Programming Fundamentals

Due Date: (See Slate\ Assignments)
Date: Week 3
Type: **Individual Assignment**
Weight: 5%

Summary

In this assignment, we are writing a program that allows us to practice the fundamentals of building computer programs: the process of building a program using an IDE, program structure using modules and program flow using sequences, branches and loops. While practicing these *fun*damentals, we will make sure not to forget the fun part: programming, while challenging at times, it is fun and rewarding. And in this assignment, you will devise and develop a simple text based adventure game.

Submission checklist

A ZIP file containing:

1. An assignment report submitted as a Word Document that has
   a. A title page with your name,
   b. Date and assignment title and
   c. Clear Structure to identify the answer of each of the assignment parts.

   Requirements for the report are provided in each section of the assignment.

2. The program folder including all source files and GIT repository.

   IMPORTANT NOTE: *Submission is done in electronic format* ***in SLATE. DO NOT email your submission.***

Detailed Requirements

Part I (20%): Program Inception. Invent a simple text-based adventure game that allows the user to choose their actions in a story to determine their success in the game. The requirements are purposefully flexible such that each student can develop their own game that is fun and challenging. You can stick to the basics or take it as far as you want. The requirements for the game are as follows:

1. The user shall be presented with a story that would allow the user to pick one out of two roles. The user would play the chosen role in the story. Unleash your creativity and design roles that are interesting to you. Examples roles are countries, characters, aliens, super-heroes, chefs, scientists, detectives, etc.

2. Each role shall have *three attributes*. For examples: Strength (ST), Dexterity (DX), Intelligence (IQ) and Health (HT). Identify the attribute names and assign the two roles hard-coded values of your choice in the following range: -2, -1, 0, +1, +2. For example, a "*barbarian"* could excel at strength and dexterity while a *"wizard"* could excel at dexterity and intelligence.

3. Based on the story, define the *quest* that the user, in their role, will have to go on (e.g., finding a treasure, destroying a monster, win MasterChef, solve a mystery). The quest shall be made of minimum *three challenges*. Each challenge shall be based on a different attribute of the role, such that all their attributes are used.

4. Define *the game play rules* which are dice based. You can pick the number of dice and the range of each dice (e.g., two six-sided dice with a range of 2-12). As the game proceeds the outcome of each challenge would depend upon the *roll value* (given by the sum of the number rolled on dice) and the attribute value related to the challenge. Associate the total *roll value* with different results e.g., critical loss, loss, win, critical win. These results could have the following meaning:
    a. Critical Loss (e.g., 2 - 3): challenge is lost and the attribute that is based on is decreased
    b. Loss (e.g., 4-7): challenge is lost, no change in the character's attributes
    c. Win (e.g., 8-10): challenge is won, no change in the character's attributes
    d. Critical Win (e.g., 11-12): challenge is won and the attribute that is based on increases

5. Define the game win/loss criteria. For example, the user might win if they succeed in all challenges.

Report Requirements

In your assignment report, introduce the background story of game. Explain the roles, the quest, the challenges and clearly state all the rules of the game you have invented.

Part II (20%): Program Structure. Create a program using Visual Studio Code and Python that will implement the game and have the following structure that ensures a basic separation of concerns:

1. The program shall use GIT for version control. **Ensure the GIT repository is initialized as soon as the program folder is created.**
2. The program shall at least have the following modules:
    a. App.py: implements the interactivity with the user.
    b. Game.py: implements the game data and logic.
    c. <Role1>.py: implements the data and logic associated with the first role (e.g., wizard.py)
    d. <Role2>.py: implements the data and logic associated with the second role (e.g., barbarian.py)

IMP: All the user interaction, i.e., the *input* and the *print* statements, must be in the module App. The other modules should NOT interact with the end user. Any data required in these modules should be exchanged through function parameters and/or return values.

3. Each module shall be described using a doc string at the top of the source file
4. Each module function shall be described using a doc string to explain its purpose
5. Each module variable shall be described using a comment to explain its purpose

Report Requirements

In your assignment report, identify and describe the program structure, clearly stating the purpose of each module and function.

Part III (40%): **Program Logic / Flow**. Implement the game logic to allow the following functionality:

1. The program shall print a "welcome" message that provides and short overview of the game including its name and goal. Be creative and have fun as you are describing the game story.
2. The program shall allow the user to choose one of the two roles implemented
3. For each of the three challenges of the game:
    a. Present the challenge to the user
    b. Allow the user to ask the program to roll the dice and display the result
    c. The program determines the outcome of the challenge and its impact on the user's role
4. After all challenges have been played display a win or loss game over message

Report Requirements

In your assignment report, provide significant screenshots of the game implemented: the welcome screen, each of the challenges and the game over message.

Part IV (20%): **Program Development Process**. The project is to be developed iteratively in small increments. Code must be version controlled using GIT. Each milestone must have at a minimum a commit at the end of each milestone. For best evaluation ensure changes are committed often (more than once per milestone) and the commit messages are informative.

Milestone 1. *Project Creation*. Create the initial program folder and add the project to version control using GIT. Use GIT effectively throughout the development of the project.

Milestone 2. *Game Intro*. The program prints the game overview

Milestone 3. *Role Choice*. Allow the user to choose between the characters implemented in the game and confirm their choice

Milestone 4. *Role 1*. Implement the challenges of the first role. Each challenge shall be implemented separately and committed separately. At the end of the milestone the user should be able to complete the game using Role 1

Milestone 5. *Game Over*. The program prints the game over message (win or loss)

Milestone 6. *Role 2*. Implement the challenges of the second role. Each challenge shall be implemented separately and committed separately. At the end of the milestone the user should be able to complete the game using Role 2

Milestone 7. *Bug Fixing and Polishing*. Test the program and fix any problems you have detected

Report Requirements

In your assignment report, describe your experience as you progress through the development process. What are difficulties encountered in each milestone and how did you overcome them. What limitations does the game have that you would like to address in future versions of this program?

Notes:

1. The **professionalism of your submission**, clarity of written **communication** is extremely important. The ability to communicate your knowledge is as important as the knowledge itself.

   a. Up to 20% of the mark for any written work can be deducted due to poor presentation / communication: *title page (5%)*, *document organization (5%)*, *layout (5%)* and *grammar and spelling* (5%).

   b. Up to 20% of the mark for a program can be deducted due to poor presentation / communication: quality of names according to our *naming and coding conventions (10%)* and *comments (10%)*.

2. **All assignments shall be submitted by the deadline.** Late submissions will be penalized with 10% per day for up to 3 calendar days after which the assignment cannot be submitted anymore. **An email must be sent** should you choose to submit a late assignment. **Assignments are not accepted after the deadline.** See Academic Procedures for Evaluations **http://academic.fast.sheridanc.on.ca/r/ac_academic_procedures_for_evaluations.pdf**

3. This assignment shall be **completed individually**. Remember that completing the assignment by yourself will ensure your success on the midterm and final exam. See the Academic Honesty at Sheridan.

4. Submission is done in electronic format **using SLATE**. **DO NOT email your submission.** Please read the Dropbox Submission Guidelines in SLATE.