The Word Homophone Dictionary

One common use of a python dictionary is to hold related information. You are going to write a program which allows the user to add, update, remove, search and display a list of homophones (words that sound the same). The list contains only a original word and its homophone. The original word is the key for the dictionary entry and the homophone is the value. The list is only kept while the program is running; when the user exits, the list disappears.

To do:

1. Write the IPO diagrams for the **Word Homophone dictionary** application in the table provided here. You must create a separate IPO diagram for each piece of functionality (add, update, remove, search and display).

Program Input	Process	Program Output
Add Homophone	1. Read original word 2. Read homophone 3. Add the original word to the dictionary 4. Assign the original word as key 5. Add the homophone to the dictionary 6. Assign homophone as the value 7. Print msg that it was successful 8. Stop	New key and value homophone pair
Update homophone	1. Read original word 2. Check if it exists 3. If it does, print to type the homophone to update 4. If it does not exist, print an error statement telling user that the word does not exist 5. Read new homophone 6. Overwrite the current homophone with the new homophone 7. Print that the update is successful to the user 8. Stop	Updated Homophone
Remove homophone	1. Read original word 2. Check if it exists 3. Print an error that the word does not exist, if the inputted word is not in the dictionary 4. If it has one, the original word and the homophone is then deleted 5. Print a msg that tell the user that the original word and homophone is deleted 6. return	Key and value homophone pair deleted
Search	1. Read original word	The search

Homophone	2. Check if the original word exists 3. Print and error statement if it does not exist 4. If it does, print the original word and the homophone in a nice format 5. Return	original word and homophone are displayed
Display homophone	1. Check the dictionary 2. Check if there is any original word and homophone 3. Print an error statement that tell there is no homophone words, if the dictionary is empty 4. If there is any original word and homophone inside the dictionary, print the entire content of the dictionary in a nice format 5. Return	A print of the entire homophone list from the dictionary
Display Menu	1. Print a triple quote string showing the menu 2. return	Menu display

2. Complete the **Expected output** columns in test cases below.

Test Plan for Homophones Program			
Test Case		***Input Data***	***Expected output***
1.	Add a new homophone	Name = 'to' Homophone = 'two'	A new homophone is added which is 'dict= {'to': 'two'}'
2.	Update a homophone	Original Word = 'their' Homophone = "there"	The 'their' homophone is updated to be 'dict= {'their': 'there'}
3.	Remove a homophone	Original Word = 'to'	The original word 'to' and its homophone should be deleted
4.	Search for a homophone	Original Word = 'they're'	It would display an error saying the word 'they're' does not exist. 'their' on the other hand, will be display in the format I put.
5.	Print out the entire Homophone list		Display the entire content of the dictionary in a nice format.
6.	Update a homophone, but original word does not exist	Original word = 'too'	Display an error that tell user that the original word does not exist, and return to menu.

Test Plan for Homophones Program			
Test Case		***Input Data***	***Expected output***
7.	Remove a homophone, but the homophone does not exist from the start.	Original Word = 'too'	Display an error that tell user that the original word does not exist, and return to menu.
8.	Search Homophone, but the homophone does not exist	Original Word: 'flour'	Display an error that tell user that the original word does not exist, and return to menu.
9.	In the menu, the user input a non-numeric value	Input: 'abcd'	Display an error that only Positive Integer is valid, and loop back for the user to input again.
10.	In the menu, the user enter an integer, but it's over 6 or under 1	Input: 7 or Input: 0	Display an error telling user only integer from 1-6 is valid, and ask the user again for what function to use.

The program works like this:

1. Prompt the user with a menu with options to add, update, delete, search, display the list or exit the program.
2. If the user chooses to add, prompt for the Original Word and Homophone. Then add the value to a python dictionary object with the key as the Original Word and the Homophone as the value {'to': 'two'}
3. If the user chooses to update, prompt for the Original Word and search for the Original Word.
    a. If the Original Word is found, prompt for the Homophone and update the value in the dictionary
    b. If the Original Word is not found, display a message and return to menu
4. If the user chooses to delete, prompt for the Original Word and search for the Original Word.
    a. If the Original Word is found, remove the key-value pair from the dictionary
    b. If the Original Word is not found, display a message and return to menu
5. If the user chooses to search, prompt for the Original Word and search for the Original Word.
    a. If the Original Word is found, display it nicely formatted along with the homophone
    b. If the Original Word is not found, display a message and return to menu
6. If the user chooses to display the Homophone list, display all entries in the list formatted nicely displaying BOTH the key (Original Word) and the value (Homophone).
7. If the user chooses to exit, exit the program.
8. An example of the Console window for the program is shown here.

```
Welcome to your Homophone list. You may:
1. Add a new homophone
2. Update a homophone
3. Remove a homophone
4. Search for a homophone
5. Display your homophones
```

```
6. Exit the program
Please enter your choice: 1
Original word: to
Homophone: two
New entry added
You may:
1. Add a new homophone
2. Update a homophone
3. Remove a homophone
4. Search for a homophone
5. Display your homophones
6. Exit the program
Please enter your choice: 2

Homophone's name: their

No homophone named their found

You may:
1. Add a new homophone
2. Update a homophone
3. Remove a homophone
4. Search for a homophone
5. Display your homophones
6. Exit the program
Please enter your choice: 2

Homophone's name: there

New homophone for there: they're

Etc.etc.etc.
```

9. Test the program using the test cases above.

10. Some extra things to do to challenge yourself:
    a. Allow the user the option of adding more than one value for a given key. Remember, the key must be unique, so the multiple values would need to be added as a list of values on the same key. So: {"to": ["too", "two"], "beat": "beet"}
    b. Confirm with the user before removing a homophone entry (when deleting a record, it is always good to confirm with the user).
    c. Use colours for the display to make the user experience better.
    d. Personalize error and status messages with the names entered.
    e. Make the search be able to search on partial names (use startswith)
    f. Make the search NOT case sensitive so BOB finds bob or Bob or BOB