07 Practical

DocTests

Recall that a doctest looks like this:

```
import doctest

def add(x, y):
    """
    Add two numbers
    >>> add(5, 2)
    7
    >>> add(2, 2)
    4
    >>> add(9, 12)
    21
    """
    return x + y

doctest.testmod(verbose=True) # Run the tests
```

1. **Copy the function definition below and write some doctests for it.** Think about what you expect the result to be if y is 0.

```
def divide(x, y):
    """
    Divide two numbers
    ** Write some tests **
    """
    pass
```

2. Now that you have written the tests, fill in the body of the function (i.e. the part that does stuff).
   This is called test driven development!

Unit Tests

Recall that a unit test is written like so:

```
import unittest

class TestAdd(unittest.TestCase):
   def test_add(self):
       self.assertEqual(add(3, 4), 7)
       self.assertTrue(add(3,4) == add(6,1))

unittest.main()  # Run the tests
```

3. Write a similar unit test for the divide function you wrote previously. Have a think about why unit tests might be better than DocTests
4. Write some unit tests for last lessons functions - would those tests be better suited to unit tests or doctests?