Please use the time you are not using to solve the problems in the lab to work, individually, on your Project 1. You can get help from the TAs.

Problem 1.

The solutions for the calculator lab are linked from the course webpage. Download those solutions. Create the file `test_calc.py`, and add in some testing code for the calculator lab in the `if __name__ == '__main__'` block of `test_calc.py` (remember to `import lab02`). Set up the testing such that in `test_calc.py`, you print `"Test 1 failed"` or `"Test 1 passed"`, `"Test 2 failed"` or `"Test 2 passed"`, etc., depending on whether the tests fail or pass. In order to do that, you should definte a new function in `lab02.py`, called `get_current_value()`. You can then have code along the lines of

```
initialize()
add(42)
if get_current_value() == 42:
  print("Test 1 passed")
else:
  print("Test 1 failed")
```

Now, run `test_calc.py` and make sure that the tests pass. Also make sure to be able to point out to the TA which 'main' block is run, and which isn't.

To run a program that consists of multiple files, place all the files in the same folder, and then use `Run->Run file as script` in Pyzo when the main file that you want to run is active.

Problem 2.

Here is a function that computes the sum of a list of numbers.

```
def sum_nums(L):
  s = 0
  for num in L:
    s += num

  return s
```

Write a function with the signature `def count_evens(L)` that returns the number of even integers in the list `L`. assume `L` only contains integers.

Problem 3.

You can use str() to convert objects to strings:

```
>> str(42)
42
```

In particular, you can obtain the string representation of a list list0 by using str()

```
>> list0 = [1, 2, 3]
>> str(list0)
[1, 2, 3]
```

Without using `str()` with arguments that are lists (using it with arguments that are not lists is fine), write a function `list_to_str(lis)` which returns the string representation of the list `lis`. You may assume `lis` only contains integers.
Reminder:

```
>> "hello" + "python"
"hellopython"
```

Problem 4.

You can compare lists using the == operator:

```
>> l1 = [1, 2, 3]
>> l2 = [4, 5, 6]
>> l3 = [1, 2, 3]
>> l1 == l2
False
>> l1 == l3
True
```

Without using the `==` operator to compare lists (you can still compare individual elements of the lists), write a function `lists_are_the_same(list1, list2)` which returns `True` iff `list1` and `list2` contain the same elements in the same order. You'll need to use a loop (either while or for)