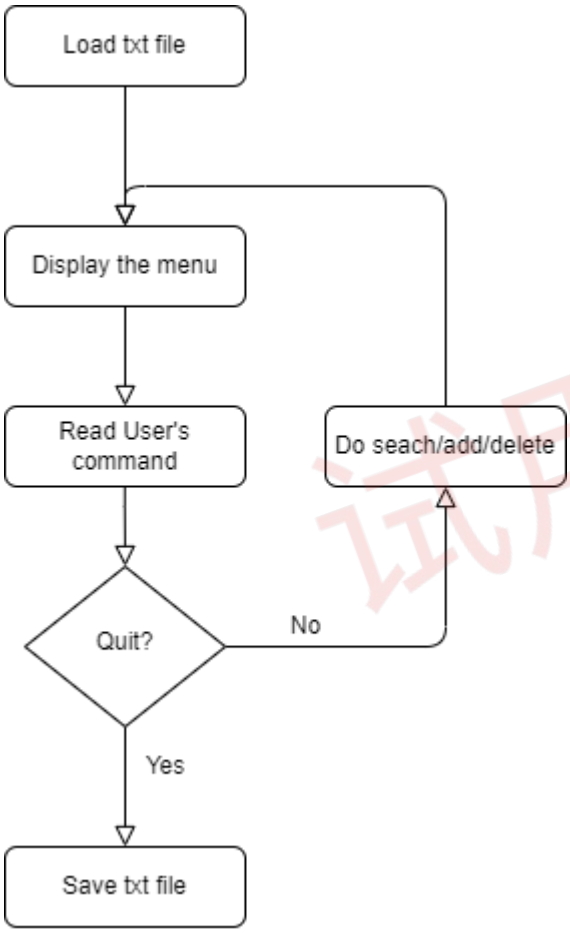
Mini-project

Part 1 Dictionary

1. Overview

In this part, you are required to implement an English-to-Chinese dictionary with search, add, and delete functions. The program should also support loading and saving dictionary data to a local .txt file. Before starting the detailed requirements, refer to the flowchart in Figure 1 for guidance.



2. Requirements

(a) Before-running (10pts)

It should first read a txt representing local storage. This file, stored in the same folder as the program, consists of several lines. An English word + (space) + a Chinese word make up each line. For example, “apple 苹果”

After loading the txt file, you should output a welcome message including your student ID and name.

(Hint1) You can use `encoding="utf-8"` as the parameter in the `open` function as the txt file contains Chinese characters.

(Hint2) You had better close the file as long as you have loaded all contents, since you need to write back to the same file when the program terminates.

(Hint3) Use the `global` keyword to modify variables outside of functions if necessary.

(b) In-running (80pts)

(I) Menu (20pts)

After completing any operation (e.g., search, add, or delete) or loading the .txt file, the program should display a menu with four commands: `quit, search, add,` and `delete`. The program should validate the user input, re-prompting for a valid command if necessary.

(Hit-point) You need to handle the case that the user input is **NON-ALL** lowercase characters. You can still execute these commands.

(Statement) Each input contains only one word (no other spaces).

(Hint) You can use a while loop for input(). Quit the program if the user enters `quit`.

(II) Search (20pts)

When the `search` command is entered, prompt the user for an English word. If the input contains invalid characters (e.g., Chinese characters or digits), print an error message and terminate the search. Otherwise, display the corresponding Chinese translation.

(Hit-point) You need to handle the case that the word doesn't exist. Printing an error message is required in this case.

(Statement) Each input contains only one word (no Chinese or other spaces).

(Hint) The solution to check if a string is made of letters is mentioned in the class of week 5.

(III) Add (20pts)

The `add` command will lead to this stage. You need to prompt the user to enter a new pair in the format English word + (space) + Chinese word. Then, you need to store that pair in your mini dictionary.

(Hit-point) You need to handle the case that the pair already exists. If two pairs with the same English word have different Chinese meanings, you should merge them; otherwise, simply ignore the one pair.

(Statement) The input always follows the format English word + (space) + Chinese word (no other spaces).

(Hint) If the same key is assigned twice, the previous value will be overwritten.

(IV) Delete (20pts)

The `delete` command will lead to this stage. First, you need to prompt the user to enter an English word.

You need to delete that pair and inform the users the process is done.

(Hit-point) You need to handle the case that the word doesn't exist. Printing a message is required in this case.

(Statement) Each input contains only one word (no Chinese or other spaces).

(Hint) You may use the `del` function.

(c) After-running (10pts)

The `quit` command will lead to this stage. You should save all the contents to the same txt file. The format should also be the same as the file in the *Before-running* stage. Finally, you should close the file.

(Hint1) You can use `encoding="utf-8"` as the parameter in the `open` function as the txt file contains Chinese characters.

(Hint2) To avoid overwritten, you may save it to another txt file different from the load one when testing. However, **you should change it back to `dict.txt` before submitting.**

3. Grading Criteria

Score	Requirement
60	Fully Submitted (Failed in interpretation)
80	Implemented basic functions (Hit-point excluded)
90	Implemented all functions with a poor interface*
100	Implemented all functions with a nice interface*

Besides, you must contain every task to a function (**-10 points** if not). Your overall score is the weighted sum of sub-scores for each task. You can attach a brief report (no extra credit).

**Remark* : You do not need to design an actual user interface using extra module. A nice user interface example is showed in the appendix.

4. Submission

You need to rename your program and sample txt file as `dict.py` and `dict.txt,` respectively.

Part 2 Translation Application

1. Overview

You are required to create a graphical user interface (GUI) translation program using Python's Tkinter library and Baidu Translate API. The program should allow the user to input English text, and upon clicking a button, the program will translate the text into Chinese and display the result on the interface.

2. Requirement

(I) Use Tkinter to Create the GUI:

a) A text input field where the user can enter English text.

b) A button that triggers the translation process.

c) A text box or label to display the translated Chinese text.

(II) Integrate Baidu Translate API or other APIs:

a) Use the Baidu Translate API to perform the translation from English to Chinese.

b) You will need to sign up for the **Baidu Translate API** and obtain your **APP ID** and **Secret Key**, which will be used in the program for authentication. The API sign up link is as follow:

http://api.fanyi.baidu.com/choose

Choose "通用文本翻译" to start your application, you only need to apply for the free version.

(Hint1) Tkinter is a built-in library in Python, so no additional installation is required. Please refer to the Tkinter official document: https://docs.python.org/3/library/tkinter.html

(Hint2) Refer to the Baidu Translate API Documentation to learn how to make API requests and handle JSON responses. http://api.fanyi.baidu.com/doc/21

(Hint3) Use the requests library to send HTTP requests to the Baidu Translate API.

(III) Program Functionality:

a) The user should be able to input English text into the Tkinter text box.

b) When the user clicks the "Translate" button, the program should send the text to Baidu Translate API and retrieve the Chinese translation.

c) The translated Chinese text should be displayed on the GUI.

3. Grading Criteria

60	Fully Submitted (Failed in interpretation)
80	Implemented basic functions
90	Implemented all functions with a poor interface*
100	Implemented all functions with a nice interface*

**Remark* : A nice user interface example is showed in the appendix.

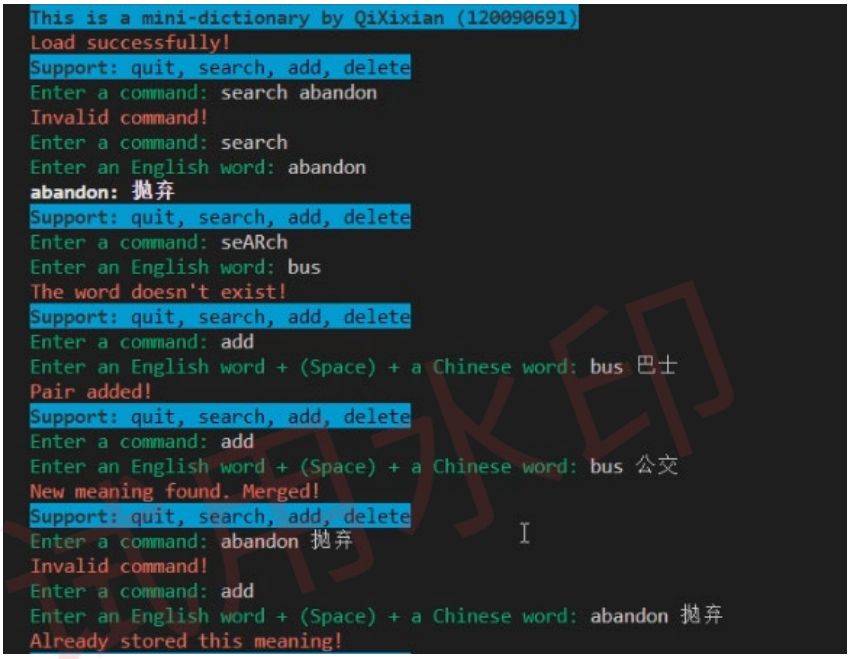
4. Submission

You need to rename your program file as translator.py. You can attach a brief report (no extra credit but can help us run and test your program smoothly).

Then, you should compress all files (include part1-dictionary and part2-translator) into a single zip file and name it using your student id, for example,
`project1_LiuXiaoliang_121090352.zip`

Appendix

1. Example user interface for Part 1*



*You don’t have to implement the color of each output line in your program.

2. Example user interface for Part 2