4CS001 - Coding Challenge 4

Individual Programming Project

Due: Sunday 3rd January 2021 at 14:00

This assignment is worth 40% of the overall module grade

Introduction: 2
Task Overview: 2
Getting Started: 2
Requirements: 3
Requirements (Advanced): 3
Structure and Documentation 4

Introduction:

This coding challenge will assess your knowledge of Python programming and computational problem solving, this is your chance to showcase what you've learned on the module.

The marking scheme for this task is on Canvas, make sure you check back on a regular basis as you work through the assessment. **There is NO additional challenge segment for this task.**

This task should be completed on an individual basis.

Task Overview:

Text adventures (or interactive fictions) are one of the oldest computer game genres, dating back to the 60's and 70's. Hardware limitations at the time made developing graphical interfaces challenging. Instead, games were controlled through textual interfaces, with commands provided by the player in the form of simple words or phrases. These inputs can be parsed and used to update the internal game state, which is then relayed to the player via textual output. This interaction forms the main gameplay loop and often involves players making their way through an in-game world made up of various locations, obtaining items, solving puzzles and much more. The possibilities are endless…

To get a better idea of what these games are all about, it's worth checking out some popular games in the genre such as: 'Adventure', 'Dreamhold' and 'Zork'. You can access several of these games online at the following link: http://zorkonline.net/.

Your task is to create your own text adventure game in Python or Java.

Getting Started:

There is no template file for this task, you will need to create a new program from scratch. Make sure you add your name and student number to the top of your submission.

Requirements:

You will develop a basic text adventure game. You do not need to include any puzzles or worry about writing compelling stories/adding witty responses. Players should be presented with snippets of narrative and asked to make choices to progress the game. This should continue until the player completes the game or fails/dies. To achieve a passing grade, you will need to demonstrate knowledge of data structures, functions, selection and iteration, as well as documenting your code. **Refer to the mark scheme on Canvas for more details.**

Here is the overarching behaviour we expect of a basic text adventure game:

1. Present players with the name of the game and an introduction on launch.
2. Check if the player has won or lost the game You should either:
   a. Inform the player they've lost and end the game.
   b. Congratulate the player and end the game.
3. Presented players with some narrative to progress the game.
4. Give the player several choices to pick from that allow them to interact with the world.
5. Players should pick a choice or enter a verb/command. This might allow them to move to a new location, pick up or drop an item, attack enemies etc.
6. Check if the user has entered a valid input. If not, continue to prompt the user until they enter a valid response, otherwise, proceed to the next step.
7. Parse the players input, update the internal game state and proceed to Step 2.

Requirements (Advanced):

The above section describes the basic requirements needed to attain a passing grade. However, to achieve the maximum possible mark, you will need to add more complex systems and functionality, showcasing the knowledge and programming skills you have developed over the course of the module. Below are examples of more advanced functions that you may wish to add to your game, implementing these will attract additional marks.

- Rather than giving players fixed choices to select from, allow them to enter more complex inputs/verbs. This should help to give the impression that more choice is available to players and make the game more complex.
- Several difficulty levels that change the game in some form (easy, medium and hard).
- Persistent scoring system to track players progress (reading/writing files).
- Menu systems with various options (play, leader boards etc.)
- Randomised aspects of the game, for example different enemies or weapons could spawn, you could start in different locations etc.

- The game could be fleshed out to include various branching paths rather than having players follow a linear route through to completion. For example, the narrative could be effected by the choices players make.

Structure and Documentation

The structure of your code and documentation will be analysed and assessed.

This will be done using a static analysis tool called **Pylint**. This software checks the code in your program, ensures that it follows Python conventions and that all functions, classes and modules have been documented. You can read more about it here: https://www.pylint.org/.

Python has an official style guide named PEP8, which is where most Python conventions/coding standards originate from. Example checks that Pylint carries out to ensure that the PEP8 coding standard is followed include things such as:

- checking line-code's length
- checking if variable names are well-formed (snake case)
- checking if imported modules/functions are used
- checking if variables/function parameters are used

It is a good idea to run these checks on your code at regular intervals and before submitting. There are several free websites that allow you to do this e.g. https://pythonbuddy.com/.

Note: Marks will be deducted for warning and errors detected in your code.