The Movie Genre 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 movies and their genre. The list contains only a movie’s name and its genre. The movie’s name is the key for the dictionary entry and the genre is the value. The dictionary is only kept while the program is running; when the user exits, the list disappears.

**To do:**

1. Write the IPO diagrams for the **Movie Genre 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).

* 1. Adding a movie

|  |  |  |
| --- | --- | --- |
| Program Input | Process | Program Output |
| Movie name, genre | If movie name does not exist in dictionary, then add the movie name as the key and genre as the value to the dictionary. | Confirmation: "Movie added successfully!” OR Error message: "This movie already exists." |

* Updating a movie

|  |  |  |
| --- | --- | --- |
| Program Input | Process | Program Output |
| Movie name, new genre(s) | * If the movie name exists in the dictionary (case-insensitive and partial match check), update its genre. * If the movie name does not exist, display an error message. | * Confirmation: "Movie genre for [movie\_name] updated to [genre(s)]!" * Error message: "[movie\_name] is not found in the movie dictionary." |

* Deleting a movie

|  |  |  |
| --- | --- | --- |
| Program Input | Process | Program Output |
| Movie name | * If the movie name exists in the dictionary (case-insensitive and partial match check), ask for confirmation, then remove the movie entry. * If the movie name does not exist, display an error message. | * Confirmation: "Movie [movie\_name] deleted successfully!" * Error message: "[movie\_name] is not found in the movie dictionary." |

* Searching for a Movie’s Genre

|  |  |  |
| --- | --- | --- |
| Program Input | Process | Program Output |
| Movie name | * If the movie name exists in the dictionary (case-insensitive and partial match check), fetch its genre. * If the movie name does not exist, display an error message. | * Display the genre of the movie. * Error message: "[movie\_name] is not found in the movie dictionary." |

* Display the List of All Movies and Their Genres

|  |  |  |
| --- | --- | --- |
| Program Input | Process | Program Output |
| Movie name, genre | Fetch all movie names (keys) and their corresponding genres (values) from the dictionary. | Display the nicely formatted list of movies and their genres |

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

| **Test Plan for Movie Genre Program** | | | |
| --- | --- | --- | --- |
| ***Test Case*** | | ***Input Data*** | ***Expected output*** | |
|  | Add a new movie | Title = ‘Wall-E’  Genre = ‘Animated’ | "Movie added successfully." Or “This movie already exists."” | |
|  | Update a movie | Title = ‘Life of Brian’  Genre = ‘Comedy’ | "Movie genre for Life of Brian updated to Comedy’" Or “Life of Brian is not found in the movie dictionary” | |
|  | Remove a movie | Title = ‘Wall-E’ | "Movie Wall-E removed successfully." Or “There's no movie called Wall-E in our list” | |
|  | Search for a movie | Title = ‘Holy Grail’ | "The genre for ‘Holy Grail’ is: “Comedy” | |
|  | Print out the entire movie list |  | Wall-E - Animated  Life of Brian - Comedy  The Matrix - Sci-Fi  Toy Story - Animated … | |

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 a title and genre. Then add the value to a python dictionary object with the key as the title and the genre as the value {“Star Wars’: ‘Sci-Fi’}
3. If the user chooses to update, prompt for a title and search for the title.
   1. If the title is found, prompt for the genre and update the value in the dictionary
   2. If the title is not found, display a message and return to menu
4. If the user chooses to delete, prompt for a title and search for the title.
   1. If the title is found, remove the key-value pair from the dictionary
   2. If the title is not found, display a message and return to menu
5. If the user chooses to search, prompt for a title and search for the title.
   1. If the title is found, display it and the genre nicely formatted
   2. If the title is not found, display a message and return to menu
6. If the user chooses to display the list, display all entries, formatted nicely displaying BOTH the key (title) and the value (genre).
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 movie list. You may:

1. Add a new movie
2. Update a movie
3. Remove a movie
4. Search for a movie
5. Display your list
6. Exit the program

Please enter your choice: 1

Movie’s name: Holy Grail

Movie’s phone number: Comedy

New entry added

You may:

1. Add a new movie
2. Update a movie
3. Remove a movie
4. Search for a movie
5. Display your list
6. Exit the program

Please enter your choice: 2

Movie’s name: Life of Brian

No movie named Life of Brian found

You may:

1. Add a new movie
2. Update a movie
3. Remove a movie
4. Search for a movie
5. Display your list
6. Exit the program

Please enter your choice: 2

Movie’s name: Holy Grail

New number for bob: Drama

Etc.etc.etc.

1. Test the program using the test cases above.
2. Some extra things to do to challenge yourself:
   1. 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: {“bob”: [“1234567890”, “9998887777”], “frank”: “2342434566”}
   2. Confirm with the user before removing a phonebook entry (when deleting a record, it is always good to confirm with the user).
   3. Use colours for the display to make the user experience better.
   4. Personalize error and status messages with the names entered.
   5. Make the search be able to search on partial names (use startswith)
   6. Make the search NOT case sensitive so BOB finds bob or Bob or BOB