"Rock, Paper, Scissors" Exercise

Learning Objectives

- Gain familiarity with the Python programming language, focusing on variables, functions, and conditional logic.
- Practice processing and validating user inputs in Python.
- Practice using a text editor to edit and save files of Python code.
- Learn how to incorporate version control practices into your development process.
- Learn how to import and access functionality provided by Python modules.

Instructions

Iteratively develop a Python application which will allow a human user to play a game of Rock-Paper-Scissors against a computer (AI) opponent. The game's functionality should adhere to the "Basic Requirements" below.

Before attempting to implement the basic requirements, take some time to configure your project repository according to the "Setup" instructions below. After doing so, you'll have a remote repo on GitHub.com and a local copy on your computer within which to develop.

When developing, as you reach key milestones, use the command-line or GitHub Desktop software to intermittently "commit", or save new versions of, your code. And remember to push / sync / upload your work back up to your remote project repository on GitHub.com at least once before you're done.

Setup

Repo Setup

Use the GitHub.com online interface to create a new remote project repository called something like "rock-paper-scissors-exercise". When prompted by the GitHub.com online interface, let's get in the habit of adding a "README.md" file and a Python-flavored ".gitignore" file (and also optionally a "LICENSE") during the repo creation process. After this process is complete, you should be able to view the repo on GitHub.com at an address like `https://github.com/YOUR_USERNAME/rock-paper-scissors-exercise`.

After creating the remote repo, use GitHub Desktop software or the command-line to download or "clone" it onto your computer. Choose a familiar download location like the Desktop.

After cloning the repo, navigate there from the command-line:

```
cd ~/Desktop/rock-paper-scissors-exercise
```

Use your text editor or the command-line to create a file in that repo called "game.py", and then place the following contents inside:

```
# game.py

print("Rock, Paper, Scissors, Shoot!")
```

Make sure to save Python files like this whenever you're done editing them. After setting up a virtual environment, we will be ready to run this file.

Environment Setup

Create and activate a new Anaconda virtual environment:

```
conda create -n my-game-env python=3.7 # (first time only)
conda activate my-game-env
```

From within the virtual environment, demonstrate your ability to run the Python script from the command-line:

```
python game.py
```

If you see the "Rock, Paper, Scissors, Shoot!" message, you're ready to move on to project development. This would be a great time to make any desired modifications to your project's "README.md" file (like adding instructions for how to setup and run the app like you've just done), and then make your first commit, with a message like "Setup the repo".

Basic Requirements

Processing User Inputs

The application should prompt the user to input, or otherwise select, an option (i.e. "rock", "paper", or "scissors") via command-line interface (CLI).

> HINT: use the `input()` function to capture user inputs

Validating User Inputs

The application should compare the user's selection against the list of valid options (i.e. "rock", "paper", "scissors") to determine whether the user has selected a valid option.

If the selection is invalid, the program should fail gracefully by displaying a friendly message to the user, and preventing further program execution. The program should not try to further process an invalid input, as it may lead to runtime errors.

> HINT: use the `exit()` or `quit()` keywords to stop the program

Simulating Computer Selection

The application should select one of the options (i.e. "rock", "paper", or "scissors") at random, and assign that as the computer player's choice.

> HINT: use the `choice()` function provided by the `random` module

Determining the Winner

The application should compare the user's selection to the computer player's selection, and determine which is the winner. The following logic should govern that determination:

1. Rock beats Scissors
2. Paper beats Rock
3. Scissors beats Paper
4. Rock vs Rock, Paper vs Paper, and Scissors vs Scissors each results in a "tie"

> HINT: use one or more `if` statements (recommended approach), or it may also be possible to use a pre-configured dictionary object containing all possible outcomes

Displaying Results

After determining the winner, the application should display the results to the user. Desired information outputs include:

- A friendly welcome message
- The user's selected option
- The computer's selected option
- Whether the user or the computer was the winner
- A friendly farewell message

Example desired output:

```
-------------------
Welcome to my Rock-Paper-Scissors game...
-------------------
Please choose either 'rock', 'paper', or 'scissors': rock
You chose: 'rock'
The computer chose: 'paper'
-------------------
Oh, the computer won. It's ok.
-------------------
Thanks for playing. Please play again!
```