**Question 1: String Operations                                                                              20**

**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.

**Deliverable:**Submit the Python script and a screenshot of the output.

**Solution:-**

**Python script:**

# Take input from the user

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

# Reverse the string

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

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

# Convert to uppercase

uppercase\_string = user\_input.upper()

print("Uppercase String:", uppercase\_string)

# Convert to lowercase

lowercase\_string = user\_input.lower()

print("Lowercase String:", lowercase\_string)

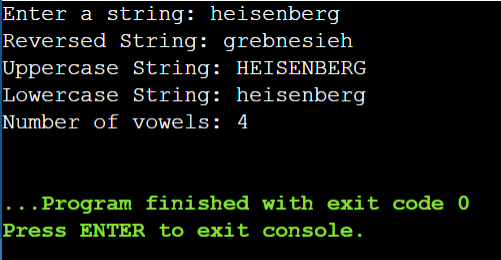
# Count the number of vowels

vowels = "aeiouAEIOU"

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

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

**Output:**

****

**Question 2: List and Tuple Operations                                                               20**

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.

**Deliverable:**  
Submit the Python script and screenshots of outputs.

**Solution:-**

**Python script:**

# Take a list of integers as input from the user

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

# Find and print the largest and smallest numbers in the list

largest\_num = max(num\_list)

smallest\_num = min(num\_list)

print("Largest number:", largest\_num)

print("Smallest number:", smallest\_num)

# Convert the list into a tuple and display it

num\_tuple = tuple(num\_list)

print("Tuple:", num\_tuple)

# Access and print the third element of the tuple (if it exists)

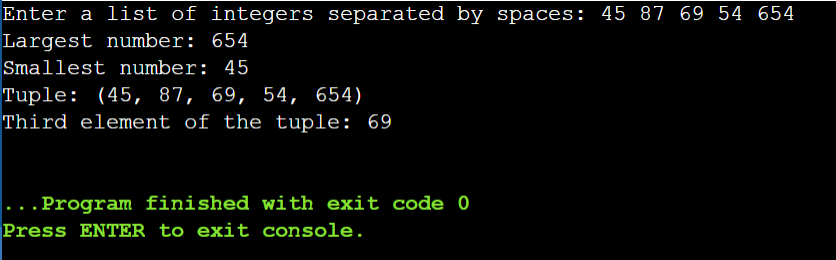
if len(num\_tuple) >= 3:

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

else:

print("The tuple does not have a third element.")

**Output:**



**Question 3: Tic-Tac-Toe Mini-Game                                                                   30**

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.

**Deliverable:**Submit Python scripts for both games and screenshots of gameplay results.

**Solution:-**

**Python script:**

def print\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 9)

def check\_winner(board, player):

# Check rows and columns

for i in range(3):

if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j in range(3)):

return True

# Check diagonals

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

def is\_full(board):

return all(cell in ['Dipon', 'Opponent'] for row in board for cell in row)

def tic\_tac\_toe():

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

players = ['Dipon', 'Opponent']

turn = 0

while True:

print\_board(board)

row, col = map(int, input(f"Player {players[turn % 2]}, enter row and column (0-2): ").split())

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

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

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

print\_board(board)

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

break

if is\_full(board):

print\_board(board)

print("It's a tie!")

break

turn += 1

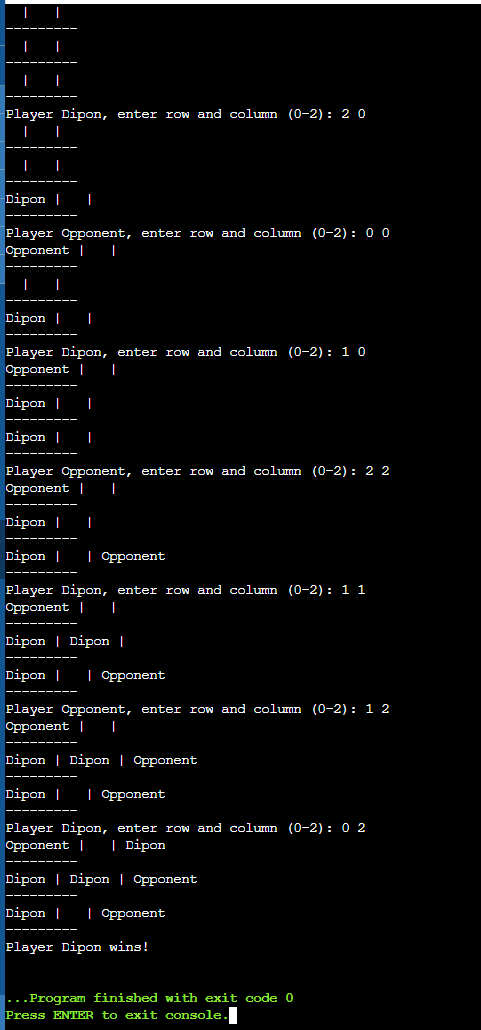
else:

print("Invalid move, try again.")

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

tic\_tac\_toe()

**Output:**



**Question 4: Rock-Paper-Scissors Mini-Game                                                    30**

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.

**Deliverable:**Submit Python scripts for both games and screenshots of gameplay results.

**Solution:-**

**Python script:**

import random

def get\_computer\_choice():

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

def get\_user\_choice():

while True:

choice = input("Enter rock, paper, or scissors: ").strip().lower()

if choice in ["rock", "paper", "scissors"]:

return choice

print("Invalid choice, please try again.")

def determine\_winner(user, computer):

if user == computer:

return "It's a draw!"

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

(user == "scissors" and computer == "paper") or \

(user == "paper" and computer == "rock"):

return "You win!"

else:

return "Computer wins!"

def play\_game():

user\_choice = get\_user\_choice()

computer\_choice = get\_computer\_choice()

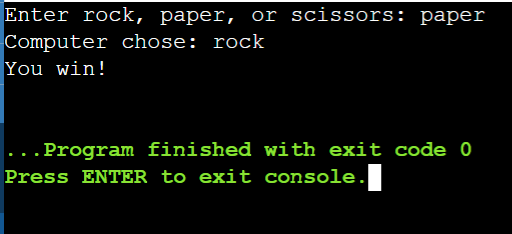
print(f"Computer chose: {computer\_choice}")

print(determine\_winner(user\_choice, computer\_choice))

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

play\_game()

**Output:**

****