Until now, I didn't find any bugs from my testing library. In order to make sure that there is no bugs, the first strategy is using the logging to record the former status and the latter status. Also, I printed out the results so that I can check it manually and it showed that nothing went wrong. To make it more particular, I set m = 40 which means there will be about 400 executes and TSTL gave positive result. Furthermore, I also consider the special case which do the replace and pop when the heap is empty. I tried in the python environment and it shows the "index error" and in the TSTL, I got the same result as it caught the expectation as "index error" then stopped automatically. To be more confident, I checked the out.txt (the file store the final result) and figured out that the error occurred when the next step is pop or replace and the heap is empty which confirm my assumptions.

My progress data is as following:

01/23/ - 01/31: Reading the TSTL paper and get a general idea about how to coding for TSTL.

02/01/ - 02/07: Ending by finishing initializing the heap and building the pools the test needs.

02/08/ - 02/15: Having an initial version of testing code which contains the pop and push functions.

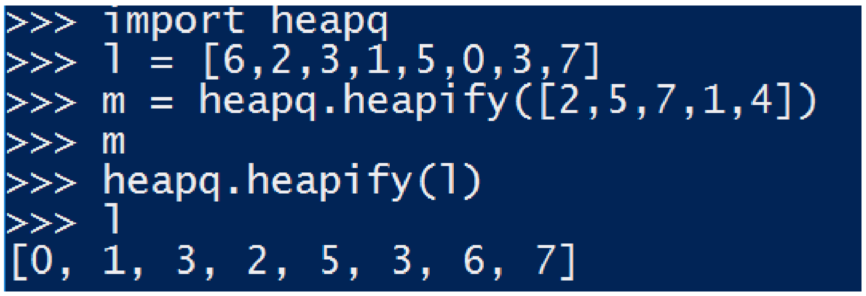
02/15/ - 02/22: Adding more functions such as poppush, pushpop, replace, minheap and so on.

02/23/ - Now: Trying to figure out the heap which mixing with int and char.

By the end of term, ideally I can finish testing almost all the functions in heapq library. For each function the input will be the list mix with integer and char. If the time admits I will probably testing the heappushpop compare to heappush followed by a separate call to heappop. I am pretty sure that the heappushpop will be more effiency. After finishing this project, I hope I can have a better understanding on how to test systems and expertly using TSTL.

The quality of the software under testing is kind of good. Here I mean that it indeed let us get rid of a lot of trouble by endless input a lot of single test. TSTL helps doing this time consuming things. However, not all the case can be test. I heard from one of my classmate that the library he testing is used to transform the messy code to the ordinary one. However, what is the messy code TSTL can figure out? How can he use the python library? Also, the list of string, the iterator and a lot of types TSTL cannot handle very well. On the other hand, TSTL, in some aspects, is an easy language that a beginner can learn fairly quickly. In my opinion, TSTL needs more works but it is promising in the future.

For the code coverage, TSTL didn't do well for my testing library. Because no matter how I modify my code, the percentage of covered is 0 all the time and there is the warning said that no data was collected. The reason why this happened is because there is a function called heapq.heapify(x) which used to transform a list to a heap. However, users cannot input a list as input but must input the name of the list. That's why I cannot implement this in TSTL. All the time when I call the function from library, the input must be the heap type, however, TSTL shows it is just a list so that there is no coverage in my code. Also, my special library shows no need to initialize the heap. We need create the empty list first and use heapify function to transform which is impossible to achieve using TSTL. The figure below can illustrate the situation very well.



```
>>> import heapq
>>> l = [6,2,3,1,5,0,3,7]
>>> m = heapq.heapify([2,5,7,1,4])
>>> m
>>> heapq.heapify(l)
>>> l
[0, 1, 3, 2, 5, 3, 6, 7]
```