

PART 1 - 2 Player Game

In part 1, you will be creating rock-paper-scissor game. The game will consist of **2** players, both of whom will bet money on the game. You will run a simulation of **n** games, ending if either the number of **n** games has been reached, or if a player has run out of money. You will then display the stats of the game.

Instructions

1. Define a class to represent the simulation (**Simulation**)
 2. Define a class to represent the simulation stats (**SimStats**) DO THIS LATER
 3. Define a class to represent a game (**Game**)
 4. Define a class to represent a Player (**Player**)
 5. Create the required functions for all 3 classes
 6. Run the simulation for 10 games, where each player starts with \$4 and bets \$1 on each game (even though I should be able to run it for any amount of games)
 7. Print the result of the simulation (see the format at the bottom)
-

Class **Simulation**

Fields

- **player1** - A **Player** class
- **player2** - A **Player** class
- **games** - A array of **Game** class
- **stats** - A **SimStats** class

Required Funcions

__init__(self)

- initializes the class with an empty array for **self.games** and **None** in the **self.stats** field.

run_simulation(self, num_games, player1, player2, bet_amt)

- use a for loop to run **n** (num_games) games.
 - for each game, create a new **Game** class, and call the **play_game** function to run a new game
 - at the end of each game:
 - add the **Game** to the array of **self.games**
 - check the balance of the player, if the balance is 0 for either player, break the loop
- after the loop is completed call the **self.game_stats** to calculate the stats

game_stats(self)

- calculates the stats of the game by looping through the array and calculating the stats and save a **SimStats** class to **self.stats**

`output_stats(self)`

- prints the game stats as formatted at the bottom of the page
-

Class `Game`

Fields

- `game_number` - which number game this is
- `game_results` - an array representing the player who won each round in the game (the `player.number` should indicate which player won)
- `winner` - the `player.number` that won the game

Required Funcions

`__init__(self, game_number)`

- initializes a game
- `self.game_number` should be set to `game_number`
- the other 2 fields should be initialized as an empty array for `self.game_results` and `None` for `self.winner`

`play_game(self, bet_amt, player1, player2)`

- plays 1 game, first to 3 points wins
- use a for loop to player 5 round of each game
 - for each round in the loop, call `play_round` (which returns the winning `player.number`)
 - add the `player.number` to `self.game_results`
 - check the number of each `player_number`, if there are more than 3 for either player, break the loop and set the player number to `self.winner`
- update the player balances by `player.update_balance` on each player

`play_round(self, bet_amt, player1, player2)`

- for each use a while loop and randomly pick `R`, `P`, `S` for each player until someone wins and return the winning `player.number`
-

Class `Player`

Fields

- `player name`
- `player number`
- `balance`

Required Funcions

`__init__(self, player_name, player_number, balance)`

- initializes a new player

`update_balance(self, bet_amt, result)`

- updates `self.balance` depending on the `result`
 - if `result` is `true` then add the `bet_amt` to `self.balance`, otherwise subtract it