zip

The **zip** built-in takes a series of iterables and aggregates the elements from each of them. Let's see what that means:

```
keys = ['x', 'y', 'z']
values = [5, 6, 7]
print (zip(keys, values))
#<zip object at 0x7faaad4dd848>

print (list(zip(keys, values)))
#[('x', 5), ('y', 6), ('z', 7)]
```

In this example, we have two lists of equal size. Next we zip them together with the zip function. This just returns a zip object, so we wrap that in a call to the **list** built-in to see what the result is. We end up with a list of tuples. You can see here that zip matched up the items in each list by position and put them into tuples.

One of the most popular use cases for zip it to take two lists and turn them into a dictionary:

```
keys = ['x', 'y', 'z']
values = [5, 6, 7]
my_dict = dict(zip(keys, values))
print (my_dict)
#{'x': 5, 'y': 6, 'z': 7}
```

When you pass a series of tuples to Python's **dict** built-in, it will create a dictionary as you can see above. I've used this technique from time to time in my own code.

Wrapping Up

While this was a brief tour, I hope it gave you a good idea of just how much power Python's built-ins give you. There are many others that you use every day, such as list, dict and dir. The built-in functions that you learned about in this chapter probably won't get used every day, but they can be extremely helpful in certain situations. I am actually devoting an entire chapter to a special built-in known as **super**, which you can read about later on in the book. Have fun with these and be sure to look up the documentation to see what other jewels exist.