Mini-project description — Rock-paper-scissors-lizard-Spock

Rock-paper-scissors is a hand game that is played by two people. The players count to three in unison and simultaneously "throw" one of three hand signals that correspond to rock, paper or scissors. The winner is determined by the rules:

- Rock smashes scissors
- Scissors cuts paper
- Paper covers rock

Rock-paper-scissors is a surprisingly popular game that many people play seriously (see the Wikipedia article (http://en.wikipedia.org/wiki/Rock_paper_scissors) for details). Due to the fact that a tie happens around 1/3 of the time, several variants of Rock-Paper-Scissors exist that include more choices to make ties less likely.

Rock-paper-scissors-lizard-Spock (RPSLS) is a variant of Rock-paper-scissors that allows five choices. Each choice wins against two other choices, loses against two other choices and ties against itself. Much of RPSLS's popularity is that it has been featured in 3 episodes of the TV series "The Big Bang Theory". The Wikipedia entry (http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock) for RPSLS gives the complete description of the details of the game.

In our first mini-project, we will build a Python function `rpsls(name)` that takes as input the string `name`, which is one of `"rock"`, `"paper"`, `"scissors"`, `"lizard"`, or `"Spock"`. The function then simulates playing a round of Rock-paper-scissors-lizard-Spock by generating its own random choice from these alternatives and then determining the winner using a simple rule that we will next describe.

While Rock-paper-scissor-lizard-Spock has a set of ten rules that logically determine who wins a round of RPSLS, coding up these rules would require a large number (5x5=25) of `if`/`elif`/`else` clauses in your mini-project code. A simpler method for determining the winner is to assign each of the five choices a number:

- 0 — rock
- 1 — Spock
- 2 — paper
- 3 — lizard
- 4 — scissors

In this expanded list, each choice wins against the preceding two choices and loses against the following two choices (if rock and scissors are thought of as being adjacent using modular arithmetic).

In all of the mini-projects for this class, we will provide a walk through of the steps involved in building your project to aid its development. A template for your mini-project is **available here** (http://www.codeskulptor.org/#examples-rpsls_template.py). Please work from this template.

Mini-project development process

1. Build a helper function `name_to_number(name)` that converts the string `name` into a number between 0 and 4 as described above. This function should use a sequence of `if`/`elif`/`else` clauses. You can use conditions of the form `name == 'paper'`, etc. to distinguish the cases. To make debugging your code easier, we suggest including a final `else` clause that catches cases when `name` does not match any of the five correct input strings and prints an appropriate error message. You can test your implementation of

`name_to_number()` using this name_to_number testing template (http://www.codeskulptor.org/#examples-name_to_number_template.py). (Also available in the Code Clinic tips thread).

2. Next, you should build a second helper function `number_to_name(number)` that converts a number in the range 0 to 4 into its corresponding name as a string. Again, we suggest including a final `else` clause that catches cases when `number` is not in the correct range. You can test your implementation of `number_to_name()` using this number_to_name testing template (http://www.codeskulptor.org/#examples-number_to_name_template.py).
3. Implement the first part of the main function `rpsls(player_choice)`. Print out a blank line (to separate consecutive games) followed by a line with an appropriate message describing the player's choice. Then compute the number `player_number` between 0 and 4 corresponding to the player's choice by calling the helper function `name_to_number()` using `player_choice`.
4. Implement the second part of `rpsls()` that generates the computer's guess and prints out an appropriate message for that guess. In particular, compute a random number `comp_number` between 0 and 4 that corresponds to the computer's guess using the function `random.randrange()`. We suggest experimenting with `randrange` in a separate CodeSkulptor window before deciding on how to call it to make sure that you do not accidently generate numbers in the wrong range. Then compute the name `comp_choice` corresponding to the computer's number using the function `number_to_name()` and print an appropriate message with the computer's choice to the console.
5. Implement the last part of `rpsls()` that determines and prints out the winner. Specifically, compute the difference between `comp_number` and `player_number` taken modulo five. Then write an `if/elif/else` statement whose conditions test the various possible values of this difference and then prints an appropriate message concerning the winner. If you have trouble deriving the conditions for the clauses of this `if/elif/else` statement, we suggest reviewing the "RPSLS" video which describes a simple test for determine the winner of RPSLS.

This will be the only mini-project in the class that is not an interactive game. Since we have not yet learned enough to allow you to play the game interactively, you will simply call your `rpsls` function repeatedly in the program with different player choices. You will see that we have provided five such calls at the bottom of the template. Running your program repeatedly should generate different computer guesses and different winners each time. While you are testing, feel free to modify those calls, but make sure they are restored when you hand in your mini-project, as your peer assessors will expect them to be there.

The output of running your program should have the following form:

```
Player chooses rock
Computer chooses scissors
Player wins!

Player chooses Spock
Computer chooses lizard
Computer wins!

Player chooses paper
Computer chooses lizard
Computer wins!

Player chooses lizard
Computer chooses scissors
Computer wins!

Player chooses scissors
Computer chooses Spock
Computer wins!
```

Note that, for this initial mini-project, we will focus only on testing whether your implementation of `rpsls()` works correctly on valid input. For more helpful tips on implementing this mini-project, please visit the Code Clinic tips (https://class.coursera.org/interactivepython1-003/wiki/rpsls_tips) page for this mini-project.

Grading rubric — 18 pts total (scaled to 100 pts)

Your peers will assess your mini-project according to the rubric given below. To guide you in determining whether your project satisfies each item in the rubric, please consult the video that demonstrates our implementation of "Rock-paper-scissors-lizard-Spock". Small deviations from the textual output of our implementation are fine. You should avoid large deviations (such as using the Python function `input` to input your guesses). Whether moderate deviations satisfy an item of the grading rubric is at your peers' discretion during their assessment.

Here is a break down of the scoring:

- 2 pts — A valid CodeSkulptor URL was submitted. Give no credit if solution code was pasted into the submission field. Give 1 pt if an invalid CodeSkulptor URL was submitted.
- 2 pts — Program implements the function `rpsls()` and the helper function `name_to_number()` with plausible code. Give partial credit of 1 pt if only the function `rpsls()` has plausible code.
- 1 pt — Running program does not throw an error.
- 1 pt — Program prints blank lines between games.
- 2 pts — Program prints `"Player chooses player_choice"` where `player_choice` is a string of the form `"rock"`, `"paper"`, `"scissors"`, `"lizard"` or `"Spock"`. An example of a complete line of output is `"Player chooses scissors"`. Give 1 pt if program prints out number instead of string.
- 2 pts — Program prints `"Computer chooses comp_choice"` where `comp_choice` is a string of the form `"rock", "paper", "scissors", "lizard"` or `"Spock"`. An example of a complete line of output is `"Computer chooses scissors"`. Give 1 pt if program prints out number instead of string.
- 1 pt — Computer's guesses vary between five calls to `rpsls()` in each run of the program.
- 1 pt — Computer's guesses vary between runs of the program.

- 3 pts — Program prints either `"Player and computer tie!"`, `"Player wins!"` or `"Computer wins!"` to report outcome. (1 pt for each message.)
- 3 pts — Program chooses correct winner according to RPSLS rules. Please manually examine 5 cases for correctness. If all five cases are correct, award 3 pts; four cases correct award 2 pts; one to three cases correct award 1 pt; no cases correct award 0 pts.