# Mini Project: Rock-paper-scissors game

I'm sure you know the simple game of rock, paper and scissors! Just in case, here is a Wikipedia article about it: Rock paper scissors

Your task is to implement this game as a console game in Python **using object-oriented programming**. Remember that there are many different ways you can write the program and many different ways you can design your classes: experiment around to see which option looks best.

In this game, there will be one computer player (random choices generated by Python) and one human player (choices typed on the keyboard).

## 1. Game start

At the start of the game, show its title and ask the user how many rounds they would like to play.

```
--- Rock Paper Scissors Game ---
How many rounds would you like to play?
```

If the user enters something that is not a number, simply keep asking until you get a number.

## 2. Playing a round

In each round, ask the player for their choice: a single letter **r**, **p** or **s** representing rock, paper or scissors, respectively. Then, get a random computer choice, print both choices to the output and print the outcome of the round. If one of the players won, give that player 1 point. Samples:

```
Rock, paper or scissors [r/p/s]? r
You: r  | Computer: r
This round is a tie
```

```
Rock, paper or scissors [r/p/s]? s
You: s  | Computer: r
You lost this round!
```

```
Rock, paper or scissors [r/p/s]? p
You: p  | Computer: r
You won this round!
```

## 3. Final score after all rounds

After all the rounds, write a short game summary indicating the score of the human player and the computer player. Then, print who won (if anyone) and exit the game. Samples:

```
[Game summary] Your points: 2  | Computer points: 2
It was a tie.
```

```
[Game summary] Your points: 0  | Computer points: 1
Computer won.
```

```
[Game summary] Your points: 1  | Computer points: 0
You won.
```

## 4. Other details

If there is anything else about the program not specified in this instruction, you are free to solve it as you considered appropriate. Remember that in programming there are always different ways you can solve the same problem. :)

## 5. Sample program output

```
--- Rock Paper Scissors Game ---
How many rounds would you like to play? 3
Rock, paper or scissors [r/p/s]? r
You: r  | Computer: s
You won this round!

Rock, paper or scissors [r/p/s]? p
You: p  | Computer: s
You lost this round!

Rock, paper or scissors [r/p/s]? p
You: p  | Computer: r
You won this round!

[Game summary] Your points: 2  | Computer points: 1
You won.
```

## 6. Hints & tips

1. You can split the game into classes in many different ways. I encourage you to experiment with different approaches, but in case you have no clue where to start, try the following approach:

   a. superclass **Player** and subclasses **HumanPlayer** and **ComputerPlayer**: keep the player score inside and provide a method to choose the move; obviously, ComputerPlayer will choose their move using a function from the **random** module while HumanPlayer will need to provide their choice with the keyboard.

   b. class **Game**: a bigger class that will keep the players and the number of rounds. Inside this class, write a method named **play():** this method will be responsible for carrying out the rounds, calculating the scores, summarising the game etc.

2. When you get the moves for both players, you need to decide who won the round. There are a few approaches here. You can have a long **if...elif...elif...else** statement, you can use **nested ifs**, or you can use a **dictionary**. The dictionary can have tuples consisting of both moves as keys and the results of the given pairs of moves as values. For example:

```
{('rock', 'rock'): 'tie',
 ('rock', 'paper'): 'lost'
 ... }
```

If you are still not sure how to write the program, remember that you can always take a look at the sample solution that I've prepared for you. It is available as a downloadable resource in the same lecture.