Lab 1: Python Classes, Recursion, and Testing!

Part 1

1. 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.

```
class Location:
    def __init__(self, name, lat, long):
        self.name = name      # string for name of location
        self.lat = lat        # latitude in degrees (-90 to 90)
        self.long = long      # longitude in degrees (-180 to 180)
```

2. Without modifying the code, run location.py in whatever environment you wish (again, reference the Getting Started document if you need help in doing this)
3. Note the information that is printed out for each Location object – you should see something like this: `Location 1: <__main__.Location object at 0x000001F6A2E0C7B8>`
4. 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 `loc1` and `loc4` ?
5. Also note the result of the equal comparisons between the locations, in particular `loc1==loc3` and `loc1==loc4`. Make sure you understand why the results are what they are.
6. Now modify the location.py code, adding in the methods (`__eq__()` and `__repr__()`). See the location_tests.py to figure out what the repr method should look like. You can also use the boilerplate generator as shown in class – you can find the link in PolyLearn.
7. Run the `location.py` code with the modifications made above.
8. Now review the information printed out for each location. The `__repr__` method of `Location` is now being used when printing the object.
9. Examine the results of the equal comparisons. How are they different from before the `__eq__` method was added?
10. Now that __eq__ and __repr__ functions have been added to the Location class, execute the location_tests.py file and observe the results. Add additional tests that will test the __init__, __eq__ and __repr__ functions.
11. Submit your updated location.py and location_tests.py files to PolyLearn for this part of the lab, and make sure you understand what the above questions are driving at, and if you have any uncertainties, ask! We'll also go over the answers later during lecture or lab time.

Part 2

1. In the lab1.py file, complete the **iterative function** to find the maximum integer in a list of integers.

```
def max_list_iter(int_list):  # must use iteration not recursion
    """finds the max of a list of numbers and returns the value (not the index)
    If int_list is empty, returns None. If list is None, raises ValueError"""
```

2. In the lab1.py file, complete the **recursive function** to reverse a list of integers:

```
def reverse_rec(int_list):   # must use recursion
    """recursively reverses a list of numbers and returns the reversed list
    If list is None, raises ValueError"""
```

3. In the lab1.py file, complete the **recursive function** to search a list of integers using binary search along with test cases. If the **target** of the search is in the list, the function returns its index.

```
def bin_search(target, low, high, int_list):  # must use recursion
    """searches for target in int_list[low..high] and returns index if found
    If target is not found returns None. If list is None, raises ValueError """
```

Test Cases

Many people tend to focus on writing code as the singular activity of a programmer, but testing is one of the most important tasks that one can perform while programming. Proper testing provides a degree of confidence in your solution. Systematic testing helps you to discover and then debug your code. 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, testing does not 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. I recommend watching the first 20 minutes or so of the following video if you need more guidance on testing in Python. https://www.youtube.com/watch?v=6tNS--WetLI

Using your editor/IDE of choice, open the *lab1_test_cases.py* file. This file defines, using code that we will treat as a boilerplate for now, a testing class with a single testing function.
In the *test_expressions* function you will see some test cases already provided. You must add additional test cases to verify that your functions (max_list_iter, reverse_rec, bin_search) are correct.

Submission/Grading

Ensure that the following file have been submitted to PolyLearn by the due date:

- **location.py**
  - Updated per the instructions in Part 1
- **location_tests.py**
  - Comprehensive test cases as described in Part 1
- **lab1.py**
  - Correct and well documented iterative **max_list_iter**, recursive **reverse_rec**, and recursive **bin_search** functions based on the template provided
- **lab1_test_cases.py**
  - A complete set of test cases for the functions above. Your test cases should test boundary conditions and other possible errors based on the structure of your program. For each test provide a comment (or docstring) that explains what it is testing. Your tests cases will be tested with known incorrect (buggy) versions of the functions in lab1.py and will also be tested for 100% code coverage.