(/) BerkeleyX: CS169.1x Software as a Service

txporter (/dashboard)

Courseware (/courses/BerkeleyX/CS169.1x/2013_Spring/courseware) Course Info (/courses/BerkeleyX/CS169.1x/2013_Spring/info)

Syllabus (/courses/BerkeleyX/CS169.1x/2013_Spring/syllabus/)

Textbook & VM (/courses/BerkeleyX/CS169.1x/2013_Spring/textbook_vm/)

Tutorials & Resources (/courses/BerkeleyX/CS169.1x/2013_Spring/tutorials_resources/)

Discussion (/courses/BerkeleyX/CS169.1x/2013_Spring/discussion/forum)

Wiki (/courses/BerkeleyX/CS169.1x/2013_Spring/course_wiki) Progress (/courses/BerkeleyX/CS169.1x/2013_Spring/progress)

HW 1-2: ROCK-PAPER-SCISSORS

In a game of rock-paper-scissors (RPS), each player chooses to play Rock (R), Paper (P), or Scissors (S). The rules are: R beats S; S beats P; and P beats R. We will encode a rock-paper-scissors game as a list, where the elements are themselves 2-element lists that encode a player's name and a player's selected move, as shown below:

```
[ ["Armando", "P"], ["Dave", "S"] ] # Dave would win since S > P
```

Part A: Write a method `rps_game_winner` that takes a two-element list and behaves as follows:

- If the number of players is not equal to 2, raise `WrongNumberOfPlayersError`.
- If either player's strategy is something other than "R", "P" or "S" (case-insensitive), raise `NoSuchStrategyError`.
- Otherwise, return the name and move of the winning player. If both players play the same move, the first player is the winner.

We'll get you started:

```
class WrongNumberOfPlayersError <  StandardError ; end
class NoSuchStrategyError <  StandardError ; end

def rps_game_winner(game)
    raise WrongNumberOfPlayersError unless game.length == 2
    # your code here
end
```

Part B: We will define a rock-paper-scissors tournament to be an array of games in which each player always plays the same move. A rock-paper-scissors *tournament* is encoded as a bracketed array of games:

```
[
    [
        [ ["Armando", "P"], ["Dave", "S"] ],
        [ ["Richard", "R"],  ["Michael", "S"] ],
    ],
    [
        [ ["Allen", "S"], ["Omer", "P"] ],
        [ ["David E.", "R"], ["Richard X.", "P"] ]
    ]
]
```

In the tournament above Armando will always play P and Dave will always play S. This tournament plays out as follows:

- Dave would beat Armando (S>P),
- Richard would beat Michael (R>S), and then
- Dave and Richard would play (Richard wins since R>S).

Similarly,

- Allen would beat Omer,
- Richard X would beat David E., and
- Allen and Richard X. would play (Allen wins since S>P).

Finally,

- Richard would beat Allen since R>S.

Note that the tournament continues until there is only a single winner.

Tournaments can be nested arbitrarily deep, i.e., it may require multiple rounds to get to a single winner. You can assume that the initial tournament is well-formed (that is, there are 2^n players, and each one participates in exactly one match per round).

Write a method `rps_tournament_winner` that takes a tournament encoded as a bracketed array and returns the winner (for the above example, it should return `["Richard", "R"]`).

Choose Files No file chosen

```
On Time
#rps_game_winner
  should be defined
  should raise WrongNumberOfPlayersError if there are not exactly two players [1 point]
  should raise NoSuchStrategyError if there is some strategy that is not R, P, or S [4 points]
  should return the correct winner in a simple RPS game with a clear winner [15 points]
  should return the first player in the case of a tie [10 points]

#rps_tournament_winner
  should be defined
  should still be able to handle the case where a tournament is just one game [10 points]
  should pass the example given in the homework of an 8-player tournament [5 points]
  should pass a basic test case of 8 players [15 points]
  should return the correct winner in the cases of 16 and 32-man tournaments [40 points]

Finished in 0.01844 seconds
10 examples, 0 failures
```

Check

Show Discussion

New Post