

The image shows a Python IDE window with a toolbar at the top containing icons for file operations, running, and debugging. The main editor displays a Python script named `city_functions.py`. The script defines a function `format_city_country` that takes `city` and `country` as arguments, creates a formatted string, and returns it. Below the function definition, three print statements call the function with the city-country pairs: Athens, Greece; Beijing, China; and Berlin, Germany. A `Shell` window at the bottom shows the output of these print statements, each on a new line. The prompt `>>>` is visible at the bottom of the shell window.

```
1 # Function that accepts city and country as parameters and returns a formatted string
2 def format_city_country(city, country):
3     """Generate a formatted city and country couple."""
4     city_location = f"{city}, {country}"
5     return city_location.title()
6
7 # Call the function with different city, country values
8 print(format_city_country("Athens", "Greece"))
9 print(format_city_country("Beijing", "China"))
10 print(format_city_country("Berlin", "Germany"))
11
```

Shell

Athens, Greece  
Beijing, China  
Berlin, Germany

>>>

city\_functions.py × test\_cities.py \* ×

```
1 import unittest
2 from city_functions import format_city_country
3
4 class CitiesTestCase(unittest.TestCase) :
5     """Tests for 'format_city_country.py'."""
6
7     def test_city_country(self):
8         city_location = format_city_country('athens', 'greece')
9         self.assertEqual(city_location, 'Athens, Greece')
10
11 if __name__ == '__main__':
12     unittest.main()
13
14
```

Shell ×

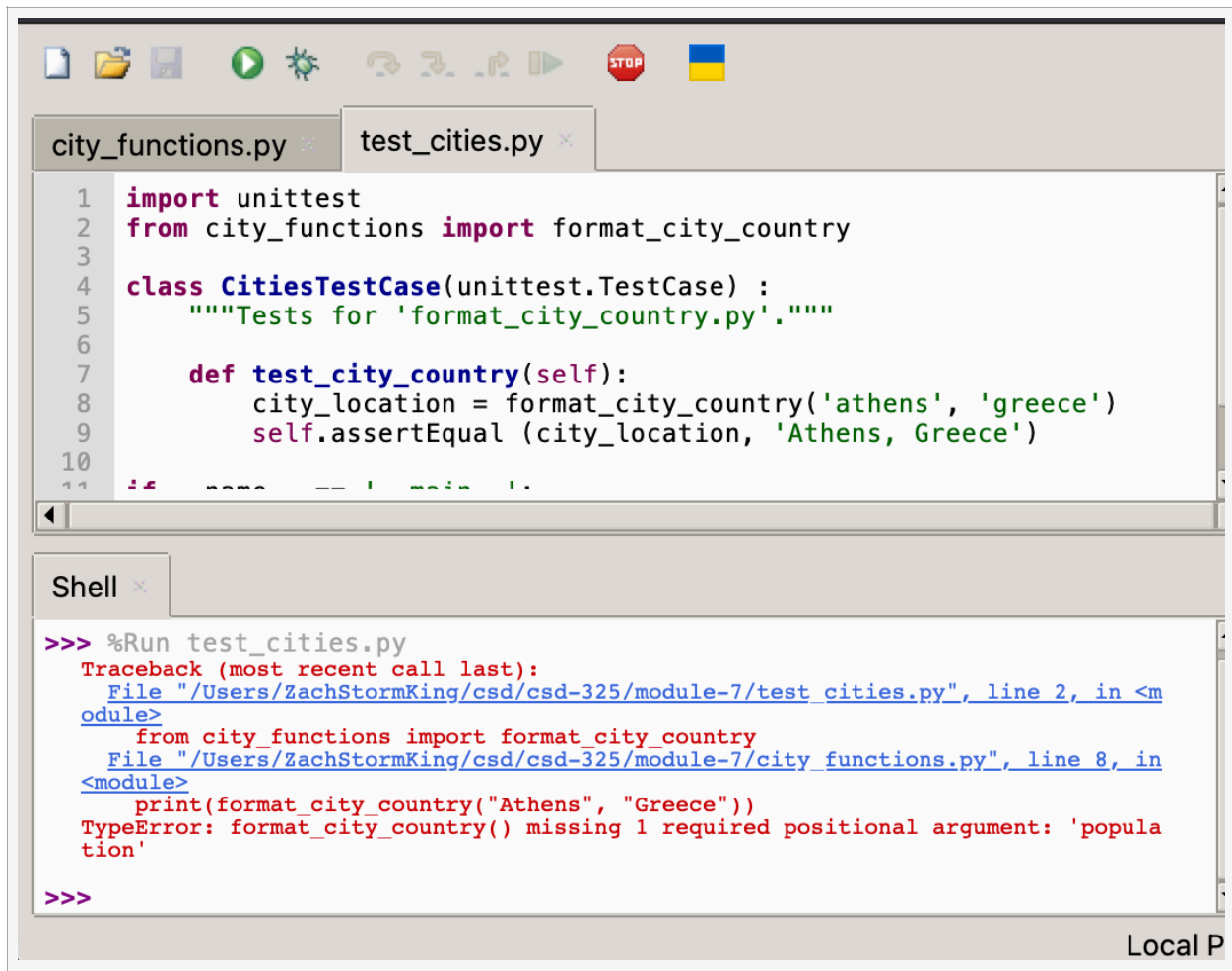
```
>>> %Run test_cities.py

Athens, Greece
Beijing, China
Berlin, Germany
.
-----
Ran 1 test in 0.000s

OK

Process ended with exit code 0.
```

Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.fra  
>>>



The screenshot shows a Python IDE with two tabs: `city_functions.py` and `test_cities.py`. The `test_cities.py` tab is active, displaying the following code:

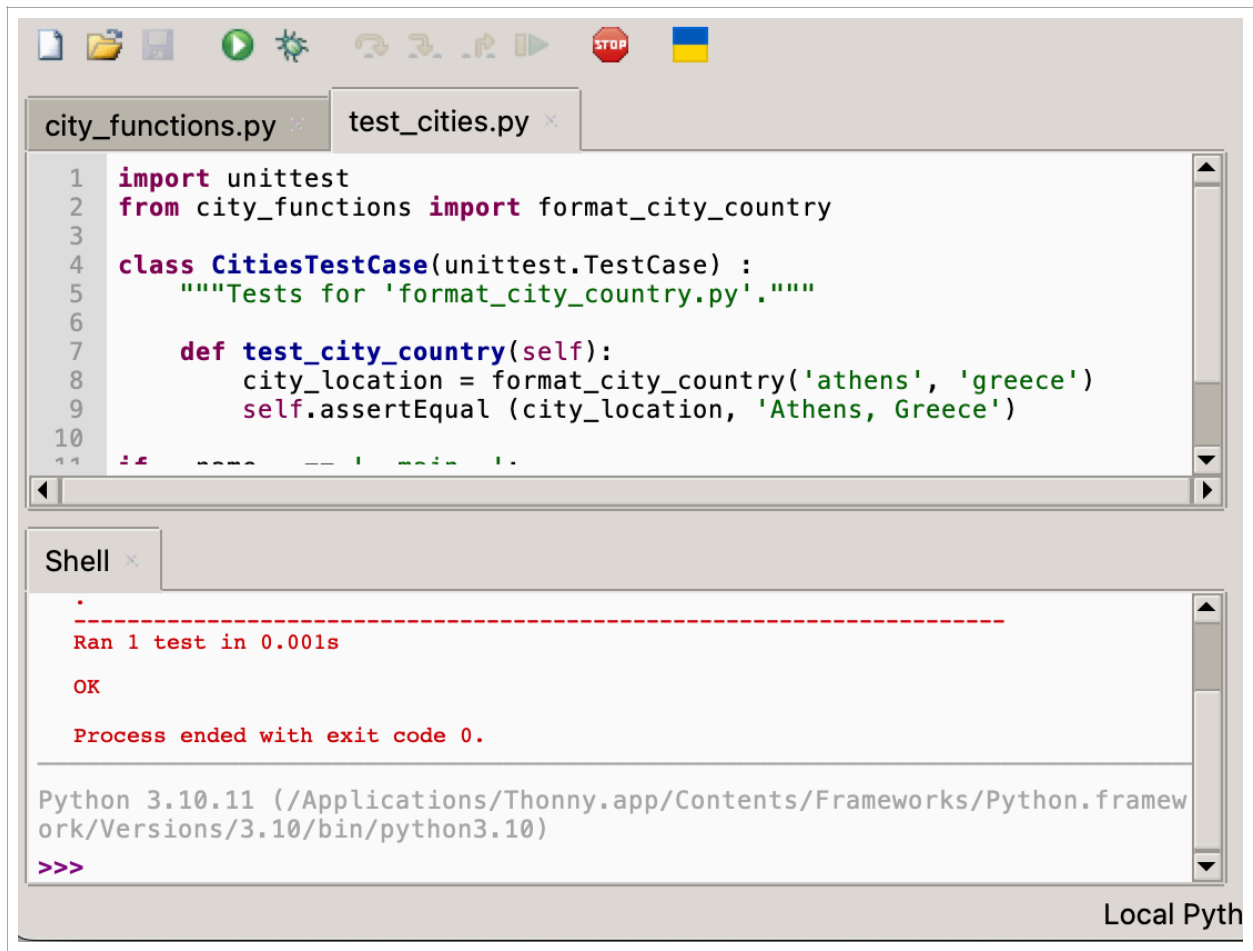
```
1 import unittest
2 from city_functions import format_city_country
3
4 class CitiesTestCase(unittest.TestCase):
5     """Tests for 'format_city_country.py'."""
6
7     def test_city_country(self):
8         city_location = format_city_country('athens', 'greece')
9         self.assertEqual(city_location, 'Athens, Greece')
10
11 if __name__ == '__main__':
```

Below the code editor is a `Shell` window showing the execution of `test_cities.py`. The output is as follows:

```
>>> %Run test_cities.py
Traceback (most recent call last):
  File "/Users/ZachStormKing/csd/csd-325/module-7/test_cities.py", line 2, in <module>
    from city_functions import format_city_country
  File "/Users/ZachStormKing/csd/csd-325/module-7/city_functions.py", line 8, in <module>
    print(format_city_country("Athens", "Greece"))
TypeError: format_city_country() missing 1 required positional argument: 'population'

>>>
```

The text "Local P" is visible in the bottom right corner of the IDE window.



The image shows a screenshot of the Thonny Python IDE. At the top, there is a toolbar with icons for file operations, running, and debugging. Below the toolbar, two tabs are open: 'city\_functions.py' and 'test\_cities.py'. The 'test\_cities.py' tab is active, displaying the following Python code:

```
1 import unittest
2 from city_functions import format_city_country
3
4 class CitiesTestCase(unittest.TestCase):
5     """Tests for 'format_city_country.py'."""
6
7     def test_city_country(self):
8         city_location = format_city_country('athens', 'greece')
9         self.assertEqual(city_location, 'Athens, Greece')
10
11 if __name__ == '__main__':
```

Below the code editor is a 'Shell' window. It shows the output of running the test:

```
.
-----
Ran 1 test in 0.001s

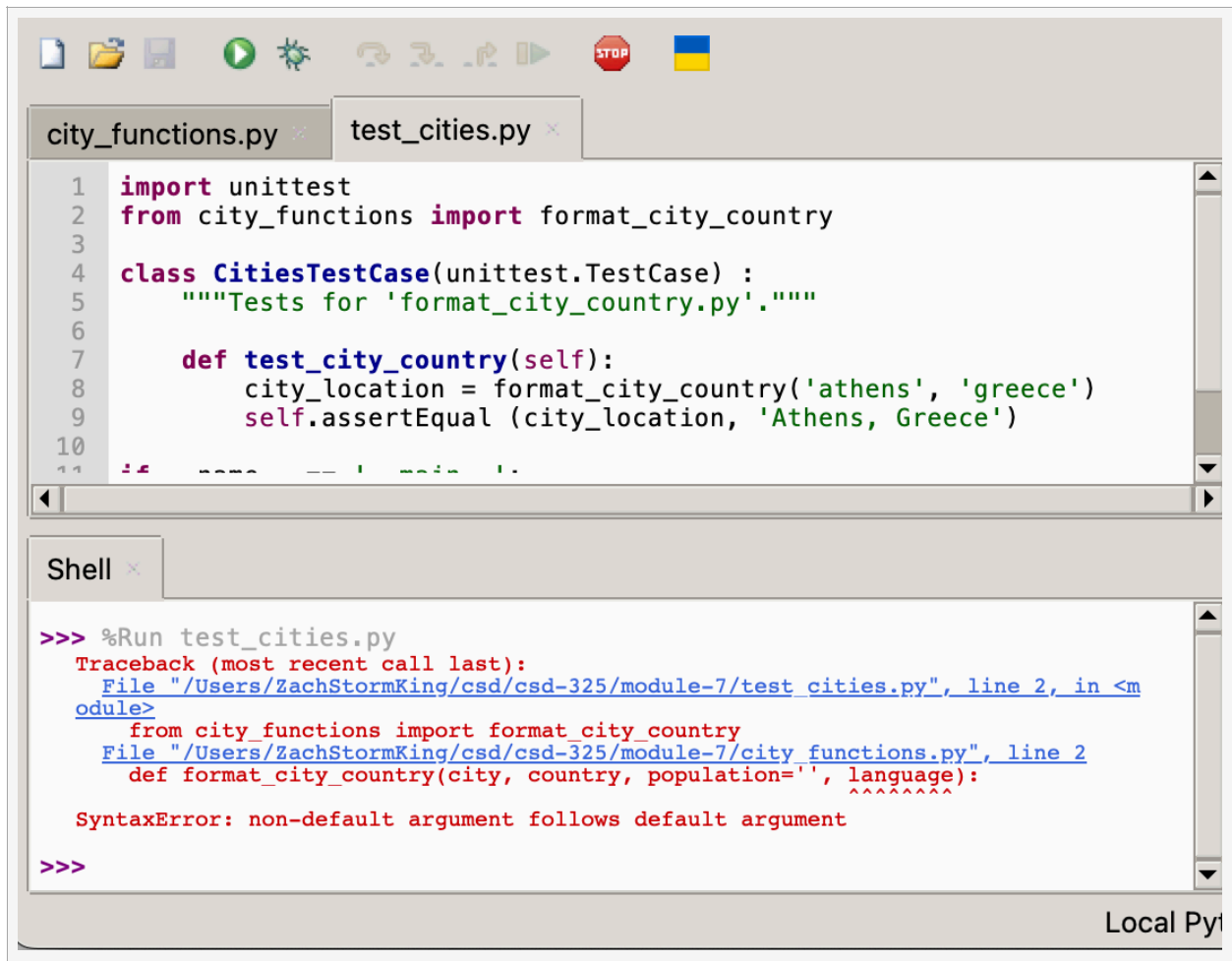
OK

Process ended with exit code 0.
```

At the bottom of the Shell window, the Python version and path are displayed:

```
Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin/python3.10)
>>>
```

The bottom right corner of the IDE window is labeled 'Local Pyth'.



The screenshot shows a Python IDE with two tabs: `city_functions.py` and `test_cities.py`. The `test_cities.py` tab is active, displaying the following code:

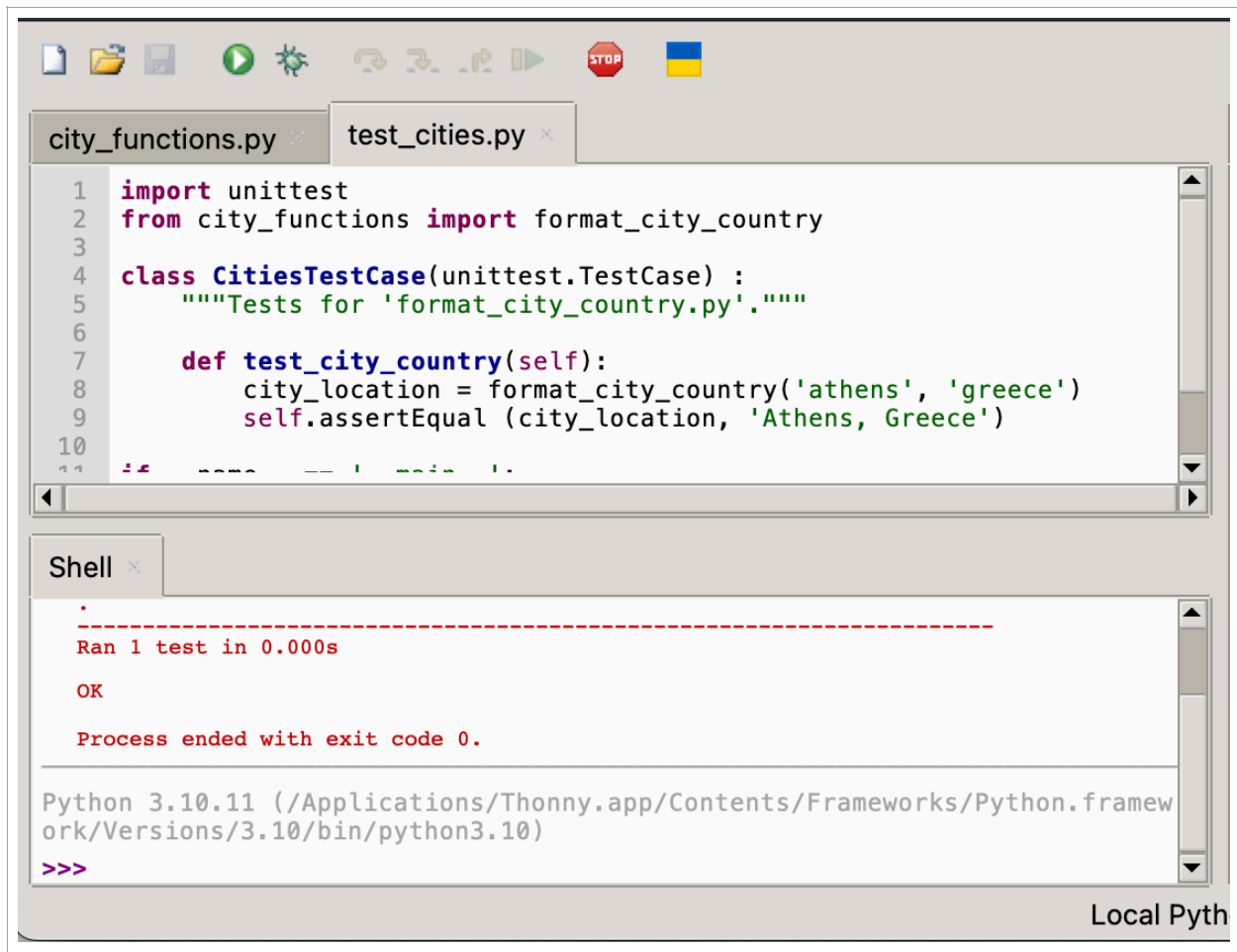
```
1 import unittest
2 from city_functions import format_city_country
3
4 class CitiesTestCase(unittest.TestCase):
5     """Tests for 'format_city_country.py'."""
6
7     def test_city_country(self):
8         city_location = format_city_country('athens', 'greece')
9         self.assertEqual(city_location, 'Athens, Greece')
10
11 if __name__ == '__main__':
```

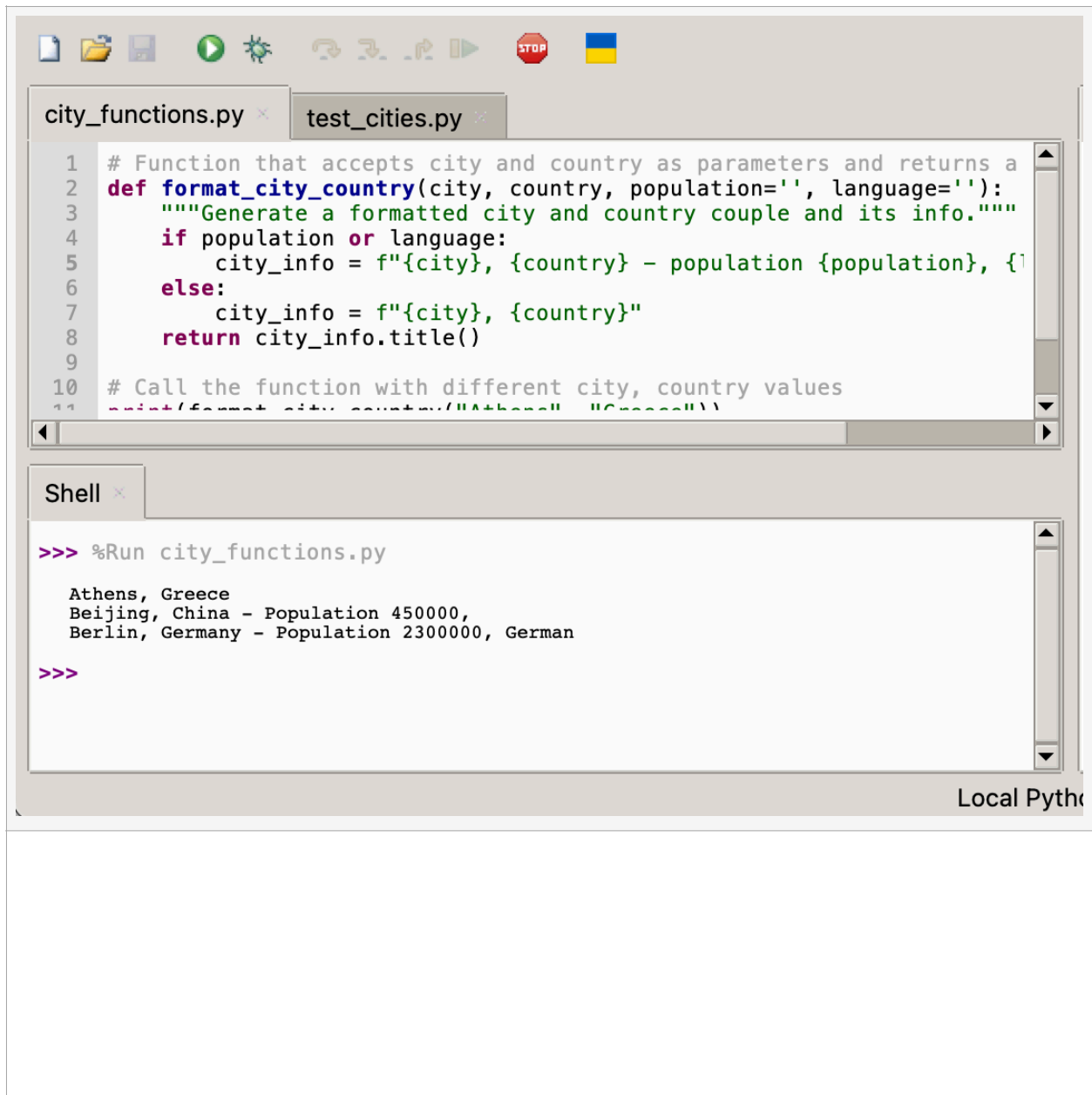
Below the code editor is a `Shell` window showing the output of running `test_cities.py`:

```
>>> %Run test_cities.py
Traceback (most recent call last):
  File "/Users/ZachStormKing/csd/csd-325/module-7/test_cities.py", line 2, in <module>
    from city_functions import format_city_country
  File "/Users/ZachStormKing/csd/csd-325/module-7/city_functions.py", line 2
    def format_city_country(city, country, population='', language):
                                ^^^^^^^^^
SyntaxError: non-default argument follows default argument

>>>
```

The error message indicates a `SyntaxError` in `city_functions.py` at line 2, where a non-default argument (`language`) follows a default argument (`population=''`).





The screenshot shows a Jupyter Notebook interface with two tabs: `city_functions.py` and `test_cities.py`. The `city_functions.py` tab is active, displaying a Python script. The script defines a function `format_city_country` that takes `city`, `country`, `population`, and `language` as parameters. It uses f-strings to format the output. The function is then called with three different city/country pairs. The output of the script is displayed in the Shell window below the code editor.

```
1 # Function that accepts city and country as parameters and returns a
2 def format_city_country(city, country, population='', language=''):
3     """Generate a formatted city and country couple and its info."""
4     if population or language:
5         city_info = f"{city}, {country} - population {population}, {language}"
6     else:
7         city_info = f"{city}, {country}"
8     return city_info.title()
9
10 # Call the function with different city, country values
11 print(format_city_country("Athens", "Greece"))
```

Shell

```
>>> %Run city_functions.py

Athens, Greece
Beijing, China - Population 450000,
Berlin, Germany - Population 2300000, German

>>>
```

Local Python