05 Team Activity: Testing and Fixing Functions

Instructions

Arrange a one hour synchronous meeting with your team for this activity. Online students should coordinate a video-sharing meeting. Campus students will use class time for this meeting. You should prepare for this meeting by completing the preparation material and the individual checkpoint assignment beforehand.

Purpose

Writing and running test functions often help a software developer find mistakes in code. During this assignment, you will write three test functions. Use `pytest` to run the test functions and use the output of `pytest` to help you find and fix errors in some program functions.

Problem Statement

Most people around the world today have at least two names, a family name and a given name. In the United States, we usually write a person's given name followed by his family name. However, when a computer lists names in alphabetical order, it is convenient to list the family name first and then the given name like this:

- Harrison; Anna
- Harrison; William
- Washington; George
- Washington; Martha

When writing a program that alphabetizes names, it is often helpful to have the following three functions.

> `make_full_name`
> Combines a person's given name and family name into one string with the family name first
> `extract_family_name`
> Extracts a person's family name from his full name
> `extract_given_name`
> Extracts a person's given name from his full name

A programmer has already written those three functions. However, there are mistakes in at least two of the three functions.

Assignment

Write three test functions named `test_make_full_name`, `test_extract_family_name`, and `test_extract_given_name`. Each of the test functions will test one of the three previously written functions. Use `pytest` to run the test functions and find and fix the mistakes, if any, that are in previously written functions.

Helpful Documentation

- The prepare content for this lesson explains how to use `pytest`, `assert`, and `approx` to automatically verify that functions are correct. It also contains an example test function and links to additional documentation about `pytest`.

- This short video (20 minutes) shows a BYU-Idaho faculty member writing two test functions and using `pytest` to run them.

Steps

Do the following:

1. Download the names.py Python file and save it in the same folder where you will save your Python test program. Then open the downloaded file in VS Code. Notice that the downloaded file contains three small functions named: `make_full_name`, `extract_family_name`, and `extract_given_name`. Notice also that each function has a documentation string (a triple quoted string immediately below the function header) that describes what the function does. Read the documentation strings for all three functions. The code in the functions may contains some mistakes.

2. Using VS Code, open a new Python file and copy and paste the following import statements at the top of the file.

   ```
   from names import make_full_name, \
       extract_family_name, extract_given_name
   import pytest
   ```

   Save the file with the name `test_names.py` in the same folder where you downloaded and saved the `names.py` file.

3. In `test_names.py`, write a function named `test_make_full_name` that takes no parameters. Write `assert` statements inside this function to test the value returned from the `make_full_name` function. If you are not sure what the `make_full_name` function does or how to test it, read the documentation string that is at the top of the `make_full_name` function in the `names.py` file.

4. In `test_names.py` write a function named `test_extract_family_name` that takes no parameters. Write `assert` statements inside this function to test the value returned from the `extract_family_name` function.

5. In `test_names.py` write a function named `test_extract_given_name` that takes no parameters. Write `assert` statements inside this function to test the value returned from the `extract_given_name` function.

6. At the bottom of your `test_names.py` file, write a call to the `pytest.main` function like this:

   ```
   # Call the main function that is part of pytest so that
   # the test functions in this file will start executing.
   pytest.main(["-v", "--tb=line", "-rN", __file__])
   ```

7. Save your `test_names.py` file and run it by clicking the green run icon in VS Code.

8. Read the output from `pytest` and use the output to help you find and fix any errors that are in your test functions or the `make_full_name`, `extract_family_name`, and `extract_given_name` functions.

9. Repeat steps 7 and 8 until you have found and fixed all the mistakes and your three test functions pass.

Core Requirements

1. Write `test_make_full_name` so that it tests `make_full_name` with various names, including long names, short names, and hyphenated names. Fix the mistake in the `make_full_name` function.

2. Write `test_extract_family_name` so that it tests `extract_family_name` with various names, including long names, short names, and hyphenated names.

3. Write `test_extract_given_name` so that it tests `extract_given_name` with various names, including long names, short names, and hyphenated names. Fix the mistake in the `extract_given_name` function.

Stretch Challenges

If your team finishes the core requirements in less than an hour, complete one or more of these stretch challenges. Note that the stretch challenges are optional.

1. In the United States, mailing addresses are supposed to be written in this form: *number street*, *city*, *state zipcode*
   For example: 525 S Center St, Rexburg, ID 83460

   Download and save this Python file named address.py. Open a new Python file named `test_address.py` and write a test function named `test_extract_city` that verifies that the `extract_city` function works correctly.

2. Write a test function named `test_extract_state` that verifies that the extract_state function works correctly.

3. Write a test function named `test_extract_zipcode` that verifies that the extract_zipcode function works correctly.

Testing Procedure

Before you fix the mistakes in the `make_full_name` and `extract_given_name` functions, `pytest` will print output similar to this:

```
> python test_names.py
======================== test session starts ========================
platform win32--Python 3.8.6, pytest-6.1.2, py-1.9.0, pluggy-0.13.1 --
rootdir: C:\Users\cse111\lesson05
collected 3 items

test_names.py::test_make_full_name FAILED                      [ 33%]
test_names.py::test_extract_family_name PASSED                 [ 66%]
test_names.py::test_extract_given_name FAILED                  [100%]

============================ FAILURES ============================
C:\Users\cse111\early-functions\docs\lesson05\teach_solution.py:16:
AssertionError: assert 'Smith-Jones;Ava' == 'Smith-Jones; Ava'
C:\Users\cse111\early-functions\docs\lesson05\names.py:31:
ValueError: substring not found
==================== 2 failed, 1 passed in 0.14s ====================
```

Repeat steps 7 and 8 in the Steps section above until you have found and fixed all the mistakes in the `names.py` file and your tests pass. After you fix the mistakes in the `make_full_name` and `extract_given_name` functions, `pytest` will print output similar to this:

```
> python test_names.py
======================== test session starts ========================
platform win32--Python 3.8.6, pytest-6.1.2, py-1.9.0, pluggy-0.13.1 --
rootdir: C:\Users\cse111\lesson05
collected 3 items

test_names.py::test_make_full_name PASSED                      [ 33%]
test_names.py::test_extract_family_name PASSED                 [ 66%]
test_names.py::test_extract_given_name PASSED                  [100%]

========================= 3 passed in 0.12s =========================
```

Sample Solution

Please work diligently with your team for the one hour meeting. After the meeting is over, please compare your approach to the sample solution [↓] and the stretch solution [↓]. Please *do not look at the sample solutions* until you have either finished the program or diligently worked for at least one hour. At the end of the hour, if you are still struggling to complete the assignment, you may use the sample solution to help you finish.

Ponder

During this assignment, you downloaded three program functions that work with a person's name. You wrote three test functions that each test one of the program functions. You used `pytest` to run your test functions and used the output of `pytest` to help you find and fix the mistakes in the program functions. How will writing and running test functions help you write better programs?

Submission

When you have finished the activity, please report your progress via the associated I-Learn quiz. When asked about which of the requirements you completed, feel free to include any work done during the team meeting or after the meeting, including work done with the help of the sample solution, if necessary. In short, report on what you were able to accomplish, regardless of when you completed it or if you needed help from the sample solution.