Option Flags and Directives

The doctest module comes with several option flags that you can use to control doctest's behavior. There are quite a few of them, so we well only look at a couple of examples of one flag to show how you might use them. For full details, be sure to read the official documentation on doctest!

One of the easiest ways to use an option flag is with a directive. A doctest directive is a special Python comment that follows an example's source code. One of the most popular option flags to use is the **ELLIPSIS** flag. Let's put the following code into a file and name it **test_directives.py**:

```
"""
print(list(range(100))) # doctest: +ELLIPSIS
#[0, 1, ..., 98, 99]

class Dog: pass
Dog() #doctest: +ELLIPSIS
#<__main__.Dog object at 0x...>
"""

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

The ELLIPSIS flag allows us to cut out part of the output and still pass the test. For the first example, we create a list with 100 elements, 0-99. Since this is a lot of data that we don't want to store in our docstring, we use the ELLIPSIS option flag. This allows us to put an ellipsis in the middle of the result to represent the values that we're not showing.

In the second example, we want to create an instance of the Dog class, which will return an object with a name that changes every time you run this test. Since we can't put something dynamic into our doctest, we can use the ELLIPSIS option flag to tell doctest to ignore most of the object's name.

Wrapping Up

Now you should know the basic uses for the doctest module. You can write self-testing docstrings with this knowledge as now the docstrings not only document your code, but they also allow you to test it. If you find yourself needing to write more in-depth documentation, then you can actually write it in a simple mark-up language known as ReStructuredText, which is what the official Python documentation itself is written in. Then you can run doctest against your documentation files and make sure you haven't broken your code when you inevitably need to update it. The doctest module really is a handy little tool to keep at hand.

https://docs.python.org/3/library/doctest.html https://pymotw.com/2/doctest/ http://www.blog.pythonlibrary.org/2014/03/17/python-testing-with-doctest/