튜플 자료형

튜플은 어떻게 만들까?

튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.

- 리스트는 [ ]으로 둘러싸지만 튜플은 ( )으로 둘러싼다.
- 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.

튜플의 모습은 다음과 같다.

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

리스트와 모습은 거의 비슷하지만 튜플에서는 리스트와 다른 2가지 차이점을 찾아볼 수 있다. t2 = (1,)처럼 단지 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙여야 한다는 것과 t4 = 1, 2, 3처럼 괄호( )를 생략해도 무방하다는 점이다.

얼핏 보면 튜플과 리스트는 비슷한 역할을 하지만 프로그래밍을 할 때 튜플과 리스트는 구별해서 사용하는 것이 유리하다. 튜플과 리스트의 가장 큰 차이는 값을 변화시킬 수 있는가 여부이다. 즉 리스트의 항목 값은 변화가 가능하고 튜플의 항목 값은 변화가 불가능하다. 따라서 프로그램이 실행되는 동안 그 값이 항상 변하지 않기를 바란다거나 값이 바뀔까 걱정하고 싶지 않다면 주저하지 말고 튜플을 사용해야 한다. 이와는 반대로 수시로 그 값을 변화시켜야할 경우라면 리스트를 사용해야 한다. 실제 프로그램에서는 값이 변경되는 형태의 변수가 훨씬 많기 때문에 평균적으로 튜플보다는 리스트를 더 많이 사용한다.

튜플의 요소값을 지우거나 변경하려고 하면 어떻게 될까?

앞에서 설명했듯이 튜플의 요솟값은 한 번 정하면 지우거나 변경할 수 없다. 다음에 소개하는 두 예를 살펴보면 무슨 말인지 알 수 있을 것이다.

1. 튜플 요솟값을 삭제하려 할 때

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0]
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
```

튜플의 요소를 리스트처럼 del 함수로 지우려고 한 예이다. 튜플은 요소를 지우는 행위가 지원되지 않는다는 메시지를 확인할 수 있다.

2. 튜플 요솟값을 변경하려 할 때

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c'
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

튜플의 요솟값을 변경하려고 해도 마찬가지로 오류가 발생하는 것을 확인할 수 있다.

튜플 다루기

튜플은 값을 변화시킬 수 없다는 점만 제외하면 리스트와 완전히 동일하므로 간단하게만 살펴보겠다. 다음 예제는 서로 연관되어 있으므로 차례대로 수행해 보기 바란다.

인덱싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0]
1
>>> t1[3]
'b'
```

문자열, 리스트와 마찬가지로 t1[0], t1[3]처럼 인덱싱이 가능하다.

슬라이싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[1:]
(2, 'a', 'b')
```

t1[1]부터 튜플의 마지막 요소까지 슬라이싱하는 예이다.

튜플 더하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t2 = (3, 4)
>>> t1 + t2
(1, 2, 'a', 'b', 3, 4)
```

튜플을 더하는 방법을 보여 주는 예이다.

튜플 곱하기

```
>>> t2 = (3, 4)
>>> t2 * 3
(3, 4, 3, 4, 3, 4)
```

튜플의 곱하기(반복) 예를 보여 준다.

튜플 길이 구하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> len(t1)
4
```

튜플의 길이를 구하는 예이다.