Lab 2 Test Code

By Dr. J

I have written a collection of test codes that will examine your four programs (`dashes.py, patterns.py, reading.py, series.py`) and will tell you if the program you wrote matches the requirements in the Lab 2 handout. The goal is to help you avoid easy mistakes by checking that your output exactly matches the output given in the Lab handout. ***Your program will still be graded by your teacher for other factors such as comments, spacing and efficiency.***

What is a test code?

Very briefly, a test code (or *test suite*) is a computer program whose job is to run another computer program, feed it inputs, and inspect its outputs. Someday later in the course, we can talk more about *test-driven development*, which is a common approach to software development that is often used in industry built around using test suites to guide your development.

How do I use it?

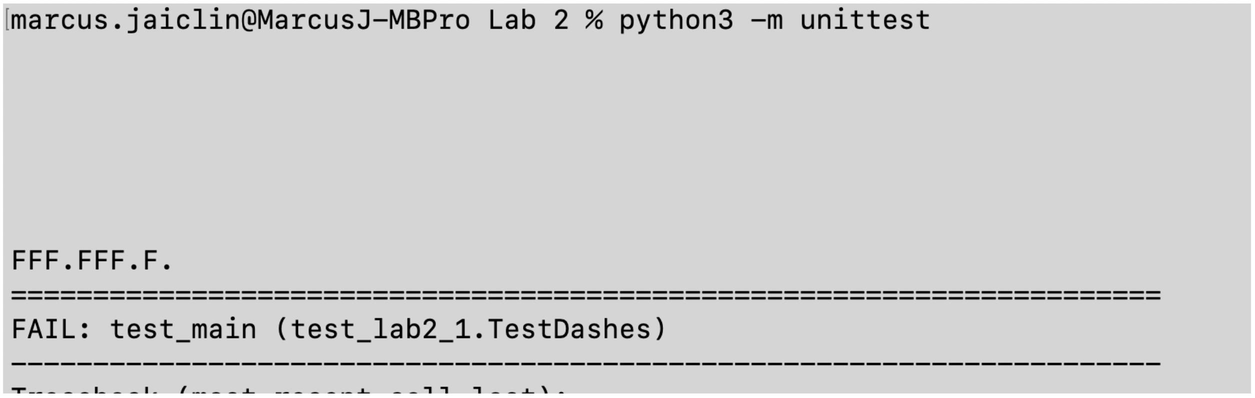
There are three Python programs in the folder with this document: `test_lab2_1.py, test_lab2_2.py, test_lab2_3.py`. You will need to copy these into the same folder with the programs you wrote. You will need to make sure that the filenames or your programs are exactly the same as described in the handout: `dashes.py, patterns.py, reading.py, series.py`. Then, in your Terminal or Command Prompt window, in the same way you would run your code, run:

`python3 -m unittest` (or whatever command you use to run Python: `python, py`)

How does it work?

It will run 10 tests on your code, matching the examples given in the Lab handout. Two of the programs get tested twice, two of them get tested three times. It will probably print out a lot of text, showing the results of each test. The first thing to do is to scroll up (it may be a few screens) until you see something that looks like this:

(Continued on next page)



```
marcus.jaiclin@MarcusJ-MBPro Lab 2 % python3 -m unittest





FFF.FFF.F.
======================================================================
FAIL: test_main (test_lab2_1.TestDashes)
----------------------------------------------------------------------
```

You should see the command you just ran at the top of the section you are looking for, followed by several blank lines.

The first line after that should have 10 characters, one for each test. There are three choices for each character:

- F = Failed Test
- E = Error
- . = Passed Test

So, in this test run, I have 7 failed tests, no errors, and 3 passing tests.

Each Failed Test or Error will have a block of text, I'll show you a couple of examples of what to look for:

Example 1:

```
======================================================================
FAIL: test_main (test_lab2_1.TestDashes)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/unitte
st/mock.py", line 1369, in patched
    return func(*newargs, **newkeywargs)
  File "/Users/marcus.jaiclin/Documents/PowerfulPython/TDD1/my_lab_tests/Lab 2/t
est_lab2_1.py", line 35, in test_main
    input_call.assert_called_with('Enter text: ')
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/unitte
st/mock.py", line 919, in assert_called_with
    raise AssertionError(_error_message()) from cause
AssertionError: expected call not found.
Expected: input('Enter text: ')
Actual: input('Enter Text: ')
```

Starting at the top, the first line of text shows that this was a Failed test, not an error. If you go across that line, you see that it ends with `TestDashes`. This means that this failed test was in `dashes.py`. Now, skip to the bottom, a lot of the stuff in between is not that useful to you at this point in your career.

The last two lines say:
`Expected: input('Enter text: ')`
`Actual: input('Enter Text: ')`

So, the test failed because something is different from what it was supposed to be. It might take you a second to notice the difference: the program used a capital `T` on Text instead of a lower-case `t`. OK, I can fix that really easily. Notice that you will see this Failed test twice, because it is testing `dashes.py` twice. So, we can fix two of the seven failed tests just by changing the `T` to a `t`.

Example 2:

```
======================================================================
FAIL: test_main (test_lab2_1.TestReading)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/unitte
st/mock.py", line 1369, in patched
    return func(*newargs, **newkeywargs)
  File "/Users/marcus.jaiclin/Documents/PowerfulPython/TDD1/my_lab_tests/Lab 2/t
est_lab2_1.py", line 94, in test_main
    self.assertEqual(input_calls[1].args[0], 'Days left? ')
AssertionError: 'Days Left? ' != 'Days left? '
- Days Left?
?      ^
+ Days left?
?      ^
```

Again, check the top line, that shows us that we are testing `reading.py`. Then, go to the bottom, and we see that we used a capital `L` instead of a lower-case `l`. Another really easy fix.

For this set of four programs, all the Failed tests were copies of the two above, or this last example, so I only have to fix three things to change all 7 tests to passing:

Example 3:

```
======================================================================
FAIL: test_main (test_lab2_1.TestPatterns)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/unitte
st/mock.py", line 1369, in patched
    return func(*newargs, **newkeywargs)
  File "/Users/marcus.jaiclin/Documents/PowerfulPython/TDD1/my_lab_tests/Lab 2/t
est_lab2_1.py", line 56, in test_main
    self.assertEqual(print_calls[0].args[0], '1 2 3')
AssertionError: '1 2 3 ' != '1 2 3'
- 1 2 3
?      -
+ 1 2 3
```

Check the top, we are looking at `patterns.py`.

At the bottom, hmmm, those look the same to me. But, look at the line above the two patterns:

```
AssertionError: '1 2 3 ' != '1 2 3'
```

If you look carefully at the pattern on the left, there is an extra space at the end of that one, compared to the one on the right. Now, if you look at the third line from the bottom again, you can see that there is a dash under that extra space at the end – that's telling you that you have an extra space there. Notice that the third to last line starts with a `-` and the last line starts with a `+`. The one with the minus is what you should *take out* of your code, the one with the plus is what you should *add into* your code. So, you need to figure out how to get your code to not produce that extra space at the end of your pattern, and then you're all set!

What happens if I get an Error?

I didn't show any examples of Errors. That's because they are rare and all different. You should have gotten all the Errors out just by running your code without the test code.

If you get an Error from the test code, try to run your code as usual and see if it runs correctly or not. If it does, see if you can understand what the error message means. If you need help understanding it, ask your teacher. (**Pro tip:** If you are going to ask for help over email, make sure you attach the entire Python code for your program, and also a screenshot of the entire error message, starting with the line with `===========` and ending with the next line of `===========`.)