|  |  |
| --- | --- |
| **AGREED ALTERNATIVE ASSESSMENT** | |
| **Module Title: Programming** | **Module Code: 4COM1037** |
| **Assignment Title: Basic Game Development in Python** | **Semester: B, Individual Assignment** |
| **Tutor**: Dr Barry Ip | **Internal Moderator**: Dr Peter Lane |

|  |  |
| --- | --- |
| Student ID Number **ONLY**: | Year Code: |
|  |  |

|  |  |
| --- | --- |
| Marks Awarded %: | Marks Awarded after Lateness Penalty applied %: |
| Penalties for Late Submissions   * Late submission of any item of coursework for each day or part thereof (or for hard copy submission only, working day or part thereof) for up to five days after the published deadline, coursework relating to modules at Levels 0, 4, 5, 6 submitted late (including deferred coursework, but with the exception of referred coursework), will have the numeric grade reduced by 10 grade points until or unless the numeric grade reaches or is 40. Where the numeric grade awarded for the assessment is less than 40, no lateness penalty will be applied. * Late submission of referred coursework will automatically be awarded a grade of zero (0). * Coursework (including deferred coursework) submitted later than five days (five working days in the case of hard copy submission) after the published deadline will be awarded a grade of zero (0). * Where genuine serious adverse circumstances apply, you may apply for an extension to the hand-in date, provided the extension is requested a reasonable period in advance of the deadline. | |
| Please refer to your student handbook for details about the grading schemes used by the School when assessing your work. Guidance on assessment will also be given in the Module Guide. | |
| Guidance on avoiding academic assessment offences such as plagiarism and collusion is given at this URL: <http://www.studynet.herts.ac.uk/ptl/common/LIS.nsf/lis/citing_menu> | |

|  |  |  |
| --- | --- | --- |
| **ASSIGNMENT BRIEF** | | |
| **This Assignment assesses the following module Learning Outcomes:**   * LO 1: Have knowledge and understanding of sufficient features of a high level programming language to develop solutions to simple programming problems; * LO 2: Have knowledge and understanding of the concepts of data declaration and operations, control flow (sequence, selection, iteration, subroutine call) and modularisation; * LO 3: Have knowledge and understanding of the terminology used in describing programs and programming. * LO 4: Design and implement solutions to simple programming problems in a given programming language; * LO 5: Execute, test and de-bug programs; * LO 6: Document programs to an agreed standard. | | |
| **Assignment Brief:**  Please see full details as set out below. | | |
| **Submission Requirements:**  Please note the following essential points:   * All submissions need to be made on Canvas, via the Assignments link on the module page. You should submit all program files as a single zipped file. * **No** additional modules should be included in your code other than those covered in this module (e.g. turtle, random, constants). This is to ensure markers can run your program with minimal errors and compatibility issues. * Your program should be written in Python 3. | | |
| This assignment is worth 40**%** of the overall assessment for this module.  **Marks awarded for:**   |  |  | | --- | --- | | **Components** | **Marks** | | Programming Technique   |  |  | | --- | --- | | *Appropriate data types and declarations* | *10%* | | *Appropriate program control techniques* | *10%* | | *Modular programming* | *10%* | | *Attempt of stretch features, if any* | *10%* | | *Appropriate comments on the program* | *10%* | | 50% | | Program Execution   |  |  | | --- | --- | | *Suitable input and output* | *5%* | | *Appropriate drawing and usage of ambulance on game screen* | *10%* | | *Correct display of Player 1’s chosen word on game screen* | *5%* | | *Correct display of wrong guesses on game screen* | *5%* | | *Correct display of ‘Win’/’Lose’ message on game screen* | *5%* | | *Correct display of remaining guess attempts* | *5%* | | *Correct game restart on end of each instance* | *5%* | | *Successful execution of stretch features, if any* | *10%* | | 50% | | **Total** | **100%** |   Also see Page 7 for general grading criteria.  Please note:   1. For undergraduate modules, a score above 40% represent pass performance at honours level. 2. For postgraduate modules, a score of 50% or above represents a pass mark. 3. Modules may have several components of assessment and may require a pass in all elements. For further details, please consult the relevant Module Guide or ask the Module Leader. | | |
| Typical (hours) required by the student(s) to complete the assignment: **70** hours | | |
| **Date Work handed out:**  Week commencing 30th March 2020 | **Date Work to be handed in:**  2pm, Thursday, 7th May 2020 | **Target Date for the return of the marked assignment:**  Thursday 11th June 2020 |
| **Type of Feedback to be given for this assignment:**  Individual feedback will be given to each student. | | |

**Basic Game Development in Python**

**Demo Video:**

A showcase demonstration video (Demo Video – Guess the Word.mov) can be found on Canvas, which can be used as guidance for the development of this coursework.

**Requirements:**

For this assignment, you are tasked to develop a game called “*Guess the Word*”, which allows two players to participate in turn. The game should contain the following key elements of play:

1. The game should start by prompting Player 1 to enter a word of their choice via the Python console screen for Player 2 to guess (assume that Player 2 looks away and does not see).
2. The Python console should then be cleared so as to hide the word from Player 2.
3. Player 2 must now guess the word by typing one letter into the Python console following a suitable prompt.
4. For each guess, the game needs to determine if the guessed letter is present in the word that has been set by Player 1. If the guess is incorrect, one element of an ambulance should be drawn onto the game/Turtle screen (see video for example of how elements may be drawn and further detail below).
5. The process then repeats from Step 3 (i.e. Player 2 is asked to enter another letter, assuming they have not used up their total number of guesses, as below).
6. Player 2 is allowed a total of 8 incorrect guesses, and hence will lose the game if all 8 attempts have been used.
7. As consistent with the above, the drawing of the ambulance should take 8 strokes in total to complete on the game/Turtle screen. Therefore, if the picture of the ambulance is fully drawn, this means Player 2 loses the game, and a message is displayed on the screen to confirm.
8. The game should provide a suitable graphical interface to enhance the user’s experience and increase the game’s appeal.

The layout of the game screen should look something like this:

A screenshot of a cell phone

Description automatically generated

Several key aspects of information must be shown on the game/Turtle screen to aid the operation of the game (as shown in above example):

* Each letter of Player 1’s chosen word should be masked with a correct number of stars, \*, to indicate the number of letters in the word (relevant letters will then become visible with a correct guess).
* There should be a suitable message displayed to indicate if Player 2 has won or lost the game at the correct point (i.e. when all the letters of the word have been guessed correctly, or the number of attempts has run out).
* Correct and appropriate drawing of the ambulance (a total of 8 strokes to match the number of attempts allowed).
* Display incorrectly guessed letters.
* Display the number of remaining attempts.

User input may take place in the Python console. The console should be cleared for the benefit of game play, i.e. at the beginning of each new game, and after Player 1 has entered their word. The game should repeat automatically on the completion of each instance (i.e. when Player 2 wins or loses).

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

**Design Strategy:**

There may be numerous approaches to implement this program. Here is one possible pathway to cover essential points of development:

1. Write a simple main file. Display the welcome information in a user-friendly format, allowing Player 1 and Player 2 to input at the correct time, and clear screen where necessary.

(Refer to Units 11 and 12 for basic Python program structure, and Unit 13 for input and output.)

1. Store the word that Player 1entered as a list.

(Refer to Unit 16 for the use of list.)

1. Use a loop to repeatedly ask Player 2 to input a letter.

(Refer to Unit 14 for the use of loops, and refer to Unit 13 for user input.)

1. Search the letter that Player 2 has entered in the list (Point 2 above) and determine if there is a match for later use.

(Refer to Unit 16 for the use of list.)

1. Add a counting feature that allocates a total of 8 guesses.

(Refer to Unit 18 for arithmetic operations as well as Units 13 and 14 for use and control of variables.)

1. Determine and implement a drawing process for the ambulance.

(Refer to Unit 15 for Python graphics.)

1. Split the code for the drawing of the ambulance into 8 distinct parts according to key strokes (watch the demo video as an example) and call them as required depending on Player 2’s progress.

(Refer to Unit 15 for Python graphics. Refer to Unit 14 for program control.)

1. Display relevant messages on the game screen.

(Refer to Unit 15 and Video Lecture that details this assignment to write text on canvas.)

1. Enhance the documentation of program.

(Refer to Unit 2 for information on adding suitable comments to the code.)

**Stretch Abilities:**

If you wish to extend this project further, you may consider adding relevant additional features as you deem appropriate. Some examples may include:

* increasing the number of strokes/number of attempts to make the drawing of the ambulance more detailed;
* keeping track of score (i.e. the number of games won or lost);
* allowing for player information to be entered and displayed;
* some other feature(s) you feel may be interesting for a game of this type.

You can be as creative as you wish in order to enhance the quality and appeal of the game. However, I suggest you do this only when all of the above core features are complete.

For any additional or standard features, please **do not** use any other libraries other than those covered in this module. This is to ensure that the tutors can run your programs with minimal errors and compatibility issues.

Please ensure you include all relevant images and program files in your submission as a single Zip file.

**Grading Criteria**

|  |  |  |
| --- | --- | --- |
| **Marks** | **Programming Technique (50%)** | **Program Execution (50%)** |
| 80 – 100 (outstanding) | Solutions are novel & fully addresses task within context. Profound depth of engagement and incisive ideas development. Decision making is in perceptive; techniques are used in an outstanding manner. | Outstanding use of appropriate technology as applied to the problem domain. Consistently accurate and outstanding application of skills and techniques. Components are fully functional as required and occasionally stepping beyond expectations using sophisticated solutions. |
| 70 – 79 (Excellent) | Solutions are innovative and partially address the task within its context. Considerable depth of engagement and successful ideas development. Clear rationale for decisions; techniques are used in an excellent manner. | Excellent use of appropriate technology as applied to the problem domain. Minor errors in technique and/or application with little or no impact on deliverables. Required components are functional as required, with a generally excellent and highly accurate application of skills and techniques. |
| 60 – 69  (Very good) | Solutions relate directly to task and may step beyond conventions.  Strong engagement with subject material and processes, evaluation of alternatives, solutions come from process. Strong use of techniques to derive solutions. | Very good use of appropriate technology as applied to the problem domain. Understanding is demonstrated. Small errors in technique and/or application with little impact on deliverables. Most required components are functional, with a high level and accurate application of skills and techniques. |
| 50 – 59  (Good) | Solutions are appropriate to task, work well within conventions. Good use of subject material and processes. Experimentation is in evidence to support implementation within conventions. Good use of techniques to derive solutions. | Good use of appropriate technology as applied to the problem domain. Good and reasonably accurate application of skills and techniques demonstrated. Some errors in technique and/or application with minor impact on deliverables; required components may not be fully functional but may be easily resolved. |
| 40 – 49 (Satisfactory) | Solutions limited to task and address conventions. Solutions found or adopted but completion is rushed. Some experiments but limited alternatives. Techniques are applied to derive solutions but steps are missed. | Satisfactory use of appropriate technology as applied to the problem domain. Satisfactory application of skills and techniques demonstrated but with minor inaccuracies. Errors in technique and/or application with some impact on deliverables; required components may suffer from notable errors that affect overall functionality, but which can be rectified by minor improvement. |
| 30 – 39 (Marginal fail) | Solutions frame task inappropriately and do not address conventions. Basic use of strategies, few alternatives, limited evaluation with limited experimentation. Limited use of techniques to derive solutions. | Limited use of appropriate technology as applied to the problem domain. Limited application of skills and techniques demonstrated. Many errors in technique and/or application with high impact on deliverables; required components may suffer from critical errors, and may require significant improvement. |
| 20 – 29  (Clear fail) | Lacking in appropriate solutions with very limited use of strategies, no evaluation and little evidence of ideas development. Little use of techniques to derive solutions. Certain elements of academic misconduct. | Very little use of appropriate technology as applied to the problem domain. Very little skill and application of techniques demonstrated. High number of errors with very high impact on deliverables; required components may not function in any discernable way and/or are entirely missing. Certain elements of academic misconduct. |
| 0 – 19  (Fail – nothing of merit) | No or completely inappropriate solution. No use of strategies, no planning and no experimentation. No application of techniques to derive solutions. Academic misconduct. | No use of appropriate technology as applied to the problem domain. No skill and application of technique demonstrated. Very high number of errors in deliverable or no deliverable submitted. Required components are under developed, missing, or inappropriate. Academic misconduct. |