Team 6D – Bing Chilling Documentation

Student ID	Members
1006874	Peh Cheng Ye
1006864	Tan Yan Zu, Joe
1007009	Thirunavukkarasu Harshini
1007088	Rout Bishmit
1006867	Neo Yau Jun Lucius

The game requires 4 libraries:

- **Tkinter** , as tk, our main graphic user interface, including messagebox
- **Turtle** , for animations
- **Time,** for functions which require the use of a timer and waiting
- **Random,** for choice function to randomly choose an index from a list.

Global Functions

convert_data_from_file()
Extracts data from .csv file. This is done with the help of a number of empty lists as .csv files are simply elements separated with just a comma.

We created 2 lists, ls_key and ls_value, and iterated the process of mapping the “Chinese” Values and “English” Values to their respective Keys (with zip function). Every iteration creates a dictionary, dict_in, in which it is appended into a final list, ls_out. This is so that we can retrieve specific dictionaries by referencing the respective index in ls_out, since dictionaries cannot be referenced by its index (only by its key).

time_convert (sec)[1]
This function takes in the number of seconds, and converts it into the number of hours, minutes in integers, and the number of seconds rounded off to 2 decimal places. It returns a string into the format of “{hours};{minutes}{seconds}”

Class

Each window screen is a **class** for easier reference. Usage of similar variable names are isolated in the class unless the **global** variable is declared, which will be updated subsequently in the functions below.

_ _ init _ _ (self)
This function is to initialize the attributes of the tkinter window.

[1] Modified code from Problem Set 2B: seconds_to_hours(seconds)

1)Across all the windows is setting the dimensions of the window size (**tk.geometry**) and the title (**tk.Title**) of the window.

2) The user interface is then placed with multiple widgets like texts(**tk.Label**), buttons(**tk.Button**), a white board **(tk.Canvas**).

3) To place the widgets into the window, we used 2 methods, the pack(**tk.pack**) for widgets in the 1 column, and grid (**tk.grid**) to place widgets in a certain order based on their rows and columns.

Start screen

_ _ init _ _ (self)

The window takes in a user input for their name (**tk.Entry)**, where they will be prompted again with a messagebox when they click “Enter’

update(self)

User is prompted of their choice of username in a messagebox. If user clicks “yes”, user will be redirected to the Main Menu with their name input, player_name stored as a global variable. The previous windows are automatically closed.

Enter_shortcut(self,event)

To bind the “Enter” key to the window, so user can easily press the “Enter” key to go to the messagebox

Main Menu Screen

_ _ init _ _ (self)

Initializes the window, which has 3 buttons, which will lead to the Gameplay Screen, the Credits Screen and the exit.

exit(self)

When user presses “Exit” button.
User is prompted of their choice to leave the game. If user clicks “yes”, the program stops running.

to_gameplay(self)

Automatically closes main menu, goes to Gameplay Screen.

Gameplay Screen

_ _ init _ _ (self)

Initializes the window, which has a canvas and 4 buttons, 3 of which are hidden, while the “Start” button is shown. The window also displays instructions on how to play the game.

start_flashcards(self)

When user presses “Start” button,
The function calls in the global variable current_card, which uses the random.choice function on the list of dictionaries, test_dict. The random.choice function takes in a random index in the given list to return a dictionary element, where the 2 keys are “Chinese” & “English”.
The keys map to their respective values of the Chinese and English translation of a given word.

Example of dictionary element = {“Chinese”:”Ping Guo”,”English”:”Apple”}

The canvas texts are configured to the “Chinese” key and its respective value.
The “Reveal” button appears and the timer starts.

reveals_answer(self)
When user presses “Reveal” button
Based on the dictionary element before, this function configures the canvas to show texts of the “English” key and its respective value.
The “Reveal” button disappears while the “Correct” and “Test again” button appears.

test_yourself_again(self)
When user presses “Test again” button,
This function increases the retry_count by 1.
The canvas then shows another random pair of flashcards through the random.choice function in start_flashcards(self), and the “Reveal” button is shown again.

remove_from_list(self)
When user presses the "Correct" button,
This removes the current_card dictionary from the list, test_dict, and appends the current_card dictionary to an initially empty list, learnt_words. The canvas then shows another current_card dictionary based on the random.choice function and the “Reveal” button is shown again. This process continues until there are no more dictionary elements in test_dict.

If the number of elements in the flashcard deck, test_dict is 0, an IndexError will occur, as the random.choice function is unable to find an Index in the now empty list, test_dict. After taking into account of this exception, the timer is stopped, where the time taken to finish the deck is calculated. The gameplay window is closed and will proceed to the Report Screen.

Report_Screen

_ _ init _ _ (self)
Initialises a window, which has a canvas, showing an animation and a corresponding text.
If retry count < 3, a happy face (**display_happy**) turtle animation is shown, with 3 stars.
If 3 <= retry count < 7, a neutral face (**display_neutral**) turtle animation is shown, with 2 stars.
If retry count > 7, a sad face(**display_sad**) turtle animation is shown, with 1 star.
All face animations are made with turtle animation, using the same base face (**face_base**) with difference in the mouth animation, along with animation of the corresponding number of stars (**display_stars**)

The animation is shown for 2 seconds before the canvas is deleted.
The report screen also shows statistics of user progress by their 1) retry count, 2) time taken to complete, and 3) summary of words learnt, and the exit button.

exit_to_Main_Menu(self)
When user presses “Exit” button,
Closes Report screen and go to the Main menu, updates all the global variables except for player_name to their initial values

simplify_list(self,words)
The function takes in a list of dictionaries of learnt_words. Iterating through the dictionary elements of the learnt_words, an empty list is appended with the values of the “Chinese” and “English” keys in pairs.

The elements in this list are inserted into the box (**tk.Listbox)**, showing the summary of words learnt.
For example,
Input: [{“Chinese”:”Ping Guo”,”English”:”Apple”}]
Output: Ping Guo, Apple

Credits_Screen

_ _init_ _(self)
Initializes a window with texts of our names and Student IDs in our project.