# Assignment 2 Auto-graded - RPSLS

**Due** Jan 28, 2016 by 12pm **Points** 66 **Submitting** an external tool
**Available** until Jan 30, 2016 at 12pm

This assignment was locked Jan 30, 2016 at 12pm.

Rock-paper-scissors is a common 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 following 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 more unlikely.

Rock-paper-scissors-lizard-Spock (RPSLS) is a variant 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 for RPSLS (http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock)** gives the complete description of the details of the game.

- **http://rice.codeskulptor.org/#comp130_assignment_rpsls.py (http://rice.codeskulptor.org/#comp130_assignment_rpsls.py)**
- **Test (http://codeskulptor.appspot.com/owltest/?urlTests=comp130.assignment_rpsls_tests.py&urlPylintConfig=comp130.pylint_config.py)**
- See submission instructions at the bottom of the page.

---

There are a variety of ways to encode the rules of RPSLS into a program. However, many of them result in long, error-prone code. We could write a long series of 5×5=25 `if`/`elif`/`else` clauses, one for each combination of the two player's choices. However, by recognizing patterns in these rules, we can reduce this, simplifying the code. For example, by noting that duplicated choices are always ties, we could instead have 5×4+1=21 `if`/`elif`/`else` clauses.

With some insight, we can discover the following pattern. Think of the following list as a circle wrapping from bottom to top. Then, each choice wins against the preceding two choices and loses against the following two choices.

- rock
- Spock
- paper
- lizard
- scissors

To easily implement that idea, we'll assign a number to each of the five choices. Then, as described later, we'll be able to use some math to determine the winner.

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

**(10 points code style)** See the **course code style guidelines**.

**(14 points correctness)** Write a function `choice_to_number(choice)` that takes a string `choice`, which is one of `"rock"`, `"paper"`, `"scissors"`, `"lizard"`, or `"Spock"`. It returns the corresponding integer between -1 and 4, i.e., 0 through 4 as described above, plus -1 if the string is not one of the valid values. This function should use an `if`/`elif`/`else` with 6 cases. You can use conditions of the form `name == 'paper'`, etc., to distinguish the cases.

**(14 points correctness)** Write a function `number_to_name(num)` that takes an integer in the range 0 to 4 and returns the corresponding string.

**(14 points correctness)** Write a function `winner(player, computer)` that takes two integers each between 0 and 4. It returns the appropriate string: `"Player wins!"`, `"Computer wins!"`, or `"Player and computer tie."`.

The winning test is actually very simple if you apply modular arithmetic (`%` in Python) to the difference between `computer` and `player`. Experiment with the math to understand its behavior, and you might also review our video of remainder and modular arithmetic.

**(14 points correctness)** Write a function `rpsls(choice)` that takes a string `choice`. If a player's choice is not one of the five acceptable values, your program returns an error string, as shown below. Otherwise, the function then simulates playing a round of Rock-paper-scissors-lizard-Spock. It generates its own random choice from these alternatives, using the function `random.randrange()`. Experiment with this function to make sure that you do not accidentally generate numbers in the wrong range. It should not cheat, so the computer's choice cannot depend on the player's choice. It uses the previous helper functions to determine the winner and produce the appropriate string, as shown below.

To combine two strings, you'll want to use the `+` operator.

We have provided some sample calls that should produce results of the following form:

```
Player chooses rock.  Computer chooses scissors.  Player wins!
Player chooses paper.  Computer chooses lizard.  Computer wins!
Player chooses scissors.  Computer chooses Spock.  Computer wins!
```

```
Player chooses lizard.  Computer chooses scissors.  Computer wins!
Player chooses Spock.  Computer chooses lizard.  Computer wins!
Player chooses rocket.  Improper player choice.
```

Use exactly this form of wording. Each sentence should end in punctuation — a period or exclamation mark. Sentences should be separated by two spaces, while words within a sentence should be separated by one space. Strings shouldn't have any leading or trailing spaces. Capitalization matters.