University of Sussex Programming in Python (823G5) Coursework Instructions

This assignment is worth **40%** of the total marks for this module.

Due:	Check Canvas
Format:	ZIP file. Electronic submission only (Canvas)

General instructions

1. Answer all parts of the brief.
2. Do not copy the work of another student. Plagiarism is a very serious matter. Discussion between students is to be encouraged – copying is an academic disciplinary matter.
3. Hand your submission in on time. There are penalties for late submission.
4. If I cannot read your submission, I cannot mark it. It is your responsibility to ensure that the presentation of your submission is appropriate for a university student.
5. If you do not understand the brief, you can get help at the workshop sessions.
6. Ensure your candidate number is on your final submission. It is surprising how many students forget this basic information.

Coursework: “Moving through Space” Adventure Game Specification

Design and implement your own adventure game scenario. The game can be anything that has as its basic structure the idea of a player moving through different locations.

Some possible examples:

- You are a pirate searching for gold in the convoluted network of caves on Treasure Island.
- You are a plumber moving though a labyrinth of underground water pipes and chambers.
- You are a knight, searching through a series of rooms in a medieval castle, gathering magic items and slaying monsters to save your Queen.

Use the “Adventure World” game that you can download from Canvas as your starting point. The game is provided as a `zip` file with 3 Python class files in it. Just create a project in PyCharm and import the 3 files into the project. The `main()` method can be found in the `Game` class. You can use as much or as little of this code as you like – it is just to give you some ideas on how to get started. Feel free to adapt and extend the code in any way that you deem appropriate.

Make sure that you are following good design principles, such as *low coupling*, *high cohesion*, *no code duplication*, and *responsibility-driven design*. To gain more marks (see Marking table below) you can increase the complexity of the game by adding, for example, items for the player to pick up or to put down in each room, up-down movement, “instant transportation” to some chosen location, etc. However, please note that the main purpose of this assignment is to create a very clean, responsibility-driven design, following good software engineering practices such as low coupling and high cohesion, with clear documentation for your code.

Marking

This assignment is worth 40% of the total module marks. The marking scheme for this assignment allocates marks out of 100 according to the following breakdown:

Aspect	**Marks available**
Design and coding style	30
Complexity of the game	20
Project report	15
Error prevention and recovery	15
Class & method documentation and inline commenting	10
Logging of player moves and outcomes	10
Total	**100**

Suggestions

The following list provides some suggestions as to the functionality that should be added for a good grade. Note this is not an exhaustive or prescriptive list – it is provided for guidance only. The precise functionality will depend on the game scenario that you choose:

- Choose a fun scenario and modify the starter code base accordingly.
- Have at least 10 locations to make the game play space more interesting.
- Add the ability to store game items in some of the rooms.
- Add a new command to allow the user to pick up items when in a room.
- Provide the player with a backpack to store the items picked up.
- Set weight limits on the number of game items that the player can carry at any time.
- Add an inventory command so that the player can easily find out what they have in their backpack.
- Add locks to some rooms so that they may only be entered when the user has and uses a key item picked up elsewhere in the game space.
- Re-engineer the Game class to make it more cohesive.
- Consider adding a Player class to represent the Player entity properly.
- Add a meaningful objective for the player to achieve to win the game.
- Add automated unit tests for each class.
- Consider adding clues, such as notices that can be read when the player provides an appropriate command.
- Add new directions and different levels to the game e.g., directions such as "upstairs".
- Consider adding a facility where a user can purchase items within rooms for use elsewhere in the game.
- Consider adding a stochastic element to randomly generate puzzles, items and layouts (being careful to record the random seed for reproducibility).

Class Documentation

Class documentation should include:

- A brief text description of the class and its purpose
- A list of any formal parameters (inputs) using `:param`
- Details of any return values using `:return`

Look at the starter code on Canvas for examples of how to properly document a class.

Project Report

Your submission should include a short report that includes:

1. A cover page with your **candidate number** and your game's title.

2. The problem statement and overview of the game. This should be a brief description of the problem the program addresses. You should describe the game scenario and explain what can be done within the game, what the end

goal is (win conditions), and how many locations the player can travel through.

3. Clear instructions on how to launch the game and play through the adventure (especially if you require additional packages to be installed, start the game other than by running `game.py`, or used an alternative IDE to PyCharm).

4. A map of the game world.

5. A UML class diagram for your project.

6. A brief description of how the starting program was modified. You do not have to describe the starting program or any of its classes but you should say how these classes have changed. Also describe any new classes. The description of new classes should be brief and at a high level.

7. Discussion of any interesting design features, such as:
    a. puzzles / mini games to obtain items or unlock doors within the game
    b. unusual data structures
    c. creative game mechanics

8. A description of any problems or issues that you encountered during this part of the coursework.

9. Any evidence of testing that you have performed, either as automated unit testing or system level testing.

Your report should consist of no more than 1,500 words (excluding any appendices if you have them). You can write your report in any word processing software you like, just make sure to convert to pdf before submission.

Submission Checklist

Put all your files together in to one `zip` file and upload it using the electronic submission point on Canvas. Your `zip` file must contain the following files and folders:

1. The project report (as described above) saved as a pdf file.
2. A folder with all the python source files (and environment files if necessary*) required to run the game.

* If you are using PyCharm, just zip up the entire PyCharm project folder and all your code will be safely inside. If you are using an alternative IDE, check that you have included all code files in your submission.

Double check that your zip file unzips correctly and contains all necessary files before submission and/or download and open your submission after uploading. **It is your responsibility to ensure that all required files are submitted before the deadline**. Sadly, I sometimes get empty files because students have not checked – empty files do not grade well!

Academic Misconduct

You may get feedback and advice from your tutors in the practical classes and discuss ideas with fellow students. However, **you must not copy other people's/AI's work**. We have sophisticated ways of detecting this and penalties can be severe!

If you are unsure what the forms of Academic misconduct are (including plagiarism, collusion and personation), see the Student Hub to check what is and is not acceptable. You can also get help to avoid Academic Misconduct through resources at the Skills Hub: https://www.sussex.ac.uk/skillshub/?id=287.

Remember, even if you are struggling, it is better to submit your *own work* and get *some credit* for it, rather than *nothing* for *someone else's work*.

On a more positive note, I wish you the best of luck with your projects and we look forward to playing them!

Dr. Benjamin Evans
B.D.Evans@sussex.ac.uk
University of Sussex