Course Learning Outcomes:
Upon completion of this assignment you should be able to:

CLO1	Translate simple problem statements into programmable solutions using flow chart/pseudo code (C3, PLO2).
CLO2	Comprehend knowledge of basic and advanced programming concepts (C2, PLO1).
CLO3	Show the ability to write computer programs for a given problem statement (P4, PLO3).

1.0 INDIVIDUAL ASSIGNMENT DESCRIPTION

SPEED BOAT TICKETING SYSTEM FOR ISLAND HOPPING

A speed boating renting company has just purchased a computer for its new automated ticketing system. The company director has asked you to design the new system to assign seats for each tour trip of their 4-seater Business Class and 8-seater Economy Class speed boats. Assume the company has 10 speed boats (i.e. two 4-seater boats and eight 8-seater boats) each with a different Boat ID. The boats are scheduled at two-hour intervals from 8 am to 2pm daily (i.e. 8am, 10am, 12noon and 2pm). At every two-hour interval, starting from 8am everyday, the company plans to schedule one 4-seater boat and four 8-seater boats as each tour trip takes around 4 hours to complete.

BASIC REQUIREMENTS OF THE SYSTEM

1. **Main Menu**
   Your initial program design should display the following menu alternatives:

   SPEED BOAT TICKETING SYSTEM
   P – to Purchase Ticket
   V –to View Seating Arrangement
   Q – to Quit the system

2. **Submenu**
   The following submenu will be displayed when P is selected:

   PURCHASING MODULE
   B – to purchase ticket for Business class
   E – to purchase ticket for Economy class
   M – to return to Main Menu

3. **Assigning Seats**

If the person types B, then your program should assign a seat in the business class (seats B1 to B4). If the person types E, then your program should assign a seat in the economy class (seats E1 to E8). The following submenu will be displayed when V is selected:

SEATING ARRANGEMENT MODULE
S- to select Boat ID
T- to select Trip Time

4. **Boarding Ticket**

Your program should then print a boarding ticket indicating the person's name, seat number, whether it is in the business or economy class, date and time of departure and Boat ID.

5. **Seating Chart**

Use a list to represent the seating chart of the boat, indicating the availability of the seats within each trip of the speed boats. Initialize all the elements of the list to 0 to indicate that all seats are empty. As each seat is assigned, set the corresponding elements of the list to 1 to indicate that the seat is no longer available. Your program should never assign a seat that has already been assigned. The Boat ID will be requested when V is selected from the main menu and the seating arrangement for that boat will be displayed in a tabular form, e.g. if Boat ID of 01(or 02) is selected:

```
*************************************************************
*       Boat ID: 01                Date: 19 Sep 2020    Time: 8.00am *
*************************************************************
*       BUSINESS CLASS                                      *
*************************************************************
*             B1 - 1              *             B2 - 0      *
*************************************************************
*             B3 - 0              *             B4 - 1      *
*************************************************************
```

and if Boat ID of 03 (or 04 to 10):

```
*************************************************************
*       Boat ID: 03                Date: 19 Sep 2020     Time: 8.00am *
*************************************************************
*       ECONOMY CLASS                                       *
*************************************************************
*             E1 - 1              *             E2 - 1        *
*************************************************************
*             E3 - 0              *             E4 - 1        *
*************************************************************
*             E5 - 0              *             E6 - 0        *
*************************************************************
*             E7 - 1              *             E8 - 1        *
*************************************************************
```

2.0 REQUIREMENTS

i. You are required to carry out extra research for your system and document any logical assumptions you made after the research.

ii. Your program should use symbolic constants where appropriate. Validations need to be included to ensure the accuracy of the system. State any assumptions that you make under each function.

iii. You are required to store all data in text files. There is no limit on the number of text files that can be used but they should be kept minimum.

iv. You are expected to use list and functions in your program. Your program must embrace modular programming technique and should be menu-driven.

v. You may include any extra features which you may feel relevant and that add value to the system.

vi. There should be no need for graphics in your program, as what is being assessed, is your programming skill not the interface design. The marking

scheme for the assignment has been provided so that you clearly know how the assessment for this assignment would be done.

vii. You should include the good programming practice such as comments, variable naming conventions and indentation**.**

viii. In a situation where a student:

- ***Failed to attempt the assignment demonstration, overall marks awarded for the assignment will be adjusted to 50% of the overall existing marks.***
- ***Found to be involved plagiarism, the offence and will be dealt in accordance to APU regulations on plagiarism.***

ix. You are required to use Phyton programming language to implement the solution. Use of any other language like C/C++/Java is not allowed. Global variable is not allowed.

x. Results of a comprehensive testing is to be included in your document. The tests conducted shall take into consideration of all valid inputs and negative test cases.

3.0 DELIVERABLES

You are required to submit a softcopy of:

i. Program coded in Python – submitted as .py file.
   - Name the file under your name and TP number (e.g. KATHY_SIERRA_TP123456.py)
   - Start the first two lines in your program by typing your name and TP number (e.g. as follows):
     #KATHY SIERRA
     #TP123456

ii. Text files created through test data – submitted as .txt files.

iii. A documentation of the system – submitted as NAME_TPNUMBER.pdf file - that incorporates basic documentation standards such as header and footer, page numbering and includes:

- Cover page
- Table of contents
- Introduction and assumptions
- Design of the program – using pseudocode **and** flowcharts – which adheres to the requirements provided above
- Program source code and explanation
- Screenshots of sample input/output and explanation
- Conclusion
- References (if any) using Harvard Name Referencing

4.0 ASSESSMENT CRITERIA

i. Design (Pseudocode and Flowchart) 30%

Detailed, logical and accurate design of programmable solution.

ii. Coding / Implementation (Python code) 30%

Application of Python programming techniques (from basic to advance); good programming practices in implementing the solution as per design; and adequate validation meeting all system requirements with all possible additional features.

iii. Documentation 25%

Adherence to document standard format and structure; screen captures of input/output with explanation; and inclusion of generated text files.

iv. Demonstration 15%

Ability to run, trace code, explain work done and answer questions.

5.0 PERFORMANCE CRITERIA

Distinction (80% and above)

This grade will be assigned to work which meets all of the requirements stated in the question. The program runs smoothly when executed. There is clear evidence and application of Python concepts up to advanced level. The program solution is unique with excellent coding styles and validation. The program implemented maps completely against the design (pseudocode and flowchart) as seen in the documentation. The design of the solution varies in styles and has unique logic with hardly any errors / omissions. The documentation does not have any missing components. Sample inputs/outputs documented have clear explanation. Student must be able to provide excellent explanation of the codes and work done, show additional concepts / new ideas used in the solution, able to answer all questions posed with accurate / logical answers / explanation provided with sound arguments and clear discussion. Overall an excellent piece of work submitted.

Credit (65%-74%)

This grade will be assigned to work which is considered to be of good standard and meets most of the requirements stated in the question. The program runs smoothly when executed. There is clear evidence and application of Python concepts up to at least intermediate level. The program solution is unique with good coding styles and validation. The program implemented maps well against the design (pseudocode and flowchart) as seen in the documentation. The design of the solution varies in styles and has unique logic with minor errors / omissions. The documentation does not have any missing components. Sample inputs/outputs documented with some explanation. Student must be able to provide good explanation of the codes and work done, answer most questions posed with mostly accurate / logical answers / explanation. Overall a good assignment submitted.

Pass (50%-64%)

This grade will be assigned to work which meets at least half of the basic requirements (approximately 50%) stated in the questions. The program runs smoothly when executed. There is clear evidence and application of Python concepts at basic level. The program solution is common with basic coding styles and validation. The program implemented somewhat maps with the design (pseudocode and flowchart) as seen in the documentation. The design of the solution is average in terms of logic and style with some errors / omissions. The documentation has some missing components. Sample inputs/outputs documented but without any explanation. Student must be able to explain some codes and work done and able to answer some questions posed with some accurate / logical answers / explanation. Overall an average piece of work submitted.

Fail (Below 50%)

This grade will be assigned to work which achieved less than half of the requirements stated in the question. The program is able to compile but not able to execute or with major errors. The program solution has only basic coding styles with no validation. The program solution has little or no mapping with the design. The design of the solution has major / obvious errors / omissions. The documentation has some missing essential components. Student is barely able to explain the codes / work done and answer given on the questions posed but with mostly inaccurate / illogical answers / explanation. Overall a poor piece of work submitted.