NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Laboratory Manual
for
SC1003
Introduction to Computational Thinking and Programming

Practical Exercise #2:
Basic Python Programming -
Pseudo Code, Flowcharts, Data Types, Variables, Input, and Output

COLLEGE OF COMPUTING AND DATA SCIENCE

NANYANG TECHNOLOGICAL UNIVERSITY

Ex. #2 – Basic Python Programming

Learning Objectives

- Understand and practice writing pseudo code.
- Learn how to create flowcharts.
- Gain knowledge about data types and variables.
- Practice input and output operations in programming using the context of a Battleships game.

Equipment and accessories required

PC/notebook with python compiler (eg python IDLE)

Battleships is a strategic game where two players, user and computer, place and attack ships on a grid representing the ocean. The objective is to sink all of the opponent's ships before they sink yours.

Setting Up the Game

1. **Board Layout:**
    - The game board is a 10x10 grid with two layers: the surface (depth = 1) and underwater (depth = 0).
    - Rows are labeled A-J (or a-j), and columns are numbered 1-10.
2. **Ships:**
    - Each player has a set of ships to place on the board.
    - Example ships: Submarine (3 units long) and Carrier (4 units long).
    - The Submarine can be placed on both layers, while the Carrier can only be placed on the surface.

Placing Ships

1. **User Ship Placement:**
    - The user is prompted to place their ships by specifying coordinates and orientation (vertical or horizontal).
    - Example input format: A,4,1 (row, column, depth).
    - The coordinates must be valid, and the ships cannot overlap or go out of bounds.
    - The user is shown their board after each placement.
2. **Computer Ship Placement:**
    - The computer randomly places its ships on the board.
    - The placement is validated to ensure no overlaps or out-of-bound positions.

Ex. #2 – Basic Python Programming

Taking Turns

1. **User Move:**
    - The user is prompted to enter the coordinates for their attack.
    - Example input format: A,4,1 (row, column, depth).
    - The result of the attack is displayed: "hit", "miss", or "redundant" (if the area was already attacked).
    - If all parts of a ship are hit, the ship is marked as sunk.
2. **Computer Move:**
    - The computer randomly selects coordinates to attack.
    - The result of the attack is displayed: "hit" or "miss".
    - If all parts of a ship are hit, the ship is marked as sunk.

Winning the Game

- The game continues in turns until one player sinks all of the opponent's ships.
- The first player to sink all of the opponent's ships wins the game.

Additional Rules

- **Depth Representation:**
    - Depth 0 represents the underwater layer.
    - Depth 1 represents the surface layer.
- **Input Validation:**
    - Coordinates must be entered in the correct format.
    - Row inputs must be between A-J (or a-j), column inputs must be between 1-10, and depth inputs must be either 0 or 1.
    - Invalid entries prompt the user to re-enter the coordinates.
- **Ship Orientation:**
    - Ships can be placed vertically or horizontally.
    - Vertical placement extends downwards from the starting coordinate.
    - Horizontal placement extends rightwards from the starting coordinate.

Understand the logic of Battleship rules and complete the following exercises.

Exercises:

1. Pseudo Code:
    - Write pseudo code for initializing the game board.

- Write pseudo code for placing a ship on the board.

2. Flowcharts:
   - Create a flowchart for initializing the game board.
   - Create a flowchart for the process of placing a ship on the board.
3. Data Types and Variables:
   - Define the data types and variables required for the Battleships game (e.g., board, ships, coordinates).
   - Write a short program to declare these variables and print their initial values.
4. Input and Output Operations:
   - Take user input for attack coordinates and display the result.

If you are new to Python programming, copy the following hint code to your IDE, e.g., IDLE, follow the TODO task lists and sample output as follows to complete the exercises.

```
# TODO : 1. There are two players in the game. Initialize their names as string

# Add you code of TODO 1 here


# TODO : 2. There are two layers in the sea. Initialize the integer variable layer

# Add you code of TODO 2 here


# TODO : 3. The battleship board is 10 by 10 dimension. Initialize the integer variables

#  width and height

# Add you code of TODO 3 here


# TODO : 4. Initialize a boolean variable that will be used to indicate

#  whether user input is valid or not, two boolean variables hit and miss that indicate

#  whether the ship is hit or missed

# Add you code of TODO 4 here
```

Ex. #2 – Basic Python Programming

```
# TODO : 5. The ships have only two orientation, either vertical or horizontal.

#  Initialize a variable as ship orientation

# Add you code of TODO 5 here


# TODO : 6. The coordinates of where to put ship or where to hit on the board have a

#  specific format. Create a string variable as coordinates

# Add you code of TODO 6 here


# TODO : 7. There are only two types of ships. Initialize two string variables as ship names

# Add you code of TODO 7 here


# TODO : 8. Initialize a string variable holding a welcome message that can be displayed to user

# Add you code of TODO 8 here


print(player1, player2)

print(layer)

print(width, height)

print(valid, hit, miss)

print(ori)

print(coor)

print(ship1, ship2)

print(welmes)


# TODO : 9. Take user input for attack coordinates and display the result.

# Add you code of TODO 9 here
```

Ex. #2 – Basic Python Programming

Sample output:

Computer User

2

10 10

False False False

v

A,4,1

submarine carrier

Welcome to Battleships! Hit ENTER to continue.

Please enter coordinate of the attack center point in following this format (row,col,depth). E.g. A,4,1

Note: depth = 0 represents the subsea layer, and depth = 1 represents the surface level.

Enter coordinates: B,3,0

Hit at area centering B,3,0