Tuples

Chapter 10

Tuples are like lists

les are another kind of sequence that functions much
- they have elements which are indexed starting at 0

```
 ('Glenn', 'Sally', 'Joseph')
nt x[2]

 ( 1, 9, 2 )
nt y
2)
nt max(y)
```

```
>>> for it
...     pr
...
1
9
2
>>>
```

t... Tuples are “immutabl

nlike a list, once you create a tuple, you cannot alte
ntents - similar to a string

```
, 8, 7]
 6
x
6]
```

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

```
>>> z = (
>>> z[2]
Traceback
object do
not suppo
Assignmen
>>>
```

hings not to do with tuple

```
 (3, 2, 1)
ort()
ck:
teError: 'tuple' object has no attribute 'so
ppend(5)
ck:
teError: 'tuple' object has no attribute 'ap
everse()
ck:
teError: 'tuple' object has no attribute 're
```

A Tale of Two Sequences

```
= list()
r(l)
nd', 'count', 'extend', 'index', 'insert', '
e', 'reverse', 'sort']

= tuple()
r(t)
t', 'index']
```

Tuples are more efficient

nce Python does not have to build tuple structures to
odifiable, they are simpler and more efficient in term
emory use and performance than lists

o in our program when we are making "temporary var
e prefer tuples over lists

Tuples and Assignment

e can also put a tuple on the left-hand side of an signment statement

e can even omit the parentheses

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

Tuples and Dictionaries

ms() method in
aries returns a
key, value)

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

Tuples are Comparable

he comparison operators work with tuples and othe
quences. If the first item is equal, Python goes on
ext element, and so on, until it finds elements tha

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

Sorting Lists of Tuples

e can take advantage of the ability to sort a list of tup
t a sorted version of a dictionary

rst we sort the dictionary by the key using the items()
ethod

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

ing
ted()

o this even
ctly using the
nction sorted
a sequence as a
r and returns a
quence

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

rt by values instead of k

could construct
of tuples of the
(value, key) we
d sort by value

lo this with a for
that creates a list
ples

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

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

t()
val in counts.items():
ppend( (val, key) )

reverse=True)

key in lst[:10] :
 key, val
```

The
com

Even Shorter Version

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

t sorted( [ (v,k) for k,v in c.items(

), (10, 'a'), (22, 'c')]
```

comprehension creates a dynamic list. In this cas
make a list of reversed tuples and then sort it.

Summary

Tuple syntax

Immutability

Comparability

Sorting

- Tuples in assignment statements
- Sorting dictionaries by either key or value