Tuples

Courtesy of Software-Carpentry.

What is a "tuple"?

- A *list* is a mutable heterogeneous sequence

- A *tuple* is an *immutable* heterogeneous sequence

- I.e., a list that can't be changed after creation

- You need to know about them

- They have their uses

Using tuples

Create tuples using () instead of []

Still index using [] (because everything does)

```
>>> primes = (2, 3, 5, 7)
>>> print primes[0], primes[-1]
2 7
>>> empty_tuple = ()
>>> single_item_tuple = (5,)
            # Because (5) is ambiguous
```

One of Python's few syntactic warts…

Don't need parentheses if context is enough

```
>>> primes = 2, 3, 5, 7
>>> print primes
(2, 3, 5, 7)
>>>
```

Can use on the left of assignment

```
>>> left, middle, right = 2, 3, 5
>>> print left, right
2 5
```

Allows functions to return multiple values

```
>>> def bounds(values):
...         low = min(values)
...         high = max(values)
...         return (low, high)
...
>>> print bounds([3, -5, 9, 4, 17, 0])
(-5, 17)
>>> least, greatest = bounds([3, -5, 9, 4, 17, 0])
>>> print least
-5
```

Provides a quick way to swap variable values

>>> left, right = 0, 10

>>> right, left = left, right

>>> print right, left
0 10

And an easy way to unpack a list

```
>>> colours = ['yellow', 'magenta', 'lavender']

>>> left, middle, right = colours

>>> print left, middle, right
yellow magenta lavender
>>>
```

Number of values must be the same

Often used in loops

```
>>> pairs = [(1, 10), (2, 20), (3, 30), (4, 40)]
>>> for (low, high) in pairs:
...         print low + high
...
11
22
33
44
>>>
```

The "enumerate" function

The enumerate function produces (index, value) pairs

```
>>> colors = ['yellow', 'magenta', 'lavender']
>>> for (i, name) in enumerate(colors):
...         print i, name
...
0 yellow
1 magenta
2 lavender
```

Prefer this to range(len(values))