Decomposition:

The assignment was broken down into 4 functions to accomplish the task.

Recipe:

1. Write a function choice_to_number that returns a numerical value for each of the 5 strings that can be played in the game RPSLS.
    a. Inputs: any string ‘choice’ that represents rock, Spock, paper, lizard, or scissors
    b. Output: any integer 0, 1, 2, 3, 4 respectively (each specific to a certain string)
        i. 0 - rock
        ii. 1 - Spock
        iii. 2 - paper
        iv. 3 - lizard
        v. 4 - scissors
        vi. However, if an invalid choice is entered, have an error (value = -1)
2. Write a function number_to_name that essentially operates exactly in the reverse of the first function. It should return a string with the name of the choice for each integer ‘num’ from 0 to 4. While this may seem useless at first (why are we coding the exact same function in reverse?), it will be necessary later in the assignment because the final function will call both this function and its apparent opposite.
    a. Inputs: Integers 0, 1, 2, 3, 4
    b. Output: a string containing the name of the choice that corresponds the the input integer

3. Write a function winner that takes two inputs and determines the winner. The game is designed such that each input ties itself, wins against two choices, and loses against two choices. Find a relationship between two inputs that can be used to determine whether the matchup is a win, loss, or tie for the first input (representing the user).

   Inputs: 2 strings - player choice, computer choice

   Outputs: a string with one of three messages - indicating win, loss, or tie

   a. Calculate the difference “player integer minus computer integer”
   b. Convert to modulo 5
   c. Result:
      i. 0 = tie
      ii. 1,2 = win
      iii. 3,4 = loss
4. The final function will make use of the three functions written above to create a game taking one user input and randomly generating a computer input.

   Input: player choice string

   Output: string containing player choice, randomized computer choice, and result of the game

   a. The original player choice is converted into an integer using the first function.
   b. The computer choice is randomized as an integer from 0 to 4 using random.randrange
   c. The winner function is used to determine the win/loss/tie of the pair of choices

d. The 2nd function is used to convert the integers back into their choice strings for display.

e. The string displays the player choice, the computer choice, and the final result

Docstrings:

Beginning

"""

This is a program used to simulate the popular "RPSLS" game (rock-paper-scissors-lizard-Spock) from the

TV-show "The Big Bang Theory."

"""

choice_to_number

"""

This function converts each possible move in RPSLS (rock, paper, scissors, lizard, Spock) into a numerical value (0, 1, 2, 3, 4) to make it easier for the program to "play" the game. If the given move is not accepted by the game, it will return -1.

"""

number_to_name

"""

This function converts each numerical value (0, 1, 2, 3, 4) back into its corresponding move in RPSLS (rock, paper, scissors, lizard, Spock).
"""

winner
"""
The following function will decide the winner (or determine if there is a tie) between two RPSLS moves.
"""

rpsls
"""
This function will take a string 'choice' and simulate a round of RPSLS. The computer's choice is randomized and is not dependent on the decision of the human player. This function controls the actual gameplay.
"""

Concepts:

The rpsls function depends on another called function, random.randrange. This function compares the user choice Spock against a randomized computer choice. If rpsls('Spock') were to be run twice, there is a chance that it could produce the same result, but because the elements are randomized, it is possible that the results are different as well.

Only the final “rpsls” function returns different results on separate calls with the same inputs. This is because the first 3 functions (choice_to_number, number_to_name, and winner) rely on the user to enter both the user choice and the computer choice. As a result, the inputs are exactly the same, and the results are exactly the same. However, for the final “rpsls” function, because only the user choice is inputted (the computer choice is randomized), the function can return different results with the same user inputs.

Winning: Each choice can win against two other choices of the five given. Therefore, 2/5 = 40%.
Losing: Each choice can also lose against two other choices of the five given. Therefore, 2/5 = 40%.
Tie: Each choice ties against itself. Therefore, 1/5 = 20%.

Modular Computations:

I attempted to figure out a relationship with the following comment present in my code. I computed the value of player-computer for each relationship. Because the computation was in modulo 5, I converted the results to mod 5.

#The following was just my work to try and find the mathematical relationship between RPSLS moves.

#-rock 0

#rock tie 0-0 = 0

```
#Spock lose 0-1 = -1
#paper lose 0-2 = -2
#lizard win 0-3 = -3
#scissors win 0-4 = -4
#
#-Spock 1
#rock win 1-0 = 1
#Spock tie 1-1 = 0
#paper lose 1-2 = -1
#lizard lose 1-3 = -2
#scissors win 1-4 = -3
#
#-paper 2
#rock win 2-0 = 2
#Spock win 2-1 = 1
#paper tie 2-2 = 0
#lizard lose 2-3 = -1
#scissors lose 2-4 = -2
#
#-lizard 3
#rock lose 3-0 = 3
#Spock win 3-1 = 2
```

```
#paper win 3-2 = 1

#lizard tie 3-3 = 0

#scissors lose 3-4 = -1

#

#-scissors 4

#rock lose 4-0 = 4

#Spock lose 4-1 = 3

#paper win 4-2 = 2

#lizard win 4-3 = 1

#scissors tie 4-4 = 0
```

From the results I could see that the user only won when the modulo 5 difference was 1 or 2. Likewise, the computer only won then difference was 3 or 4. The final value of 0 only occurred when the value was subtracted from itself, i.e. a tie.

I coded for 5 instances:

```
if (player - computer) % 5 == 0:

    return 'Player and computer tie.'

elif (player - computer) % 5 == 1:

    return 'Player wins!'

elif (player - computer) % 5 == 2:

    return 'Player wins!'

elif (player - computer) % 5 == 3:
    return 'Computer wins!'

elif (player - computer) % 5 == 4:

    return 'Computer wins!'
```

This code incorporated the pattern I had discovered into the RPSLS game that I was creating.