CSE 231, Fall 2007

Programming Project 04

Assignment Overview

This assignment is worth 30 points (3.0% of the course grade) and must be completed and turned in before 11:59pm on Monday, September 24, 2007. The purpose of this project is to familiarize you with the use of Boolean logic, branching statements and loops. For this assignment, you will create a program that plays the game "Rock, Paper, Scissors."

Background

Rock, Paper, Scissors (also known by several other names, see http://en.wikipedia.org/wiki/Rock_paper_scissors) is an extremely popular hand game most often played by children. Often, it is used as a method of selection similar to flipping a coin or throwing dice to randomly select a person for some purpose. Of course, this game is not truly random since a skilled player can often recognize and exploit the non-random behavior of an opponent; for instance, if you notice that your opponent chooses Paper most frequently, you may choose Scissors (which beats Paper) most often in an effort to win.

Rules of the Game:

The objective of Rock, Paper, Scissors is to defeat your opponent by selecting a weapon that defeats their choice under the following rules:

- Rock smashes (or blunts) Scissors, so Rock wins
- Scissors cut Paper, so Scissors win
- Paper covers Rock, so Paper wins
- If players choose the same weapon, neither win and the game is played again

Program Specifications

This project requires you to use:

- raw_input to prompt the user
- print to print results
- at least one branching mechanism (if statement)
- at least one loop (while loop)
- Boolean logic

Your program will allow a human user to play Rock, Paper, Scissors with the computer. Each round of the game will have the following structure:

- The program will choose a weapon (Rock, Paper, Scissors), but its choice will not be displayed until later so the user doesn't see it.
- The program will announce the beginning of the round and ask the user for his/her weapon choice

- The two weapons will be compared to determine the winner (or a tie) and the results will be displayed by the program
- The next round will begin, and the game will continue until the user chooses to quit
- The computer will keep score and print the score when the game ends

The computer should select the weapon most likely to beat the user, based on the user's previous choice of weapons. For instance, if the user has selected Paper 3 times but Rock and Scissors only 1 time each, the computer should choose Scissors as the weapon most likely to beat Paper, which is the user's most frequent choice so far. To accomplish this, **your program must keep track of how often the user chooses each weapon.** Note that you **do not** need to remember the order in which the weapons were used. Instead, you simply need to keep a count of how many times the user has selected each weapon (Rock, Paper or Scissors). Your program should then use this playing history (the count of how often each weapon has been selected by the user) to determine if the user currently has a preferred weapon; if so, the computer should select the weapon most likely to beat the user's preferred weapon. **During rounds when the user does not have a single preferred weapon, the computer may select any weapon.** For instance, if the user has selected Rock and Paper 3 times each and Scissors only 1 time, or if the user has selected each of the weapons an equal number of times, then there is no single weapon that has been used most frequently by the user; in this case the computer may select any of the weapons.

At the beginning of the game, the user should be prompted for his/her input. The valid choices for input are:

- R or r (Rock)
- P or p (Paper)
- S or s (Scissors)
- Q or q (Quit)

At the beginning of each round your program should ask the user for an input. If the user inputs something other than r, R, p, P, s, S, q or Q, the program should detect the invalid entry and ask the user to make another choice.

Your program should remember the game history (whether the user wins, the computer wins, or the round is tied).

At the end of the game (when the user chooses 'q' or 'Q'), your program should display the following:

- The number of rounds the computer has won
- The number of rounds the user has won
- The number of rounds that ended in a tie
- The number of times the user selected each weapon (Rock, Paper, Scissors)

A sample executable file (proj04.pyc) is provided for you to use to explore how your final program should function.

Deliverables
You must use handin to turn in a file called **proj04.py** – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj04.py file to your H drive as a backup.

Assignment Notes
1. `raw_input` should be used for prompting. It returns a string containing the user's choice.
2. There is a string method called lower. It converts the string to all characters to lower case. Thus:
   `'ABC'.lower()` ⇒ `'abc'`
   This might prove helpful for user input checking.

Getting Started
1. Do all the standard startup things. Create a new file called proj04.py. Put your comments in at the top, save it.
2. Now you need to break the problem down into parts. Read the description and identify the subtasks that need to be solved. For example, one subtask would be to get proper user input. Mark in the empty program, using comments, all the subtasks you need to solve.
3. Now address one subtask, getting user input. Do this in stages as well. Can you:
   a. Prompt for and get a choice (a string) from the user?
   b. Once you can do that, can you repeatedly prompt for a character until you see a 'q' or 'Q' for quit?
   c. Once you can do that, can you check for "legal" character responses from the user, and print an error message when an illegal response is given?
   d. Next, can you check for legal responses that are in both upper and lower case?

   Once you can do all that, move on to the next subtask.
4. Remember, save the file and run it all the time! It will make debugging the program easier.