Assignment 5

Testing and Debugging

EC602 Fall 2016

Contents

1 Introduction

1.1 Assignment Goals

The assignment goals are to

- provide experience with designing tests
- provide experience with handling buggy code using exceptions
- introduce the unittest module of python

1.2 Due Date

This assignment is due 2016-10-10 at midnight.

1.3 Submission Link

You can submit here: week 5 submit link

2 Background: Testing

Testing is a critical component of all engineering endeavors.

Here are some examples of electrical and computer engineering items which can be tested:

- research results
- research equipment
- prototypes
- chips

Of course, software also can and must be tested.

Software can be tested for

- meeting its specifications
- robustness to user action or error
- speed or efficiency
- compatibility with prior versions
- compatibility with different versions of hardware, operating systems, browsers

A very prominent design methodology is test driven development. No software is written until the test for the software is written.

3 Background: Inheritance

Python, C++ and virtually all other languages that support objects include an import technique and concept called *inheritance*

The class inherited from is called the parent or base class, and the class which inherits its properties and code is called the child or derived class.

Here is a simple python example

```
class Animal():
    pass

class Dog(Animal):
    pass
```

4 Python's unittest

Python includes a module called `unittest` which provides a framework fo building tests.

4.1 Example: testing_complex

Here is an example of how to use `unittest` to test the `Complex` class which was provided as part of HW 4.

```
"""
Example of using unittest to test a class.
The class begin tested is Complex
"""
import unittest

from model_complex import Complex

class ComplexTestCase(unittest.TestCase):
    """unit testing for polynomials"""

    def setUp(self):
        pass

    def test_init(self):
        z = Complex()
        self.assertIsInstance(z,Complex)


    def test_eq(self):
        z = Complex(3,5)
        w = 3+5j
        self.assertEqual(z,w)

    def tearDown(self):
        "tear down"

if __name__ == '__main__':
```

```
    unittest.main()
```

Here is a link to the code: testing_complex.py

5 JSON

The results of this weeks assignment will be stored in JSON format

Here is an example of the format week5_results.json

```
{
    "authors": [
        "test@bu.edu",
        "jbc@bu.edu",
        "brower@bu.edu"
    ],
    "failed": [
        "poly2.py"
    ],
    "passed": [
        "poly1.py"
    ]
}
```

6 Tester

The following tester program shows how to

- import all the modules to test (with names `poly*.py`)
- use the `loader` and `results` features of `unittest`

Your task is to complete the class `PolynomialTestCase` in a separate file called `w5_testpoly.py`

You are provided a testing framework, which you do not need to modify or hand in. The tester, shown below, is also available here: w5_tester.py

```
import unittest
import importlib
import glob
import io
import sys
import json

import w5_testpoly
```

```
suppress_output = True

def check_all_files():
    passed,failed = [],[]

    Trials = glob.glob('poly*.py')

    for file_name in Trials:
        loader = unittest.loader.TestLoader()
        results = unittest.result.TestResult()

        try:

            if suppress_output:
                s = io.StringIO()
                sys.stdout = s

            module_tested = importlib.import_module(file_name[:-3])
            w5_testpoly.Polynomial = module_tested.Polynomial
            tests = loader.loadTestsFromTestCase(w5_testpoly.PolynomialTestCase)
            tests.run(results)

            tests_passed = results.testsRun - len(results.failures) - len(results.errors)

            if results.wasSuccessful():
                passed.append(file_name)
            else:
                failed.append(file_name)
            if suppress_output:
                sys.stdout = sys.__stdout__

        except Exception as e:
            if suppress_output:
                sys.stdout = sys.__stdout__

            print('exception',file_name,e)
            failed.append(file_name)


    return passed,failed

if __name__ == "__main__":
    passed,failed = check_all_files()
    Results={'failed':failed,'passed':passed,'authors':w5_testpoly.authors}
    with open('week5_results.json','w') as f:
        json.dump(Results,f,indent=4)
```

This program reads in all the files called `poly*.py` and checks them against a collection of test cases you have designed and put into `w5_testpoly.py`

The results are stored in a JSON file `week5_results.json` which you can submit to the website for checking.

The following code can be added to the tester so that your test case results are printed as well:

```
    print('Run {} tests'.format(results.testsRun))

    print('you passed {} tests'.format(tests_passed))
    for test,output in results.failures:
         print(">>",test)
         print(">>",output)
    for test,output in results.errors:
         print(">>",test)
         print(">>",output)
```

Be sure to set the flag `suppress_output` to `False` so that you can see the results.

7 Template

Starting this week, we are submitting results using JSON format, and you must include the author list as part of the JSON file. The following program w5_testpoly.py shows how to do this:

```
# AUTHOR test jbc@bu.edu
# AUTHOR jbc jbc@bu.edu
# AUTHOR brower brower@bu.edu
import unittest

authors=['test@bu.edu','jbc@bu.edu','brower@bu.edu']

class PolynomialTestCase(unittest.TestCase):
    "empty class"
```

Note that the tester includes the author list as part of the JSON file output.

8 The assignment

Write a python program `w5_testpoly.py` that tests a directory full of possibly correct implementations of a class Polynomial.

The programs to be tested are available here There are one hundred files.

Here are the requirements of the `Polynomial` class which you should be testing:

- implement a constructor which takes a sequence and assigns the coefficients in the natural (descending order). So `Polynomial([4,-9,5.6])` should make $4x^2 - 9x + 5.6$
- implement addition and subtraction of polynomials using `+` and `-`
- implement multiplication of polynomials using `*`
- implement testing for equality of polynomials using `==`
- implement an efficient mechanism for handling sparse polynomials
- implement negative powers in the polynomial, i.e. you should be able to handle $p(x) = x^{-1}$
- implement evaluation of the polynomial using a `eval` method, like this: p.eval(2.1)
- implement accessing and modifying the coefficients using `[]`. So `p[2]` should be the coefficient of x^2 and `p[8] = 12` should set the coefficent of x^8 to 12.
- implement a derivative method `p.deriv()` which returns the derivative of the polynomial.

Your program should be called `w5_testpoly.py`