Game Show

psp-07-02

1 Overview

Below is the source code for a simple trivia game. In this lab you will used similar code to implement a full featured game that simulates a game show.

```
questions = [
              {"question": "Who␣played␣the␣first␣Batman?",
               "answers": [
                           "George␣Clooney",
                           "Adam␣West",
                           "Louis␣G␣Wilson",
                          "Christian␣Bale"],
               "correct": "3"},
              {"question": "What's␣the␣answer␣to␣life ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣the␣universe␣and␣everything?",
               "answers": [
                           "Cheese",
                           "42",
                           "Tacos",
                           "Baseball"],
               "correct": "2"}]

print("Welcoem␣to␣our␣CS126␣Trivia␣Game")
print("Win␣MILLIONS!!\n")

for question in questions:
   print(question["question"])
   for i, choice in enumerate(question["answers"]):
     print(str(i + 1) + ".␣" + choice)
   answer = input("Choose␣an␣answer␣1-4:")
   if answer == question["correct"]:
     print("Good␣job.␣␣Confetti␣falls␣in␣the␣background.")
   else:       print("Not␣correct.␣␣The␣audience␣boos.")
```

2 Learning Outcomes

By the end of this project students should be able to:

- read and write programs that use user input;
- read and write programs that use iteration, for-loops, while-loops;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before coming to lab:

- Read Problem Space Chapter 7: Iteration.
- Brainstorm at least one theme for a game show that you find interesting. Try and make your theme as unique and interesting as possible. Write down a short paragraph explaining your idea.
- Research the random.shuffle() method on the python webstie and write down how you can use this method for this lab. Give an example.
- Be prepared to show your work to the lab aide at the beginning of lab.

4 Lab Instructions

Do this part in lab:

Your game has the following requirements:

1. Your game show needs a theme. Consider games like "Family Feud", "So you want to be a millionaire?", or "Are you smarter than a fifth grader?" The hook to each show is some kind of theme that makes the relatively simple trivia core seem more exciting. Simply adding some extra print statements can do a lot to add ambiance to your game. Edit the welcome message in the example above to reflect your theme.
2. Your game needs at least 10 questions. All of your questions should not have the same number of possible answers (i.e., some may have two answer to choose from while other may have four). Think about how your theme might reflect on the questions to the game. Don't use the two examples given above.
3. The questions should be given in random order and no question should be given twice.

4. Add a "main menu" to your game where the user can (1) play the game, (2) view the game credits (i.e., who created the game), or (3) quit (be sure to thank the user for playing before exiting). Each menu option should be implemented in a function. After the user finishes answering all questions or viewing the credits, the main menu should appear again.

5. Update the example code so that when users pick an invalid letter or response (not a wrong answer, but an invalid selection such as 3 for a questions with only 2 possible answers), they are prompted to enter a valid answer before moving on.

6. Keep track of the users score and print the current score after each question. For example, "Your current score is: 2 out of 4."

Extra Credit:

1. Add a "View high score table" to your main menu. The high score table only needs to exist for one execution of your module (i.e., scores don't have to be saved to disk). Users should be prompted for their name after they finish answering all of the questions. If the user attempts to view the high score table when it is empty, you must let them know it is empty. Otherwise, the high score table (including names and scores) should be output from highest to lowest (ties may be printed in any order). After viewing high scores, the main menu should appear.

When you have completed the lab run pep8 against your code until all formatting errors have been corrected and your code is PEP 8 compliant. See the Getting Started lab if you need instructions on running the program, or the pep8 documentation found here.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as a .py file)
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or 1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indcates that your partner met the class expectations of them. Rating your partner at -1 means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. If a student recieves three ratings of -1 they must attend a mandatory meeting with the instructor to dicuss the situation, and recieving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

Colophon

This project was developed by Dr. James Dean Palmer of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.