1, 2, 'hello')
list(T), T=(1,2,3,4,5)  [100, 2, 3, 4, 5]
```

예시 3 # Tuple 사용 용도 -> 고정된 값을 쌍

```
d = {'one':1, 'two':2}
print (d.items() )  # 3번째
```

결과

```
---------------------------
dict_items([('two', 2), ('one', 1)])
```

3] Tuple 사용 용도

- 함수가 하나 이상의 값을 리턴하는 경우
  ① 리턴을 할 때 2개의 원소 같이 리턴 → 튜플 괄호 숨어있음
  ② 리턴한 결과가 튜플한 뒤 언패킹 진행
  ③ 튜플은 동시에 여러 개의 값 리턴 가능

- 고정된 값을 쌍으로 표현하는 경우
  ① 사전의 원소 기준 → 콤마(,)
  ② 원소 : item
  ③ 결과는 리스트, 원소는 튜플

# 2. Python 집합 자료형

1] **집합 자료형**
- set 내장 함수를 사용한 집합 자료 생성(인자->리스트, 튜플, 사전 가능)
- 변경 가능(Mutable)한 객체이다.
- 각 원소간에 순서는 없다.
- 각 원소는 중복될 수 없다.
-시퀀스 자료형이 아니다(인덱싱, 슬라이싱 안 됨)
- set 안에 튜플이 들어가도 출력은 리스트로 됨

2] set의 주요 메소드

| set 연산 | 동일 연산자 | 내용 |
|---|---|---|
| S issubset(t) | s <= t | s가 t의 부분집합인가? |
| S issuperset(t) | s >= t | s가 t의 슈퍼집합인가? |
| s union(t) | s\|t | 새로운 s와 t의 합집합 |
| s intersection(t) | s & t | 새로운 s와 t의 교집합 |
| s difference(t) | s-t | 새로운 s와 t의 차집합 |
| s symmetric_difference(t) | s * t | 새로운 s와 t의 배타집합 |
| s copy() | | 집합 s의 shallow 복사 |

| set 연산 | 동일 연산자 | 내용 |
|---|---|---|
| s.update(t) | s \|= t | s와 t의 합집합을 s에 저장 |
| s.intersection_update(t) | s &= t | s와 t의 교집합을 s에 저장 |
| s.difference_update(t) | s -= t | s와 t의 차집합을 s에 저장 |
| s.symmetric_difference_update(t) | s ^= t | s와 t의 배타집합을 s에 저장 |
| s.add(x) | | 원소 x를 집합 s에 추가 |
| s.remove(x) | | 원소 x를 집합 s에서 제거, 원소 x가 집합 s에 없으면 예외 발생 |
| s.discard(x) | | 원소 x를 집합 s에서 제거 |
| s.pop() | | 임의의 원소를 집합 s에서 제거, 집합 s가 공집합이면 예외 발생 |
| s.clear() | | 집합 s의 모든 원소 제거 |