**Chapter 47: Collections module**

The built-in collections package provides several specialized, ﬂexible collection types that are both high- performance and provide alternatives to the general collection types of dict, list, tuple and set. The module also deﬁnes abstract base classes describing diﬀerent types of collection functionality (such as MutableSet and ItemsView).

**Section 47.1: collections.Counter**

[Counter is](https://docs.python.org/2/library/collections.html#collections.Counter) a dict sub class that allows you to easily count objects. It has utility methods for working with the frequencies of the objects that you are counting.

**import** collections

counts = collections.Counter([1,2,3])

the above code creates an object, counts, which has the frequencies of all the elements passed to the constructor. This example has the value Counter({1: 1, 2: 1, 3: 1})

**Constructor examples**

Letter Counter

>>> collections.Counter('Happy Birthday')

Counter({'a': 2, 'p': 2, 'y': 2, 'i': 1, 'r': 1, 'B': 1, ' ': 1, 'H': 1, 'd': 1, 'h': 1, 't': 1})

Word Counter

>>> collections.Counter('I am Sam Sam I am That Sam-I-am That Sam-I-am! I do not like that Sam-I- am'.split())

Counter({'I': 3, 'Sam': 2, 'Sam-I-am': 2, 'That': 2, 'am': 2, 'do': 1, 'Sam-I-am!': 1, 'that': 1,

'not': 1, 'like': 1})

**Recipes**

>>> c = collections.Counter({'a': 4, 'b': 2, 'c': -2, 'd': 0})

Get count of individual element

>>> c['a']

4

Set count of individual element

>>> c['c'] = -3

>>> c

Counter({'a': 4, 'b': 2, 'd': 0, 'c': -3})

Get total number of elements in counter (4 + 2 + 0 - 3)

>>> sum(c.itervalues()) *# negative numbers are counted!*

3

Get elements (only those with positive counter are kept)

[GoalKicker.com – Python® Notes for Professionals 275](https://goalkicker.com/)