

Lesson 7

Collection 자료형 튜플(Tuple)



목차

1. 튜플

- 튜플 생성
- 팩킹과 언팩킹

튜플(tuple)

- 리스트와 유사
- **저장된 값을 변경(추가, 수정, 삭제)할 수 없는 리스트 자료형. 즉, 읽기 전용.**
- 읽기 전용만 제외하면 리스트와 동일함.(인덱싱, 슬라이싱,...모두 가능함)
- **순서O, 중복O, 수정X, 삭제X**
- 튜플은 언제 사용하면 좋은가?
 - 프로그램이 실행되는 동안 추가, 삭제, 수정 작업이 필요없는 데이터를 사용할 때.



튜플을 사용하는 이유

메모리를 적게 사용하며 속도가 빠르다. 리스트는 변경 가능성에 대비해서 더 많은 메모리를 사용하고 속도가 느리다. 튜플은 변경을 할 필요가 없으므로 내부 구조가 단순하고 읽는 속도가 빠르다.

튜플의 메모리 크기와 생성 속도 확인

코드

```
import sys

li=[1,2,3,4,5]
tu=(1,2,3,4,5)
print(sys.getsizeof(li))
print(sys.getsizeof(tu))
```

120
80



sys 모듈 (System-specific parameters and functions)

파이썬의 내장 모듈(built-in-module)로 시스템 관련 정보를 얻을 수 있다. 인터프리터와 상호작용하는 변수와 함수를 직접 제어하고 경로도 제공해준다.

```
print(sys.version)
print(sys.path)
```

코드

```
import timeit
print(timeit.timeit('[5,6,7,8]'))
print(timeit.timeit('(5,6,7,8)'))
```

0.10939849994610995
0.017613000003620982

- 소괄호 () 또는 tuple()함수로 생성
- 값을 1개만 보관하는 튜플 생성할 때 → 변수=(값1,) or 변수=값1,

코드

```
t1=(1,2,3)
t2=4,5,6
t3=tuple([10,20,30])      # 리스트를 튜플로 변경
print(t1)                 # (1, 2, 3)
print(t2)                 # (4, 5, 6)
print(t3)                 # (10, 20, 30)
t4=(100)                  # 튜플 자료가 아님
print(type(t4))           # <class 'int'>
t4=(100,)                 # 값이 1개인 튜플. 100, 와 동일.
print(type(t4))           # <class 'tuple'>
```

코드

```
tu1=(1,1,2,3)
tu2=('a','b','c','d','e')
print(tu1+tu2)          # (1, 1, 2, 3, 'a', 'b', 'c', 'd', 'e')
print(tu2[-1])          # e
print(tu2[0:3])         # ('a', 'b', 'c')
```

`index()` : 찾는 요소의 위치
`count()` : 찾는 요소의 개수

코드

```
tu1=(1,1,2,3)
```

```
print(tu1.index(3)) # 4    요소 3의 위치.  
print(tu1.count(1)) # 2    요소 1의 개수.
```


튜플 변경, 팩킹(packing)과 언팩킹(unpacking)

코드

```
# 값 변경  
tu=(1,2,3,4,5)  
tu[0]=100      # 오류
```

코드

```
# 팩킹과 언팩킹  
tu=1,2,3,4      # 팩킹  
a,b,c,d=tu      # 언팩킹  
x,y,z=1,2,3     # 언팩킹  
print(a,b,c,d)  
print(x,y,z)
```



튜플/리스트 팩킹(packaging)과 언팩킹(unpacking)

코드

```
a, b, c = (1, 2, 3, 4, 5)
```


오류.  변수 개수와 데이터 개수가 동일해야 함

```
a, b, *c = (1, 2, 3, 4, 5) # *는 여러 개를 의미함.
```

```
print(a) # 1
```

```
print(b) # 2
```

```
print(c) # [3, 4, 5]
```

 리스트도 튜플과 같은 방식으로 팩킹과 언팩킹 가능함.

 언팩킹시 결과는 리스트 형태로 반환됨.

튜플 변경, 팩킹(packing)과 언팩킹(unpacking)

코드

```
# 두 변수의 값 교환
```

```
a,b=10,20
```

```
print("교환전:",a,b)      # 교환전: 10 20
```

```
a,b=b,a
```

```
print("교환후:",a,b)      # 교환후: 20 10
```



전통적인 교환 방식

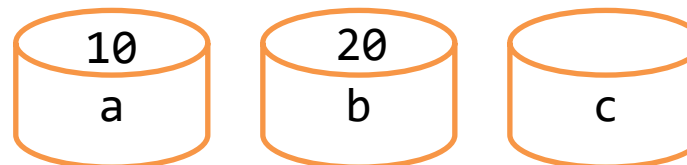
```
a=10
```

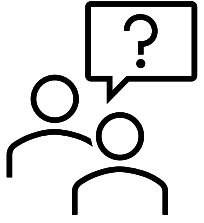
```
b=20
```

```
c=a
```

```
a=b
```

```
b=c
```





1. 튜플은 순서가 (있다 없다).
2. 튜플은 요소를 추가할 수 (있다 없다).
3. 튜플은 요소를 수정할 수 (있다 없다).
4. 튜플은 중복된 요소를 가질 수 (있다 없다).
5. 튜플을 생성하려면 ()를 사용하면 된다.

→ ()

tuple()