CSE101: Introduction to Programming
Lab 4
Section 1: Defining your own module

Create a file with ***lab4.py*** and implement the following methods in it:

1. **second_max(a,b,c,d):** A function that takes a,b,c,d integers. This function should return 2nd maximum from four integers.
2. **admission(p,c,m,avg):** A function that takes marks in physics (p), chemistry (c), mathematics (m) and average marks (avg) and returns eligibility of admission on the basis of all individual and average marks which should be greater or equal to 80. The function returns 1 if eligibility is satisfied otherwise returns -1.
3. **triangleType(s1,s2,s3):** A function to check whether a triangle is Equilateral, Isosceles, or Scalene. It would return 3, 2 or 1 if triangle is Equilateral, Isosceles or Scalene, respectively.
4. **validSides(a,b,c):** A function to check whether a triangle can be formed or not with the given sides. It would return True if triangle can be formed and False otherwise.
5. **characterCheck(a):** A function to check whether a character is an alphabet, digit or special character. It would return 1, 2, 3 if character is special character, digit, alphabet respectively.

Once you have implemented the methods in Section 1, the next stage is to 'test' if your
implementation behaves according to your expectations or not. If there are errors in your implementation, it can be caught through careful testing of the code.

Unit testing, specifically tests a single "unit" of code in isolation. A unit could be an entire module, a single class or function, or almost anything in between. What's important,
however, is that the code is isolated from other code we're not testing (which itself could
have errors and would thus confuse test results). The unittest module in Python provides a rich set of tools for constructing and running tests.

You may take a look at the documentation on unittest:
https://docs.python.org/3/library/unittest.html

We have shared a test code, called ***testing.py***, along with this document. You have to save lab4.py and testing.py in same directory. Open the testing.py in an editor. Notice that testing.py has a test code to test the five functions defined in first module. The

testing.py module imports lab4 module. There are various useful functions defined in the unittest module. Let's consider an important one that you will be required to use in this lab.

assertEqual(first, second, msg=None)

Tests that first and second are equal. If the values do not compare equal, the test will fail. Normally, we give first as the result returned from the unit we are testing. In our
case, the first argument would be a call to the function we are testing. The second argument is the output expected by the programmer. If the two differ, it implies, the function being tested fails our test.

In testing.py module, we have imported the lab3 module, as we would need its functions
for testing. Ignore unfamiliar keywords(like class, self) in the module for now. We have already defined a test method, called second_max, to test the second_max function.

There are three test cases defined in it
1. self.assertEqual(second_max(1,2,3,4),3)
2. self.assertEqual(second_max(-1,-2,-3,-4),-2)
3. self.assertEqual(second_max(-1,20,-30,100),20)

For the first case, we are passing a,b,c,d as 1, 2, 3, 4 respectively. We expect function
to return 3. If the function returns 3, the test passes, else it fails.
Similarly, for second it should return -2 and for third it should return 20.

Now here are your tasks to perform on the testing.py. You have to run testing.py and check if your lab4 runs perfectly.