Assignment 5: Testing and Debugging

EC602 Design by Software

Fall 2017

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 Group Size

For this assignment, the maximum group size is 3.

1.3 Due Date

This assignment is due 2017-10-13 (Friday) at midnight.

1.4 Points

This assignment is worth 7 points (4 for JSON file, 3 for python script)

1.5 Submission Link

You can submit here: collision tester 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. In this methodology, 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 important 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 for 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

4.2 Examples: the checkers

Please look for examples in

- polyops_checker.py
- bigint_checker.py
- system_checker.py
- modeling_checker.py
- ec602lib.py

5 JSON

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

Here is an example of the format collisiontest_results.json

```
collision_tester: more collisiontest_results.json
{
    "authors": [
        "?@bu.edu",
        "??@bu.edu",
        "???@bu.edu"
    ],
    "passed": [
        "collision_tester.py",
        "collisiontester_checkall.py"
    ],
    "failed": [
        "collision_tester.html",
        "collision_tester.md",
        "collision_tester.pdf",
        "collisiontest_results.json"
```

```
    ]
}
```

6 Tester

The following tester program shows how to

- find all the programs to test (with names `collision*`)
- use the `loader` and `results` features of `unittest`

Your task is to complete the class `CollisionTestCase` in a separate file called `collision_tester.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: collisiontester_checkall.py

This program finds all files called `collisionc*` and checks them against a collection of test cases you have designed and put into `collision_tester.py`

The results are stored in a JSON file `collisiontest_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 collision_tester.py shows how to do this:

```
"""this is the main part of the assignment"""
```

```
# Copyright ? ?@bu.edu
# Copyright ?? ??@bu.edu
# Copyright ??? ???@bu.edu
import unittest
import subprocess

#please change this to valid author emails
AUTHORS = ['?@bu.edu', '??@bu.edu', '???@bu.edu']

PROGRAM_TO_TEST = "collision"

def runprogram(program, args, inputstr):
    coll_run = subprocess.run(
        [program, *args],
        input=inputstr.encode(),
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE)

    ret_code = coll_run.returncode
    program_output = coll_run.stdout.decode()
    program_errors = coll_run.stderr.decode()
    return (ret_code, program_output, program_errors)


class CollisionTestCase(unittest.TestCase):
    "empty class - write this"
    def test_one(self):
        strin = "one 20 10 -2 1"
        correct_out = "3\none 14 13 -2 1\n"
        (rc,out,errs) = runprogram(PROGRAM_TO_TEST,["3"],strin)
        self.assertEqual(rc,0)
        self.assertEqual(out,correct_out)
        self.assertEqual(errs,"")

    def test_programname(self):
        self.assertTrue(PROGRAM_TO_TEST.endswith('.py'),"wrong program name")

def main():
    "show how to use runprogram"

    print(runprogram('./test_program.py', ["4", "56", "test"], "my input"))
    unittest.main()

if __name__ == '__main__':
    main()
```

A simple test program was written to help us check `runprogram`: test_program.py

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

8 The assignment

Write a python program `collision_tester.py` that tests a program for correct implementation of assignment 6 on collisions.

You should submit both your python script and the JSON file `collisiontest_results.json` coming from its use with `collisiontester_checkall.py`

8.1 The programs to test

The following zip file contains 47 example programs to test.

- collisionc_files.zip

They are compiled inside the devbox and should run there.