**Explain Documentation**

**Question A**

Your goal for this question is to write a program that accepts two lines (x1,x2) and (x3,x4) on the x-axis and returns whether they overlap. As an example, (1,5) and (2,6) overlaps but not (1,5) and (6,8).

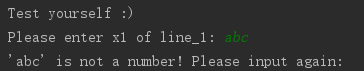
**Solution:**

Step 1: set two lines

We have to ensure all inputs are numbers before invoking the ‘is\_overlaps’ function.

If there is an illegal input, like letter or operator (e.g. ‘a’, ‘#’), it’s not a point value. So, I define a ‘is\_number’ function to verify the inputs. If the input is illegal, it requires input again until it’s valid.

We assume that the ordinate points should be int or float type, for example, the line should be (4, 5) or (1.2, 7.8). So, (4, 5) would be treated as (4.0, 5.0)

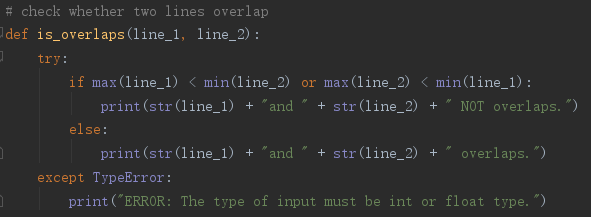


Step 2:

Obviously, it is complicated to decide the conditions of overlaps.

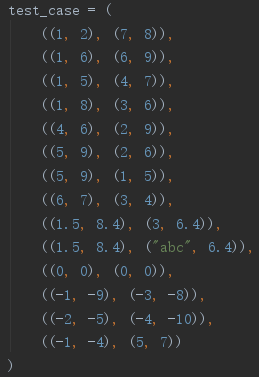
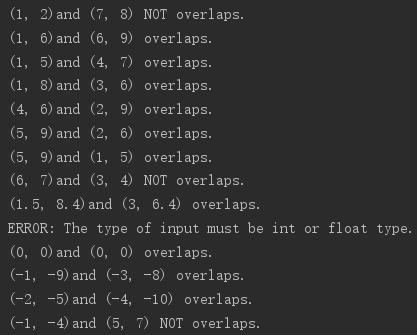
But it would be easier to consider the conditions of NOT overlaps.

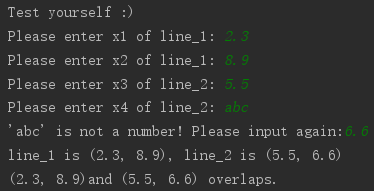
We assume that the large value is ending point and the small value is starting point. So, as long as the ending point of one line is lower than the starting point of another line, they are not overlapped.



By the way, for stability, we have to catch the exception if someone invoke the method illegally.

**Results:**



**Question B**

The goal of this question is to write a software library that accepts 2 version string as input and returns whether one is greater than, equal, or less than the other. As an example: “1.2” is greater than “1.1”. Please provide all test cases you could think of.

**Solution:**

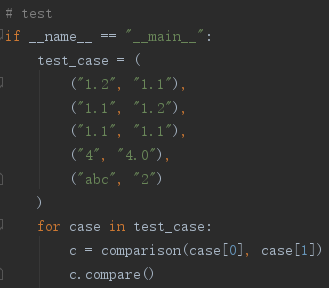
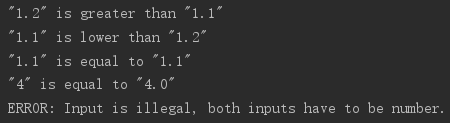
If we need to compare two input which is greater, equal or less, they must be number. Otherwise, the letters or operators cannot be compared. Although they can be compared by ACSII code, it’s not greater or less in nature.

Thus, the first step is to verify the inputs are number. Otherwise, illegal input will invoke an exception.

The second step is to transform the string type to float type. Although the string type of number can be compared correctly in Python, because Python is a weakly typed language. However, I hold my perception that they should be compared by float type, for consideration of stability, like Java.

The final step is to return whether one is greater than, equal, or less than the other.

**Result:**

**Question C**

At Ormuco, we want to optimize every bits of software we write. Your goal is to write a new library that can be integrated to the Ormuco stack. Dealing with network issues everyday, latency is our biggest problem. Thus, your challenge is to write a new Geo Distributed LRU (Least Recently Used) cache with time expiration. This library will be used extensively by many of our services so it needs to meet the following criteria:

1 - Simplicity. Integration needs to be dead simple.

2 - Resilient to network failures or crashes.

3 - Near real time replication of data across Geolocation. Writes need to be in real time.

4 - Data consistency across regions

5 - Locality of reference, data should almost always be available from the closest region

6 - Flexible Schema

7 - Cache can expire

**Design:**

The Geo Distributed LRU cache is implemented by a dictionary and a list.

In the dictionary, the key is an index, while the value is an object which can store data and its latest time. We can add anything useful into the object in the future development.

In the list, it stores the keys of dictionary, and it has a constant capacity.

When a new key comes, if the size doesn’t hit to the capacity, list appends the new key, in the meantime dictionary add a new key-value pair. If the size is equal to the capacity, the key on list[0] will be removed, and it and its value will also be removed in dictionary.

When an old key comes, it will be found out from list and moved to the rear, while its data and timestamp will change in dictionary.

**Missing:**

2 - Resilient to network failures or crashes.

In my opinion, it should have a rollback mechanism, like Transaction in SQL.

**Result:**

