Test-driven development Lab

Assignment purpose

This lab is designed to help you develop competence in unit testing and code coverage in Python. You will also learn about the test-driven development (TDD) method discussed in the lecture.

Lab environment installation and setup:

Please follow the following steps to prepare the Python enjoinment to complete the lab.

1. Download and Install python (https://www.python.org/downloads/)
2. Download and Install Visual Studio Code (https://code.visualstudio.com/download)
3. Install Python code coverage tool (https://coverage.readthedocs.io/en/7.3.2/)
4. Please make sure the coverage tool is installed and added to the path before running it (https://coverage.readthedocs.io/en/7.2.7/cmd.html)
5. Please see the commands below for installation. If the coverage command is not found after installation, you will need to add the path (see below).
    a. `python3 -m pip install coverage`
    b. `export PATH="$PATH:paste your path here" (see the example below).`
    c. `export PATH="$PATH:/Library/Frameworks/Python.framework/Versions/3.11/bin"`
    d. `check the coverage version with the command mentioned in the screenshot below.`

6. Download the Zipped lab folder from Canvas with all the python files in it and run them in your VS code. Four Python files are in the zip folder, and the description of the files are given below.
    a. ***Task1_Rover.py*** (The code is already written)
    b. ***Task1_Unittest_rover.py*** *(*You are required to write test cases in this file)
    c. ***Task2.py*** *(*You are required to write code in it using the TDD method.)
    d. ***Task2_unittest.py*** *(*You are required to write unit test cases for the code, i.e., test-driven development fashion, written in Task2.py)

Note: Please note that you are required to write code in the given python code files only. Do not add new python files.

Task 1:

The ***Task1_Rover.py*** code converts strings to and from "rövarspråket" – all consonants are duplicated, and an "o" is inserted between them. Vowels, numbers, and other characters are kept as they are. "rovar" thus becomes "rorovovaror".

> ***Rövarspråket*** (English: The Robber Language) is a Swedish language game. It became popular after the books about Kalle Blomkvist by Astrid Lindgren, where the children use it as a code, both at play and in solving actual crimes.
> The principle is easy enough. Every consonant (spelling matters, not pronunciation) is doubled, and an *o* is inserted in-between. Vowels (and other characters, e.g. numbers) are left intact. It is quite possible to render the *Rövarspråket* version of an English word as well as a Swedish, *e.g.*:
> *sos-tot-u-bob-bob-o-ror-non* or *sostotubobboborornon*
> that syllable chain would mean *stubborn*. Needless to say, the code is not very useful in written form, but it can be tough when spoken by a trained (and thus quick) user. On the other hand, for an untrained speaker, a word or phrase can often be something of a tongue-twister or a shibboleth.
> Today, the books (and subsequent films) are well known in Sweden, so the language has become integrated in the culture of schoolchildren. Most Scandinavians are familiar with it.

Create a set of test cases (Equivalence Partitioning yields minimum 3+3 test cases, (3 tests for each of *enrov* and *derov* testing: null, empty string, and non-empty string), but then the equivalence class of "non-empty string" will be a very long string, so it pays off to split this into smaller chunks, thus creating more test cases) that tests the method(s) of the tested code. Use different test methods to find suitable test data to be used. Make sure you have considered the following:

- Have you:
  - Checked "non-empty string", containing:
  - Checked all characters a..ö (**lower case** and **UPPER CASE**)?
  - Checked all Numbers (0-9)?
  - Checked a reasonable set of other characters (e.g.!" #€%&/(),.)?
- Have you:
  - checked "empty string"?
  - checked "null pointer"?

Please use the following website to see how it works. http://www.xn--rvarsprket-75a1r.se/

Answer the following questions for Task 1:

1. Please run the test coverage module, generate the HTML report, and attach the screenshot of test coverage (see the lecture for reference)
2. What types of coverage were measured?
3. What is the code coverage you achieved with the test cases?
4. Please state and describe the defect found in the code. (There is a bug!).
5. Which test case (s) found the defect (copy and paste the test case to answer the question)?

6. How did you fix the defect(s) found in the code?

Task 2: Text analysis

In this task, you will use Python to demonstrate your skills in text manipulation problems. Write a Python program and unit test cases for the following user stories. Please document your code and include comments explaining how each user story is accomplished.

1. Convert all words to lowercase.
2. Extract all the email addresses mentioned in the text.
3. Find and count all unique hashtags (words or phrases starting with #) used in the document.
4. Identify and list all URLs mentioned in the text.
5. Calculate the average word length in the document.
6. Determine the top 3 most frequent words in the text.
7. Find the longest word in the text.
8. Identify the sentences in the document that contain the word "Python."
9. Remove all punctuation and special characters from the text.
10. Convert all numerical figures (1-10) within the text into their respective written-out forms (for example, “This is sample text 1, 2, 3” shall become “This is sample text one, two, three”).

Note: In this task, you must write the code using the Test-driven development method. You are supposed to write the test case first and then implement the code.

Answer the following questions for Task 2:

1. Please run the test coverage module, generate the HTML report, and attached the screenshot of test coverage (see the lecture for reference)
2. What types of coverage were measured?
3. How many test cases were written to test the user stories mentioned in task 2?
4. What is the code coverage you achieved with the test cases?

Submission guidelines

Submit the following items as a zip file and ensure the code is fully functional and runs without errors.

1. Please do the lab in pairs using pair programming and clearly write your names in the report and code comments of your submission.
2. Your Python project directory contains the following items. Please submit a zipped file on Canvas
    a. Submit fully functional code files for task 1 and task 2.
        i. Task1_Rover.py
        ii. Task1_Unittest_rover.py
        iii. Task2.py
        iv. Task2_unittest.py
        v. Generated HTML report of test coverage for **Task 1** and **Task 2**

b. A brief reflection report (2-3 pages max) with answers to questions in **Task 1** and **Task 2** with screenshots (wherever applicable).

Good Luck!