Python - Tuples

Tuples are like lists

-- Tuples are another kind of sequence, which function much like a list - they have elements which are indexed starting at 0

```
>>> 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
>>>
```

..but.. Tuples are "immutable"

-- Unlike a list, once you create a tuple, you cannot alter its contents - similar to a string

```
>>> 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
>>>
```

Things not to do with tuples

```
>>> 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'
>>>
```

A Tale of Two Sequences

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

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

Tuples are more efficient

-- Since Python does not have to build tuple structures to be modifiable, they are simpler and more efficient in terms of memory use and performance than lists

Tuples and Assignment

-- Put a tuple on the left hand side of an assignment statement

-- We can even omit the parenthesis

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

Tuples and Dictionaries

-- The items() method in dictionaries returns a list of (key, value) tuples

```
>>> 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
[('csev', 2), ('cwen', 4)]
```

Tuples are Comparable

-- The comparison operators work with tuples and other sequences if the first item is equal, Python goes on to the next element, and so on, until it finds elements that differ.

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

Sorting Lists of Tuples

-- Take advantage of the ability to sort a list of tuples to get a sorted version of a dictionary

-- First sort the dictionary by the key using the items() method

```
    >>> d = {'a':10, 'b':1, 'c':22}
    >>> t = d.items()
    >>> print t
[('a', 10), ('c', 22), ('b',1)]
>>> t.sort()
>>> print t
[('a', 10), ('b',1) , (‘c’, 22) ]
```

Using sorted()

-- Even more directly using the built-in function sorted that takes a sequence as a parameter and returns a sorted sequence

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

Sort by values instead of key

--Construct a list of tuples of the form (value, key) by sorting the value

--Do this with a for loop that creates a list of tuples

```
>>> 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.sort(reverse=True)
>>> print tmp
[(22, 'c'), (10, 'a'), (1, 'b')]
```

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

lst = list()
for key, val in counts.items():
        lst.append( (val, key) )
lst.sort(reverse=True)
for val, key in lst[:10] :  print key,
        val
```

The top 10 most common words.

Even Shorter Version (adv)

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

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

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

List comprehension creates a dynamic list. In this case, we make a list of reversed tuples and then sort it.