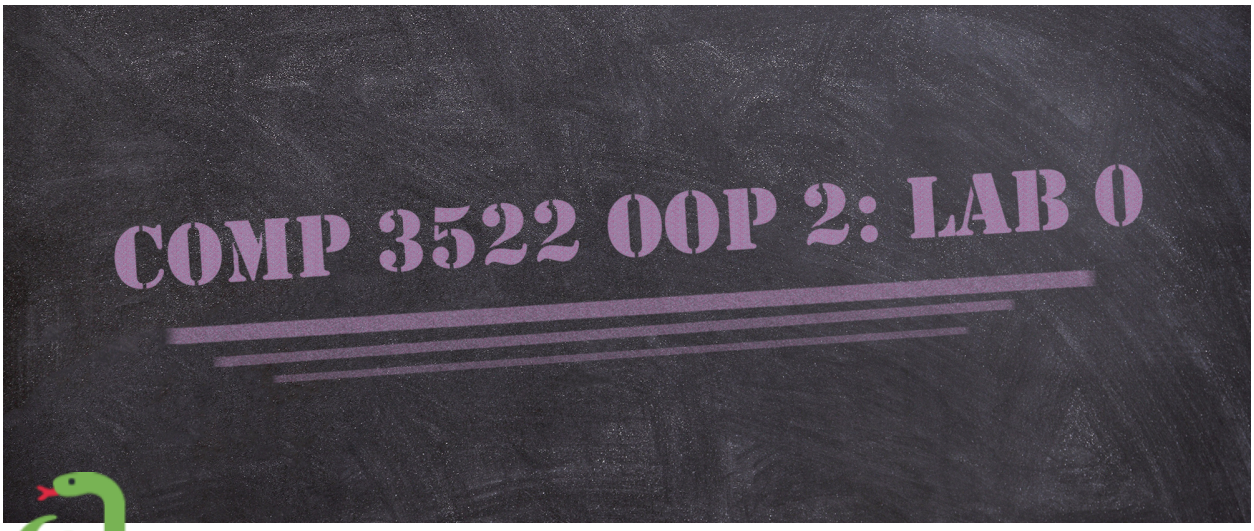



# Lab 0: Rock, Paper, Scissors, Python, Spock

**COMP 3522 Object Oriented Programming 2- Fall 2020**
**Rahul Kukreja - rkukreja1@bcit.ca**
**BCIT - CST**

## Welcome!

Welcome to the first COMP 3522 Lab! In todays lab, you will:

1. Ensure that you have your python development environment up and running
2. Access your course labs and assignments via GitHub Classroom.
3. Write a simple python program and start coding!

## Grading

This lab isn't graded, but future labs will be graded out of 5. If submitted I will be more than happy to provide some feedback.


**IF this were a graded lab**, the grading structure would look something like this:

- **(2 Marks)** Correctly implement all functionality and behaviour outlined in this document below.
- **(2 Marks)** Follow good coding principles. This includes implementing such things as meaningful identifier names, writing atomic functions, focusing on code readability, documentation, etc.
    - You must also adhere to all the standards set within PEP - 8: Style Guide for Python Code and PEP - 257: Docstring Conventions.
- **(1 Mark)** Correctly use git and Github so I can grade your work. This means committing often with meaningful commit messages. **- / 1 Mark**

## Getting Started

Before we can start writing code in Python please do the following:

- Set up your python development environment. We will be working with `Python 3.8.5`.
    - You can find the instructions on D2L under ***Week 1 → Lab 0: Getting Started With Python***. This involves installing Python, the Pycharm IDE and Jupyter Notebook.
- Familiarize yourself with the PyCharm IDE. Follow along with the **"Using PyCharm"** video on D2L under ***Week 1 → Lab 0: Getting Started With Python***.
- Complete **Module 1: Python Funamentals 1**. This involves watching all the video lectures and following along with any code examples.
- Similarly, please complete **Module 2: Python Fundamentals 2** as well.

## GitHub, and COMP 3522

In order to prepare you for industry and to ensure that you continue building good version control habits, we will be using GitHub repositories as our version

control of choice to manage all lab and assignment submissions.

Ensure that git is installed on your laptop. If it isn't, install it now from https://git-scm.com/downloads . To check and see if you have git installed, try running `git --version` from a terminal window or the command line.

Oh, and also create a GitHub Account if you don't have one already.



This course assumes that you have previously worked with Git and are familiar with basic version control (stage, commit, push, pull and merging). If you have any trouble with this then please don't hesitate to reach out to me. You are allowed to use any of the version control GUI interfaces such as Fork, SourceTree or GitKraken

### GitHub Classroom

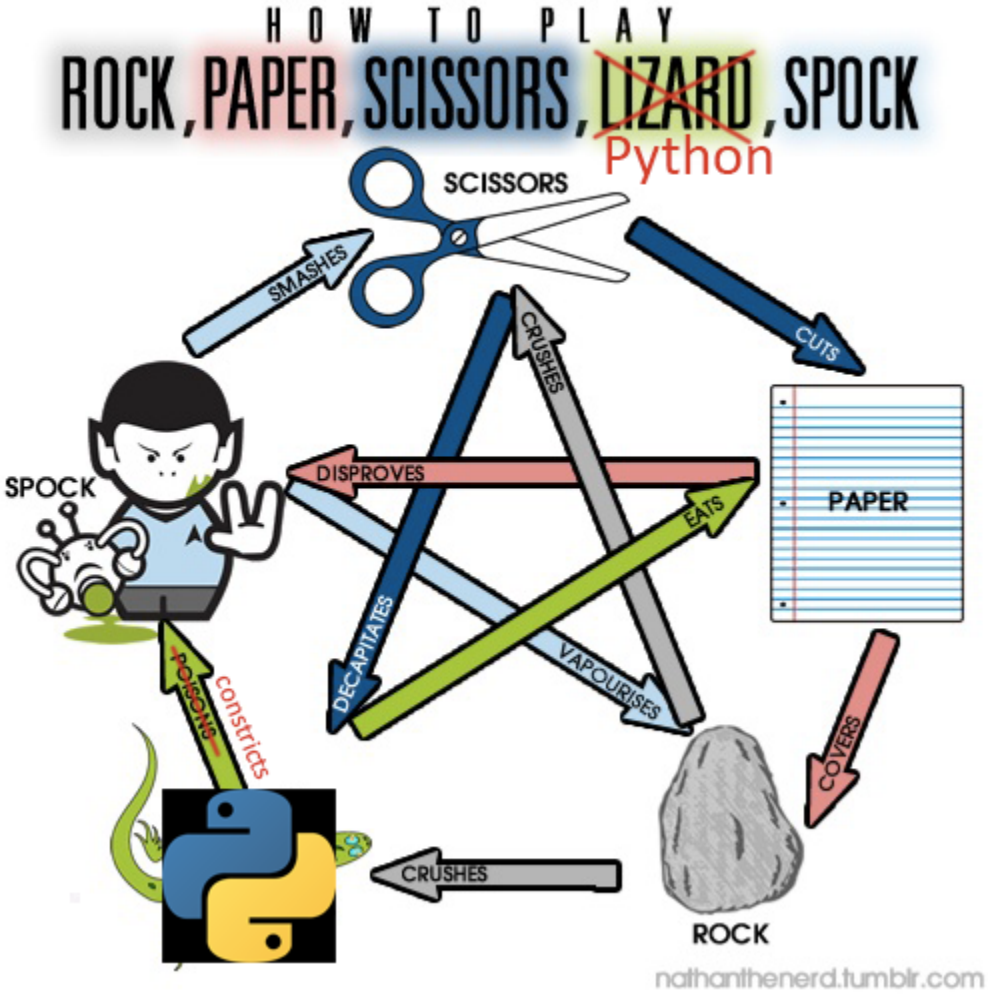
Private repo's for all the labs and assignments will be provided to each student via **GitHub Classroom**. I will have access to each student's private repo to monitor progress, view your code and help answer any questions that you may have. In the event of a group assignment, students will form teams and each team will be provided a private repo.

Each lab and assignment will have an invite link on D2L. Clicking on that link will enroll you into the lab/assignment and create a private repository with starter code and files (if applicable) for you to work with.  This only needs to be done once for each assessment.



The invitation link for Lab 0 can be found on D2L by navigating to ***Content → Week 1 → Lab 0: Getting Started with Python → Course Materials and Recordings → Lab 0 GitHub Classroom Invitation Link*** .

## Developing a Simple Python Program: Rock Paper Scissors Python Spock




Original image by Nathan the Nerd.

Let's create a highly *original* and *non-plagiarized* game which is definitely not appropriated from The Big Bang Theory: **Rock Paper Scissors Python Spock**. This program will simulate a game played between two players. The game rules are:

| Rock | Paper | Scissors | Python | Spock |
|---|---|---|---|---|
| Crushes Scissors | Covers Rock | Cuts Paper | Constricts Spock | Smashes scissors |
| Crushes Python | Disproves Spock | Decapitates Python | Eats Paper | Vaporizes Rock |

Your program in Lab 0 should run a simulation of this game. It should:

- Accept the names of the two players

- Let the players select one of 3 modes:
  - Best of 3
  - Best of 5
  - Best of 7
- Randomly select one of `Rock`, `Paper`, `Scissors`, `Python` or `Spock` for each player and pit them against each other. If a round results in a tie, the round is re-attempted until a winner is found.
- Print the result of each round to the console in addition to the result of the tournament.
- Allow the user to re-run the simulation.
- Give the player an option to exit the game. The simulation should maintain a history of all tournament winners and print the log before exiting the program.

Usually for a program this simple it would be fine to code it all out in one module. It's easy to maintain it as well. For this lab I would like you to get a hang of how the import statement works and as such you will be splitting the program across two modules that have been pre-created for you in the repo — `game.py` and `lab0_driver.py`

### `game.py`

This module should contain all the game logic and game processing. Please implement the following:

- A dictionary mapping that maps each selection (Rock, Paper, Scissors, Python, Spock) to a list of a selections that it can defeat.
- A function that executes a round. It should take the names of the players as it's argument and return the name of the player that won the round. In the event of a tie this function should re-attempt the round until a player has won.
  - Let's call this function `simulate_game(p1: str, p2: str) -> str`
  - Each player is randomly assigned a choice between `Rock`, `Paper`, `Scissors`, `Python` or `Spock` per round.
- Any other supporting functions that you may need.

`lab0_driver.py`

This module should implement a menu that takes user input. The code should then communicate with the `game.py` module to run the simulation and execute the tournament. Please implement the following:

- A menu that let's a player start a tournament or exit the program.
- This module should keep track of all the tournament winners. Print this list of winners when the user exits the game.
- A `main()` method alongside the `if __name__ == __main__` statement that executes the `main()` method.
- This module is also responsible for printing to the console. As such it should be the only module that prints to the console. Do not print to the console from the `game.py` module.
- A function that starts and executes a tournament. This function should take the names of the players and the number of rounds as it's arguments.
  - Let's call this `simulate_tournament(p1: str, p2:str, num_rounds: int) -> str`
- Any other supporting functions that you may need

## Output Formatting

I strongly encourage you to have fun with this lab. This is probably one of the few times that you will be writing non-object oriented and simple code like this. Ignorance is bliss and once you start learning about what good code looks like you will rarely be able to write code like this again! (That's a good thing... sometimes. 😄 ).

Part of having fun in python is exploring and familiarizing yourself with the vast array of packages that have been built by other developers. One such package is `Rich`. It let's you format the output to the console and play with colours, emoji's, tables and more!

> 💡 Install `Rich` by typing `pip install rich` in the command line.
>
> A handy guide with more information and instructions on how to use the package can be found at https://pythonawesome.com/a-python-library-for-rich-text-and-beautiful-formatting-in-the-terminal/ . The official docs can be found at https://rich.readthedocs.io/en/latest/.

Format your output using emoji's and colours. Feel free to be creative in any way you want. The goal here is to have fun 🙂

## That's it!

I hope you all enjoy getting into some hands-on python coding in the first week of the course. For those of you who have never worked with Python before, I hope you feel more comfortable and not as over-whelmed with the language by the end of this lab.

Remember, the best way to learn a new language is to get some mileage writing code!