Final Project Report

GDG, 2CC

Vellore Institute of Technology

By,

K.N.Sri Dharmasastha
16BCE0956

Contents

Abstract

Create a GUI game in which whenever game starts , the game gives 10 randomly selected TV show names and jumbled , the user has to predict the correct name of the TV show and accordingly how many he guesses correct he gets 1 point. Remember the jumbled word should be different every time and also jumbled in a different way.

Introduction

Jumble is a word puzzle with a clue, a drawing illustrating the clue, and a set of words, each of which is "jumbled" by scrambling its letters. A solver reconstructs the words, and then arranges letters at marked positions in the words to spell the answer phrase to the clue. The clue and illustration always provide hints about the answer phrase. The answer phrase frequently uses a homophone or pun. But the exact word is taken as the correct answer and not he homophone or pun of the given jumbled word.

I use a python toolkit PyQt to create the GUI. You can use other toolkits for creating the GUI such as Tkinter, wxPython etc. PyQt is the easiest among all the toolkits I've tried working with.

Methodology

Building back-end logic:

First – we use the python library, **random** to pick a random TV show title from the 30 titles that are stored in a list.

Second - once the random word is picked then we jumble the letters using a built-in function called **jumble**. This function produces a new jumbled word every time you use it.

Third, display the question and collect the answer - I print the jumbled word previously created and display and ask the player to input what they think the answer is.

Fourth, validating the answer and keeping score – now that the answer is collected, we match it to the originally picked word and see if they are the same word. If they are, the player's score is increased by one. (The score will be initialized ahead of this use)

Fifth, looping 10 times – once all the above functions are running without errors, all the first four steps are put on loop for 10 times to keep score out of 10.

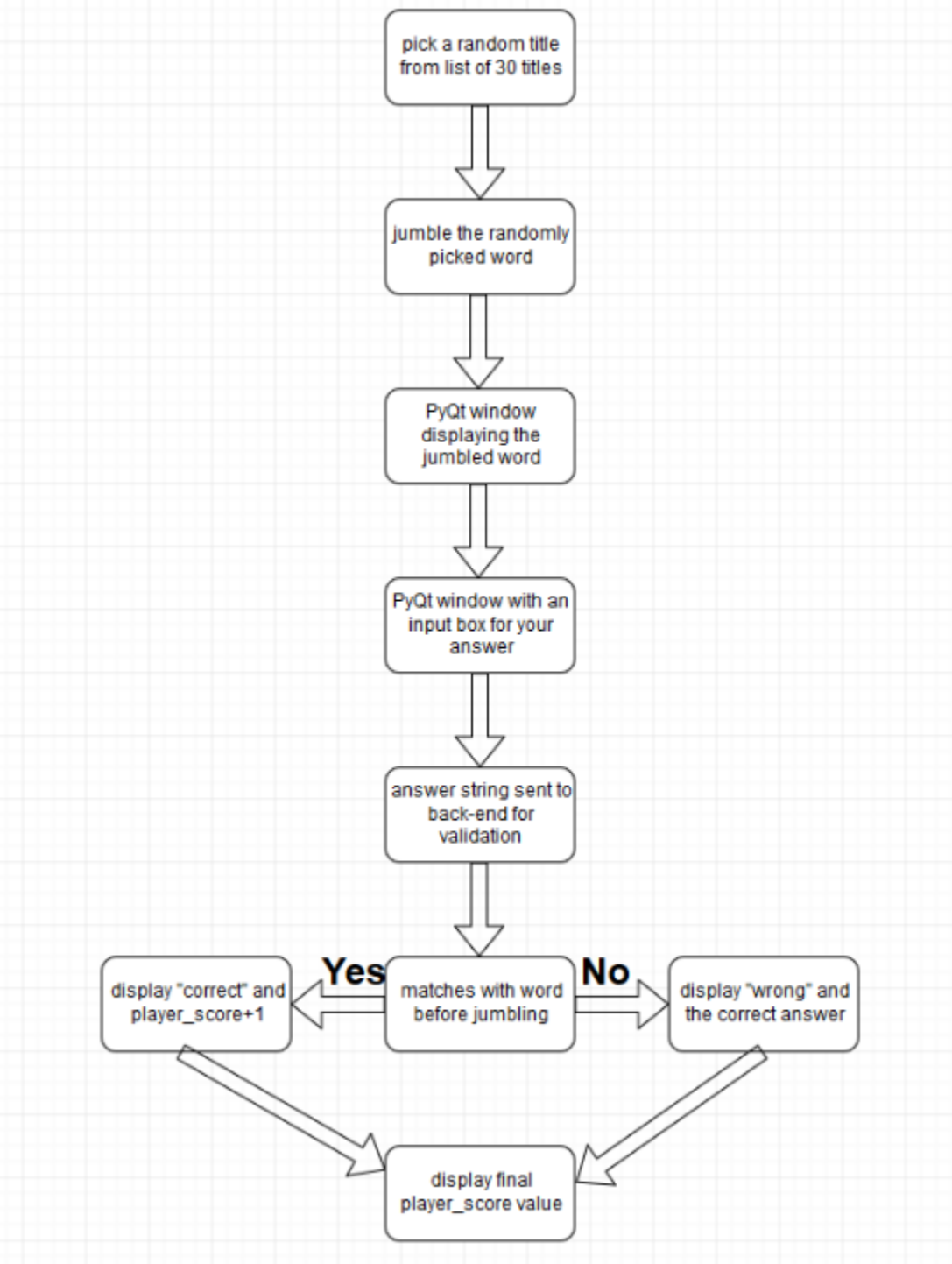
Building front-end:

I used **PyQt**, a python toolkit to create and manage GUI, to create the interface required to run the game on. All the important libraries are imported before starting as always.

Instead of putting all 10 questions on one window, I made it such a way that each question and its answer-taking windows up sequentially one after the other. This can be done by using the **messagebox** concept which is a one-line code which pops up a new window at every call.

For every answer, the string will be passed to back-end and validated at step four of it, and adds one to the player's score if it's correct and display the correct answer if it's wrong.

Block Diagram

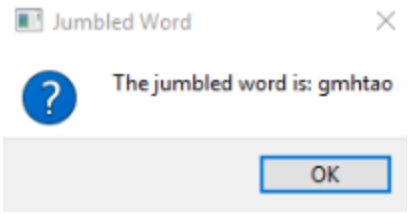


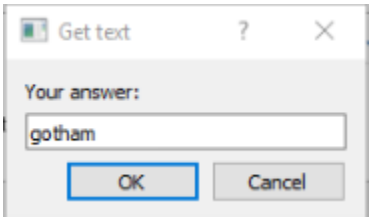
Execution

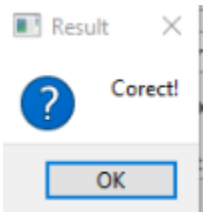
Back-end:

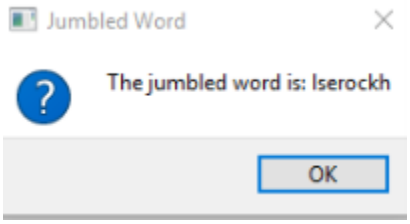
```
Jumbled word is : vigisnk
What's  your guess? vikings
Correct!
Jumbled word is : liurcfe
What's  your guess? lucifer
Correct!
Jumbled word is : aeowdodd
What's  your guess? s
Sory, the correct word is:  deadwood
Jumbled word is : cefirul
What's  your guess? lucifer
Correct!
Jumbled word is : gaorf
What's  your guess? fargo
Correct!
Jumbled word is : dwooedda
What's  your guess? deadwood
Correct!
Jumbled word is : rgafo
What's  your guess? fargo
Correct!
Jumbled word is : elsohkcr
What's  your guess? d
Sory, the correct word is:  sherlock
Jumbled word is : raerldive
What's  your guess? riverdale
Correct!
Jumbled word is : krsttera
What's  your guess? s
Sory, the correct word is:  startrek
Your score is:  7
```

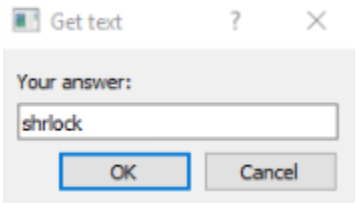
Front-end:

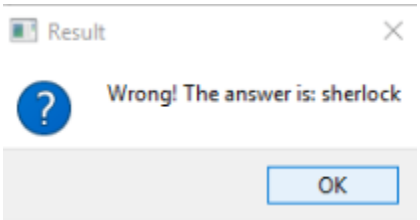

Jumbled Word
The jumbled word is: gmhtao
OK

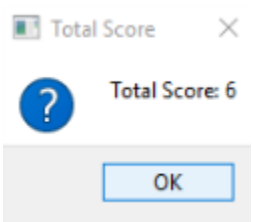

Get text
Your answer:
gotham
OK
Cancel


Result
Corect!
OK


Jumbled Word
The jumbled word is: lserockh
OK


Get text
Your answer:
shrlock
OK
Cancel


Result
Wrong! The answer is: sherlock
OK


Total Score
Total Score: 6
OK

Difficulties

The back-end was the easiest due the simple logic requirement. The difficulty was in connecting the back-end to front end. The steps to follow to pass the arguments from front-end to back-end was challenging. Depending on the PyQt version, the function syntax changes and also the proper GUI interface to choose to display the questions and collect the answers. This matters as the way of passing arguments between loops of back-end and front-end differs for different types of front-end interface chosen.

Conclusion

Creating a jumbled word game purely on python using PyQt, even at how easy to understand it might be, made me appreciate how simple and more appropriate the GUI dedicated tools are for creating something like this.

This was certainly still a great and new learning experience.

References

- https://pythonspot.com/gui/
- http://www.greenteapress.com/thinkpython/swampy/install.html