**Realtor.com Take Home Assignment**

**Task:**

Python script to return the list of available anagrams in a file for a user specified input.

**Dependencies:**

Python3

Math Module – for using ceil mathematical function to round total execution time.

Sys Module – for handling command line arguments.

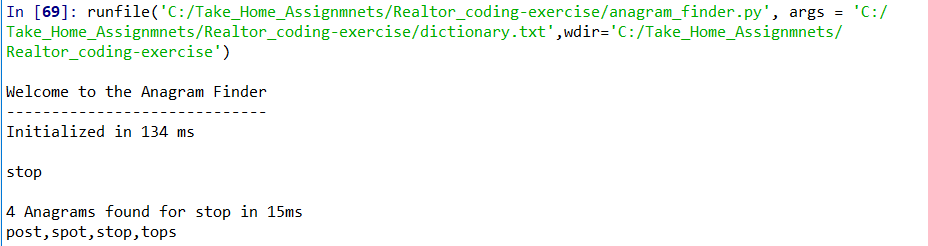
Time Module – for calculating execution time of the application.

Path Module – to validate file path.

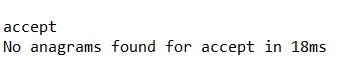
**Code Walkthrough:**

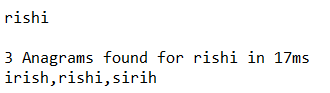
* Imported necessary modules and read file contents from the specified file path in the argument to a list using file read() operation. Striped newlines from the list and converted all the words to lower case for anagram comparison.
* Read user input and called the anagram finder function to list the available anagrams in the dictionary file.
* Created a filtered list with the words only matching the length of the given user input to limit no of searches and to make execution of the program faster.
* Looped through all the words in the filtered list and verified if the current element in the list matches the sorted user input. Used python inbuilt sorted() function to compare the elements in the filtered list and if there’s a match then incremented the counter and stored matched elements in an anagram list.
* Returned list of anagrams found and its respective execution statistics at the end.

**Test Screenshots : Anaconda 3- Spyder Application**



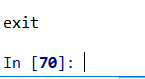
Note: I’ve checked for the word “opts” but couldn’t find it in the dictionary, not sure if there’s some typo in the sample input provided.



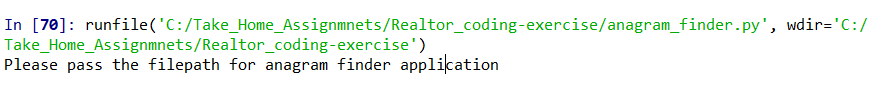




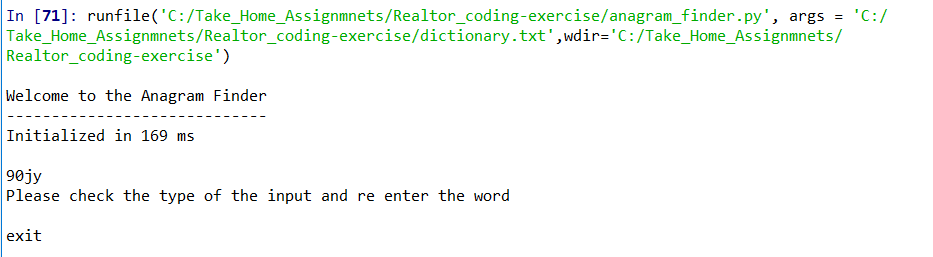


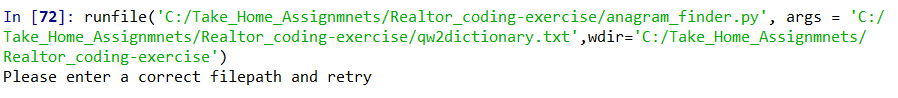
   
  
On exit the application terminates.

**Edge Cases:**



As no file path is specified in the arguments.





Notes:

The overall time I took for this assignment was around 3 to 4 hours with testing and documentation. Developing the logic for this program was simple however I took time in assessing the possible edge cases and the ways to handle them appropriately.

Approach: I started checking for the anagram match directly from the dictionary file and found program was executing way longer. Then I loaded all my file contents to list and then performed a match, results were satisfactory. However, to limit no of searches I looped only through the elements that matched the size (length) of the user input which further reduced the execution time.