Final Report

Overview:

For this project, I had decided to make a program that depicts the game “Hangman”. I chose the gaming theme to create a more interactive and a fun way to learn few functions and methods in Python programming.

To create the Hangman, I have defined set of words to start with in three different text files. These text files are read using the readlines() function. This function creates a list of words for every word, separated by “\n” i.e. new line character. The list is then used in the program to extract a word to play the game of Hangman. We then select a word, using a function which selects a random word from the list. After this, with the number of turns calculated, the code runs, and the game continues. If the guess by the user is correct, the code runs. If the guess is wrong, the number of turns decrease and tin turn if the number of turns become zero, Hangman comes!

Challenges Faced:

While writing the code, the main challenge faced was iterating the code for correct letter guesses. After several trials, the code was made to iterate till the time the user inputs the correct letters and wins. As soon as the user either wins or loses, the **break** statement is used to get out of the main **while** loop.

Initially, I could not create a fail-safe sort of situation when the user inputs wrong category name. Initially, this was tried using if-else statements, but the code could not iterate if the input was wrong. This was again tried using for loop, but the condition for the for loop could not be defined. At last, the situation was handled using **while** loop. This allowed me to iterate a set of code repeatedly for wrong input by the user.

Understanding the Code:

1. The code starts with importing three packages, “random”, “time” and “re”.
    - **random**: This package is used to run the code, “random.choice()”. In this program, the code helps to select a random index from the list and thus is useful in selecting a random string from the list, rather than just pre-defining the word for the game.
    - **time**: This package is used to create a delay in the code, if required. In this program, the delay is used to make the code interesting. With the delay, the program becomes interactive and interesting.
    - **re**: This package is for “regular expression”. I have used this package to use “re.sub” which substitutes all the special characters with a blank space, “”.

2. Next, we open the text files created using **open(“”,’r’)**, which only allows us to read the files. The reason for using this is just to read the files for accessing the words in the text files, which in turn are used in the code to play the game.
3. We then create a list **options[ ]**, which consists of the categories from which the user while play the game.

4. After this, we create a **While** loop to understand the input given by the user. Before this, we display the categories available to the user and ask the user to choose one from them. For example, let's assume the following example:
   The user is given categories: **Movies**, **Brands** and **Countries**.
   The user is then requested to choose from the above categories.
   If the user inputs a wrong choice, like "**Moviess**" or "**moovies**", the user is asked to input again using this while loop. Thus, unless user provides the way of input, the loop runs and asks the user to input correctly.
5. Also, in this **While** loop, we change the input to lowercase and capitalize the same in order to match with the list **options[ ]**. Thus, this gives me the ease to check whether the input is same or not. Thus, by comparison, we can check the input from the user.
6. Now that we have checked the input, we now choose the list. As soon as the list is selected, we choose a random word from the list using the "**random.choice()**" function.
7. Since the word will have special characters like "-", ":", ";", ".", "/" or numbers and many more, we need to change the word. This is done using the regular expression using code, "**re.sub()**". The argument given is **re.sub(r'[^a-zA-Z]', "", word)**. This removes everything except letters with a blank character.
8. We create a variable "turns" to initialize the number of turns for the game. Using a for loop, we create a output that is shown as:

```
_ _ _ _ _ _ _ _ _ _ _ _ _ _ .    Guess characters:
```

9. The for loop is used to iterate the code unless either the user has won the game and guessed all the characters or has lost the game. The loop breaks if either of the above conditions are met.
10. The loop runs from 1 to (turns-1). Also, there is another variable used here called fail. This variable is used to understand whether the player has won or lost. If the user inputs a character that is not present inside the word, then the fail variable is incremented by 1. If at the end, fail is 0, then the player has won. Else, it means the player had inputted a wrong character, thus iterating till the user inputs correct character.
11. Now, as soon as the user inputs wrong character and the fail variable has been incremented by 1, the program displays,
    "**You have", + turns, "more guesses. Keep trying!"**.
    This then iterates the code again to input the correct character.
12. At the end, if the "turns" variable is 0, then it means you have lost the game. The code displays a depiction of Hangman.

Thus, in this way, the program runs to help you to play the game of Hangman.

References:

1. Data for **Movies**: http://blog.joelberghoff.com/wp-content/uploads/2015/12/1990-2015-movie-titles.txt
2. Data for **Countries**: https://gist.github.com/kalinchernev/486393efcca01623b18d
3. Data for **Brands**: http://brandirectory.com/league_tables/table/global-500-2014

The above data has been refined using Microsoft Excel and then stored in respective text files.

GitHub Publish Link: https://github.com/vigilante-snipe/Final_Project