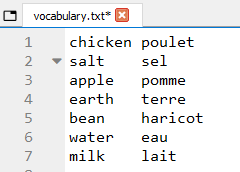
**Extendable Language Translation Using a Vocabulary File**

Once again, we find ourselves doing language translation. In our previous experiences we learned about how to work with lists and dictionaries. In the process you created programs that recognized when a word was not in the vocabulary and then added it. Unfortunately, any extensions to the vocabulary were lost when the program was closed.

This problem can be solved by using a data file to hold the vocabulary. When vocabulary words are added and saved to the vocabulary file, they are not lost when the program ends.

In this project, your program will read from and write to a vocabulary file saved in the same folder as your source code. The vocabulary file is just a simple list of word pairs in a text file like this …

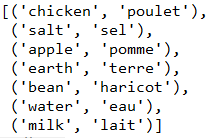


Use the file-oriented Python constructs, functions, and methods to read this file into the data structures of your choice (lists or dictionaries) and implement a user interface as described in the previous two weeks. You ought to be able to reuse most of your code. In fact, learning about code reuse is important to your development as a programmer. If you have gone through the process of breaking down programming problems into individual steps and developed your code one step at a time, you have already solved most of the problems presented here. The only thing new is the aspect of file storage.

To read the vocabulary into your program, I suggest creating a with block that opens the vocabulary file and then reads the data using the **readlines()** method. This will create a Python list of strings … one string for each line in the vocabulary file.



This will produce a list that you can convert using **split()** in a list comprehension to produce a list of tuples. Your list comprehension may need to include an **if** clause to reject any empty lines that may have been included at the end of the file. You ought to be able to produce a list comprehension using data that results in this …



Once you have this, it ought to be relatively easy to modify the structure as needed to implement language translation just as you have in the previous two weeks. If you wish to try another method, be my guest.

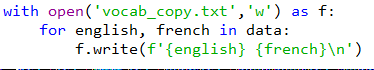
So, now the question is: what happens after the user has added a word to the vocabulary? How do I write that new information to the vocabulary file?

When the vocabulary has been extended, the easiest and safest thing to do is to write the entire data structure back to the data file. Some of you may think that using the append mode is the best way to go but, in my experience, simply re-writing the entire vocabulary gives a cleaner result.

In the code below, I open and write to a separate vocabulary file (vocab\_copy.txt) for safety. But really, to extend the vocabulary, you should write to the same file you originally read the data from. That way, the extended vocabulary is available the next time the program is run. I use a **with** block containing a **for** loop which traverses my data list. Then, I use a formatted string literal to write each item in the tuple separated by a space character and then terminate the string with a newline character.



I could also unpack the tuple in the **for** loop and make the code a bit more readable …



Your submission must:

* Use a vocabulary file named "vocabulary.txt" which is in the same directory as your source code. Your program must not …
  + include a directory path to the file in your call to **open()** (use only the filename + extension)
  + hard-code any of the vocabulary (all vocabulary must come from the vocabulary file)
* Implement a user interface that continues to translate words and/or add words to the vocabulary until the user enters a null string.
* Extend the vocabulary file whenever the user adds a new word.
* Avoid inefficiencies such as:
  + redundant/unnecessary operations
  + redundant/unnecessary variables
  + variable names that are not descriptive
  + loops that are not Pythonic