Lab 1

Contents

1 Location

Review the code in `location.py`. Note that there is a class definition for a `Location` class and an associated `__init__` method. In addition, there is code to create `Location` objects and print information associated with those objects.

Without modifying the code, run `location.py`. Note the information that is printed out for each `Location` object. You should see something like:

```
Location 1: <__main__.Location object at 0x000001F6A2E0C7B8>
```

Since we haven't provided any specific method to provide a representation for the class, Python uses a default method. What do you notice about the information for Locations 1 and 4?

Also note the result of the equality comparisons between the locations, in particular, `loc1 == loc3` and `loc1 == loc4`. Make sure you understand why the results are what they are.

Now modify the `location.py` code, adding in the methods `__eq__` and `__repr__`. See the `location_tests.py` to figure out how the `repr` should behave.

Run the `location.py` code with the modifications made above.

Now review the information printed out for each location. The `__repr__` method of `Location` is now being used when printing the object.

Examine the results of the equal comparisons. How are they different from before the `__eq__` method is added?

2 Recursion Samples

Take a look at each function in `sample.py`. Discuss with others in lab and make sure you understand all of the code in each function. Ask your instructor if you have questions about anything!

Take a look at each test in `sample_tests.py`. Discuss with others in lab and make sure you understand all of the test cases. Ask your instructor if you have questions about anything!

3 Recursion and Iteration

3.1 Implementation

In the `lab1.py` file, complete the functions specified. Pay careful attention to the details of the specification for each function.

You ***may not***:

- Use the builtin Python function `reversed`.
- Use the builtin Python function `max`.
- Reverse a list using only slicing (e.g., `my_list[::-1]`).

3.2 Testing

Testing is one of the most important task that one can perform while programming. Proper testing provides a degree of confidence in your solution. Writing high quality test cases can greatly simplify the tasks of both finding and fixing bugs and, as such, **will save you time during development**. However, no degree of testing will every *guarantee* that your program is correct.

For this part of the lab, you will practice writing some simple test cases to gain experience with the `unittest` framework.

Using the editor/IDE of your choice, open the `lab1_tests.py` file. You must add additional test cases to verify that your functions are correct.

To ensure your tests are thorough, I have incorrect versions of the functions in `lab1.py` that I will run your tests against. If your tests aren't about to expose the flaw in my broken code, then your tests are not thorough enough.

4 GitHub Submission

Push your finished code back to GitHub. Refer to Lab 0, as needed, to remember how to push your local code.