

Course:	INFO6143 Python
Professor:	Darryl Bedford
Project:	Project #1 – Jumble Puzzle Solver – Version 1.0
Due Date:	Saturday, June 22, 2024 11:30pm
Submitting:	Please see the last page for instructions

How will my project be marked?

- This project counts for 15% of your final grade and will be evaluated using the following grid:

Marks Available	What are the Marks Awarded For?	Mark Assigned
2	Good coding style including proper indentation and use of variable and object naming conventions and suitable comments.	
3	Loading of the four index files provided into a suitable Python data structure (list or dictionary) as specified in project requirements.	
2	Creation of data entry loop which processes each new anagram (word) and terminates if a blank entry is received. Convert entered words to upper case.	
3	Implementation of “permutations” function which receives a new anagram (word) and returns a unique list of all possible letter permutations.	
3	Uses the permutations list to search the word dictionary (or list) for any and all matches and displays each matched word with the corresponding part of speech (noun, verb, adjective, adverb)	
1	Implementation of timing as “hh:mm:ss” format for each new word as it’s tested (see sample output for placement)	
1	Proper zipped submission, including .py file.	
15	Total	

Project Description

The main goal of this project is to use the four modified WordNet dictionary index files and some Python code to build an application that can solve anagram word game puzzles.

The user enters a series of letters such as “BEALF” and your code will return: “fable” “noun”

...where “fable” is the decoded English word and “noun” is the part of speech. See the example output later in this PDF doc.

Note: You’ll find many “Jumble Puzzle” code solutions on the web in various programming languages; however, in this case, we want you to use the data files provided and write your own code for most parts of the project.

One of the main challenges with this project is generating all the possible permutations for a given set of characters. For example, the anagram “BEALF” has 120 different ways of combining the characters. Web sites such as Stack Overflow have multiple versions of “permutations” functions and you may use one of these code fragments in your solution so long as you include all the source code as a function in your code. Find an example which takes a single test “word” such as “BEALF” as an input parameter and returns a Python list of all possible unique permutations.

You can create your own “permutations” function if you like, however I would recommend doing that at the end because it will make debugging easier to start with something you know works.

The only role played by the WordNet index data in this case is to tell us if one of the permutations is an English word. It’s also possible that any given anagram could contain more than one matching English word.

In some cases, you won’t find a match even though the word exists. For example, the “SKURNH” anagram is intended to be “SHRUNK” however WordNet only has “shrunkn”

JUMBLE
Unscramble these Jumbles,
one letter to each square,
to form four ordinary words.

BEALF

FROEF

CUVAMU

SKURNH

©2020 Tribune Content Agency, LLC
All Rights Reserved.

Answer here: “”

THAT SCRAMBLED WORD GAME
By David L. Hoyt and Jeff Knurek

Get the free JUST JUMBLE app • Follow us on Twitter @PlayJumble

©2024 Fanshawe College

Page 2 of 5

INFO6143 Python, Winter 2024

Specific Requirements:

You can create a solution to the jumble puzzle problem using either a Python nested list or a Python dictionary. It's recommended that you use a dictionary because the performance will be better.

Begin by loading the following index files into your dictionary or nested list:

NounsIndex.txt
VerbsIndex.txt
AdjIndex.txt
AdvIndex.txt

The “key” for each entry in the dictionary should be an English word from one of the index files. You will not need to load any multi-level n-grams as the Jumble Puzzle game only uses single words.

The “value” property for each dictionary element should be the correct part of speech for the word, i.e., noun, verb, adjective, adverb.

The above is only a suggestion on how to manage the data for the project, however **it's a requirement that you use the WordNet derived index files as provided.**

Because words that are longer than five or six characters have many more possible permutations, we want to time our code to see how long the processing takes. See the output below for a demonstration of how this should look and also the “Notes” section for some help getting and formatting the current time.

Finally, creating a solution for this project is best done by starting with the sample output and doing some research to determine what it takes to create the code to do the same thing.

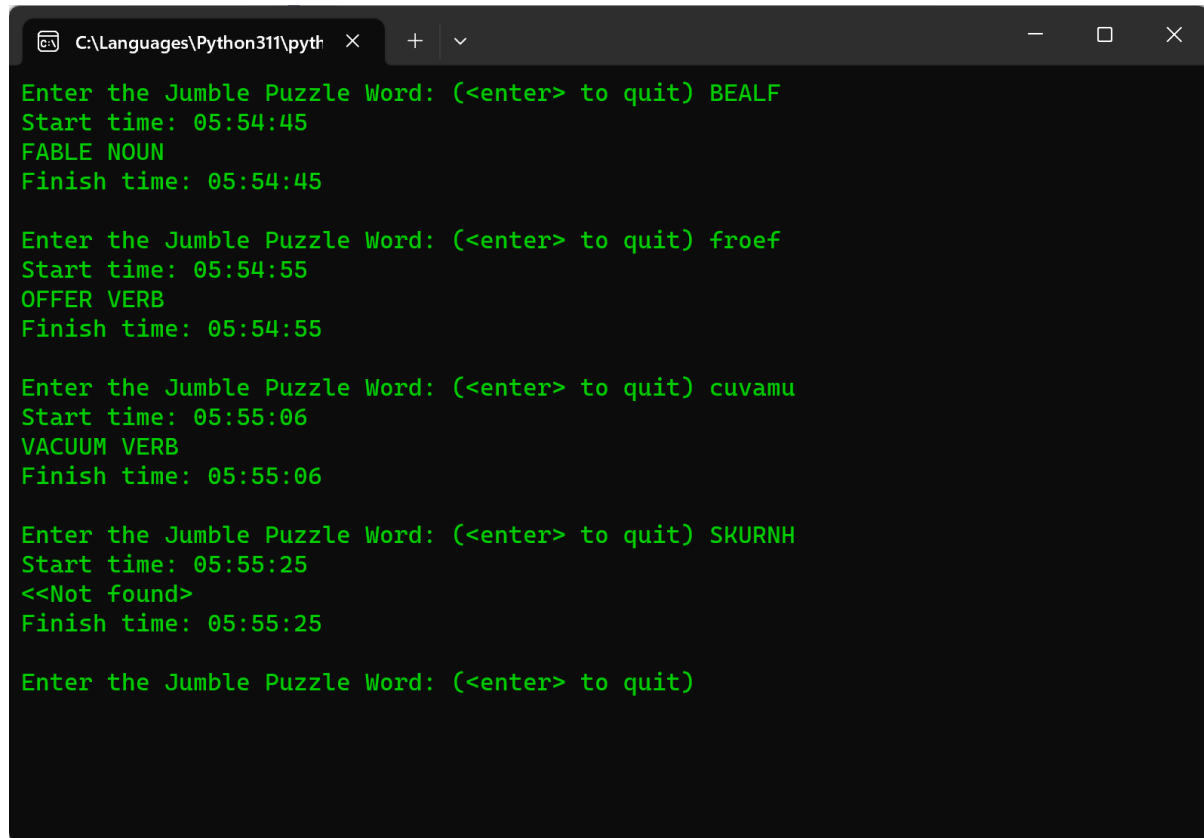
*** End of Requirements ***

Notes:

Here's some sample code for getting the current time:

```
import time
curr_time = time.strftime("%H:%M:%S", time.localtime())
```

Sample Output:



```
C:\Languages\Python311\pyth x + v
Enter the Jumble Puzzle Word: (<enter> to quit) BEALF
Start time: 05:54:45
FABLE NOUN
Finish time: 05:54:45

Enter the Jumble Puzzle Word: (<enter> to quit) froef
Start time: 05:54:55
OFFER VERB
Finish time: 05:54:55

Enter the Jumble Puzzle Word: (<enter> to quit) cuvamu
Start time: 05:55:06
VACUUM VERB
Finish time: 05:55:06

Enter the Jumble Puzzle Word: (<enter> to quit) SKURNH
Start time: 05:55:25
<<Not found>
Finish time: 05:55:25

Enter the Jumble Puzzle Word: (<enter> to quit)
```

How should I submit my project?

Electronic Submission:

Submit your program files to the *Info6143 "Project 1"* electronic submission folder in *FanshaweOnline*. These files should be submitted as a single "zip" file containing your web application's complete website.

I strongly recommend that you test your own submission to ensure that nothing has been missed.

Darryl's note: the grading and submission criteria for projects are more strict than for labs.

School of IT Policy: Submit your project on time

Project submissions must be made on time! Late projects will be subject to divisional policy on missed tests and late projects. In accordance with this policy, no late projects will be accepted without prior notification being received by the instructor from the student.

Fanshawe College Academic Integrity Policy

It is considered cheating to submit work done by another student or from another source. Helping another student cheat by sharing your work with them is also not tolerated. Students are encouraged to share ideas and to work together on practice exercises, but any code or documentation prepared for a project must be done by the individual student. Penalties for cheating or helping another student cheat may include being assigned zero on the project with even more severe penalties if you are caught cheating more than once. Just submit your own work and benefit from having made the effort on your own.