Spaceman

Description

Spaceman is a guessing game. There is a mystery word which the user tries to guess one letter at a time. A placeholder is initially shown, with the number of blanks corresponding to the number of letters in the word. If the letter is in the mystery word, the position(s) of the letter(s) are revealed in the placeholders. Guess the word before you run out of guesses!

Users win if they can guess the mystery word before the spaceman is drawn. The spaceman is made up of seven parts, and each part is drawn for each incorrect guess. If all seven parts get drawn before the user guesses the word, then they lose.

You will be given starter code to work off of. **Make sure you begin this project using this code!**

Check out an example demo of winning the game
Check out an example demo of losing the game

Learning Outcomes

By completing this project, you should be able to...

1. Build a game loop that is aware of wins and losses
2. Use control flow to decide what action your program should take based on given information
3. Use at least one of the following collection types to store data: lists, tuples, dictionary

Schedule

From the day this project is assigned, you will have **1 week** to complete this project. A sample daily outline is provided to assist you in planning your project.

Important note: "Day 01" refers to the first calendar day of the project being assigned, and subsequent days will follow this reference:

- Day 01: Project assigned, read the project spec, ask any clarifying questions you may have, begin designing how you would implement the project
- Day 03: Build out guessing functionality, scoring system
- Day 05: Establish how to traverse a word to find all instances of a letter
- Day 06: Tie all the pieces of the project together, test with short and long words, and test with words that have repeat letters (like "apple")
- Day 07: Fix any remaining bugs, make sure your code has helpful comments, and submit the project!

Requirements

Your project must meet the below requirements in order to be considered passing:

1. User must be able to enter letters to guess
2. The user must get accurate feedback on if they guessed a correct letter or an incorrect letter
3. The user is always prompted to guess a letter until they win or lose
4. The game must use the provided list of words as its source.
5. User is allowed seven (7) incorrect guesses, and they should be told how many guesses they have left after each incorrect guess
6. After guessing a letter, the user must be told the following:
    a. Correct guess: the placeholder text with the correct letter filled in.
        i. If the word is “dog” and they guess “g” as their first guess, they should see _ _ g
    b. Incorrect guess: a message telling them their guess is incorrect, and then the number of guesses they have remaining
7. If a guessed letter appears multiple times in the word, that guessed letter should appear in all valid blanks
    a. *Correct* Example (word is “apple”): p ➔ _ pp_ _
    b. *Incorrect* Example (word is “apple”): p ➔ _p_ _
8. If a user successfully guesses all the letters, the game ends, and the user is shown a message notifying them that they won
9. If a user guesses incorrectly seven (7) times, the game ends, and the user is shown a message notifying them that they lost

Stretch Requirements/Challenges (Optional)

- Alert the user if they guessed a letter they already guessed, and don’t have it count as an incorrect guess
- Users can only guess individual letters at a time. If they guess anything other than an individual letter, they should be prompted and told to input only one letter
- Prompt the user to play again after a game ends. If they say yes, then start a new game.
- Change the number of incorrect guesses allowed to match the length of the mystery word
- Show the user the mystery word when they lose
- Use ASCII art to draw the spaceman with each incorrect guess
- Sinister Spaceman: After the user guesses a correct letter, change the mystery word to be a *new mystery word* that is the same word length and uses the same correctly guessed letters
    - Example: mystery word is “car”, user guesses “a”, the user is shown “_a_”, but the word is now changed to “bar”
        - User guesses “c”, which results in an incorrect guess.

        - They then guess “b”, the user is shown “ba_”, but the word is now changed to “bat”
        - User guesses “r”, which results in an incorrect guess.
        - User guesses “t”, which results in a correct guess, and the user wins

Rubric

Link to project rubric placed here

Commit Requirements

Due to the length of this project, you are required to have a minimum of 5 commits, and they must take place throughout the entirety of the project timeline.

- **Good Example:** 5+ commits throughout the length of the project, looking for a healthy spattering of commits each week (such as 3-5 per day).
- **Bad Example:** 5 commits on one day during the course and no others. Students who do this will be at severe risk of not passing the class.
- **Unacceptable Example:** 2 commits the day before a project is due. Students who do this should not expect to pass the class.

Why are we doing this?

We want to encourage best practices that you will see working as a professional software engineer. Breaking up a project by doing a large amount of commits helps engineers in the following ways:

- It's much easier to retrace your steps if you break your project/product/code up into smaller pieces
- It helps with being able to comprehend the larger problem, and also will help with your debugging (i.e. finding exactly when you pushed that piece of broken code)
- It allows for a more streamlined, iterative communication in your team, as it's much easier to hand off a small change to someone (updating a function) than a huge one (changed the architecture of the project)

Through this requirement, we hope to encourage you to think about projects with an iterative, modular mindset. Doing so will allow you to break projects down into smaller milestones that come together to make your fully-realized solution.

Resources

Additional resources that will help with this project, or that can be used as reference

- Starter Code
- List of words to use for the game