Rock Paper Scissors in Python 3.x

**Rock paper scissors** (also known by other orderings of the three items, with "rock" sometimes being called "stone," or as **Rochambeau**, **roshambo**, or **ro-sham-bo**) is an intransitive hand game, usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "rock" (a closed fist), "paper" (a flat hand), and "scissors" (a fist with the index finger and middle finger extended, forming a V). The earliest form of "rock paper scissors"-style game originated in China and was subsequently imported into Japan, where it reached its modern standardized form, before being spread throughout the world in the early 20th century.

A simultaneous, zero-sum game, it has three possible outcomes: a draw, a win or a loss. A player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors" or "breaks scissors" or sometimes "blunts scissors"), but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cuts paper"). If both players choose the same shape, the game is tied and is usually replayed to break the tie.

Rock paper scissors is often used as a fair choosing method between two people, similar to coin flipping, drawing straws, or throwing dice in order to settle a dispute or make an unbiased group decision. Unlike truly random selection methods, however, rock paper scissors can be played with some degree of skill by recognizing and exploiting non-random behavior in opponents.

import random

import time

options = ("rock", "paper","scissors")

cpu = random.choice (options) # uses  choice method of 'random' libary to give random word from 'options' variable

player\_win = 0

cpu\_win = 0

draw = 0

rock = 0

paper = 0

scissors = 0

target = 3 # amount of times either player or cpu has to win or hit draw for game to end

print("ROCK PAPER SCISSORS!!")

This is the beginning where we import all libraries and declare variables

options = ("rock", "paper","scissors")

cpu = random.choice (options) # uses  choice method of 'random' libary to give random word from 'options' variable

options variable uses a tuple because the values does not need to change

The variable **cpu** uses the .**choice** method from **random** library to choose a random string from the **options** variable at random

while True:

   time.sleep(0.5) # sleep method of 'library' to add small delay before next line

   cpu = random.choice (options)

   player = input("""choose rock, paper or scissors

you chose: """)

   if player == "rock": #increases by 1 if player chose respective weapon

      rock = rock + 1

   elif player == "paper":

      paper = paper + 1

   elif player == "scissors":

      scissors = scissors + 1

.sleep method of time library is used to add a small display of 0.5 seconds before the next lines are read

A while loop is used to play multiple rounds of the game

   cpu = random.choice (options)

Calls the .choice method to select the weapon the cpu will choose

This resides inside the while loop so that it chooses at random for every runtime loop

player = input("""choose rock, paper or scissors

you chose: """)

Calls for the user to input their weapon using the input function making player an input variable

   if player == "rock": #increases by 1 if player chose respective weapon

      rock = rock + 1

   elif player == "paper":

      paper = paper + 1

   elif player == "scissors":

      scissors = scissors + 1

Using if and elif statements with the comparison operator == (is equal to) we will check to see which weapon the player chose

Each weapon the player uses is increased by 1 from the default 0 that is declared in the variables

We will use this to later print out the number of times the player has selected each weapon

if player == "rock" and cpu == "scissors" or player == "paper" and cpu == "rock" or player == "scissors" and cpu == ("paper"): # sets the rules for the game

      player\_win = player\_win + 1 # increase player\_win count if won

      time.sleep(0.5)

      print("cpu chose:", cpu) # print the weapon the cpu played in round

      print("Player won!")

      print("player has won:", player\_win, "cpu has won:", cpu\_win, "draws:", draw) # print number of times player won, cpu won and amount of draws

      print("rock,", rock, "paper,", paper, "scissors,", scissors) # print amount of times player played a weapon

using this if statement we define the win conditions for the game

if the player wins we increase the player\_win by 1

sleep method is called again to delay the next print statements from being displayed for clarity

first print statement shows the weapon cpu chose, as well as a string letting the user know as such

second print statement prints (if the win conditions for the player has been met) a string indicating that the player won the round

third print statement prints the amount of times the player has won, the amount of times the cpu has won, and the amount of time the round has ended in a draw including strings that show what is being printed to the user

   elif player == cpu: # check if player and cpu chose same weapon and if so prints draw and adds +1 to draws var

      time.sleep(0.5)

      draw = draw + 1

      print("cpu chose:", cpu)

      print("draw!")

      print("player has won:", player\_win, "cpu has won:", cpu\_win, "draws:", draw)

      print("rock,", rock, "paper,", paper, "scissors,", scissors)

elif statement is used with the is equal to operator to define if the round ended in a draw, with ours being that the player choice and the cpu choice are equal to each other

draw variable is increased by 1

prints the weapon cpu chose with a string indicator

next prints the result of the round, this time being a draw

*elif* cpu == "rock" and player == "scissors" or cpu == "paper" and player == "rock" or cpu == "scissors" and player == ("paper"):

      time.sleep(0.5)

      cpu\_win = cpu\_win + 1

      print("cpu chose:", cpu)

      print("cpu won!")

      print("player has won:", player\_win, "cpu has won:", cpu\_win, "draws:", draw)

      print("rock,", rock, "paper,", paper, "scissors,", scissors)

another elif statement defines the conditions needed for the cpu to win and the player to lose.

*else*:

      print("incorrect input")

If all other conditions (player win, cpu win and draw) are not met, this else statement is passed and the code assumes the players input is not recognized in the code and prints out “incorrect input”

*if* player\_win == 3: *# end the game (code) after target is reached*

      print("GAME OVER")

      time.sleep(3) *# print respective statement before break*

*break* *# break func is used to end the code*

*elif* cpu\_win == 3:

      print("GAME OVER")

      time.sleep(3)

*break*

*elif* draw == 3:

      print("GAME ENDED IN TIE")

      time.sleep(3)

*break*

these statements defines the target that is needed for the game to end and the code to break using the break function

we use comparison operator == to see if the either win variables or the draw variable is equal to 3

if any of these statements is true the game will end and a the result of the game will be printed out before the code breaks fully ending the game