Project 0: Using Pytest

Purpose

The purpose of this project is to ensure that you understand how to properly test and lint your programs using pytest and pylint, respectively. If you write programs that run, but do not pass the specified test requirements, you will not pass this class.

1. What is pytest?
   Pytest is a unit-testing module for .py files. For each project in this course, you will be provided with a spec file, the corresponding test file(s), and any necessary additional files. You must run pytest against your code for each project before submitting. It is recommended that you regularly run pytest against your program as you write it, as it may return useful information that will help you as you write your code.

2. What is pylint?
   Pylint is a linter for python files. In so many words, it checks the "cleanliness" of your program and returns a pylint score out of 10.0, with the minimum expected score for each project being 8.5. i.e. Scoring 10 on pylint will give you the same grade as if you scored an 8.5, but scoring less than 8.5 will be detrimental to your grade. Pylint returns useful information concerning which lines of code could use tidying up.

Installing Pytest and Pylint

1. (Optional) If you are using Thonny as your IDE, search online for another IDE that is compatible with Python. For instance, you might consider VSCode, PyCharm, Spyder, or Komodo, among many others. Ultimately, whichever IDE you decide to use is up to you.

2. Install python onto your machine if you have not already done so. We will be grading using python 3.8, so it is recommended that you install python 3.8. Whether you are running Linux, Mac, or Windows, you must be sure to include the appropriate directory in your system's PATH variable.

3. Find out how to install pip onto your machine, and install it. Again, be sure to include the appropriate directory in your system's PATH variable.

4. Find out how to install pytest and pylint onto your machine using pip. The directory should already be in the PATH variable, as pytest and pylint should install in the same location as pip.
5. Make sure python (or python3), pip, pytest, and pylint are all valid commands in your machine's terminal.

Problem

You are provided with a python file called *arithmetic.py*. This file contains five methods. The methods and their intended functions are as follows:

- add(num1,num2) — Adds num1 to num2 and returns the sum.
- subtract(num1,num2) — Subtracts num1 from num2 and returns the difference.
- multiply(num1,num2) — Multiplies num1 and num2 and returns the product.
- divide(num1,num2) — Divides num1 from num2 and returns the quotient.
- int_to_string(num) — Converts an int to a string and returns the string.

This file compiles and runs without error. However, running pytest against it will show that only the add() function passes the test. Running pylint against it returns a negative pylint score. Your job is threefold:

1. Rewrite each method (aside from add()) such that it passes the corresponding test case in *test_arithmetic.py*.
2. Run pylint against *arithmetic.py* to determine where it needs to be cleaned up, and then clean it sufficiently so that it returns a pylint score of 8.5 or higher.
3. Change the arguments passed into each function in main() (except for int_to_string(num)), such that the output is meaningful. e.g. Avoid identity functions, etc. What constitutes as "meaningful" is up to you. You do NOT have to use the global list provided.

Grading (100 points)

Pytest will be run on your *arithmetic.py*.
Score is the sum of:

- Percentage of test cases passed × 80 points.
- min(Coding style score/8.5, 1) × 20 points.
- Possible adjustment due to the physical inspection of the code, to make you actually implemented things and updated the arguments in main() accordingly.

What to Submit

1. *arithmetic.py*
2. A short 1-minute video showing your code running, as well as pytest running against your code. If on Youtube, make sure the link is unlisted and not searchable, and paste the link in a comment with your submission.