UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Business School
Assignment Cover Sheet

Course:	**Foundation Certificate Programme**
Unit Code and Description:	Introduction to Programming in Python – P1
Module Leader:	Mr. Sudharshan Welihinda
Assignment Number:	1
Assignment Type:	**Individual Coursework**
Issue Date:	**14th November 2022**
Hand – in – Date:	**6th December 2022**
Deadline:	**on or before 9.00 AM**
Weighting:	
Qualifying mark:	**40%**

The department is not responsible if an assignment is lost. To cover this eventuality, you are advised to take a photocopy of the assignment OR to ensure you have the means of re-creating it.

1. Procedure for submission:

- Create a folder including your coursework report (In PDF format) and all the python codes. (Python 3.x source codes)
- Name the folder as "**DOC 333 Coursework report – StudnetID**" (E.g DOC 333 Coursework report – 20220xxx)
- Then convert your folder to a ZIP file and submit it to the link given in LMS before the deadline (Link will be available under ***Coursework*** section)
- Ensure you submit your ZIP folder on time as per the given deadline else, the submission will be considered as a late submission.
- Check if you are uploading the correct ZIP file as you will be given only ***one chance*** to submit/email. Changes cannot be done.

2. **Penalties for Late Hand In:**
   - If students submit coursework late but within 24 hours (or one working day) of the specified deadline, the work will be marked and will then have 10% of the overall available marks deducted, to a minimum of the pass mark (40%).

   - If students submit coursework more than 24 hours (or one working day) after the specified deadline, they will be given a mark of zero for the work in question.

3. **Exceptional Factors Affecting your Performance:**
   - Students should submit written evidence to the Registrar's Department with a copy to the Module Leader of exceptional circumstances, which they consider having caused them to submit assessments late and for which they do not wish to attract any penalty. These must be handed over/emailed to the Registrar within four working days of the hand-in-date.

- Proper use of Python 3.x coding and language constructs is needed for a better program. You should follow good and proper programming techniques when completing this coursework.

***** Plagiarism *****

The strength of the university depends on academic and personal integrity. You must be honest and truthful. Plagiarism is the use of someone else's work, words, or ideas as if they were your own.

Plagiarism is a serious offense and will not be treated lightly.

Assignment Brief

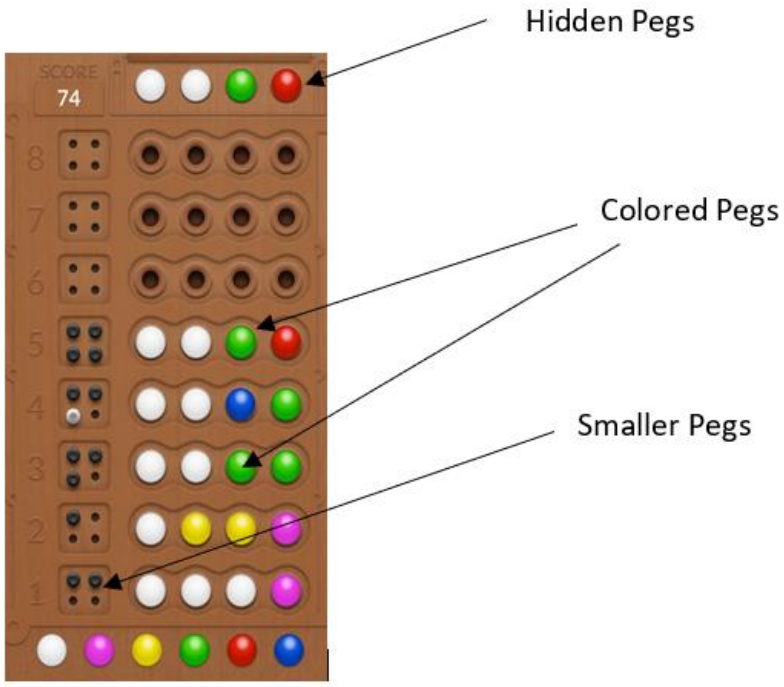

Figure 1

Guess the color of hidden pegs. A deduction game where each player takes turn making a limited number of guesses, using logic to deduce what pegs the opponent has hidden.

The code maker (Here the system will act as the code maker) secretly puts four colored pegs and put in the spaces behind a screen at once end of the game board. (Here the system will use integer numbers from 1 to 6 to represent the six colors. i.e. 1- White, 2-Blue, 3-Red, 4 –Yellow, 5-Green, 6-Purple and shall randomly generate a four-digit number. This will represent as the secretly selected four colors. E.g if the system generates a number 2651 it will represent 4 pegs as Blue, Purple, Green and White pegs)

The other player, the code breaker, makes a series of guesses by entering 4-digit number at each guess. This 4-digit number will represent four colors of his guess and the position. E.g if he enters 2413, it indicates blue color peg is in position 1, yellow color peg is in position 2, white color peg is in position 3 and red color peg is in position 4.

After each guess, the code maker uses smaller pegs (here the system shall put '0' to represent white peg and '1' to represent black peg) to tell the code breaker if the guessed pegs are the right color and in the right place or are the right color but the wrong place or are the wrong color entirely.

if the guessed pegs are the right color and in the right place then the system uses 'black' peg (here as '1') and the guessed pegs are the right color but the wrong place then the system uses 'white' peg (here as '0'). If it is wrong color entirely then the system does not use any pegs and keep it blank.

The code breaker makes another guess in the next row, building upon information from previous guesses, trying to match the pegs the code maker hid at the beginning of the game.

Hi <Student_name> Welcome to GameInt

Number to Guess - X X X X

Color Mapping:
1-White 2-Blue 3-Red
4-Yellow 5-Green 6-Purple

Attempt No	**Guess**	**Result**
1	2 4 5 3	1 1 . .
2	2 5 6 1	1 0 0 0
3	2 6 1 1	1 1 0 .
……		
…….		
8	2 6 5 1	1 1 1 1

Congratulations !!!!! You have won the game…

You have scored **XXX** points.

Do you want to play another game (Yes/No) ? __

Figure 2

Task to be performed:

Task 1:

Set up a Python project by following the instructions and routine practiced in all tutorial exercises.

Task 2:

Implement a human-computer interaction mode, where the following will be possible:

1. Visualize the user interface as depicted by figure 2.
2. The user should be in a position to mark interactively the start and end of the game by selecting start and end options from the menu.
3. The system should be able to generate a 4-digit random number where each digit is in the range of 1-6.
4. The user should be able to enter a 4-digit number where each digit is in the range of 1-6.
5. The system should validate the 4-digit number entered by the user and should NOT accept any number where the number does not fall into criteria specified in point 4. and should give an appropriate error message if the criteria do not match.
6. The user should be able to guess a maximum of 8 guesses and should be able to terminate the game without going into 8^{th} guess (if he/she wishes) by entering '0000' as the guess.

Task 3:

Design and implement your "GameInt" game as specified by the problem specification.

Task 4:

A brief report (less than 10 A4 size pages) which gives the following.

- Table of contents
- Introduction about the problem
- Flowchart or pseudocodes which explains the program
    - This can be parts of the programs if you implement modular coding
- Actual python codes for the program (final solution)
- Any other vital information you wish to present (e.g any assumptions you made.)
- Screenshots of the working program (Which includes positive and negative results)
- Conclusion

Task 5:

Demonstrate and **defend** the implemented solution at viva by

a) Visualizing the menu as of figure 1.

b) Marking the start and end of the program.

c) Demonstrate the points specified in Task 2.

d) Justifying the results.

To Submit:

The following should be submitted to the LMS as a ZIP file.

- Python source codes
- PDF version of the original report
- One or two past game play records in text files

Note:

- All codes must be written in Python 3.x version.
- A listing of the programs (ensure that your program listing is appropriately commented)
- You should submit the softcopy of the report and the project inclusive of all source code of your solution.
- **DON'T use another student's work or submit to someone else's work as your own.**

End of Coursework