튜플

제10장

Python for Everybody

www.py4e.com

튜플과 리스트

튜플은 리스트와 비슷한 기능을 하는 시퀀스
- 0부터 시작하는 인덱스에 항목을 저장

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print(x[2])
Joseph
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

그러나 튜플은 변경불가

리스트는 값을 바꿀 수 있지만 튜플은 저장된 내용를 변경 불가

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback:'str'
object does
not support item
Assignment
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple'
object does
not support item
Assignment
>>>
```

튜플이 할 수 없는 것

```
>>> x = (3, 2, 1)
>>> x.sort()
Traceback:
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback:
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback:
AttributeError: 'tuple' object has no attribute 'reverse'
>>>
```

서로 다른 두 시퀀스

```
>>> l = list()
>>> dir(l)
['append', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

튜플의 장점

- 파이썬은 튜플을 수정 가능하지 않게 저장하기 때문에 리스트와 비교하여 메모리 사용량과 성능 측면에서 훨씬 단순하고 효과적
- 그러므로, 임시 변수를 선언할 때는 리스트를 쓰는 것보다 튜플을 쓰는 것이 좋음

튜플의 선언

- 튜플을 좌변에 놓는 것으로 선언문에서 사용 가능
- 괄호는 생략 가능

```
>>> (x, y) = (4, 'fred')
>>> print(y)
fred
>>> (a, b) = (99, 98)
>>> print(a)
99
```

튜플과 딕셔너리

딕셔너리의 items() 메소드는 (키, 값)를 튜플의 형태로 리턴

```
>>> d = dict()
>>> d['csev'] = 2
>>> d['cwen'] = 4
>>> for (k,v) in d.items():
...     print(k, v)
...
csev 2
cwen 4
>>> tups = d.items()
>>> print(tups)
dict_items([('csev', 2), ('cwen', 4)])
```

튜플의 비교

튜플에서 다른 시퀀스와 비교 연산자를 사용할 수 있음. 만약 첫번째 요소가 같다면, 파이썬은 다음 요소를 비교하고, 다른 요소가 있을 때까지 비교를 계속함

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ( 'Jones', 'Sally' ) < ('Jones', 'Sam')
True
>>> ( 'Jones', 'Sally') > ('Adams', 'Sam')
True
```

Tuple로 된 Lists의 Sorting

- 딕셔너리를 정렬 하기 위해 튜플로 이루어진 리스트를 사용할 수 있음
- items() 메소드를 통해 키와 값를 얻은 후 sorted() 메소드로 딕셔너리를 정렬하면 됨

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22), ('b', 1)])
>>> sorted(d.items())
[('a', 10), ('b', 1), ('c', 22)]
```

sorted() 함수

내장된 sorted
메소드는 시퀀스를
인자로 받아 정렬된
시퀀스를 리턴.
활용도가 높고 편리

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> for k, v in sorted(d.items()):
...     print(k, v)
...
a 10
b 1
c 22
```

값을 이용한 정렬

- (키, 값) 형태의 튜플로 이루어진 리스트를 만들면 값를 기준으로 정렬할 수 있다
- for 반복문를 사용하여 튜플로 이루어진 리스트를 만들 수 있다

```
>>> c = {'a':10, 'b':1, 'c':22}
>>> tmp = list()
>>> for k, v in c.items() :
...         tmp.append( (v, k) )
...
>>> print(tmp)
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> tmp = sorted(tmp, reverse=True)
>>> print(tmp)
[(22, 'c'), (10, 'a'), (1, 'b')]
```

가장 많이 쓰이는 단어 열 개 찾기

```
fhand = open('romeo.txt')
counts = {}
for line in fhand:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0 ) + 1

lst = []
for key, val in counts.items():
  newtup = (val, key)
    lst.append(newtup)

lst = sorted(lst, reverse=True)

for val, key in lst[:10] :
    print(key, val)
```

더 짧게 쓸 수도 있음

```
>>> c = {'a':10, 'b':1, 'c':22}

>>> print( sorted( [ (v,k) for k,v in c.items() ] ) )

[(1, 'b'), (10, 'a'), (22, 'c')]
```

리스트 컴프리헨션은 동적인 리스트를 생성.
이 경우 역으로 이루어진 튜플로 리스트가 만들어지고 정렬.

http://wiki.python.org/moin/HowTo/Sorting

요약

- 튜플 문법
- 튜플 변경 불가
- 비교
- 소트

- 튜플 선언
- 키와 값을 이용한 딕셔너리 소트

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Contributor:
- 이승준
- 커넥트재단

Translation:
- Yang Incheol (inchyangv@gmail.com)
- Jeungmin Oh (tangza@gmail.com)

...