Integrating with doctest

The unittest module can be used with Python's doctest module as well. If you have created a lot of modules that have doctests in them, you will usually want to be able to run them systematically. This is where unittest comes in. The unittest module supports **Test Discovery** starting in Python 3.2. Test Discovery basically allows unittest to look at a the contents of a directory and determine from the file name which ones might contain tests. It then loads the test by importing them.

Let's create a new empty folder and inside of it, we will create a file called **my_docs.py**. It will need to have the following code:

```
def add(a, b):
    """
    Return the addition of the arguments: a + b

    >>> add(1, 2)
    3
    >>> add(-1, 10)
    9
    >>> add('a', 'b')
    'ab'
    >>> add(1, '2')
    Traceback (most recent call last):
      File "test.py", line 17, in <module>
        add(1, '2')
      File "test.py", line 14, in add
        return a + b
    TypeError: unsupported operand type(s) for +: 'int' and 'str'
    """
    return a + b

def subtract(a, b):
    """
    Returns the result of subtracting b from a

    >>> subtract(2, 1)
    1
    >>> subtract(10, 10)
    0
    >>> subtract(7, 10)
    -3
    """
```

```
    return a - b
```

Now we need to to create another module in the same location as this one that will turn our doctests into unittests. Let's call this file **test_doctests.py**. Put the following code inside of it:

```
import doctest
import my_docs
import unittest

def load_tests(loader, tests, ignore):
    tests.addTests(doctest.DocTestSuite(my_docs))
    return tests
```

The function name is actually required here for Test Discovery to work, according to the documentation for the doctest module. What we're doing here is adding a suite to the **tests** object in much the same way as we did earlier. In this case, we are using doctest's **DocTestSuite** class. You can give this class a setUp and tearDown method as parameters should your tests need them. To run this code, you will need to execute the following command in your new folder:

```
python -m unittest discover
```

On my machine, I received the following output:

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK
```

You will note that when you use unittest to run doctest, each docstring is considered a single test. If you were to run the docstrings with doctest directly, then you will notice that doctest will say there are more tests. Other than that, it works pretty much as expected.

Wrapping Up

We covered a lot in this chapter. You learned the basics of using the unittest module. Then we moved on and learned how to use unittest from the

command line. We also found out how to set up and tear down tests. You discovered how to create test suites too. Finally we learned how to turn a series of doctests into unittests. Be sure to spend some time reading the documentation on both of these fine libraries as there is a lot of additional functionality that was not covered here.