Lab 1 - Recursion

CS 2302 – Data structures summer 2019

Dilan Ramirez

INSTRUCTOR: OLAC FUENTES

TEACHING ASSISTANTS (TA): ISMAEL VILLANUEVA-MIRANDA

JUNE 17, 2019

Spring 2019

Contents

[Introduction 2](#_Toc11696051)

[Implementation 2](#_Toc11696052)

[Running Time 2](#_Toc11696053)

[Experimental results 3](#_Toc11696054)

[Conclusion 5](#_Toc11696055)

[Appendix 5](#_Toc11696056)

[Honesty Certification 7](#_Toc11696057)

# Introduction

An anagram is a permutation of the letters of a word that produces another word. Your task for this lab consists of writing a program that asks a user to input a word and then prints all the anagrams of that word. For example, if the user enters the word spot, your program should output the following words: opts, post, pots, spot, stop, tops.

Your program should first read the words in the file https://raw.githubusercontent.com/dwyl/englishwords/master/words alpha.txt, which contains over 466000 words in the English language. As the words are read, they should be stored in a set (see section 13.24 in the online textbook). Your program should then use a recursive function, similar to the ones seen in class and described in the textbook, to find the anagrams. The anagrams should be displayed in alphabetical order and contain no duplicates. Your program should also display the time it took to find the anagrams.

# Implementation

First, I write code to read the “txt” to then save it into a set called “wordSet”. I used the function “line.split()” to create a list of all these words. After that, I used my function that makes all the permutations from a word. I saved all these permutations in a list called “permList”. After that, I used two for loops. One to search into the list of the permutations, and the other to search into the list that has all the English words. If the permutation was equal to the word in the “wordSet,” it will be appended to another list call “foundPermutations” which contains all the final found permutations. Finally, I just print the elements of “foundPermutations”

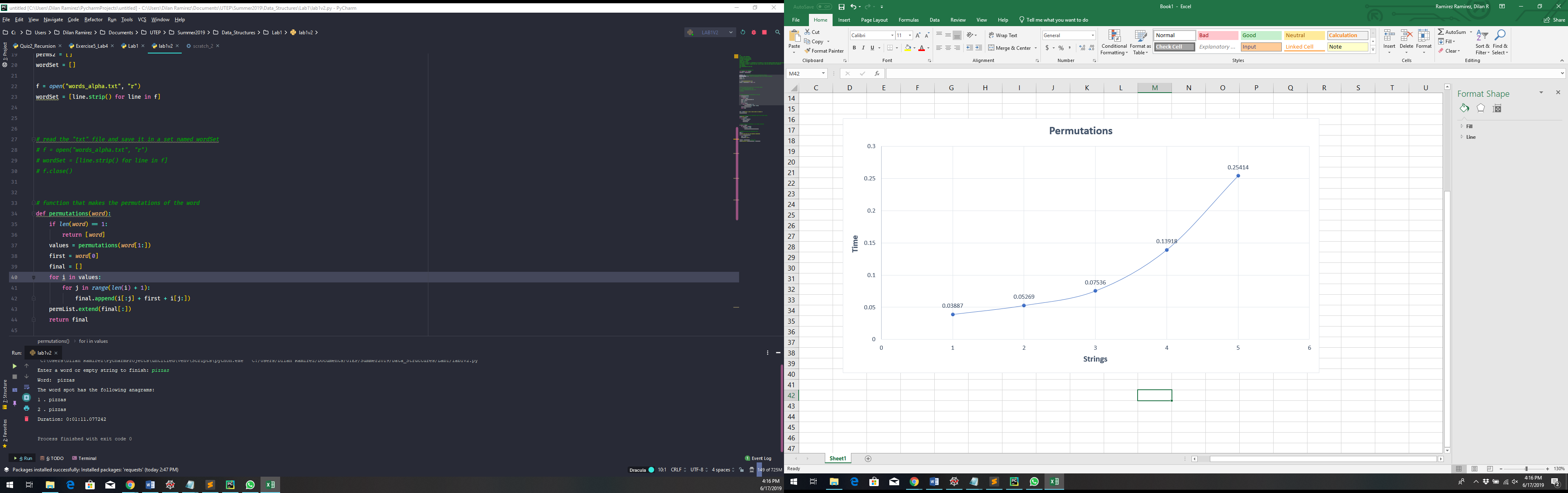
### Running Time

Big-O: O (log(n))

String 1 = “ad“ String 2 = “ bye”

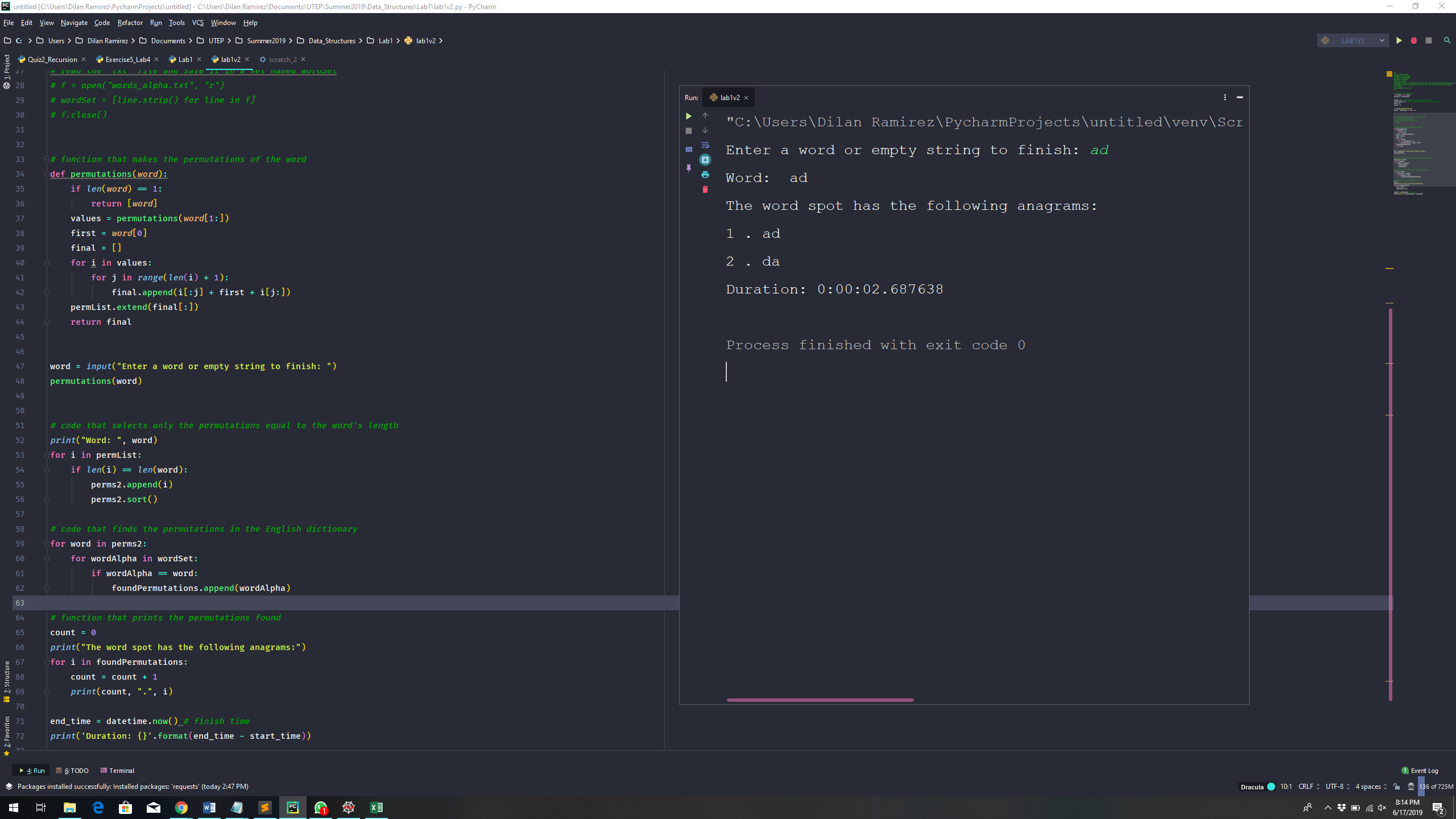
String 3 = “raul” String 4 = “dilan”

String 5 = “pizzas”

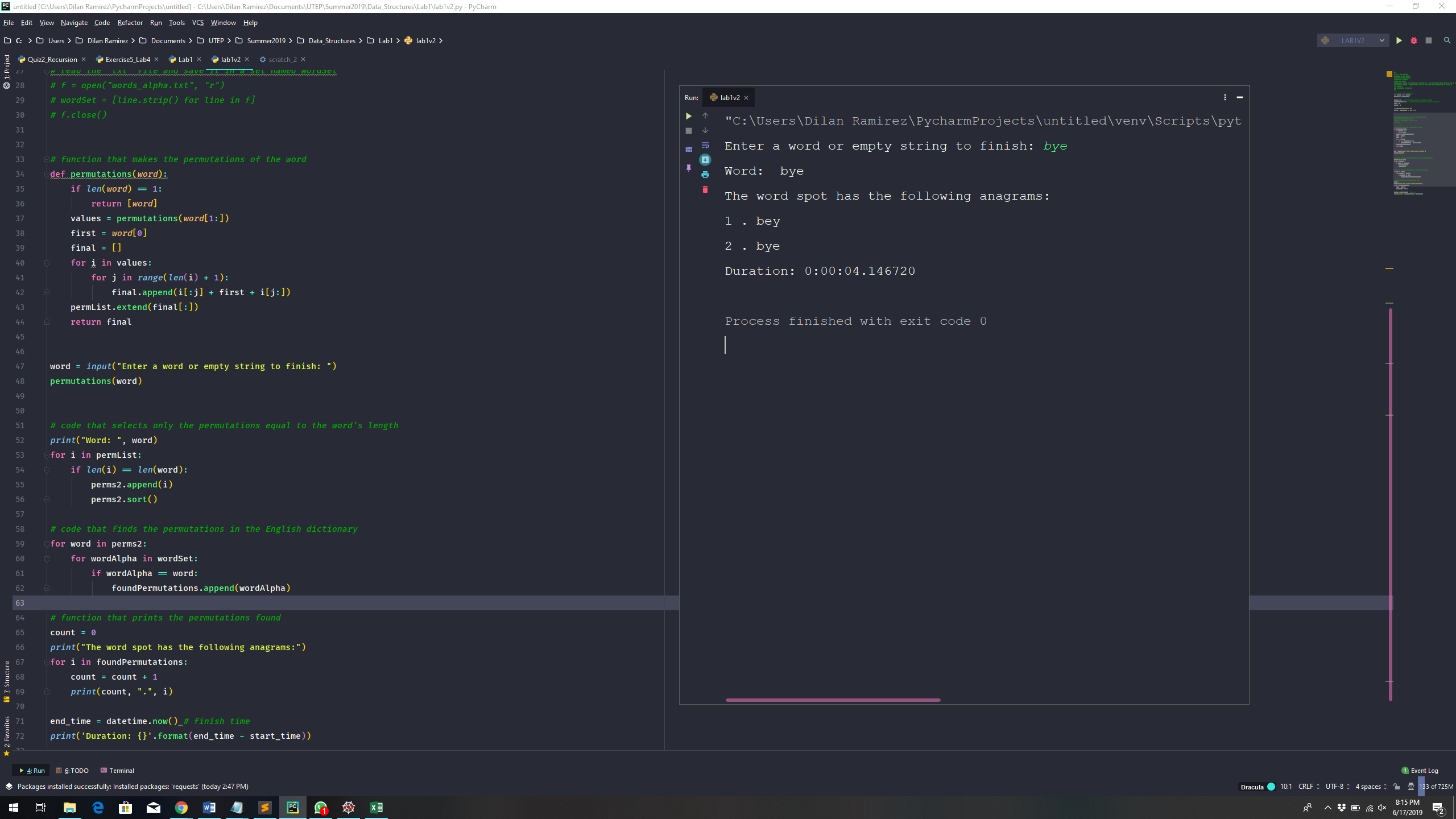


# Experimental results

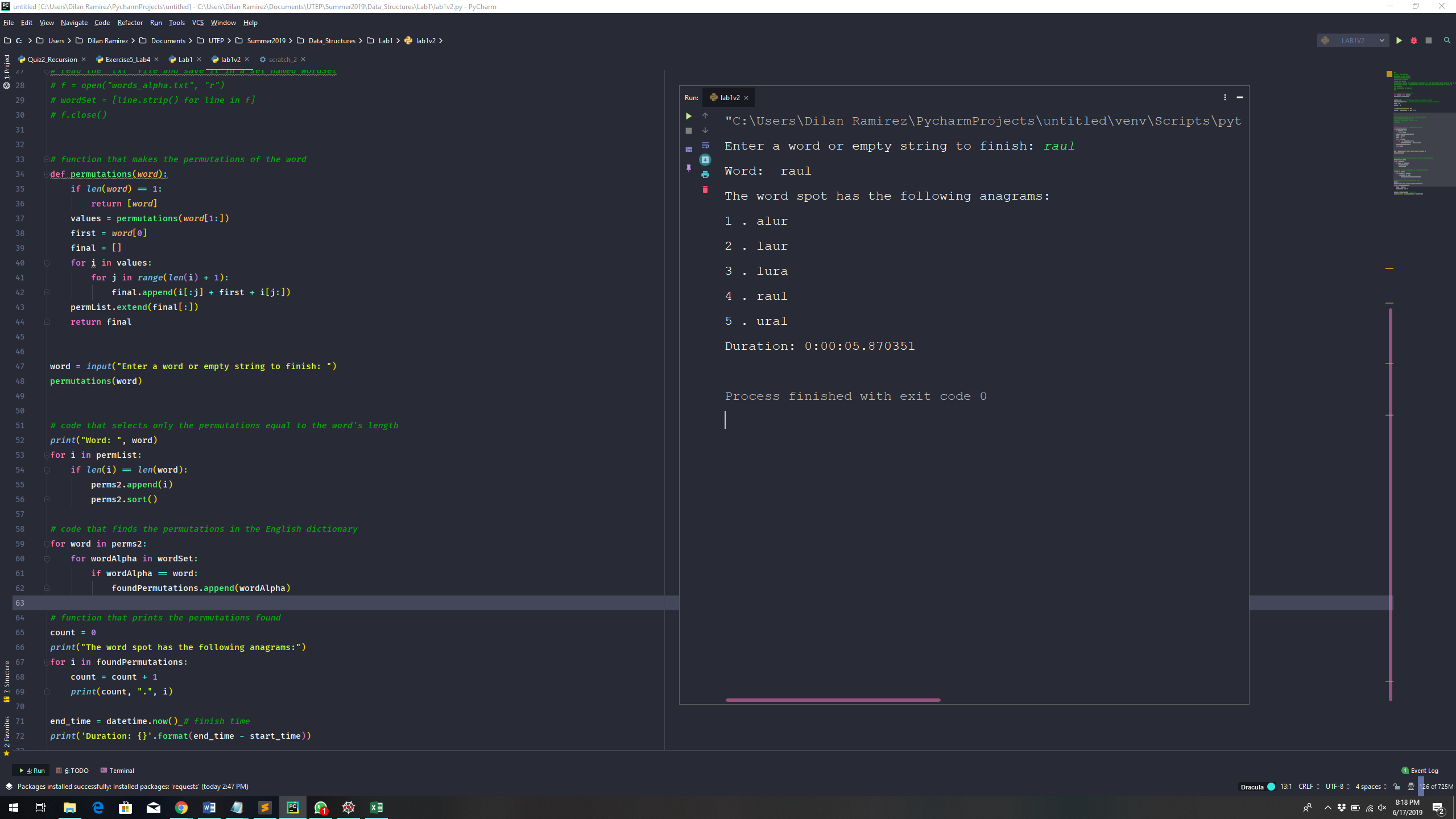
String 1



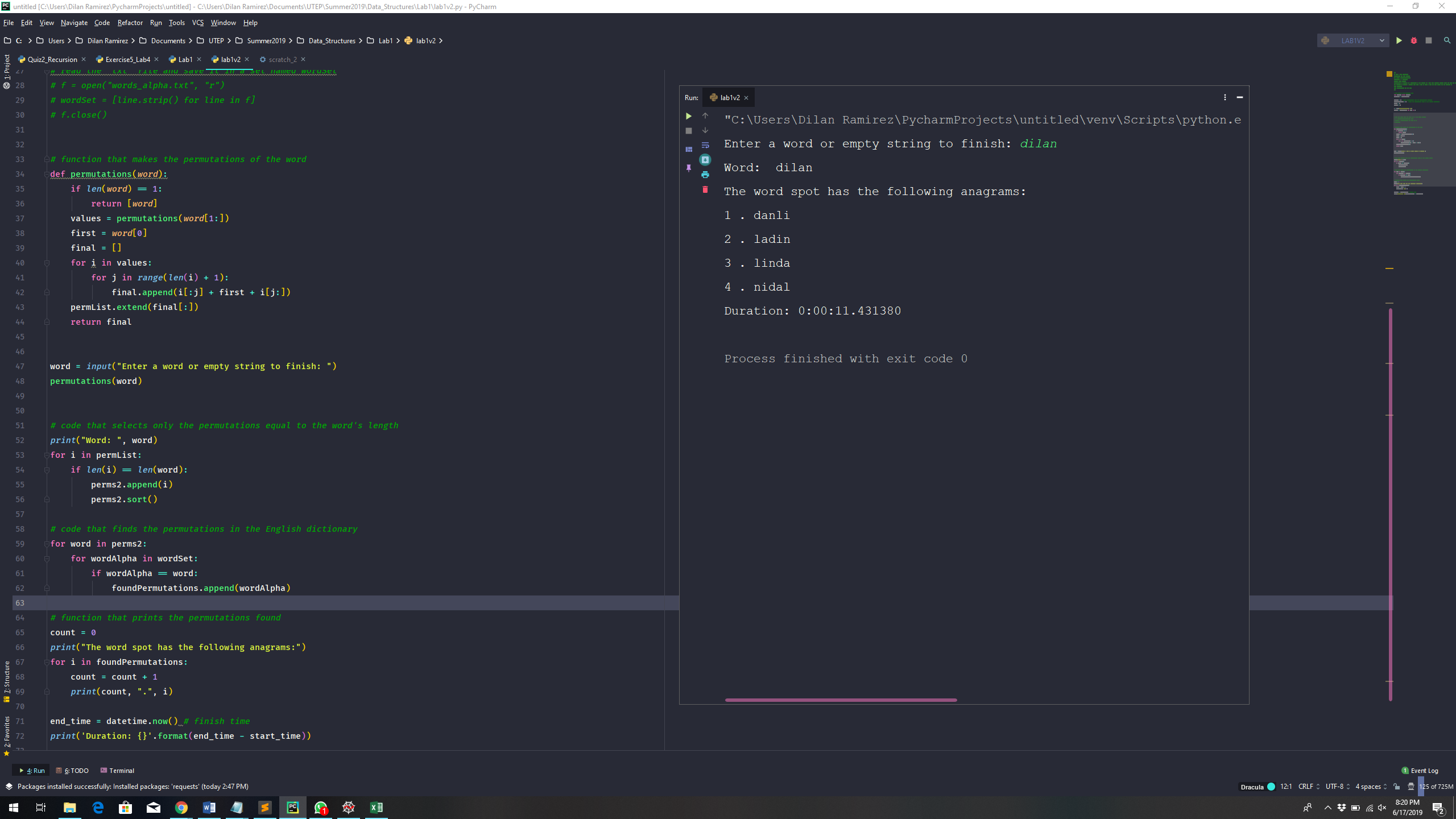
String 2



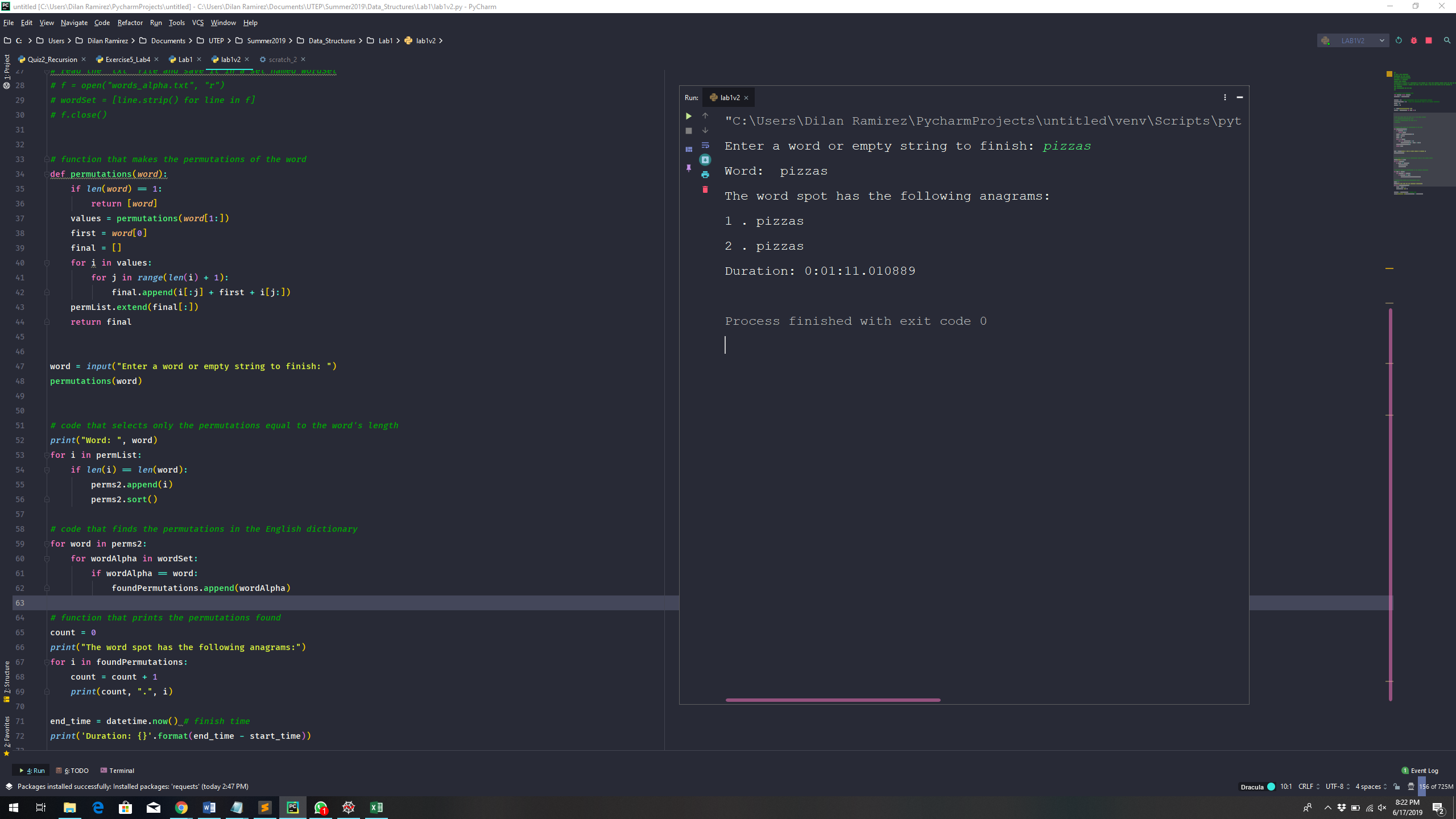
String 3



String 4



String 5



# Conclusion

To do this lab, I had to investigate how to read a txt file, and how to convert it to a set. Therefore, I had to investigate what a set is and its properties. Then, I look for a solution to compare the permutations gotten from my function and the words from the set. Besides, I investigated how to keep track of processing time of my program. The difficult part of this lab was in my recursion function that gets the permutations because it appends the permutations into a list which is inside of the function. The problem was how to use that list to make it global, and then compare the permutations of that list with the list with de English words. Finally, I go thought a solution which was to use the property extended to copy that list to another list.

# Appendix

# keep track of the program's time

from datetime import datetime

start\_time = datetime.now()

permList = [] # saves permutations from the "permutation" function

foundPermutations = [] # saves the permutations found in the English dictionary

perms2 = []

wordSet = []

f = open("words\_alpha.txt", "r")

wordSet = [line.strip() for line in f]

# read the "txt" file and save it in a set named wordSet

# f = open("words\_alpha.txt", "r")

# wordSet = [line.strip() for line in f]

# f.close()

# function that makes the permutations of the word

def permutations(word):

if len(word) == 1:

return [word]

values = permutations(word[1:])

first = word[0]

final = []

for i in values:

for j in range(len(i) + 1):

final.append(i[:j] + first + i[j:])

permList.extend(final[:])

return final

word = input("Enter a word or empty string to finish: ")

permutations(word)

# code that selects only the permutations equal to the word's length

print("Word: ", word)

for i in permList:

if len(i) == len(word):

perms2.append(i)

perms2.sort()

# code that finds the permutations in the English dictionary

for word in perms2:

for wordAlpha in wordSet:

if wordAlpha == word:

foundPermutations.append(wordAlpha)

# function that prints the permutations found

count = 0

print("The word spot has the following anagrams:")

for i in foundPermutations:

count = count + 1

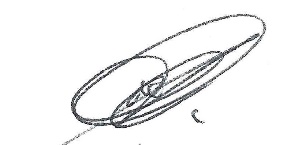
print(count, ".", i)

end\_time = datetime.now() # finish time

print('Duration: {}'.format(end\_time - start\_time))

# Honesty Certification

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provide inappropriate assistance to any student in the class.

 06 / 17 / 2019

Dilan Ramirez Date