



NOVEMBER 30, 2024

ADVANCE PYTHON

MODULE 7.2: TEST CASES

STEVE STYLIN
BELELVUE UNIVERSITY



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".
- Each call to the function is printed to the console, allowing us to see the output directly.

```
1 # city_functions.py
2
3
4
5 def city_country(city, country):
6     """Return a formatted string of the form 'City, Country'."""
7     return f"{city}, {country}"
8
9 # Calling the function with different city and country values
10 print(city_country("Santiago", "Chile"))
11 print(city_country("Paris", "France"))
12 print(city_country("Montreal", "Canada"))
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [X] ... ^ X

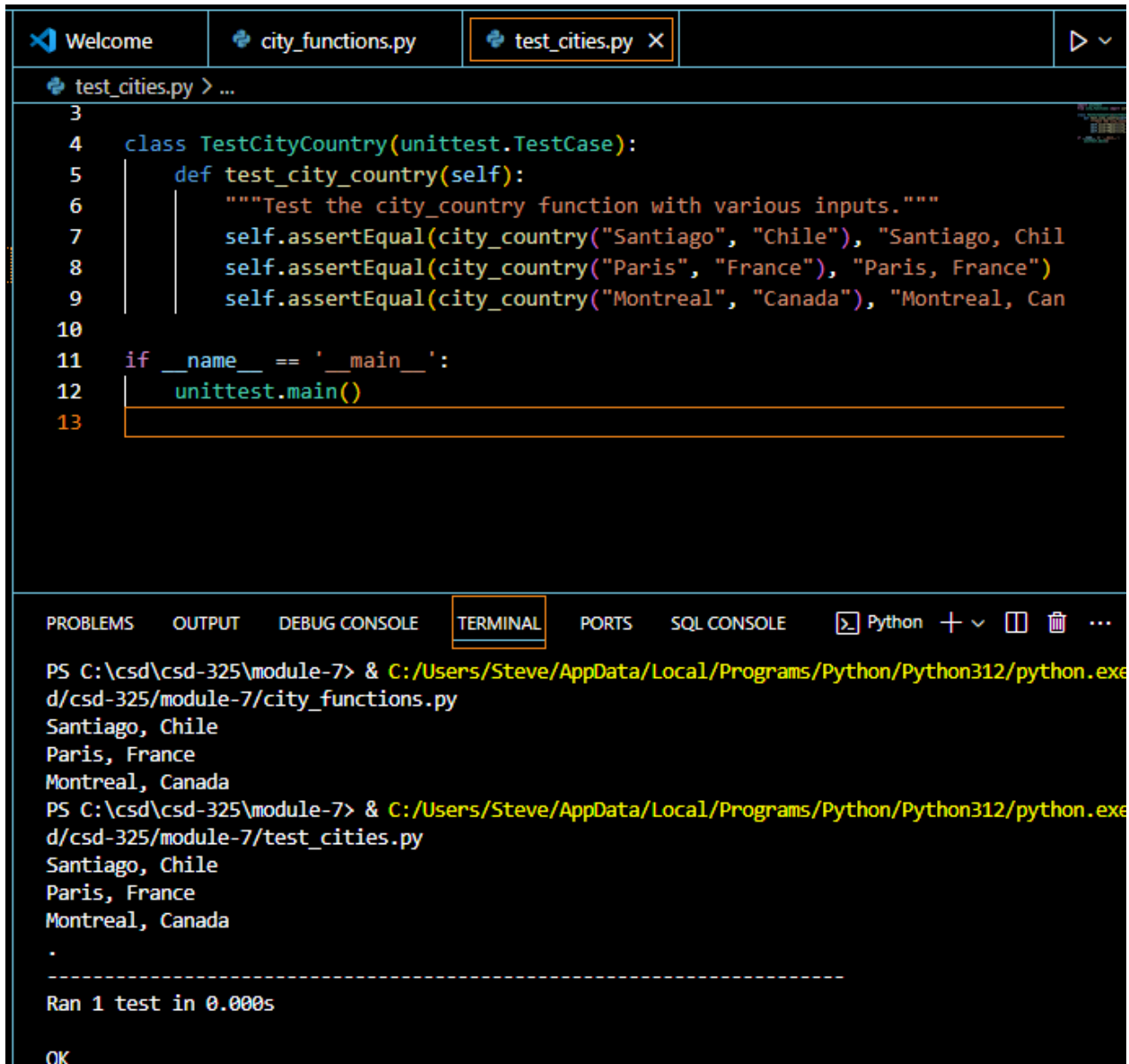
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe c:/csd/csd-325/module-7/city_functions.py

Santiago, Chile
Paris, France
Montreal, Canada

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 assertEquals to compare the output of the function with the expected result.

4. **Running Tests:** The `if __name__ == '__main__':` block ensures that the tests run when the script is executed directly.



The screenshot shows the Visual Studio Code interface. The top pane displays the file `test_cities.py` with the following code:

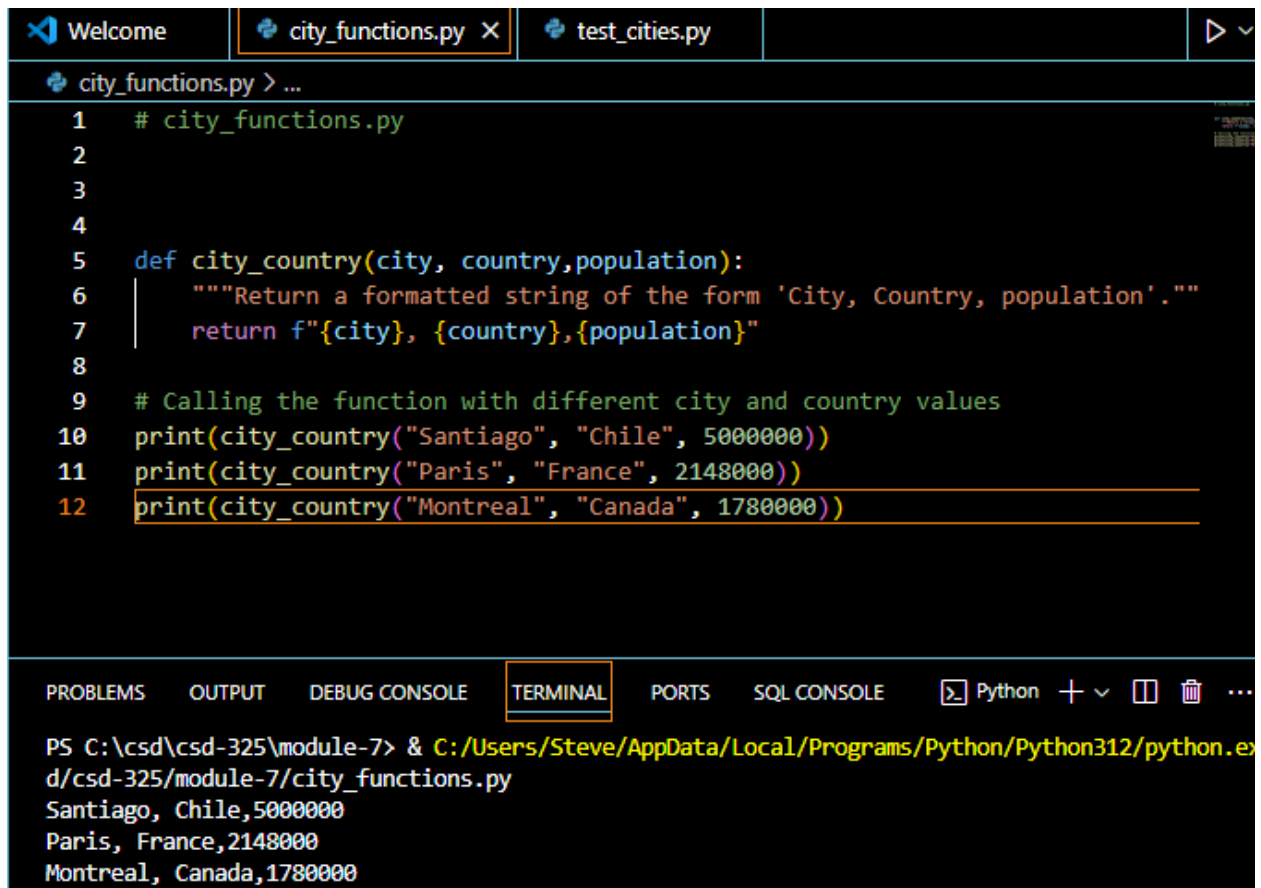
```
3
4 class TestCityCountry(unittest.TestCase):
5     def test_city_country(self):
6         """Test the city_country function with various inputs."""
7         self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chil
8         self.assertEqual(city_country("Paris", "France"), "Paris, France")
9         self.assertEqual(city_country("Montreal", "Canada"), "Montreal, Can
10
11 if __name__ == '__main__':
12     unittest.main()
13
```

The bottom pane shows the **TERMINAL** output, which displays the results of running the test script twice. The first run shows the output of the `city_functions.py` script, and the second run shows the output of the `test_cities.py` script, which includes a summary of the test results.

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe d/csd-325/module-7/city_functions.py
Santiago, Chile
Paris, France
Montreal, Canada
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe d/csd-325/module-7/test_cities.py
Santiago, Chile
Paris, France
Montreal, Canada
.
-----
Ran 1 test in 0.000s
OK
```

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.



The image shows a Python IDE with two tabs: 'city_functions.py' and 'test_cities.py'. The 'city_functions.py' tab is active, displaying a function definition and three print statements. The function 'city_country' takes three arguments: city, country, and population, and returns a formatted string. The print statements call this function with specific city and country names and their populations. The 'TERMINAL' tab at the bottom shows the command to run the script and the resulting output, which matches the expected formatted strings.

```
1 # city_functions.py
2
3
4
5 def city_country(city, country, population):
6     """Return a formatted string of the form 'City, Country, population'."""
7     return f"{city}, {country}, {population}"
8
9 # Calling the function with different city and country values
10 print(city_country("Santiago", "Chile", 5000000))
11 print(city_country("Paris", "France", 2148000))
12 print(city_country("Montreal", "Canada", 1780000))
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [X] ...

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe d/csd-325/module-7/city_functions.py
Santiago, Chile,5000000
Paris, France,2148000
Montreal, Canada,1780000
```

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.

```
Welcome | city_functions.py | test_cities.py X
test_cities.py > ...
1  import unittest
2  from city_functions import city_country
3
4  class TestCityCountry(unittest.TestCase):
5      def test_city_country(self):
6          """Test the city_country function with various inputs."""
7          self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chile")
8          self.assertEqual(city_country("Paris", "France"), "Paris, France")
9          self.assertEqual(city_country("Montreal", "Canada"), "Montreal, Canada")
10
11  if __name__ == '__main__':
12      unittest.main()
13

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SQL CONSOLE | Python +
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python38-325/module-7/test_cities.py
Santiago, Chile,5000000
Paris, France,2148000
Montreal, Canada,1780000
E
=====
ERROR: test_city_country (__main__.TestCityCountry.test_city_country)
Test the city_country function with various inputs.
-----
Traceback (most recent call last):
  File "c:\csd\csd-325\module-7\test_cities.py", line 7, in test_city_country
    self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chile")
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: city_country() missing 1 required positional argument: 'population'
-----
Ran 1 test in 0.000s

FAILED (errors=1)
PS C:\csd\csd-325\module-7>
```

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".

```
city_functions.py > ...
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4
5 # city_functions.py
6
7
8
9 def city_country(city, country, population=None):
10     """Return a formatted string of the form 'City, Country - population xx
11     if population:
12         return f"{city}, {country} - population {population}"
13     else:
14         return f"{city}, {country}"
15
16 # Function calls
17 print(city_country("Santiago", "Chile"))
18 print(city_country("Paris", "France"))
19 print(city_country("Montreal", "Canada"))
20
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - []

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python
d/csd-325/module-7/city_functions.py
Santiago, Chile
Paris, France
Montreal, Canada
```

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.

```
test_cities.py > ...
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4 # test_cities.py
5
6 import unittest
7 from city_functions import city_country
8
9 class TestCityCountry(unittest.TestCase):
10     def test_city_country(self):
11         """Test the city_country function with various inputs."""
12         self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chile")
13         self.assertEqual(city_country("Paris", "France"), "Paris, France")
14         self.assertEqual(city_country("Montreal", "Canada"), "Montreal, Canada")
15
16 if __name__ == '__main__':
17     unittest.main()
18
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [X]

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python
d\csd-325/module-7/test_cities.py
Santiago, Chile
Paris, France
Montreal, Canada
-
-----
Ran 1 test in 0.000s

OK
```

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.

```
city_functions.py > ...
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4
5 # city_functions.py
6
7
8
9 def city_country(city, country, population, language):
10     """Return a formatted string of city, country, population, and language
11     return f"{city}, {country} - population {population:,}, {language}"
12
13 # Example usage
14 if __name__ == "__main__":
15     print(city_country("Santiago", "Chile", 5000000, "Spanish"))
16     print(city_country("Paris", "France", 2148000, "French"))
17     print(city_country("Montreal", "Canada", 1780000, "English/French"))
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [] ...

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe
d/csd-325/module-7/city_functions.py
Santiago, Chile - population 5,000,000, Spanish
Paris, France - population 2,148,000, French
Montreal, Canada - population 1,780,000, English/French
```

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.

```
test_cities.py > TestCityCountry > test_city_country
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4 # test_cities.py
5
6 import unittest
7 from city_functions import city_country
8
9 class TestCityCountry(unittest.TestCase):
10     def test_city_country(self):
11         """Test the city_country function with various inputs."""
12         self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chile")
13         self.assertEqual(city_country("Paris", "France"), "Paris, France")
14         self.assertEqual(city_country("Montreal", "Canada"), "Montreal, Canada")
15
16 if __name__ == '__main__':
17     unittest.main()
18
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [] ... ^

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe c:\csd-325\module-7\test_cities.py
E
=====
ERROR: test_city_country (__main__.TestCityCountry.test_city_country)
Test the city_country function with various inputs.
-----
Traceback (most recent call last):
  File "c:\csd\csd-325\module-7\test_cities.py", line 12, in test_city_country
    self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chile")
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: city_country() missing 2 required positional arguments: 'population' and 'language'

-----
Ran 1 test in 0.000s

FAILED (errors=1)
```

city_functions.py

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

```
city_functions.py > city_country

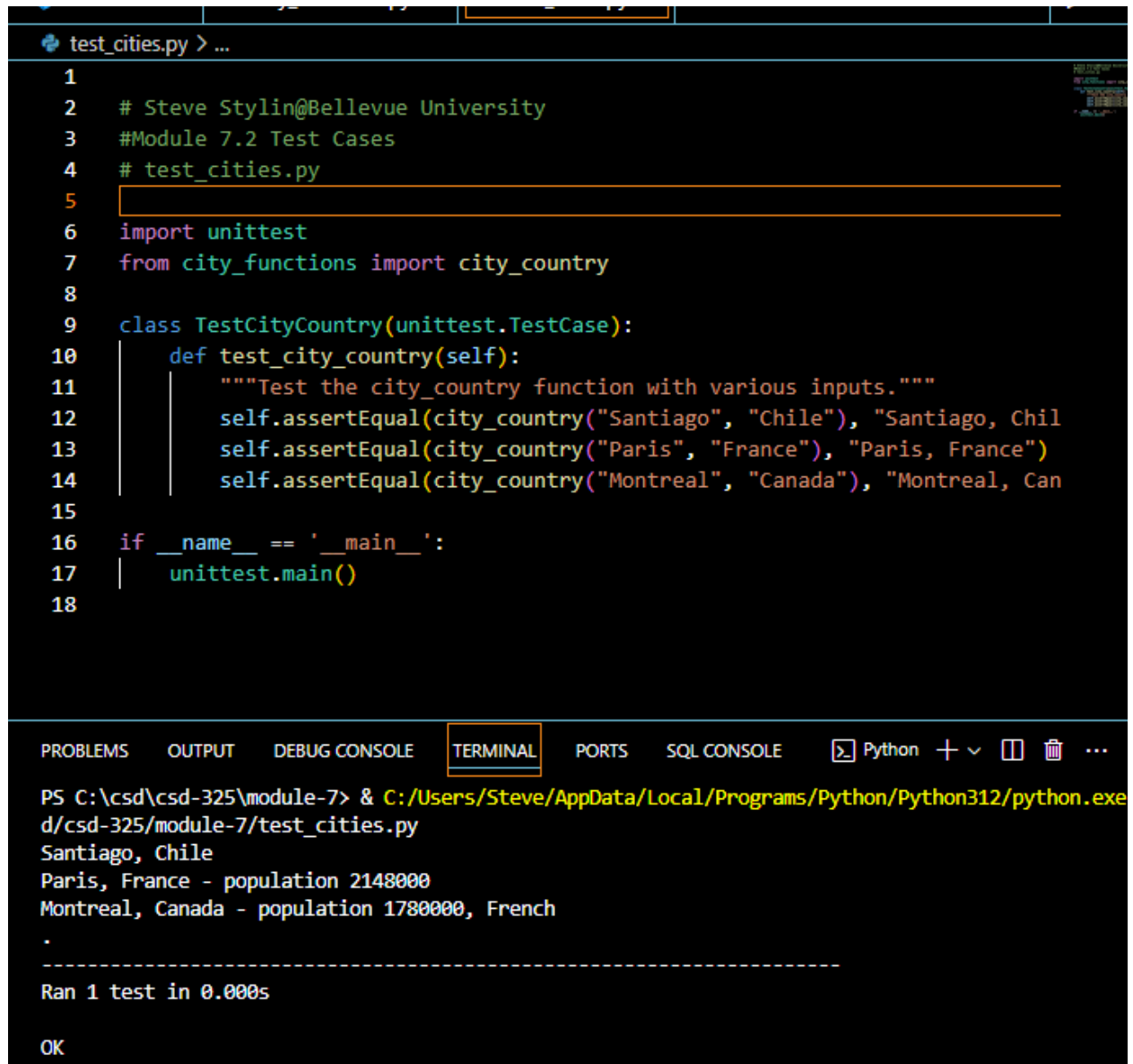
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4
5 # city_functions.py
6
7
8 def city_country(city, country, population=None, language=None):
9     """Return a formatted string of the form 'City, Country - population xx
10     if population and language:
11         return f"{city}, {country} - population {population}, {language}"
12     elif population:
13         return f"{city}, {country} - population {population}"
14     elif language:
15         return f"{city}, {country}, {language}"
16     else:
17         return f"{city}, {country}"
18
19 # Function calls
20 print(city_country("Santiago", "Chile"))
21 print(city_country("Paris", "France", 2148000))
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [] ...

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe
d/csd-325/module-7/city_functions.py
Santiago, Chile
Paris, France - population 2148000
Montreal, Canada - population 1780000, French
```

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.



The screenshot shows an IDE with a Python file named `test_cities.py` and a terminal window. The code in the file defines a test class `TestCityCountry` that tests the `city_country` function. The terminal shows the command to run the test, the output of the function for three different inputs, and the successful completion of the test.

```
test_cities.py > ...
1
2 # Steve Stylin@Bellevue University
3 #Module 7.2 Test Cases
4 # test_cities.py
5
6 import unittest
7 from city_functions import city_country
8
9 class TestCityCountry(unittest.TestCase):
10     def test_city_country(self):
11         """Test the city_country function with various inputs."""
12         self.assertEqual(city_country("Santiago", "Chile"), "Santiago, Chil
13         self.assertEqual(city_country("Paris", "France"), "Paris, France")
14         self.assertEqual(city_country("Montreal", "Canada"), "Montreal, Can
15
16 if __name__ == '__main__':
17     unittest.main()
18
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SQL CONSOLE Python + - [] [X] ...

```
PS C:\csd\csd-325\module-7> & C:/Users/Steve/AppData/Local/Programs/Python/Python312/python.exe
d\csd-325/module-7/test_cities.py
Santiago, Chile
Paris, France - population 2148000
Montreal, Canada - population 1780000, French
.
-----
Ran 1 test in 0.000s

OK
```