Discussion 5: Unit Tests and Debugging

SI 206: Data-Oriented Programming

Instructor: Dr. Barbara (Barb) Ericson
GSI: Kexuan (Michael) Huang
IA: Cristina & Jade

School of Information
University of Michigan

Fall 2023

Reminders

- Commit at least 4 times to get full credit on assignments and projects
- Please submit Python file that can be executed for assignments and projects

Deadlines

- Homework 4 due this Friday
- Midterm 1 on next week during lecture time (Wed/Thur)

Table of Contents

Table of Contents

What are Tests?

- Tests are a checklist of user inputs that your programs have to pass
- We have to make sure the programs "survive" and give expected output.

How to break this program?

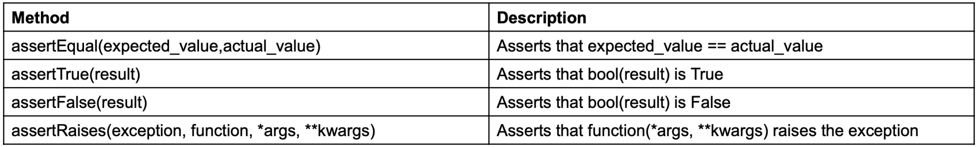
```python
def calculate_average(numbers):
    return sum(numbers) / len(numbers)

print_first_element(?????) # try giving different input here
```

How to test?

- Generate different inputs (common and edge cases)
- Calculate expected output with our brain
- Run the program with these inputs, and hope it won't throw an error
- Compare the program output with our brain output

Unit Tests

- Unit test tests individual piece of code (e.g. functions) in isolation from the rest of the program
- unittest is a library (code written by others) to write tests easily in Python



Method	**Description**
assertEqual(expected_value,actual_value)	Asserts that expected_value == actual_value
assertTrue(result)	Asserts that bool(result) is True
assertFalse(result)	Asserts that bool(result) is False
assertRaises(exception, function, *args, **kwargs)	Asserts that function(*args, **kwargs) raises the exception

Figure 1: Basic Assertions that unittest offers

Unit Tests

```python
import unittest

def calculate_average(numbers):
    return sum(numbers) / len(numbers)

class TestAll(unittest.TestCase): # make a subclass of unittest.TestCase
    def test_calculate_average(self): # start each method with "test_"
        # test normal cases
        self.assertEqual(calculate_average([1]), 1)
        self.assertEqual(calculate_average([1, 2, 3]), 2)
        # test edge cases
        self.assertEqual(calculate_average([]), "invalid input!")
        self.assertEqual(calculate_average("haha"), "invalid input!")

unittest.main() # run all tests
```

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Table of Contents

Testing and Debugging Tips

- Write tests first, then write program.
- Start small: don't wait for code get too long to test it
- Comment out things you don't need
- Use print statement (in for loop, functions ...)
- Break complicated lines in to shorter ones

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Table of Contents

Discussion 5 Exercise

Go to Canvas → Assignment → Discussion 5 and clone the GitHub repository https://classroom.github.com/a/c6GUwOIN

Tasks

- Write test cases for function `count_a()` and fix this function
- Write test cases for the `Warehouse` class

Submission

- Commit at least 4 times and push to GitHub
- Submit the repository link to Canvas by the end of this discussion