Elizabeth Oyebade

May 01, 2021

MET BU CS 521 Spring 02

[ooyebade@bu.edu](mailto:ooyebade@bu.edu)

Final Project – Rock, Paper, Scissors Game

Rock, Paper, Scissors is a game often used as a fair choosing method between two people similar to coin flipping or throwing dice in order to settle a dispute or making an unbiased group decision. Rock is when you place your hand into the form of a simple fist. Paper is when you place your hand in the form of an outstretched position. Scissors when you hold your fist with your index and middle finger pointing outwards in a V shape. Rock smashes scissors. Paper covers rock. Scissors cuts paper. Instead of a two people, my program is pinning a player against the computer. When a player run the program, they first see a menu which gives the options to either start a game, load a previous, display the high scores the users that plays the game by calculating the number of times they’ve won, loss or tied with the computer during game, or exit. When a player starts a new game, they are required to input their name and age. If the player is younger than 15, they are not allowed to play the game. If the player is 15 or older, they are allowed to the game. The game is especially useful in helping the user choose a fair method when a dispute needs to be settle or when an unbiased decision needs to be made.

In the ooyebade\_RPS.py file, def startGame(playerObj) function displays the number of choices available for the player and computer. Using ‘import randit from random’, the computer options of ‘Rock, Paper, Scissors’ is randomly chosen. It also lets the user know whether they’ve won, loss or tied the game against the computer. The def playGame(userGame, userName, playerObj) function displays the menus options after the user selects the option to start a new game. This function gives the user the choice to either play a round, view stats, or exit to the previous menu. The if \_\_name\_\_ == “\_\_main\_\_” is the main function of the program. Within this function, users are able to view the menu which gives the options to either start a game, load a previous, display the high scores the users that plays the game by calculating the number of times they’ve won, loss or tied with the computer during game by selecting option 3, or exit the game with option 4. By selecting option 1, the user has to enter a name and an age (15 or older) in order to start a game. When the username is created, the program creates and read the username. Selecting option 2, allows the user to reload and continue a previous game by inputting their name. If the user enters an option that not on the menu, the program tells them to enter the appropriate option.

In the ooyebade\_info\_user.py file, my program has two classes. The first class method is ‘class Info:’ which holds the players information. It’s a parent class. The \_\_init\_\_() method is the class constructor that takes and create username, win, loss, and tie as its parameter and initializing the wins, loss, and ties as equal to zero. The \_\_repr\_\_ method computes the string representation of the object. The method def get\_username(self) gets and return the value in the username field, def get\_userWins(self) gets the wins of users and return the value, def set\_userWins(self, win) set the wins of user from its parameter. The method def get\_user\_ties(self) get and return the value of the users ties, def set\_user\_ties(self, tie) set the ties of the users from its parameter, the method def get\_userLosses(self) gets losses of users and return its value, def set\_userLosses(self, loss) sets the losses of users from its parameter. And the method def get\_round(self) returns the wins, losses, and ties of the users. The next class method shown in this program is the ‘class User’ method describes the user, it’s a parent class. The \_\_init\_\_() method is the class constructor and within the constructor, it has a self.user\_list which stores each user in a list and self.\_\_RPS.filename that store the file the user read and write in. There are six method shown in this class. The def loadUser() read the list of users in a file using pickle, and def saveUser() writes the list of users in a file using pickle. The method def createUser(self, username) takes in the username and see if the user exists and if the user exists, return false and if it doesn't, add it to the user\_list method and return true. The method def readUser(self, username) accepts a string and returns a boolean and the players objective. The def updateUser(self, user) method takes the names in the user list and update the wins, losses, and ties of the user. The final method in this class is the def display\_highScores() takes the users in the list and creates a ranking of the wins % based on the wins divided by each round. In the User class, I imported pickle to use for the file in the loadUser and saveUser method. I also imported itemgetter from operator to help the method display\_highScores return the function in the temp\_list. To run the program, user must be on the main file which is ooyebade\_RPS.py.