Lab no: 1

Title: WAP to create a tic tac toe game in python.

Tic-Tac-Toe

Tic -tac-toe is a simple 2-player game played on a 3×3 grid. Players take turns marking a cell with either X or O. The first player to align **three of their symbols** (horizontally, vertically, or diagonally) wins. If the grid is full and no one has won, it’s a draw.

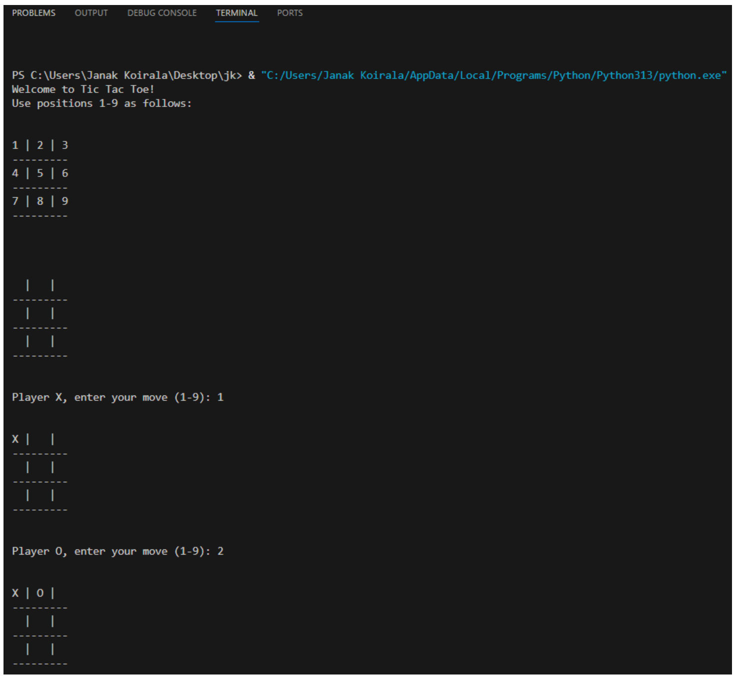
It’s a great beginner project to practice:

- Loops
- Conditionals
- Lists
- Functions

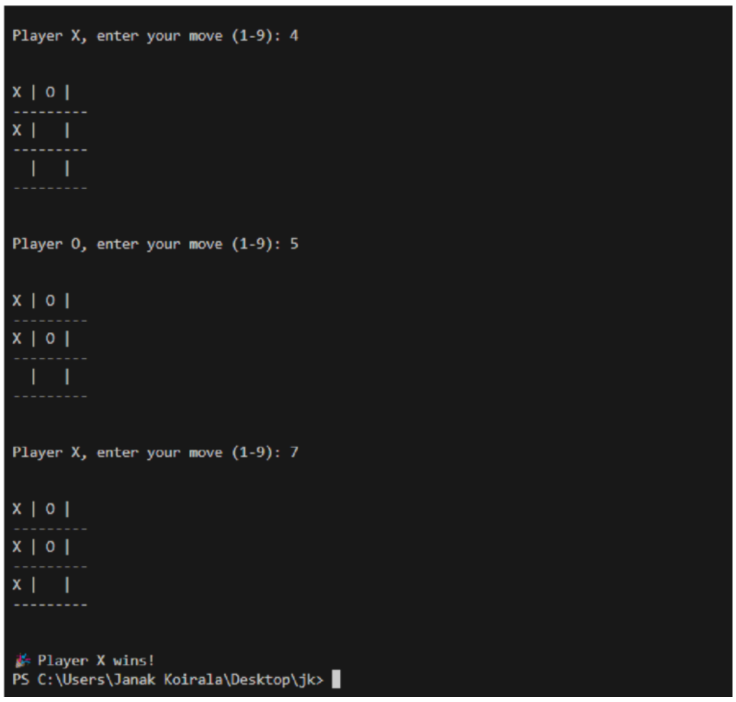
Source code:

```
def print_board(board):
    print("\n")
    for row in board:
        print(" | ".join(row))
        print("-" * 9)
    print("\n")

def check_winner(board, player):
    # Rows, columns, diagonals
    for row in board:
        if all(cell == player for cell in row):
            return True
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True
    if all(board[i][i] == player for i in range(3)):
        return True
    if all(board[i][2 - i] == player for i in range(3)):
        return True
    return False

def is_draw(board):
    return all(cell in ['X', 'O'] for row in board for cell in row)

def get_move(player, board):
    while True:
        try:
            move = int(input(f"Player {player}, enter your move (1-9): "))
            if move < 1 or move > 9:
                print("Invalid move. Choose a number between 1 and 9.")
                continue
            row = (move - 1) // 3
            col = (move - 1) % 3
            if board[row][col] == ' ':
                return row, col
            else:
                print("Cell already taken. Choose another.")
        except ValueError:
            print("Invalid input. Please enter a number between 1 and 9.")

def tic_tac_toe():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    current_player = 'X'

    print("Welcome to Tic Tac Toe!")
    print("Use positions 1-9 as follows:")
    print_board([['1','2','3'], ['4','5','6'], ['7','8','9']])

    while True:
        print_board(board)
        row, col = get_move(current_player, board)
        board[row][col] = current_player

        if check_winner(board, current_player):
            print_board(board)
            print(f" Player {current_player} wins!")
            break
        if is_draw(board):
            print_board(board)
            print("It's a draw!")
            break

        current_player = 'O' if current_player == 'X' else 'X'

# Run the game
tic_tac_toe()
```

Output:



```
PS C:\Users\Janak Koirala\Desktop\jk> & "C:/Users/Janak Koirala/AppData/Local/Programs/Python/Python313/python.exe"
Welcome to Tic Tac Toe!
Use positions 1-9 as follows:


1 | 2 | 3
---------
4 | 5 | 6
---------
7 | 8 | 9
---------



  |   |
---------
  |   |
---------
  |   |
---------


Player X, enter your move (1-9): 1

X |   |
---------
  |   |
---------
  |   |
---------


Player O, enter your move (1-9): 2

X | O |
---------
  |   |
---------
  |   |
---------
```



```
Player X, enter your move (1-9): 4

X | O |
---------
X |   |
---------
  |   |
---------


Player O, enter your move (1-9): 5

X | O |
---------
X | O |
---------
  |   |
---------


Player X, enter your move (1-9): 7

X | O |
---------
X | O |
---------
X |   |
---------


🎉 Player X wins!
PS C:\Users\Janak Koirala\Desktop\jk> ▌
```

Lab no: 2

Title: WAP to create water jug puzzle in python.

Water Jug Problem :

The **Water Jug Problem** is a classic puzzle where you're given two jugs of different capacities and an unlimited supply of water. The goal is to measure an exact amount of water using only these two jugs.

You can:

- Fill a jug completely
- Empty a jug
- Pour water from one jug to the other until one is full or the other is empty

This problem is often used to teach logical reasoning, problem-solving, and state-space search in computer science and artificial intelligence.

Problem Statement (Most Common Version)

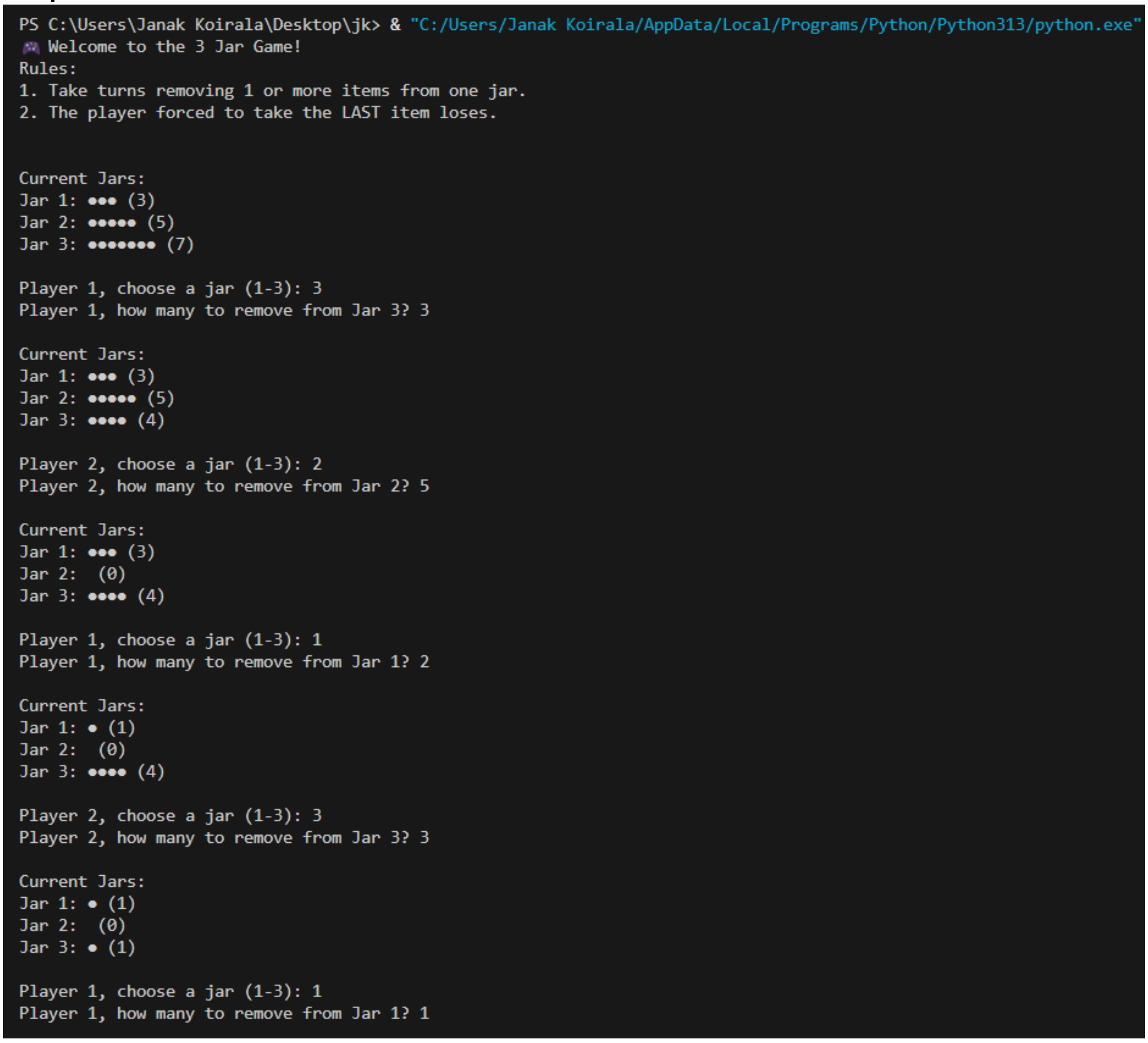
You are given:

- A **5-liter** jug
- A **3-liter** jug
- An unlimited water supply
- Both jugs are initially empty

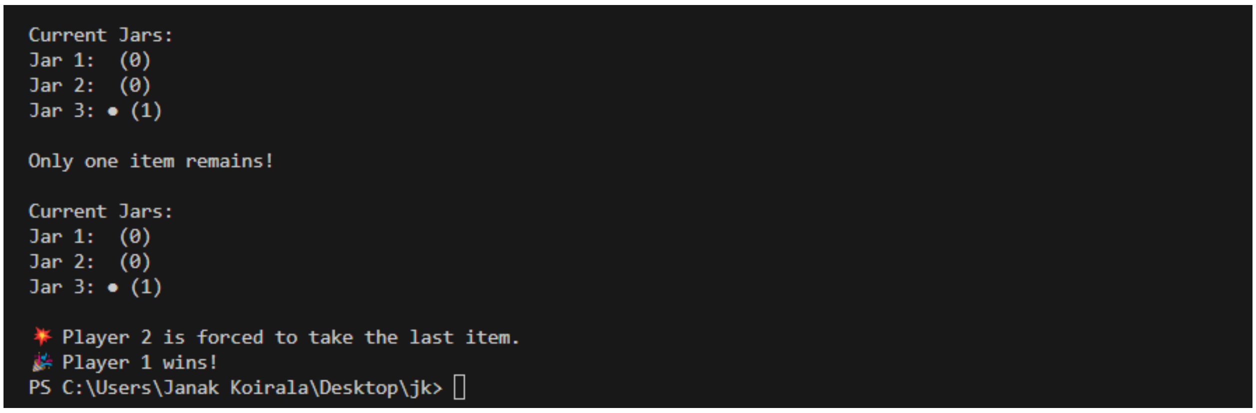
Source code:

```
def print_jars(jars):
    print("\nCurrent Jars:")
    for i, count in enumerate(jars, start=1):
        print(f"Jar {i}: {'●' * count} ({count})")
    print()

def get_move(player, jars):
    while True:
        try:
            jar = int(input(f"Player {player}, choose a jar (1-3): "))
            if jar not in [1, 2, 3]:
                print("Invalid jar number. Choose 1, 2, or 3.")
                continue
            if jars[jar - 1] == 0:
                print("That jar is empty. Choose another.")
                continue
            count = int(input(f"Player {player}, how many to remove from Jar {jar}? "))
            if not (1 <= count <= jars[jar - 1]):
                print(f"Choose between 1 and {jars[jar - 1]}.")
                continue
            return jar - 1, count
        except ValueError:
            print("Please enter valid integers.")

def is_game_over(jars):
    return sum(jars) == 1

def three_jar_game():
    jars = [3, 5, 7]
    current_player = 1

    print(" Welcome to the 3 Jar Game!")
    print("Rules:\n1. Take turns removing 1 or more items from one jar.")
    print("2. The player forced to take the LAST item loses.\n")

    while True:
        print_jars(jars)

        if is_game_over(jars):
            print("Only one item remains!")
            print_jars(jars)
            print(f" Player {current_player} is forced to take the last item.")
            print(f" Player {3 - current_player} wins!")
            break

        jar_index, remove_count = get_move(current_player, jars)
        jars[jar_index] -= remove_count
        current_player = 3 - current_player  # Switch players

three_jar_game()
```

Output:



```
PS C:\Users\Janak Koirala\Desktop\jk> & "C:/Users/Janak Koirala/AppData/Local/Programs/Python/Python313/python.exe"
🎮 Welcome to the 3 Jar Game!
Rules:
1. Take turns removing 1 or more items from one jar.
2. The player forced to take the LAST item loses.


Current Jars:
Jar 1: ●●● (3)
Jar 2: ●●●●● (5)
Jar 3: ●●●●●●● (7)

Player 1, choose a jar (1-3): 3
Player 1, how many to remove from Jar 3? 3

Current Jars:
Jar 1: ●●● (3)
Jar 2: ●●●●● (5)
Jar 3: ●●●● (4)

Player 2, choose a jar (1-3): 2
Player 2, how many to remove from Jar 2? 5

Current Jars:
Jar 1: ●●● (3)
Jar 2:  (0)
Jar 3: ●●●● (4)

Player 1, choose a jar (1-3): 1
Player 1, how many to remove from Jar 1? 2

Current Jars:
Jar 1: ● (1)
Jar 2:  (0)
Jar 3: ●●●● (4)

Player 2, choose a jar (1-3): 3
Player 2, how many to remove from Jar 3? 3

Current Jars:
Jar 1: ● (1)
Jar 2:  (0)
Jar 3: ● (1)

Player 1, choose a jar (1-3): 1
Player 1, how many to remove from Jar 1? 1
```



```
Current Jars:
Jar 1:  (0)
Jar 2:  (0)
Jar 3: ● (1)

Only one item remains!

Current Jars:
Jar 1:  (0)
Jar 2:  (0)
Jar 3: ● (1)

💥 Player 2 is forced to take the last item.
🎉 Player 1 wins!
PS C:\Users\Janak Koirala\Desktop\jk> 
```

Lab no: 3

Title: WAP to create Chatbot in python.

Chatbot:

Chatbot is a computer program that uses rules or artificial intelligence to communicate with people automatically, usually through text or voice, to answer questions or help with tasks.

- It **doesn't learn or adapt** like AI chatbots.
- Only works **within the limits** of the programmed rules.

Purpose of the Program

This chatbot, named **Jarvis**, is a basic conversational agent implemented in Python. It interacts with users by recognizing predefined keywords or phrases and responding with appropriate answers. The chatbot can greet users, respond to simple questions, tell jokes, and remember the user's name during the conversation

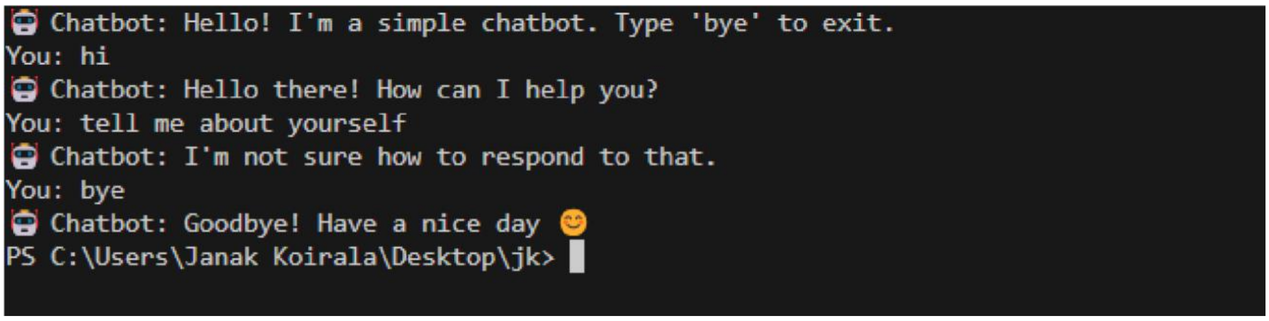
Key Features

- **Pattern Matching**: The chatbot matches user input against keys in a dictionary to generate responses.
- **State Maintenance**: Remembers the user's name once provided.
- **Simple Natural Language Handling**: Handles common phrases and can recognize commands like "my name is" or "what is my name."
- **Exit Condition**: Allows graceful exit on user command

Source code:

```
def chatbot():
    print("🤖 Chatbot: Hello! I'm a simple chatbot. Type 'bye' to exit.")

    while True:
        user_input = input("You: ").lower()

        if user_input in ['bye', 'exit', 'quit']:
            print("🤖 Chatbot: Goodbye! Have a nice day 😊")
            break
        elif "hello" in user_input or "hi" in user_input:
            print("🤖 Chatbot: Hello there! How can I help you?")
        elif "how are you" in user_input:
            print("🤖 Chatbot: I'm just code, but thanks for asking! How are you?")
        elif "name" in user_input:
            print("🤖 Chatbot: I'm a simple Python chatbot.")
        elif "help" in user_input:
            print("🤖 Chatbot: I can respond to greetings, tell you my name, and say goodbye.")
        else:
            print("🤖 Chatbot: I'm not sure how to respond to that.")

# Run the chatbot
chatbot()
```

Output:



```
🤖 Chatbot: Hello! I'm a simple chatbot. Type 'bye' to exit.
You: hi
🤖 Chatbot: Hello there! How can I help you?
You: tell me about yourself
🤖 Chatbot: I'm not sure how to respond to that.
You: bye
🤖 Chatbot: Goodbye! Have a nice day 😊
PS C:\Users\Janak Koirala\Desktop\jk>
```