Companies House Accounts pipeline construction: unittests Elliott Phillips May 15, 2020



Unit testing – What is unit testing?

"Testing" is the practice of writing code, separate from your actual application code, that **invokes** the code it tests to help determine if there are any errors.

It does not prove that code is correct – it merely reports if the conditions the tester thought of are handled correctly.



Unit testing – What is unit testing?

We unittest the smallest testable parts of a software, in isolation, to validate they perform as designed.

- Test fixture: used as a baseline for running tests, ensuring testing environment is fixed so that results are repeatable.
- Test case: a set of conditions used to determine whether a system under test functions correctly.
- Test suite: a collection of test cases, used to test and confirm a software's specified behaviours by aggregated and executing tests.
- Test runner: a component that sets up the execution of tests and provides the outcome to the user.



Unit testing – What is unit testing?

Manage test fixtures using code.

Test itself.

Possible outcomes

OK - all tests passed.

FAIL - test did not pass and an AssertionError exception is raised.

ERROR — test raises an exception other than AssertionError.

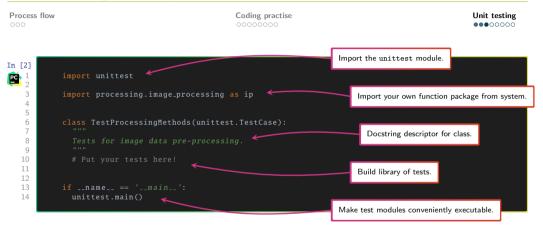


Unit testing – **Example function**

```
In [1]
P 1
           def import_files(directory: str):
   6
   8
   9
  10
             if directory is None:
               raise ValueError("This function requires the input 'directory' to be a string.")
  14
  15
             return [_file for _file in listdir(directory) if isfile(join(directory, _file))]
```



Unit testing – **Script infrastructure**



Each function for unit test name must start with the five characters 'testa'.



Unit testing – Individual test format

For each test, we adopt the given-when-then protocol:

given some input argument...

when its corresponding result is...

then assert result is aligned to expected.

Standardising approach with clear systematic structure.



Unit testing – Case 1: Equality assertion

```
Process flow
                                                   Coding practise
                                                                                                        Unit testing
                                                                                                        .....
In [3]
PC 1 2
            class TestProcessingMethods(unittest.TestCase):
              def test_import_files_count(self):
    6
    8
    Q
                # When
                result = len(ip.import_files(directory))
   10
                expected = 7
   14
                self.assertEqual(result. expected)
```

Here I confirm that len(ip.import_files(directory)) is equivalent to the expected result 7, using the .assertEqual() method.



Unit testing – Case 2: Object type error



Unit testing – Case 3: None type error



Unit testing – Case 4: Return type error

```
In [6]

class TestProcessingMethods(unittest.TestCase):

"""

Tests for image data pre-processing.

"""

def test_return_type(self):
    # Given
    text_string = "Hello world!"

# When
    result = ip.strip_punctuation(text_string)

# Then
    self.assertTrue(isinstance(result, str))
```

