Counter

The collections module also provides us with a neat little tool that supports convenient and fast tallies. This tool is called **Counter**. You can run it against most iterables. Let’s try it out with a string!

```
from collections import Counter
print (Counter('superfluous'))
#Counter({'u': 3, 's': 2, 'e': 1, 'l': 1, 'f': 1, 'o': 1, 'r': 1, 'p': 1})

counter = Counter('superfluous')
print (counter['u'])
#3
```

In this example, we import **Counter** from collections and then pass it a string. This returns a Counter object that is a subclass of Python’s dictionary. We then run the same command but assign it to the variable **counter** so we can access the dictionary a bit easier. In this case, we saw that the letter “u” occurs three times in the example string.

The Counter provides a few methods that might interest you. For example, you can call **elements** which will an iterator over the elements that are in the dictionary, but in an arbitrary order. You can kind of think of this function as a “scrambler” as the output in this case is a scrambled version of the string.

```
print (list(counter.elements()))
#['u', 'u', 'u', 'o', 'p', 'e', 'f', 'l', 'r', 's', 's']
```

Another useful method is **most_common**. You can ask the Counter what the most common items are by passing in a number that represents what the top recurring “n” items are:

```
print( counter.most_common(2))
#[('u', 3), ('s', 2)]
```

Here we just ask our Counter what the top two most recurring items were. As you can see, it produced a list of tuples that tells us “u” occurred 3 times and “s” occurred twice.

The other method that I want to cover is the **subtract** method. The subtract method accepts an iterable or a mapping and the uses that argument to subtract. It’s a bit easier to explain if you see some code:

```
from collections import Counter

counter_one = Counter('superfluous')
print (counter_one)
#Counter({'u': 3, 's': 2, 'l': 1, 'r': 1, 'e': 1, 'o': 1, 'p': 1, 'f': 1})

counter_two = Counter('super')
print(counter_one.subtract(counter_two))
#None

print (counter_one)
#Counter({'u': 2, 'l': 1, 'o': 1, 's': 1, 'f': 1, 'r': 0, 'e': 0, 'p': 0})
```

So here we recreate our first counter and print it out so we know what’s in it. That we create our second Counter object. Finally we subtract the second counter from the first. If you look carefully at the output at the end, you will notice the that number of letters for five of the items has been decremented by one.

As I mentioned at the beginning of this section, you can use the Counter against any iterable or mapping, so you don’t have to just use strings. You can also pass it tuples, dictionaries and lists! Give it a try on your own to see how it works with those other data types.

Now we’re ready to move on to the defaultdict!