|  |  |
| --- | --- |
| **Assessment Details and Submission Guidelines** | |
| **Trimester** | T2 2021 |
| **Unit Code** | BN108 |
| **Unit Title** | Programming for Networking |
| **Assessment Type** | Individual |
| **Assessment Title** | Assignment 2 - Turtles and Tricks |
| **Purpose of the assessment (with ULO Mapping)** | This assignment assesses the basic concepts of programming; students should be able to demonstrate their achievements in the following unit learning outcomes:  a. Describe the fundamental principles of object-oriented programming;  b. Interpret a user’s needs while dealing with simple program specifications;  c. Design a simple class using UML notation;  d. Create a simple application based on UML design and the incremental development process of coding, debugging, and testing;  e. Apply basic control structures – sequence, repetition, and selection – to program development;  f. Produce simple interactive applications. |
| **Weight** | **35%** |
| **Total Marks** | 80 marks |
| **Word limit** | Not applicable |
| **Due Date** | Week 11 (03/Oct/2021) |
| **Submission Guidelines** | * All work must be submitted on Moodle by the due date along with a completed Assignment Cover Page. * The assignment must be in MS Word format, 1.5 spacing, 11-pt Calibri (Body) font and 2 cm margins on all four sides of your page with appropriate section headings. * Reference sources must be cited in the text of the report, and listed appropriately at the end in a reference list using IEEE referencing style. |
| **Extension** | If an extension of time to submit work is required, a Special Consideration Application must be submitted directly on AMS. You must submit this application three working days prior to the due date of the assignment. Further information is available at:  <https://www.mit.edu.au/about-us/governance/institute-rules-policies-and-plans/policies-procedures-and-guidelines/assessment-policy> |
| **Academic Misconduct** | Academic Misconduct is a serious offence. Depending on the seriousness of the case, penalties can vary from a written warning or zero marks to exclusion from the course or rescinding the degree. Students should make themselves familiar with the full policy and procedure available at:  <http://www.mit.edu.au/about-mit/institute-publications/policies-procedures-and-guidelines/Plagiarism-Academic-Misconduct-Policy-Procedure>.  For further information, please refer to the Academic Integrity Section in your Unit Description |

**Assignment 2 Title: Turtles and Tricks**

**Assignment Description:**

There are 2 questions in this assignment. This assignment will give you practice with interactive programs, if/else statements, list and classes. You are required to submit a report as mentioned in the marking guide. Also, a 5 min demonstration recording must be submitted with the required files: 1. Python scripts, 2. Report, 3. 5 Min demo video.

**Q 1:**

**Problem Description [10+2+3+3+2 =20 Marks]**

Stuck in the lockdown, a group of friends decided to do something fun remotely. They explored different apps and games to play online but one of them suggested why not use their programming knowledge from BN108 and come up with a game of their own. Thrilled with the idea, all of them agreed to work together to program their game. They planned to share the code on a common platform where everyone could access and review changes in the code. The platform they chose for code sharing was GitHub.

A class diagram tells you two things (for starters): what an object knows and what an object can do. A code template is given in the script *assignment2\_question1.py.* Explore and analyse the code and the class diagram for module turtle to understand the given problem.

Diagram

Description automatically generated

***Figure 1: Class diagram of class Turtle in turtle module*[[1]](#footnote-1)**

**Task Requirements**

**Task 1:** Create a flowchart based on the pseudocode given below:

A picture containing text

Description automatically generated

**Figure 2:** Pseudocode for how this racing game (Race to finish line) is going to work **[[2]](#footnote-2)**

**Task 2:** Add a new turtle with a colour of your choice. That’s your turtle.

**Task 3:** Modify the code to reflect the following:

1. Race to finish line: Use a random number generator to race towards the finish line
2. Circle race: Use a random number generator to complete a circle race. The turtle which completes a circle first wins.
3. Turtle wars: Position the turtles in opposite corners. Move the turtles towards one another at a speed determined by the random number generator. As soon as the turtles arrives at the same x,y co-ordinate, remove a random turtle. The turtle that stays wins. Print the colour of the winning turtle on the screen with a message.

**Q2:**

**Problem Description**

The students at MIT have a database. It can be designed in OOP paradigm by creating a class Student. The class holds the following information about each student: **studentName**, **studentID**, **unitsEnrolledList**.

The following two students are already in the database:

* Shankar, MIT123, [BN101,BN102,BN103]
* Danny, MIT321, [BN101, BN202, BN300]

**Task 1: [6+2+4+3 = 15Marks]**

1. Design the **class Student** with appropriate attributes and methods from the given scenario. Save this as a Student module (Student.py) which will be called in following questions.
2. Add a new attribute of your choice to this class.
3. Draw the UML of the class Student.
4. Load the pre-existing data: Import the module Student in a new script called dataProcessing.py . Create two objects *obj1, obj2,* with the info of both students given above. Append the objects in a list or dictionary.

**Task 2: [5+5+5+5 = 20Marks]**

1. Create a new method in class Student, ***common\_units(…)***. This method can tell a student who else is enrolled in the same units. **obj2.common\_units(obj1)**. (Hint: Compare the **unitsEnrolledList** and find out which items are common using the *in* keyword).

This method should process the result and output the following statement:

**Dear \_\_\_ , you and \_\_\_\_ are studying similar units: \_\_\_\_\_.**

***Example****: Dear Shankar, you and Danny are studying similar units: BN101.*

1. You need to create a new method in class Student, ***search\_student(studentID)***. This method can search for a student in the database.

Provide a pseudocode for searching a student in database.

1. Implement the code for ***search\_student(studentID)***

(Hint: Compare the **studentID** against each member inside the list. To access the **studentID** use the dot operator. To compare use the **in** keyword).

This method should process the result and output the following statement:

**Student information found in the DataBase**.

1. Draw the updated UML of class Student. This should include the new methods.

**Marking criteria:**

Marking criteria is shown in following table. Marks are allocated as follows:

|  |  |  |
| --- | --- | --- |
| **Description of criteria** | **Max marks** | **Marks obtained** |
|  |  |  |
| **Q1** | **20** |  |
| Task 1: Flowchart | 10 |  |
| Task 2: Add a new turtle | 2 |  |
| Task 3: Design 3 games | 8 |  |
| **Q2** | **35** |  |
| Task 1: Design and code: class, attribute, UML and list of student objects | 15 |  |
| Task 2: Pseudocode for search method  Incremental development:  Add 2 methods in the class and update the UML | 20 |  |
|  |  |  |
| **Report (10 marks)** | **10** |  |
| Program overview for both questions | 2+2 |  |
| Screen shots of the Q2 Testing output, e.g. validate user input in any scenario | 3 |  |
| Paste code into report | 3 |  |
|  |  |  |
| **Others (15 marks)** | **15** |  |
| Use comments, whitespace/layout properly to make program more readable | 5 |  |
| Demonstration 5 min video | 10 |  |
|  |  |  |
| **Total** | **80** |  |
| **Total** | **35%** |  |

**Marking Rubric for Assignment *#:* Total Marks 80**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Grade**  **Mark** | **HD**  **64-80** | **D**  **56-63** | **CR**  **48-55** | **P**  **40-47** | **Fail**  **<40** |
|  | **Excellent** | **Very Good** | **Good** | **Satisfactory** | **Unsatisfactory** |
| **Design /20** | Algorithm and flow chart have been well designed. All scenarios have been covered. UMLs are accurate. | Algorithm and flow chart have been well designed. Most of the scenarios have been covered. UMLs are well designed. | Algorithm / flow chart has been designed but only a few scenarios have been covered. UMLs are partially designed. | One of the designs is missing from Algorithm and flow chart . UMLs are satisfactory. | Algorithm and flow chart have been poorly designed or are missing. UMLs are not well designed. |
| **Implementation of Python functions**  **/35** | Design and implementation is well done and consistent with the task requirements. | Design and implementation is consistent with most of the task requirements. | Design and implementation is mostly consistent with the task requirements. | Design and implementation misses on some task requirements | No attempted or implementation is incorrect |
| **Report**  **/10** | Presentation, structure and the content (UML and Screen outputs) of the report is well done and consistent with the task requirements. | Presentation, structure and the content of the report is consistent with the task requirements. Some test cases are missing. | Presentation, structure and the content of the report is mostly consistent with the task requirements. | Presentation, structure and the content of the report misses on some task requirements | Not attempted or implementation is incorrect |
| **Others**  **/15** | Code layout, comments, meaningful variable names & compilation: well done and consistent with the tasks requirements and demo | Code layout, comments, meaningful variable names & compilation: very good and consistent with the tasks requirements | Code layout, comments, meaningful variable names & compilation:  generally consistent but can improve | Code layout, comments, meaningful variable names & compilation:  Some inconsistencies with the requirements | Code layout, comments, meaningful variable names & compilation: poor effort and program fails to compile |

1. Source: <https://learning.oreilly.com/library/view/head-first-learn/9781491958858/ch08.html#seven_modulescomma_methodscomma_classesco> [↑](#footnote-ref-1)
2. Source: <https://learning.oreilly.com/library/view/head-first-learn/9781491958858/ch08.html#seven_modulescomma_methodscomma_classesco> [↑](#footnote-ref-2)