**Tuples**

*Ordered, immutable* sequence of objects. Since it’s ordered, a tuple can be indexed, so it’s like a list you can’t change. Doesn’t sound that useful? We’ll see about that…

**Declaration:** tup1 = (*ob1*, *ob2*, …etc.)

**Type Conversion:** tuple()

Using Tuples with Dictionaries: These two types show up together a lot.

**# Convert a list of tuples to a dictionary**

>>> tups\_list = [('a',1), ('b',2)]

>>> D = dict(tups\_list)

>>> D

{'a': 1, 'b': 2}

**# Iterating over dictionary key/value pairs with iteritems()**

**# Each returned value of iteritems() is a tuple**

>>> for i in D.iteritems():

... print i

...

('a', 1)

('b', 2)

**# Now loop with multiple assignment**

>>> for i,j in D.iteritems():

... print i,j

...

a 1

b 2

**Useful built-in functions**

Python has several built-in function which make data handling even easier. See ‘Built in Functions’ for more: <http://docs.python.org/2/library/functions.html#map>.

**enumerate():** Return an iterator, where each item (when iterated over) returns a tuple (*place in sequence, item*)

>>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']

>>> for i in enumerate(seasons):

… print i

…

(0, 'Spring')

(1, 'Summer')

(2, 'Fall')

(3, 'Winter')

>>> seasonslist = [i for i in enumerate(seasons)]

**eval():** Executes a python expression from a string of that expression.

>>> print ‘2+2’

2+2

>>> print eval(‘2+2’)

4

**map(*function*,*iterable1, iterable2…*):** Apply function to every value of iterables, and return list of results as tuples. If *function* is None, return list of tuples of corresponding items in all iterables. This is useful for stitching lists together without having to write your own code.

>>> orgs = [‘Bird’, ‘Bee’, ‘Bear’]

>>> sightings = [25, 67, 5]

>>> map(None,orgs,sightings)

[('Bird', 25), ('Bee', 67), ('Bear', 5)]

**range():** This one’s pretty straightforward

>>> range(5)

[0,1,2,3,4,5]

>>> range(1,5)

[1,2,3,4,5]

>>> range(1,10,2)

[0,2,4,6,8]

>>> for i in range(3):

… print i

…

0

1

2