**Lab Assignment 4  
Literal Translator**

|  |  |
| --- | --- |
| **Due Dates** | |
| **Wednesday Labs** | **11/04/20 @ 11:59** |
| **Async. Labs** | **11/05/20 @ 11:59** |

**Objectives**

* Review working with files
* Review handling exceptions

**Background**

One simple technique for machine translation involves going word by word in the source text and replacing it with a corresponding word in the target language. Such literal translation (cf. <http://en.wikipedia.org/wiki/Literal_translation>) usually results in a close but inaccurate translation, primarily because the rules for word order vary from language to language.

However, this literal translator actually has some good uses. First, it’s not always necessary to have a perfect translation, especially if it is much quicker and easier not to. Second, other methods use such a translation to generate a set of words, and then statistically find the most likely ordering for those words. For example, try putting the following words in the correct order (you probably can do it easily because you know that certain words generally follow other words):

have programming a seen never I language better

**Problem Specification**

Write a Python application that provides a literal translation for text entered by a user. To determine how to translate the words, you will use something called a ***parallel corpus****.* A parallel corpus is a large collection of text in two or more languages.

For this assignment the corpus will be much simpler than most ***corpora*** in that each line contains only a single word in each language, separated by a comma. All you will need to do is find the word in the first language, and replace it with its partner in the second.

* If the corpus does not contain a particular word, you should leave it **untranslated**.

The following is an example of a parallel corpus text file (French-English).

* The first line indicates the number of entries (you may assume this number is correct).
* Accents on letters are ignored (for easy keyboard input).
* Sometimes two different words might translate to the same word (“le” / ”la” both translate to “the”), but you may assume each word on the left-hand side appears only once.

**Example Input:**

8

le,the

la,the

meduse,jellyfish

garcon,boy

danse,dances

avec,with

paresseux,lazy

puante,smelly

Using this corpus, a literal translator can translate:

**Le garcon paresseux danse avec la meduse puante!**

into:

**The boy lazy dances with the jellyfish smelly!**

For this assignment, you are to create two **csv** corpus files: one to convert from English to French and another to convert from French to English. You can use the corpus provided above for the translation from French to English, and create the English to French corpus by reversing the order of the words. You can also add additional words. Your program should select the appropriate corpus file based on the user’s option (i.e., translate from English to French, or translate from French to English).

Your application should exhibit the following functionality (see the sample output below):

* Allow the user to select an option (1 or 2) for what translation is to be performed (i.e., English to French OR French to English). This determines the file name of the corpus to be used for the translation (this way your program could be used for multiple languages).
  + Display a message indicating that the corpus file was successfully read.
* Allow the user to enter some source text that is to be translated.
* When the user commits the input by pressing on the “enter” key, translate the text and display the result.
  + The source text may contain capital (uppercase) letters and/or punctuation. You should convert uppercase letters to lowercase and ignore punctuation, except in the following cases:
    - if the first character in the source is capitalized, capitalize it in the result as well.
    - if the last character in the source is punctuation (*not* a letter or digit), include it in the result as well.
  + If a source word cannot be found in the corpus, just leave it untranslated (this is good for things like proper nouns).
* Your program should loop continuously till the user indicates that they want to quit.
  + With each loop, prompt the user to:
    - re-enter an option to determine the filename of the corpus to be used;
    - enter a new source text to be translated.
* If any exceptions are encountered, display an appropriate message to the user indicating what the problem is.

Your application **must** perform exception handling using *try-except-else* statements; “finally” statements are optional. Your program should also explicitly **raise** exceptions where appropriate to validate user input.

Your program should **at a minimum** handle the following exceptions, but you may include additional exceptions if you deem it necessary. For each exception handled, an appropriate message / explanation should be displayed to the user, followed by a prompt to re-enter the requested information.

* **FileNotFoundError** 
  + in case a user tries to open a non-existent corpus file for reading
* **ValueError**:
  + in case a user tries to write to a file that has already been closed, OR
  + a user enters a non-numeric option.

Your program should also properly handle the following cases:

* the user enters nothing but presses on the ‘enter” key.
* the user’s input is numeric, but not one of the valid options (i.e., not 1 or 2)

**Example Output:**

Welcome to the Translator! Bienvenue!!

You can translate sentences from English to French and vice versa!

So, what would you like to do? Please select from one of the following:

>> English to French: (input '1')

>> French to English: (input '2')k

Invalid input! Input must be an integer, and must be 1 or 2! Try again:2.5

Invalid input! Input must be an integer, and must be 1 or 2! Try again:5

Invalid input! Input must be an integer, and must be 1 or 2! Try again:1

Input the sentence for translation: The lazy boy dances with the smelly jellyfish!

Your sentence translated is:

>> Le paresseux garcon danse avec le puante meduse!

Another translation? (y/n)2

Invalid input! Please enter 'y' or 'n':Y

So, what would you like to do? Please select from one of the following:

>> English to French: (input '1')

>> French to English: (input '2')

Invalid input! Input must be an integer, and must be 1 or 2! Try again:2

Input the sentence for translation:

No input!

Input the sentence for translation: Le garcon danse.

Your sentence translated is:

>> The boy dances.

Another translation? (y/n)n

Process finished with exit code 0

**Design** **Requirements**

Your program should have a user interface class **UserInterface** which implements the methods given below. This class interfaces with the user as follows:

* requests input,
* reads in user input, and
* uses the appropriate classes to perform the required tasks.

**class** UserInterface:  
  
 **def** \_\_init\_\_(self):  
*# Calls the necessary methods to: display a greeting, ask the user what  
 # s/he wants to do; update the  
 # corpus file name and the text to be translated based on the user’s input,  
 # uses these to initialize  
 # a Translator object, and loops until the user no longer wants  
 # to translate any sentences.* **def** run\_program(self):  
*# Displays the translation options to the user and, after reading  
 # in the option selected by the user, sets  
 # the name of the corpus file. Uses try-except to enforce correct input.* **def** get\_corpus\_filename(self):  
*# Requests the text the user wants to translate (i.e. the source text).* **def** get\_source\_text(self):  
*# Uses the Translator object to translate the source text  
 # and displays the translated text to the user.* **def** translate(self):

You should also include a **Translator** class in your project with the following methods.

class Translator:  
 def \_\_init\_\_(self, file, input):  
  
 *# Reads in the data from the input file and stores the data in a two  
 # dimensional list for easy retrieval during the translation process.* def read\_corpus(self):  
  
 *# Calls the necessary methods to perform the translation and  
 # returns the translated text.* def translate(self):  
  
 *# Uses the private method: “lookup(word)” to look up each English  
 # word in the user’s input text, creates a string representing  
 # the translation of the user’s input, and updates the corresponding  
 # data attribute with the translation of the user’s text  
 # (in French).* def english\_to\_french(self):  
  
 *# Uses the private method: “lookup(word)” to look up each French  
 # word in the user’s input text, creates a string representing the  
 # translation of the user’s input, and updates the corresponding  
 # data attribute with the translation of the user’s text (in English).* def french\_to\_english(self):  
  
 *# Accepts a word as parameter, looks up the word in the  
 # appropriate corpus, and returns the corresponding  
 # translation. If the word is not found in the  
 # corpus, it returns the word received as parameter.* def \_\_lookup(self, word):

A method to display a greeting to the user should be included in your application:

# Displays a greeting to the user and indicates briefly what

# the program does.

def greeting():

Your project should also have a main method:

# Creates an instance of the UserInterface class,

# and uses that instance to run the program.

# You should handle the FileNotFoundError in main.

def main():

**Hint(s)**

* Research into how you can split a string using multiple delimiters.

**Additional Requirements**

A proper *design* (with detailed pseudocode) and proper *testing* are essential.

*Note:* ***Correct pseudocode development*** and **proper *testing*** *will be worth* ***40%*** *of the total LA grade.*

You will also need to install **SPHINX** (if you haven’t yet done so) and use it to **generate HTML documentation** for your projects in PyCharm. This has been reviewed in the Labs. Accurately following the steps shown in class (a copy of which is available in Elearning) will result in the creation of a file “**index.html**” for your project; this file can be opened in a browser and will contain your project’s documentation.

**Coding Standards**

You must adhere to all conventions applicable to writing programs. This includes the use of white spaces and indentations for readability, the use of comments to explain the meaning of various methods and attributes, and the conventions for naming classes, variables, method parameters and methods.

**Assignment Submission**

* Generate a .zip file that contains all your files including:
  + Program Files
  + Any input or output files
  + The document containing your pseudocode and a proper testing.
* Submit the .zip file to the appropriate folder on ELearning.

**NOTE**: The Elearning folder for LA submission will remain open beyond the due date but will indicate how many days late an assignment was submitted where applicable. The dropbox will be inaccessible seven days after the due date by which time no more credit can be received for the assignment.

The penalty for late submissions as stated in the course syllabus will be applied in grading any assignment submitted late.