Introduction

**Problem A**

Your job is do design and implement a data structure called [Least Recently Used (LRU)](https://en.wikipedia.org/wiki/Cache_replacement_policies#LRU) cache. This data structure supports the following operations: get(key), put(key, value), size(), max\_capacity()

All operations MUST run in O(1) time complexity. You are free to uses Python’s set and/or dictionary data structures. If you need to use a doubly linked list (hint), you need to code it yourself.

**Problem B**

Given a list of words (strings), print the *k* most frequent elements in descending order. When you print, you have to print the word and its number of occurrences in the list.

If two words have the same frequency, the word with the lower alphabetical order comes first. Use a heap to receive credit.

Proposed solution design and implementation

Problem A

The first thing I did to approach the problem was by looking up what an LRU was and how it worked. It was essential to know how an LRU works in order to be able to solve the problem. Since the operations supported by an LRU are straightforward, it was easy to break up the lab into solving those operations one at a time. The code has all the operations necessary as well as some helper functions such as add\_node and remove\_node which help in retrieving values as well as updating the LRU as necessary. Since the LRU is a structure that you can use as you wish, there is no set input or output that can be given, it all depends on how the program is used.

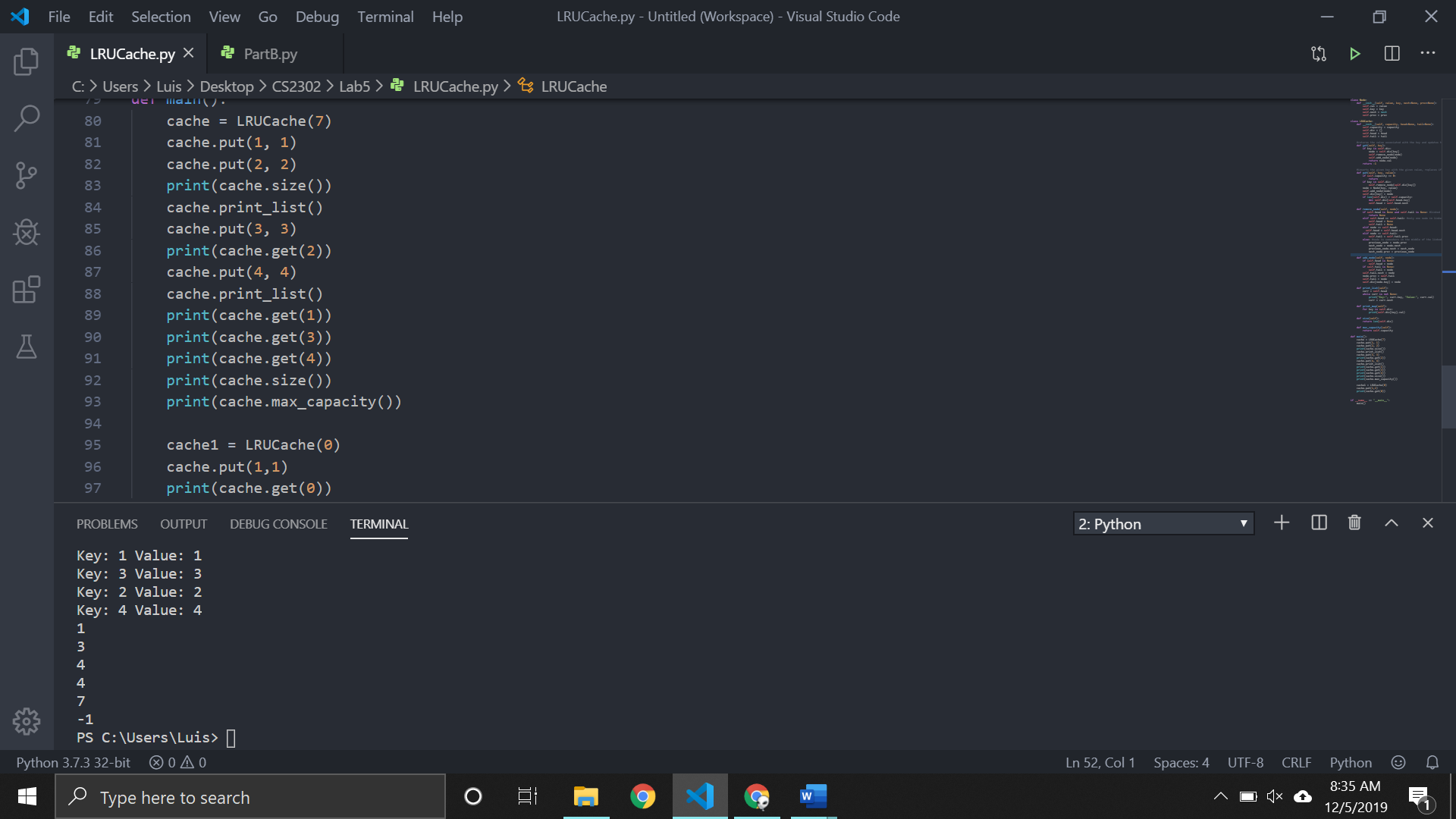
Problem B

I was not able to solve problem B. Since we were told we needed a heap, I tried to understand how a heap would make it work. I was able to understand that but I was not able to correctly implement it. What my code does is that it inserts all the words into a dictionary with the word as the key and the number of occurrences as the value and then it uses those values to insert into the Max Heap which sorts them from least to greatest. However, I was not able to come up with a way to print which words are associated with what number after they get sorted by the heap.

Experimental results

Problem A

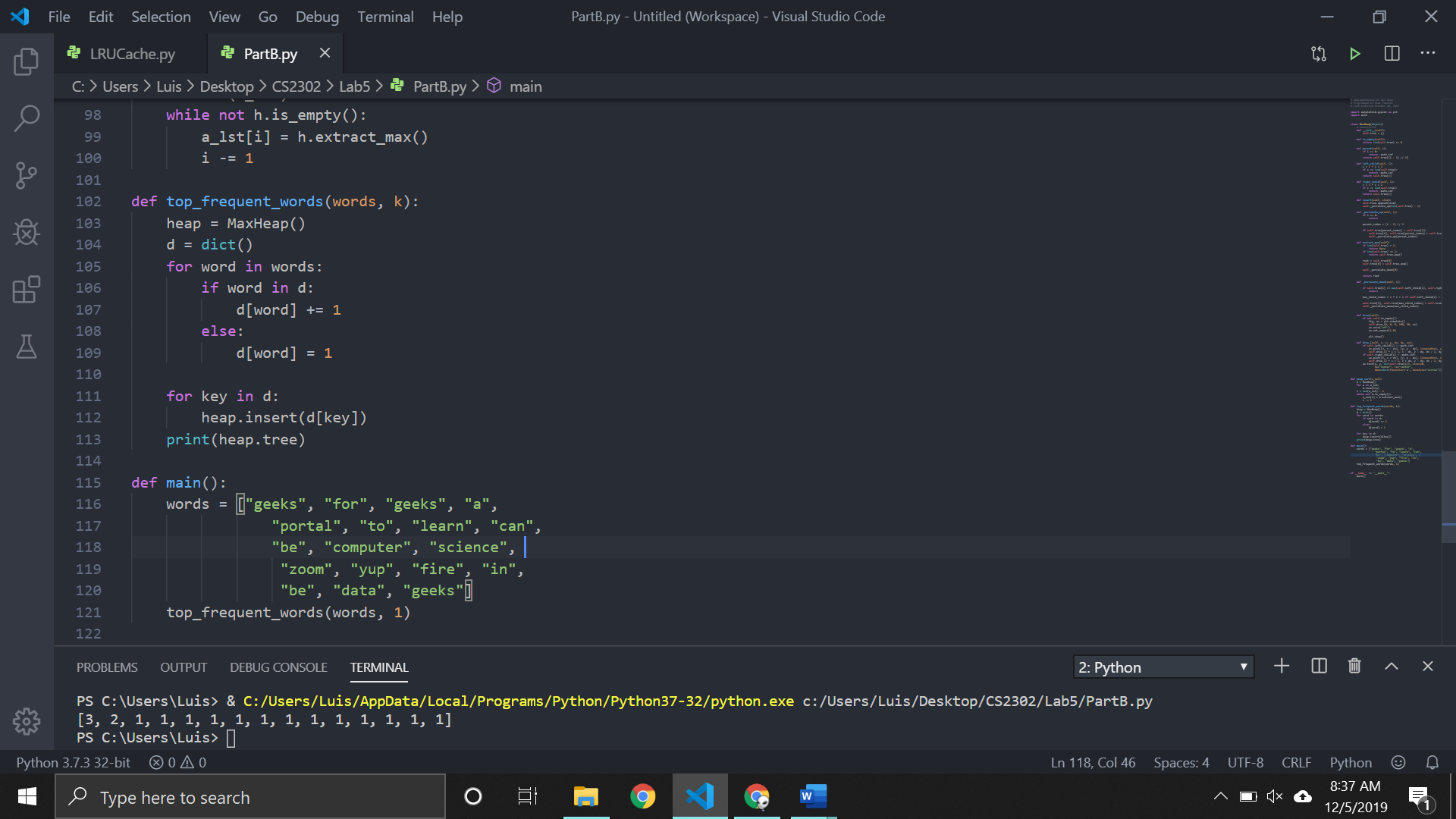
For this problem, I tried inputs such as trying to get a key that wasn’t available and trying to do insert more than the capacity allowed.



All operations can be performed in O(N)

Problem B

For problem B, I just got a sample input of words from an online source and used that to verify my output. It sorts the numbers the correct way but it doesn’t show the words that belong to those numbers



Conclusions

I think the most important thing that I learned from this lab was how to use multiple data structures at once to solve one problem. By doing that, the running time of the algorithms can be greatly improved.

Source Code:

Problem A <https://github.com/lebriones/CS2302/blob/master/LRU.py>

Problem B <https://github.com/lebriones/CS2302/blob/master/PartB.py>