PLTW COMPUTER SCIENCE

Project 3.1.5

# Artificial Intelligence: Rock Paper Scissors Simulation

### goals

- Decompose a project into smaller parts
- Apply coding fundamentals and iterative processes
- Develop a simulation as part of a Scrum team

### description of web page

Develop a program that attempts to predict the outcome of a seemingly random game - Rock, Paper, Scissors.

### Essential Questions

1. What was your role on the Scrum development team?
2. What is the purpose of your program?
3. Where does your program integrate mathematical and/or logical concepts?
4. What does one of the algorithms in your program do?
5. How does an abstraction you created manage complexity in the program?
6. What part of the code did you develop?

### essential Concepts

- Computer Science Practices, Computational Thinking, User-centered Design, Iterative Design and Testing

- Decomposition and Agile
- Algorithms, Variables, Arguments, Procedures, Operators, Data Types, Logic, Loops, and Strings
- Modeling and Simulation

**Resources**

**Project 3.1.5 Student Files**

# Rock-Paper-Scissors Simulation

In this project, you will create a Rock-Paper-Scissors **artificial intelligence (AI)**. Artificial intelligence is any program that analyzes its environment and is able to respond accordingly to achieve an end goal. You will be able to do this using only the *Python* programming you have learned so far in this class. Your AI will have access to a history of its opponent's choices as well as a history of its own choices. A program may base its decisions and next move on these histories.

**PLTW DEVELOPER'S JOURNAL** Document all project work in your PLTW Developer's Journal.

## Gather Data to Simulate

To be able to program an AI, you must first understand the environmental factors that you need to consider to determine the output based on those inputs. You are going to have a class-wide Rock-Paper-Scissors contest to see who is the best in the class and compare strategies. After that, as a class, you will identify the winning strategy. The whole class will compete a few times to see whether the same strategy wins each time, and decide why or why not.

1. Play a paper-scissors-rock competition with your class.

   **Rules of Tournament Play**

   1. Everyone pairs up and plays for two out of three wins.
   2. Count "1, 2, 3", then throw rock, paper, or scissors.
   3. Rock beats scissors, scissors beats paper, and paper beats rock.


4. Whoever wins keeps going, while the person who lost follows them to cheer them on and take note of the strategies they are using.
5. Eventually, only one winner will remain. Have him or her describe their strategy.

2. Continue your strategy research.
    - ☐ Identify strategies used in your class tournament.
    - ☐ Research strategies online.
    - ☐ Plot out a strategy.
    - ☐ Identify what coding concepts (such as loops, operators, and conditionals) will support your strategy.
    -

You will have the program file to run the simulated Rock-Paper-Scissors tournament. You will have additional files that have sample strategies that will compete in the simulated tournament. You will use these files to help develop your own strategy to use in the simulated tournament to gather data on how well your strategy would perform in more games in less time than what you could do in class.

## Running the Simulated Tournament

3. Download the **source files**.
4. Navigate to Cloud9 and log in to your account.

5. Right-click in the workspace and create a CSE_315 folder.
6. Upload all the source files to this folder. It is important that all the files are together in the same folder because they reference each other to work.

7. Execute the psrock_play.py program.

8. Examine the printed output.

    - Which team won?
    - Look at the history of the matches. Can you see any patterns for when the teams call p (paper), s (scissors) or r (rock)?

9. Read the comments before the psrock.round_robin function call and the comments in the psrock_play files.

    - The top part of the report is returned by what part of the program?
    - The bottom part of the report is returned by what part of the program?
    - Why are the reports printed in the order they are?

10. Check your understanding with the exercise below.

> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

11. Change the psrock_play.py program so that all five teams play 10 rounds against each other.
    - Which team won?
    - Does the number of games impact the winning strategy?
    - How much time was saved gathering the same data this way instead of your class doing the same number of games?
12. To create a new team, save one of the team files as “team6.py”. Save this new file in the same folder as the other source files. Run a tournament for the six teams that includes the new team.

# Creating your Own Strategy

Your teacher will give you a team number starting at 7 or higher. Use this number for saving your python strategy file. To complete this project, you will edit the contents of a team file to create a strategy for Rock-Paper-Scissors.

13. Pick a base strategy that you will modify.

    - Copy the file so you will still have that sample strategy to test your strategy against.
    - Based on your teacher’s instructions, rename the strategy_name with the team number.

Each team has different strategies. Some rely on looking at a list of the previous moves focusing on the last moves in the list, their_history[-3], while others just repeat a certain pattern. Understanding how each one works will help you develop your own strategy.

The move function used in the tournament accepts two arguments:

```
def move(my_history, their_history):
```

One argument is a string that represents the moves you made in the past. The other argument is a string that represents the moves your opponent made. You will play a series of rounds against each opponent, and in each round, the move function will be called by the psrock.py file.

14. Make sure your move function returns 'r', 'p', or 's', or your team will be disqualified, since it will not be able to compete. To test your function, run the tournament with the other existing teams and look at the long_report.

15. To check your understanding of how the move function works, change the move function for your team, run the tournament again, and examine the results.

16. With your partner, plan an algorithm to simulate playing Rock-Paper-Scissors.

    - You are not permitted to use the random module, random.choice().
    - To do well, your algorithm should try to detect the opponent's strategy and predict the opponent's next move.
    - Use what you learned during the class tournament to guide your algorithm development.

17. Explain your algorithm using diagrams or human language supported by the research you have done into effective strategies.

18. Create your algorithm in *Python*, testing frequently as you write.

    **Important**: To test your function in the *Python* interpreter, import the file and call the move function directly.

## Running a Practice Simulation

19. When you are ready to test your AI against others, you can run the tournament code as follows:
    - Make sure each team's file is in the same folder along with one psrock_play.py file and one psrock.py file.
    - Modify psrock_play.py, so that each team's file is listed in the import statement.
    - Modify the function call to round_robin, so that each team's file is listed in the arguments and then play 20 rounds.
    - Execute psrock_play.py.

## Simulated Class Tournament

20. Did the winning team win against everyone or just a couple of teams?

21. Did the winning team also have the least amount of losses and ties?

22. Identify an opponent you did well against and an opponent you did poorly against. Explain why your strategy performed well or poorly against them.

## Conclusion

1. How did you interpret and respond to the **essential questions**? Capture your thoughts for future conversations.