COSC 010 Project 2

I. Overview for Project 2

- For Project 2, you will create an adventure story game using Python. The content and structure of the story is up to you, but you must use specific programming elements (detailed in Section III)
- The goal of this project is for you to use and practice all the concepts covered so far, with particular emphasis on sequential data types and repetition structures.
- You are strongly encouraged to be creative with this project.
- Remember that the Python for Everyone textbook and Dr. Gates's textbook are great resources.
- You must complete this project **individually** and submit it to Canvas.

II. What You Will Submit

Part 1: The Python Program
You will submit your program as a .py file. See the next section below for the specifications and required elements. ***Take careful note of the following:***

- Projects must run using Python 3.
- The file must run and function to gain credit. Programs that do not run will receive zero credit.
- If the TA needs to email you or "work" with you after submission to see that your project runs, the penalty is -30%.
- Double check your work.

Part 2: The Project Report

1. The first page of your project report will contain an explanation of what your project does and how it works, including:
   - Overview of your adventure game
   - Expected inputs and outputs
   - Detailed description of how you used each of the required computing concepts
   - Limitations (e.g., do certain user inputs mess up the story or "break" your program?)
   - Future expansion options

   All this will help a person to understand your project. You may use screenshots in the explanations.
2. The second (and more as needed) page(s) will contain three (3) examples of program use – including input and output. You can screenshot these if you wish.
3. The remaining pages will contain a copy of the Python code (***You will also submit the code as .py***).

III. Project Specifications and Required Elements

1) Your adventure game can have any type of story that you would like - but it *must* be original and yours (it also must be PG-rated). Remember, part of coding is the creativity of understanding a problem itself and thinking how code can solve it. Each project will be unique (like a fingerprint).
   - **DO NOT COPY anyone else's code.**
   - **DO NOT SHARE** your ideas or code with other students (until after project grading is complete).
   - You can use the Internet and all class resources *to learn about syntax*. **However, you must write your own program from start to finish.** **DO NOT COPY** code from the Internet.

2) **Good code is readable code**. Comments are required in your code. Be sure to submit clear code.

3) **Good code is user-friendly code**. Design your code with the user in mind, with appropriate outputs.

4) **Your project must use at least the following elements**:
   - Basic requirements:
     - User input: at least one use input value must subsequently be used with a string method (e.g., upper(), split(), isalpha(), etc.)
     - User output: at least one print statement must override the default behavior of adding a newline character at the end (i.e., use end=)
     - At least 10 variables
     - At least 5 decision structures (each one starting with the if keyword and possibly also using elif and else conditions)
   - Sequential data type requirements:
     - At least 1 instance of indexing with a sequential data type (e.g., list, string)
     - At least 1 instance of adding and/or removing items in a list
     - At least 1 list comprehension, used appropriately and for an appropriate reason
   - Loop requirements:
     - At least 2 for/in loops, used appropriately
     - At least 2 while loops, used appropriately
     - At least 1 nesting of loops and/or decisions
     - At least 1 appropriate use of break and/or continue

Final Notes:
- Do not copy or repurpose code from the book or the Internet. Be original and creative. Use the book to learn. Close the book. Then create.
- This does not need to be a huge project. However, you are not limited – feel free to get creative and expansive.
- Start with an idea and plan it out on paper.
- Begin small. Then test the small bit of code to ensure correctness. Then add some more code. Then test the expanded program. Then add some more code – and so on until completion of your program.
- Section V includes a short example to get an idea of what an adventure story program might (begin to) look like.

IV. How to submit

- Enter Canvas, go to COSC-010, then Assignments and choose Project 2
- Upload your Python Program as ***P2_LastName.py.*** (Replace "LastName" with your actual last name)
- Upload your Project Report as ***P2Report_LastName.pdf***
- ** DOUBLE CHECK your submission. If the grader needs to email you, the penalty is -30%.

V. Example Program (Incomplete)

```
import time       #By importing time library, the program can have dramatic pauses

delay = 1.5       #For pauses. Set to 0.0 for testing or speed runs, larger for dramatic effect.
username = input("Who goes there? What is your name?")       #Store user's name
baddie = False    #Since we aren't suspicious people, we will first assume user is NOT a baddie
print()           #This just prints an empty line (to make the spacing in the output nicer)
print("Welcome,", username, " to Hoya Saxa, a castle of great ")
print("stones! You must be a great adventurer to dare enter.")
print()

purpose = input("For what purpose have you come, "+username+"? ") #Store user's purpose
#Decision structure 1. Evaluates purpose variable for two specific options. Also has a default response
if purpose == "adventure":
        print("You've come to the right place!")
elif purpose == "gold":
        print("Oh, really?")
        baddie = True           #Update this Boolean variable because now we're suspicious
else:                           #For any response other than "adventure" or "gold"…
        print("Well, I hope that you find what you're looking for.")
print()
print("It is time for you to enter the castle!\n\n")
print("You enter a dark entry hall. You face a great oaken door, while you ")
print("also spy a spiral staircase to your right, leading up to a tower. ")
print("To the left of the door sits a heavy chest of drawers.")
time.sleep(delay)               #This will put in a dramatic pause before the program continues
print()

choice1 = input("What do you want to inspect? [door/staircase/chest] ") #Store choice 1
print()
#Decision structure 2. Evaluates choice for the three options presented.
if choice1 == "door":           #First option
        print("You open the door and walk into a room with a knight. ")
        time.sleep(delay)
        if baddie:      #A nested if statement, using the Boolean that might have been updated above
                print("The knight shouts: \n \"Hey, you're trying to steal gold!\" ") #Note the use of  \"
                print("He grabs you and locks you into a prison cell.")
                print("You lose.")
        else:           #If we never updated the Boolean to be True, we still aren't suspicious
                print("The knight says \"Hi, how can I help you?\"")
elif choice == "staircase":     #Second option
        print("You walk up the staircase.")
        time.sleep(delay)       #Another use of the dramatic pause feature
        print("One level up, you come to a door. ")
        print("You can either keep going up, or go through the door.")
elif choice == "chest":         #Third option
        print("The chest has three drawers.")
```