Assignment 4: Classes and Unit Tests

Due: Friday October 18 at 11:59:59 PM

Purpose

For this assignment, you will (1) get practice implementing code as part of a Python class hierarchy, and (2) get practice writing unit tests for code.

Extra Background

Familiarize yourself with the Stack data structure:
https://en.wikipedia.org/wiki/Stack_(abstract_data_type)

This assignment will have you implementing and testing several different types of Stacks. In addition to push() and pop(), the stacks in this assignment also have the peek() and size() methods:

- size() - returns the number of items currently in the stack.
- peek() - returns the newest item in the stack (without removing the item).

Instructions

Download the assignment 4 starter code here:
https://github.com/mtsargent/eecs-297-397-examples/tree/master/assignment-004

The files stack.py and stack_test.py are my provided base Stack class and some unit tests for the class. These are examples and you do not have to alter these.

Implement your code in assignment4.py and assignment4_test.py. Add the functionality described to each of the three classes (UniqueStack, LimitedStack, and RotatingStack) in assignment4.py. Make sure to leave the names of the classes and the method names from the base Stack class alone (do not rename them).

Before/while you are writing your code for the three stack classes, also add unit tests that will capture the new, unique behaviors of each class. Add your tests to assignment4_test.py. Replace the existing stub tests with your own. Add as many tests as you feel are needed in order to verify that the stack classes you are writing will work as described.

Once the three stack classes are implemented and unit tested, submit assignment4.py and assignment4_test.py to Canvas.

Stack Descriptions

See the Python files for full details, but these are the three stacks you will implement in a nutshell:

- UniqueStack - This stack can only contain unique items at any point in time. An error is raised when there is an attempt to add a repeat item.
- LimitedStack - This stack can only contain up to a certain number of items at any point in time. Attempting to add an item that would cause the stack to go beyond capacity will instead raise an error.
- RotatingStack - This stack can only contain up to a certain number of items at any point in time. When pushing an item causes the stack to go beyond capacity, the push succeeds, but then the oldest item in the stack is dropped (keeping the final size after the drop equal to max size). **Note**: This class inherits from LimitedStack rather than the base Stack class.

Grading

- 60% - Submitted files contain unit tests that are all passing. Unit tests are testing for relevant behaviors for each of the stack classes.
- 30% - Passing tests demonstrate described functionality is met for each type of stack. Relevant error cases and edge cases are tested in addition to core functionality for each class and the behaviors are correct.
- 10% - Style/code layout. A large focus of this category is on choosing meaningful names throughout your program and producing readable code that takes minimal effort to understand. Try to name everything in a way that requires no extra comments to explain.

I plan on running your own submitted tests as well as my own to test for overall functionality. If all goes well, all of your submitted tests and my tests should pass.