**Code Explanation**

*city\_functions.py*

1. **Function Definition**:
   * The function is defined using the def keyword, followed by the function name and its parameters.
   * A docstring is included to describe the function's purpose.
2. **Return Statement**:
   * The function returns a formatted string that combines the city and country, separated by a comma.
3. **Function Calls**:
   * The function is called three times with different city and country pairs: "Santiago, Chile", "Paris, France", and "Montreal, Canada".
   * A computer screen shot of a program code

     Description automatically generatedEach call to the function is printed to the console, allowing us to see the output directly.

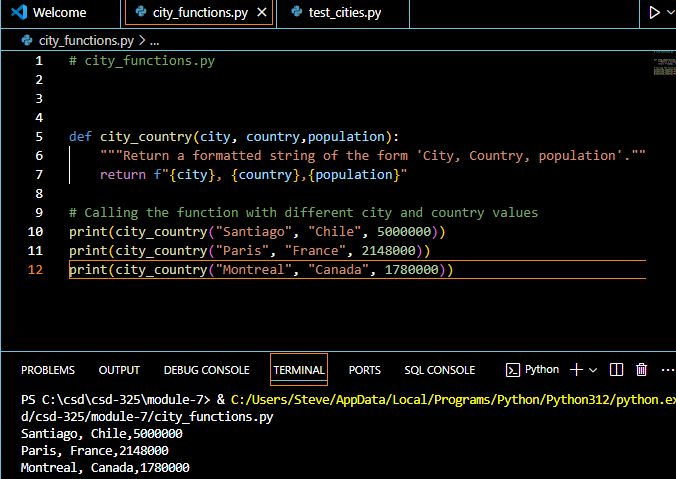
*test\_cities.py*

1. **Importing Modules**: We import the unittest module for testing and the city\_country function from the city\_functions module.
2. **Test Class**: We define a class TestCityCountry that inherits from unittest.TestCase. This class contains our test methods.
3. **Test Method**: The method test\_city\_country() checks if the city\_country function returns the expected formatted strings for the given inputs. We use assertEqual to compare the output of the function with the expected result.
4. A screenshot of a computer program

   Description automatically generated**Running Tests**: The if \_\_name\_\_ == '\_\_main\_\_': block ensures that the tests run when the script is executed directly.

*Modify city\_functions.py*

The function *city\_functions* is modified to accept a third parameter, population, and the return string is updated accordingly. This change allows us to provide more detailed information about the city.



We rerun the test\_cities.py. If the test does not match the expected output, it will fail, demonstrating how to handle changes in function behavior.

A screenshot of a computer program

Description automatically generated

*city\_functions.py*

1. **Function Definition**: The city\_country function takes three parameters: city, country, and an optional population.
2. **String Formatting**: It checks if the population parameter is provided. If it is, the function returns a string formatted as "City, Country - population xxx". If not, it returns "City, Country".

A screen shot of a computer

Description automatically generated

*test\_cities.py*

1. **Unit Testing**: This file uses the unittest framework to test the city\_country function.
2. **Test Method**: The test\_city\_country method verifies that the function returns the expected strings for various inputs, including the population.

*Modifications*

1. **Adding Population**: The function was initially modified to include a population parameter, which caused the tests to fail if it was not provided.
2. **Making Population Optional**: The population parameter was then made optional, allowing the function to pass all tests again.

A screen shot of a computer program

Description automatically generated

*city\_functions.py*

We modified the city\_country function and added a new parameter called language. This allows us to specify the language spoken in the respective city. The function now constructs a formatted string that includes the city name, country name, population (formatted with commas for readability), and language.

A computer screen shot of a program

Description automatically generated

test\_cities.py

When you run the test\_cities.py file after making these changes, it is expected to fail. This is likely because the test cases in test\_cities.py may not account for the new language parameter.

A computer screen shot of a computer program

Description automatically generated

*city\_functions.py*

The city\_country function takes four parameters: city, country, population, and language. The last two parameters are optional.

A computer screen shot of a computer program

Description automatically generated

Initially, the function was modified to include a population parameter, which caused the tests to fail. After making population optional, the tests passed again. The function was then updated to include a language parameter, which also caused the tests to fail. After making language optional, the tests passed successfully.

A computer screen shot of a program

Description automatically generated