**Question 1: String Operations**

Write a Python script that performs the following:

* Takes a string input from the user.
* Reverses the string and prints the result.
* Converts the string to uppercase and lowercase.
* Counts the number of vowels in the string.

**Answer:**

**Inputs:**

**# Taking input from the user**

user\_string = input("Enter a string: ")

**# Reversing the string**

reversed\_string = user\_string[::-1]

print("Reversed String:", reversed\_string)

**# Converting to uppercase and lowercase**

uppercase\_string = user\_string.upper()

lowercase\_string = user\_string.lower()

print("Uppercase:", uppercase\_string)

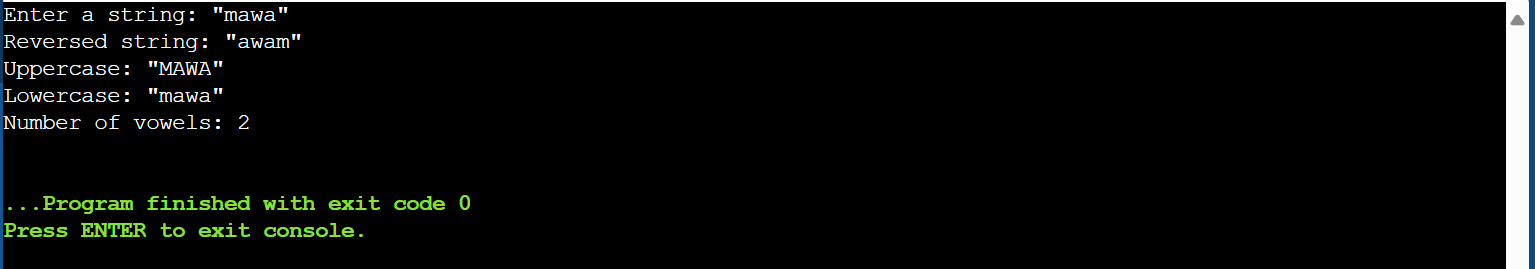
print("Lowercase:", lowercase\_string)

**# Counting the number of vowels**

vowels = "aeiouAEIOU"

vowel\_count = sum(1 for char in user\_string if char in vowels)

print("Number of vowels:", vowel\_count)

**Output:**

**Question 2: List and Tuple Operations**

Write a Python program to solve the following:

* Take a list of integers as input from the user.
* Find and print the largest and smallest numbers in the list.
* Convert the list into a tuple and display it.
* Access and print the third element of the tuple.

**Answer:**

**Inputs**

**# Taking a list of integers as input from the user**

user\_list = list(map(int, input("Enter a list of integers separated by spaces: ").split()))

**# Finding the largest and smallest numbers**

largest = max(user\_list)

smallest = min(user\_list)

print("Largest number:", largest)

print("Smallest number:", smallest)

**# Converting the list into a tuple**

user\_tuple = tuple(user\_list)

print("Tuple:", user\_tuple)

**# Accessing and printing the third element of the tuple**

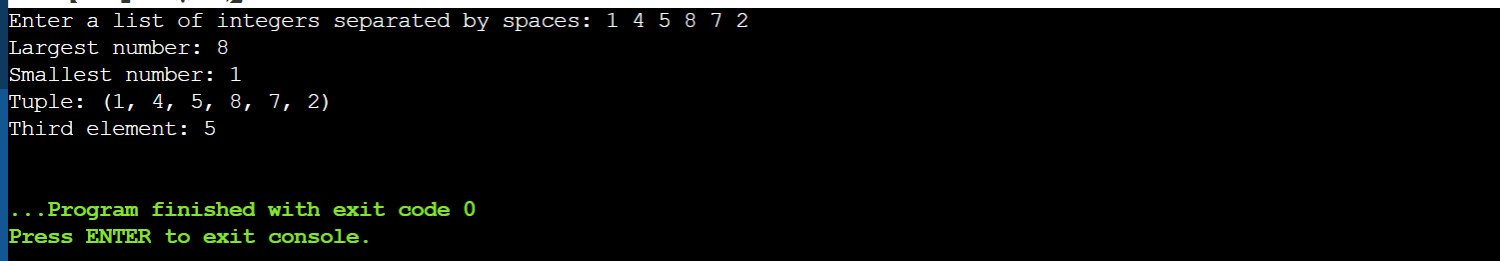
if len(user\_tuple) >= 3:

print("Third element of the tuple:", user\_tuple[2])

else:

print("The tuple has less than 3 elements.")

Output:



**Question 3: Tic-Tac-Toe Mini-Game**

Write a Python program to implement a simple two-player Tic-Tac-Toe game:

* Use a 3x3 grid represented by a list of lists.
* Allow players to take turns, entering their moves.
* Check for a winner after each move or a tie if the board is full.

**Answer:**

**Inputs**

**# Function to display the Tic-Tac-Toe board**

def display\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 9)

**# Function to check for a winner**

def check\_winner(board, player):

# Check rows, columns, and 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)) or all(board[i][2 - i] == player for i in range(3)):

return True

return False

**# Function to check if the board is full**

def is\_board\_full(board):

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

# Main game function

def play\_tic\_tac\_toe():

board = [[" " for \_ in range(3)] for \_ in range(3)]

players = ["X", "O"]

turn = 0

while True:

display\_board(board)

print(f"Player {players[turn % 2]}'s turn.")

**# Taking input from the user**

while True:

try:

row, col = map(int, input("Enter row and column (0-2, separated by space): ").split())

if board[row][col] == " ":

board[row][col] = players[turn % 2]

break

else:

print("Cell already occupied! Try again.")

except (ValueError, IndexError):

print("Invalid input! Enter row and column between 0 and 2.")

**# Check for a winner**

if check\_winner(board, players[turn % 2]):

display\_board(board)

print(f"Player {players[turn % 2]} wins!")

break

**# Check for a tie**

if is\_board\_full(board):

display\_board(board)

print("It's a tie!")

break

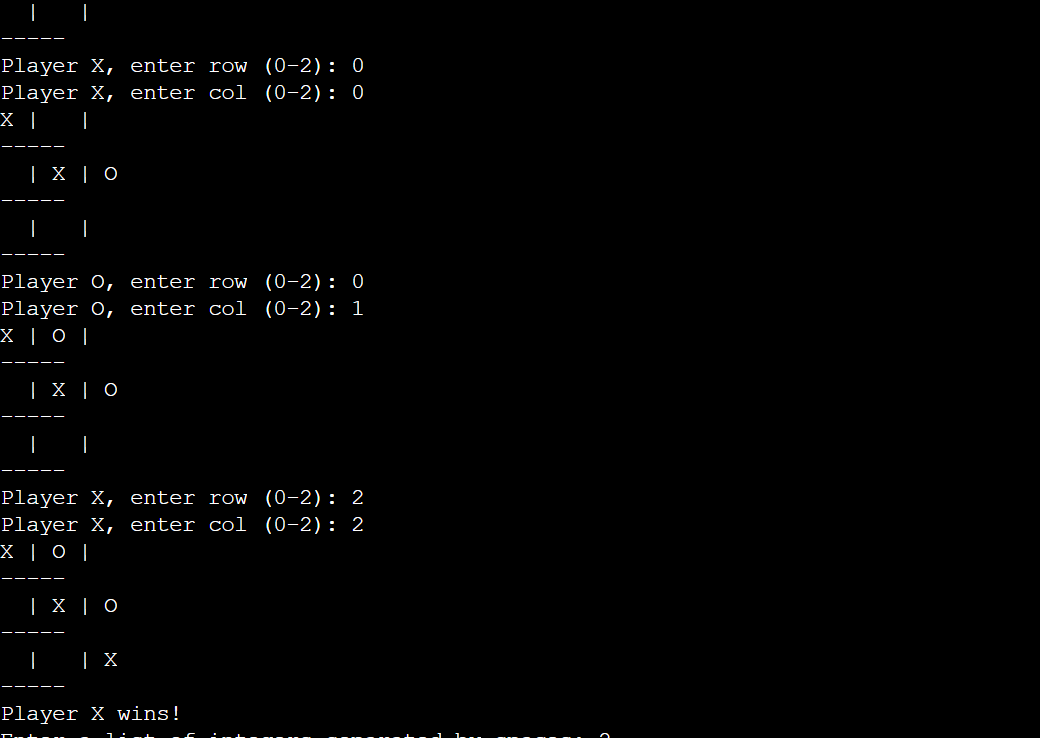
turn += 1

**# Run the game**

if \_\_name\_\_ == "\_\_main\_\_":

play\_tic\_tac\_toe()

Output:



**Question 4: Rock-Paper-Scissors Mini-Game**

Write a Python program for a Rock-Paper-Scissors game:

* Allow a user to play against the computer.
* Use random to generate the computer's choice.
* Compare the user’s and computer’s choices to determine the winner or if it’s a draw.
* Announce the result at the end of each round.

**Answer:**

**Inputs**

**# Function to get the computer's choice**

def get\_computer\_choice():

return random.choice(["rock", "paper", "scissors"])

**# Function to determine the winner**

def determine\_winner(user\_choice, computer\_choice):

if user\_choice == computer\_choice:

return "It's a draw!"

elif (user\_choice == "rock" and computer\_choice == "scissors") or \

(user\_choice == "paper" and computer\_choice == "rock") or \

(user\_choice == "scissors" and computer\_choice == "paper"):

return "You win!"

else:

return "Computer wins!"

**# Main function to play the game**

def play\_rock\_paper\_scissors():

print("Welcome to Rock-Paper-Scissors!")

choices = ["rock", "paper", "scissors"]

while True:

user\_choice = input("Enter rock, paper, or scissors (or 'quit' to exit): ").lower()

if user\_choice == "quit":

print("Thanks for playing! Goodbye!")

break

elif user\_choice not in choices:

print("Invalid choice. Please try again.")

continue

computer\_choice = get\_computer\_choice()

print(f"Computer chose: {computer\_choice}")

result = determine\_winner(user\_choice, computer\_choice)

print(result)

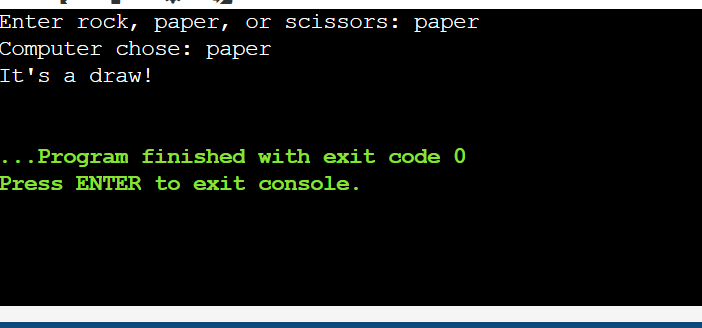
print("-" \* 30)

**# Run the game**

if \_\_name\_\_ == "\_\_main\_\_":

play\_rock\_paper\_scissors()

**Output:**



THE END