Data Science and Gen AI LLMs

Name: Dr P Sumalatha (2406DGAL147)

Subject: Python Programming

Assignment Questions:

Question 1:

Number game between user and computer. The user starts by entering either 1 or 2 or 3 digits starting from 1 sequentially. The computer can return either 1 or 2 or 3 next digits in sequence, starting from the max number played by the user. User enters the next 1 or 2 or 3 next digits in sequence, starting from the max number played by the computer. Whoever reaches 20 first wins the game.

Note:

- the numbers should be in sequence starting from 1.

- minimum number user or computer should pick is at least 1 digit in sequence

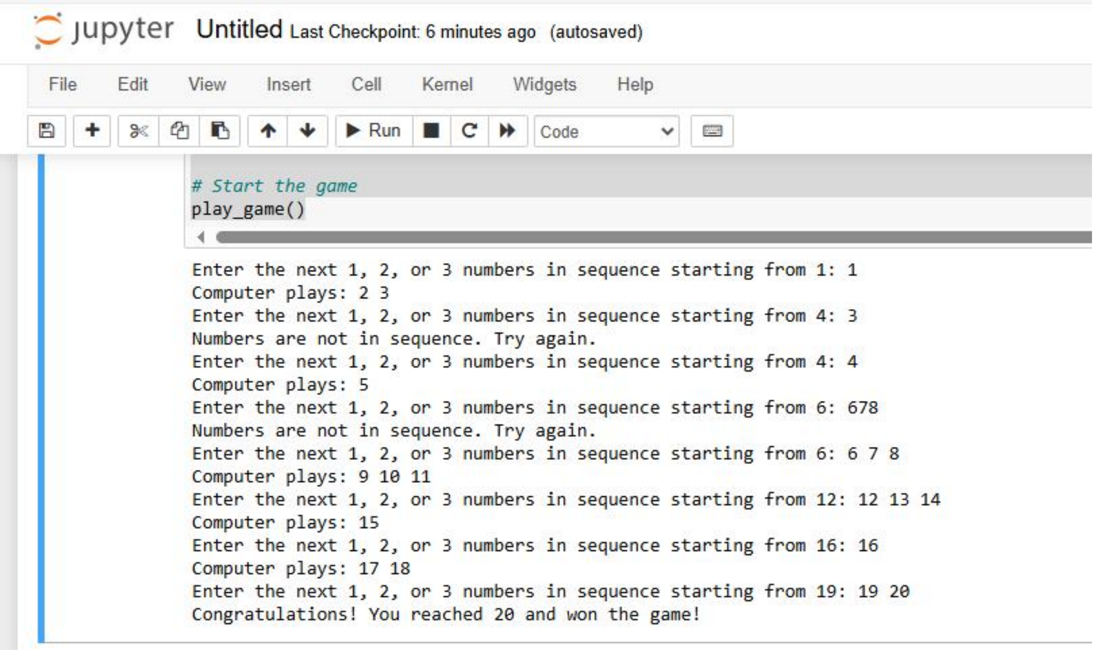
- maximum number user or computer can pick only 3 digits in sequence

Answer:

```
import random

def user_turn(current_number):
    while True:
        try:
            user_input = input(f"Enter the next 1, 2, or 3 numbers in sequence starting from {current_number + 1}: ")
            user_numbers = list(map(int, user_input.split()))
            if len(user_numbers) < 1 or len(user_numbers) > 3:
                print("You must enter 1, 2, or 3 numbers.")
                continue
            if user_numbers[0] != current_number + 1 or user_numbers != list(range(current_number + 1, current_number + 1 + len(user_numbers))):
                print("Numbers are not in sequence. Try again.")
                continue
            return user_numbers[-1]
        except ValueError:
            print("Invalid input. Enter numbers only.")
```

```
def computer_turn(current_number):
    numbers_to_play = random.randint(1, 3)
    computer_numbers = list(range(current_number + 1, current_number + 1 + numbers_to_play))
    print(f"Computer plays: {' '.join(map(str, computer_numbers))}")
    return computer_numbers[-1]
def play_game():
    current_number = 0
    while current_number < 20:
        current_number = user_turn(current_number)
        if current_number >= 20:
            print("Congratulations! You reached 20 and won the game!")
            break
        current_number = computer_turn(current_number)
        if current_number >= 20:
            print("Computer reached 20. You lose!")
            break
# Start the game
play_game()
```

OUTPUT:



```
# Start the game
play_game()
```

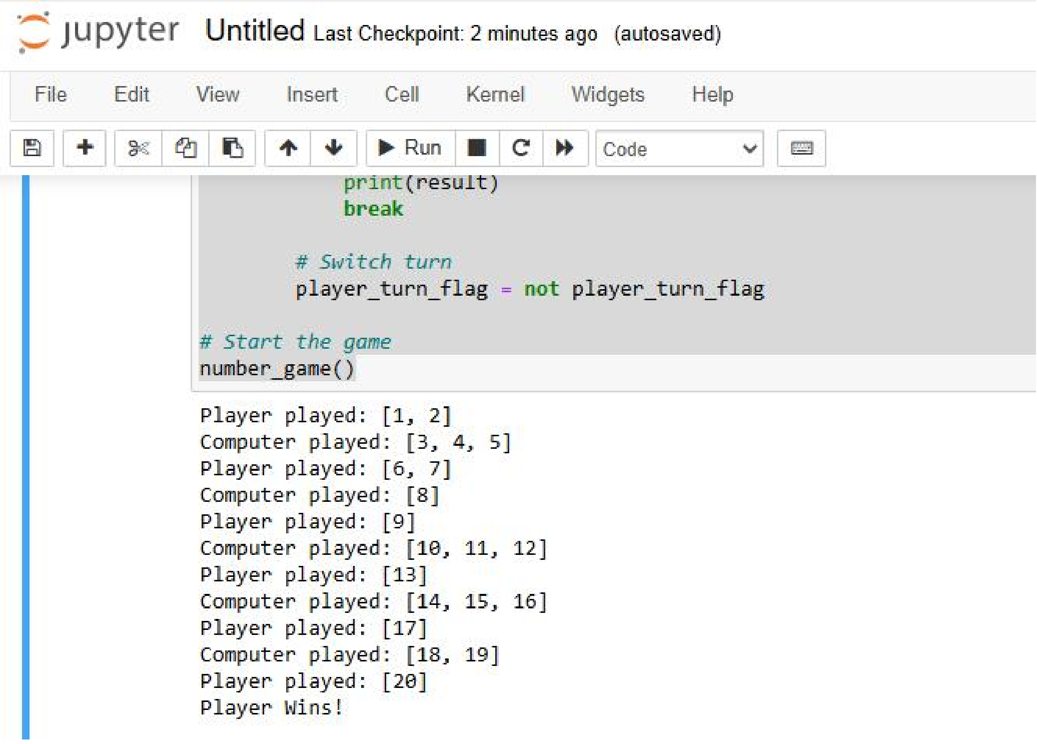
```
Enter the next 1, 2, or 3 numbers in sequence starting from 1: 1
Computer plays: 2 3
Enter the next 1, 2, or 3 numbers in sequence starting from 4: 3
Numbers are not in sequence. Try again.
Enter the next 1, 2, or 3 numbers in sequence starting from 4: 4
Computer plays: 5
Enter the next 1, 2, or 3 numbers in sequence starting from 6: 678
Numbers are not in sequence. Try again.
Enter the next 1, 2, or 3 numbers in sequence starting from 6: 6 7 8
Computer plays: 9 10 11
Enter the next 1, 2, or 3 numbers in sequence starting from 12: 12 13 14
Computer plays: 15
Enter the next 1, 2, or 3 numbers in sequence starting from 16: 16
Computer plays: 17 18
Enter the next 1, 2, or 3 numbers in sequence starting from 19: 19 20
Congratulations! You reached 20 and won the game!
```

Example 2:

```
import random
# Function to check the winner
def check_winner(last_number, player_turn):
    if last_number >= 20:
        if player_turn:
            return "Player Wins!"
        else:
            return "Computer Wins!"
    return None
# Function for player's turn
def player_turn_auto(current_number):
    # Player chooses the next 1, 2, or 3 sequential numbers
    next_count = random.randint(1, 3)
    player_input = [current_number + i for i in range(1, next_count + 1)]
    return player_input
# Function for computer's turn
def computer_turn(current_number):
    next_count = random.randint(1, 3)
    computer_input = [current_number + i for i in range(1, next_count + 1)]
    return computer_input
# Main game function
def number_game():
    current_number = 0
    player_turn_flag = True  # True means it's player's turn, False means it's
computer's turn
    while True:
        if player_turn_flag:
            player_input = player_turn_auto(current_number)
            current_number = player_input[-1]
            print(f"Player played: {player_input}")
```

```
else:
        computer_input = computer_turn(current_number)
        current_number = computer_input[-1]
        print(f"Computer played: {computer_input}")

    # Check if the game is over
    result = check_winner(current_number, player_turn_flag)
    if result:
        print(result)
        break

    # Switch turn
    player_turn_flag = not player_turn_flag

# Start the game
number_game()
```

OUTPUT:



```
            print(result)
            break

        # Switch turn
        player_turn_flag = not player_turn_flag

# Start the game
number_game()
```

```
Player played: [1, 2]
Computer played: [3, 4, 5]
Player played: [6, 7]
Computer played: [8]
Player played: [9]
Computer played: [10, 11, 12]
Player played: [13]
Computer played: [14, 15, 16]
Player played: [17]
Computer played: [18, 19]
Player played: [20]
Player Wins!
```

Question 2:

Develop a function called ncr(n,r) which computes r-combinations of n-distinct object . use this function to print pascal triangle, where number of rows is the input

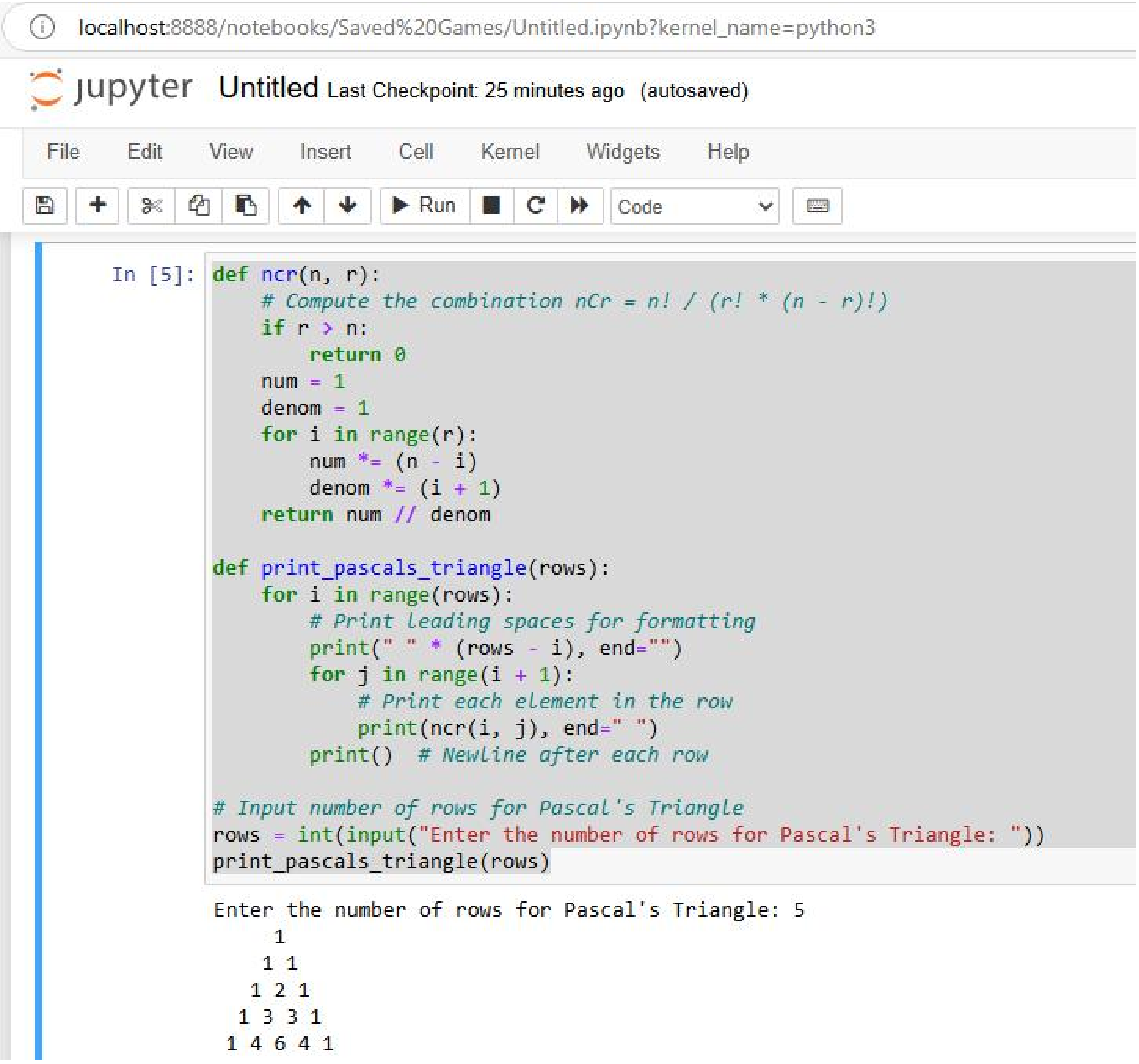
Answer:

```
def ncr(n, r):
    # Compute the combination nCr = n! / (r! * (n - r)!)
    if r > n:
        return 0
    num = 1
    denom = 1
    for i in range(r):
        num *= (n - i)
        denom *= (i + 1)
    return num // denom

def print_pascals_triangle(rows):
    for i in range(rows):
        # Print leading spaces for formatting
        print(" " * (rows - i), end="")
        for j in range(i + 1):
            # Print each element in the row
            print(ncr(i, j), end=" ")
        print()  # Newline after each row

# Input number of rows for Pascal's Triangle
rows = int(input("Enter the number of rows for Pascal's Triangle: "))
print_pascals_triangle(rows)
```

OUTPUT:

Enter the number of rows for Pascal's Triangle: 5

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```



localhost:8888/notebooks/Saved%20Games/Untitled.ipynb?kernel_name=python3
jupyter
Untitled Last Checkpoint: 25 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code
In [5]:

```
def ncr(n, r):
    # Compute the combination nCr = n! / (r! * (n - r)!)
    if r > n:
        return 0
    num = 1
    denom = 1
    for i in range(r):
        num *= (n - i)
        denom *= (i + 1)
    return num // denom

def print_pascals_triangle(rows):
    for i in range(rows):
        # Print leading spaces for formatting
        print(" " * (rows - i), end="")
        for j in range(i + 1):
            # Print each element in the row
            print(ncr(i, j), end=" ")
        print()  # Newline after each row

# Input number of rows for Pascal's Triangle
rows = int(input("Enter the number of rows for Pascal's Triangle: "))
print_pascals_triangle(rows)
```

```
Enter the number of rows for Pascal's Triangle: 5
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
```

Question 3:

Read a list of n numbers during runtime. Write a Python program to print the repeated elements with frequency count in a list.

Answer:

```
from collections import Counter
# Input number of elements in the list
n = int(input("Enter the number of elements in the list: "))
# Input the list elements
elements = []
for _ in range(n):
    num = int(input("Enter a number: "))
    elements.append(num)

# Count the frequency of each element
frequency = Counter(elements)

# Print repeated elements with their frequency
print("\nRepeated elements with frequency count:")
for element, count in frequency.items():
    if count > 1:
        print(f"{element}: {count}")
```

OUTPUT:

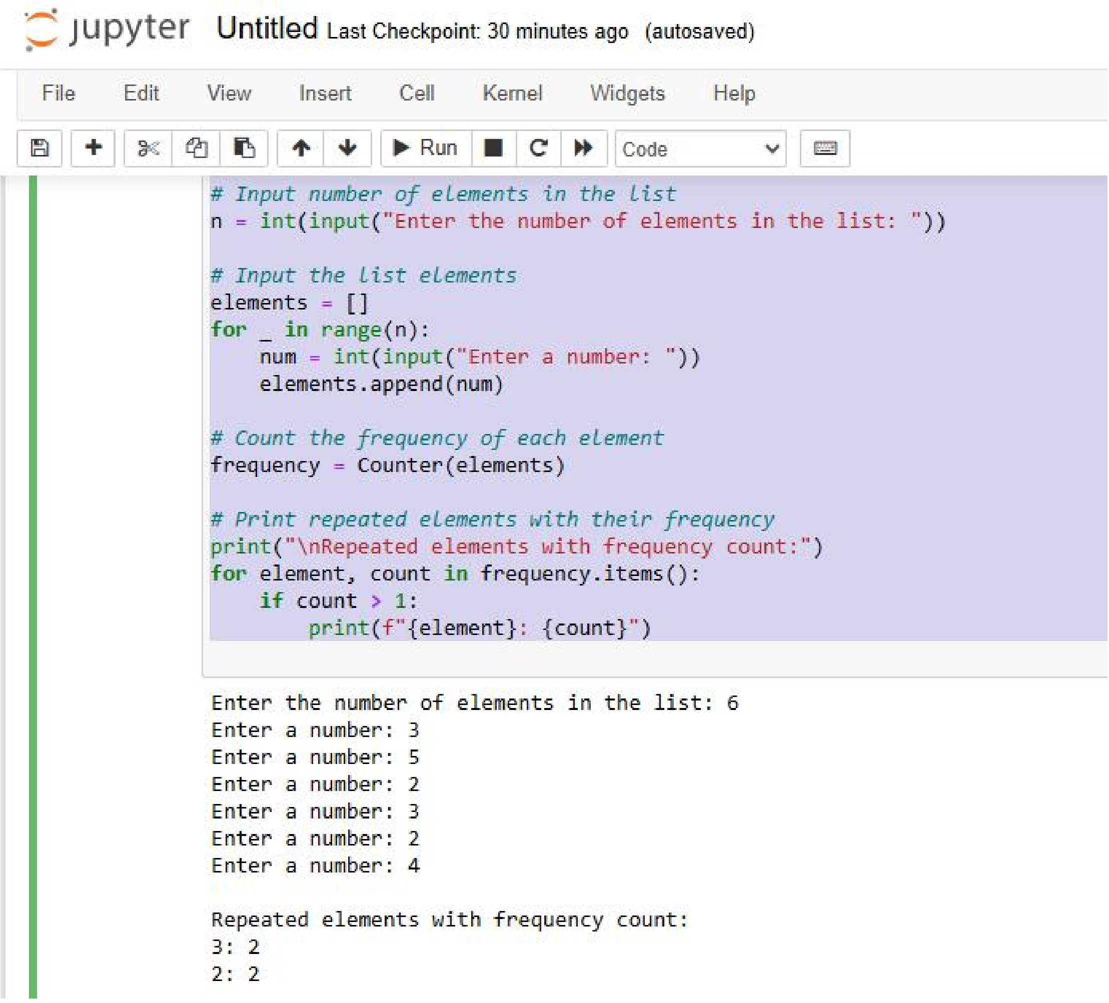
```
Enter the number of elements in the list: 6
Enter a number: 3
Enter a number: 5
Enter a number: 2
Enter a number: 3
Enter a number: 2
Enter a number: 4
```

Repeated elements with frequency count:

3: 2

2: 2

OUTPUT:



```python
# Input number of elements in the list
n = int(input("Enter the number of elements in the list: "))

# Input the list elements
elements = []
for _ in range(n):
    num = int(input("Enter a number: "))
    elements.append(num)

# Count the frequency of each element
frequency = Counter(elements)

# Print repeated elements with their frequency
print("\nRepeated elements with frequency count:")
for element, count in frequency.items():
    if count > 1:
        print(f"{element}: {count}")
```

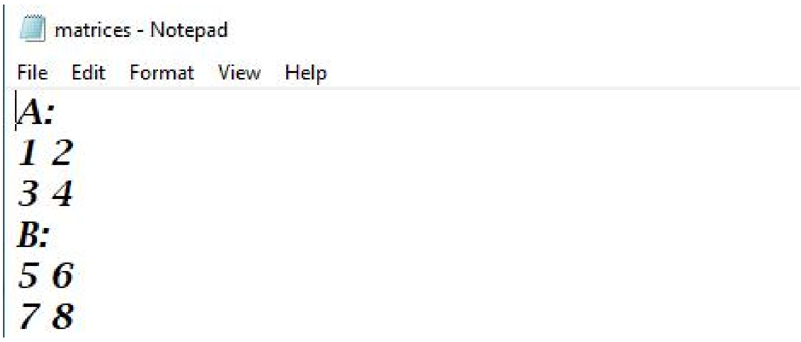
```
Enter the number of elements in the list: 6
Enter a number: 3
Enter a number: 5
Enter a number: 2
Enter a number: 3
Enter a number: 2
Enter a number: 4

Repeated elements with frequency count:
3: 2
2: 2
```

4. Develop a python code to read matric A of order 2X2 and Matrix B of order 2X2 from a file and perform the addition of Matrices A & B and Print the results.

Answer:

```
def read_matrices_from_file(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()

    matrix_a = []
    matrix_b = []
    current_matrix = None

    for line in lines:
        line = line.strip()
        if line == "A:":
            current_matrix = matrix_a
            continue
        elif line == "B:":
            current_matrix = matrix_b
            continue

        if current_matrix is not None:
            # Convert the line of numbers into a list of integers and add to the current matrix
            current_matrix.append(list(map(int, line.split())))
        return matrix_a, matrix_b

def add_matrices(matrix_a, matrix_b):
    # Element-wise addition of two 2x2 matrices
    return [
        [matrix_a[i][j] + matrix_b[i][j] for j in range(2)]
        for i in range(2)
    ]

def main():
    filename = "matrices.txt"
    matrix_a, matrix_b = read_matrices_from_file(filename)

    print("Matrix A:")
    for row in matrix_a:
        print(row)

    print("\nMatrix B:")
    for row in matrix_b:
        print(row)

    # Perform addition
    result = add_matrices(matrix_a, matrix_b)

    print("\nResult of A + B:")
    for row in result:
        print(row)

# Run the main function
if __name__ == "__main__":
    main()
```

INPUT: Matrices.txt



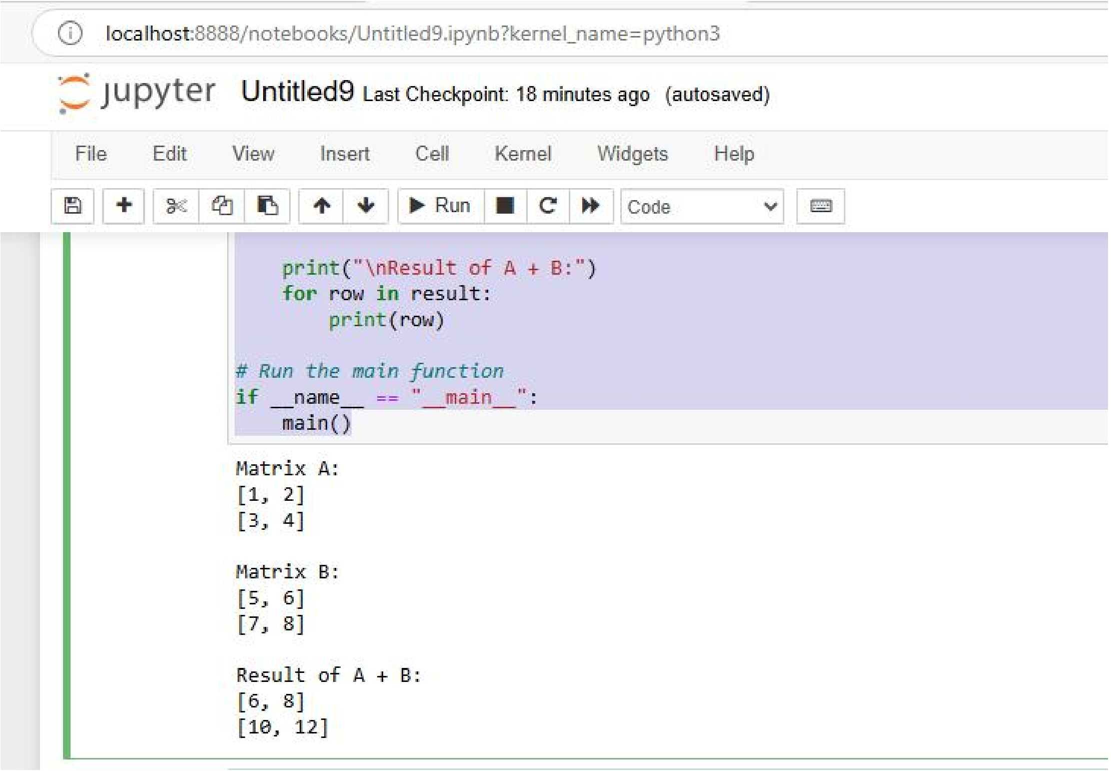
```
A:
1 2
3 4
B:
5 6
7 8
```

OUTPUT:

Matrix A:
[1, 2]
[3, 4]

Matrix B:
[5, 6]
[7, 8]

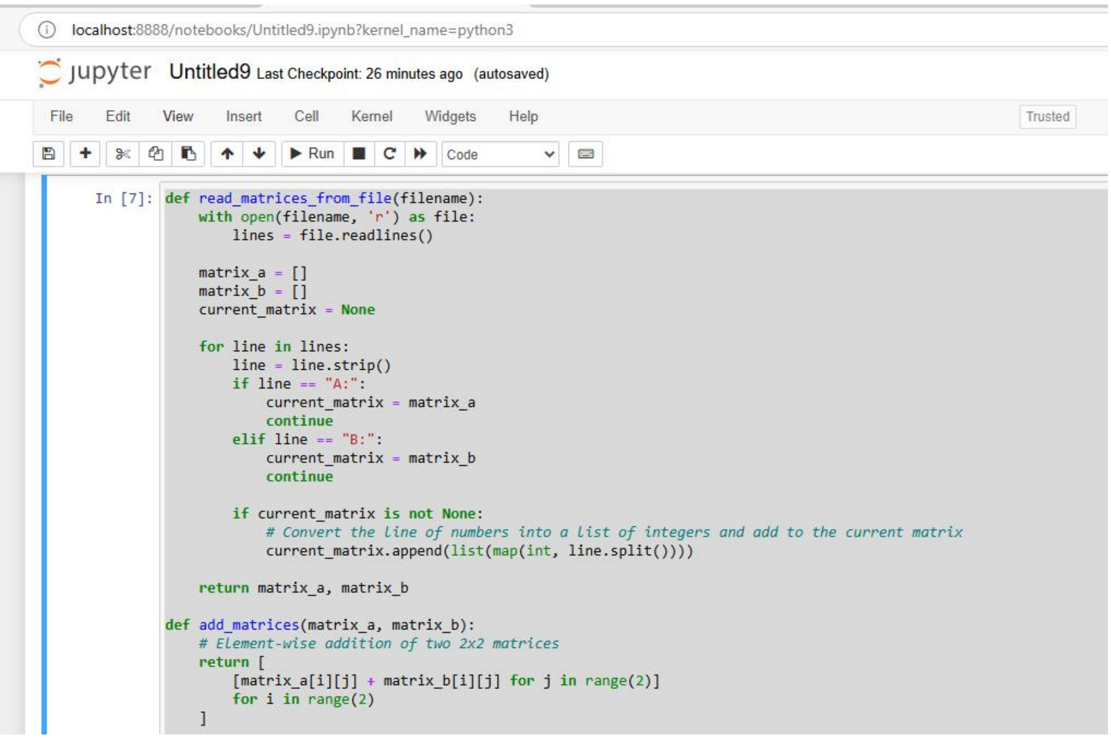
Result of A + B:
[6, 8]
[10, 12]


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter Untitled9 Last Checkpoint: 18 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

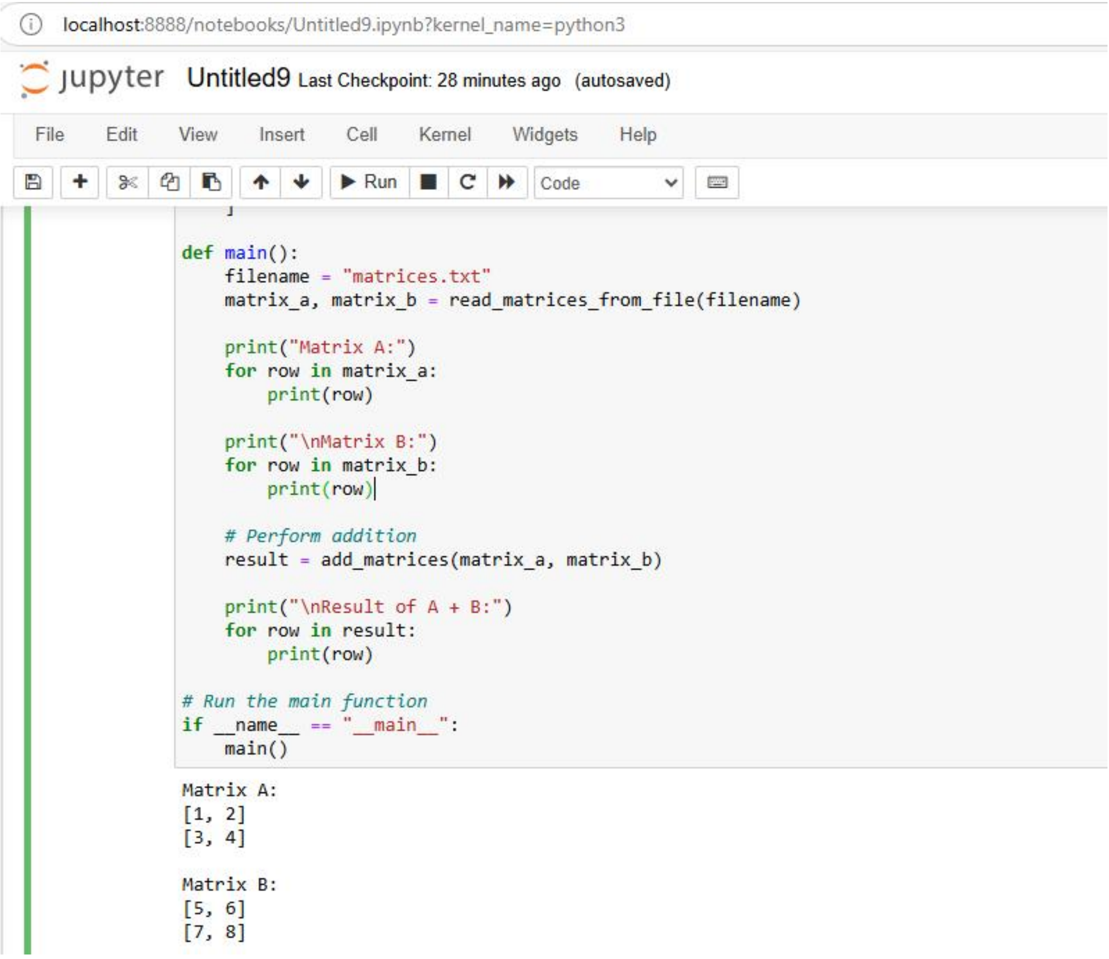
```
    print("\nResult of A + B:")
    for row in result:
        print(row)

# Run the main function
if __name__ == "__main__":
    main()
```

```
Matrix A:
[1, 2]
[3, 4]

Matrix B:
[5, 6]
[7, 8]

Result of A + B:
[6, 8]
[10, 12]
```

Program:

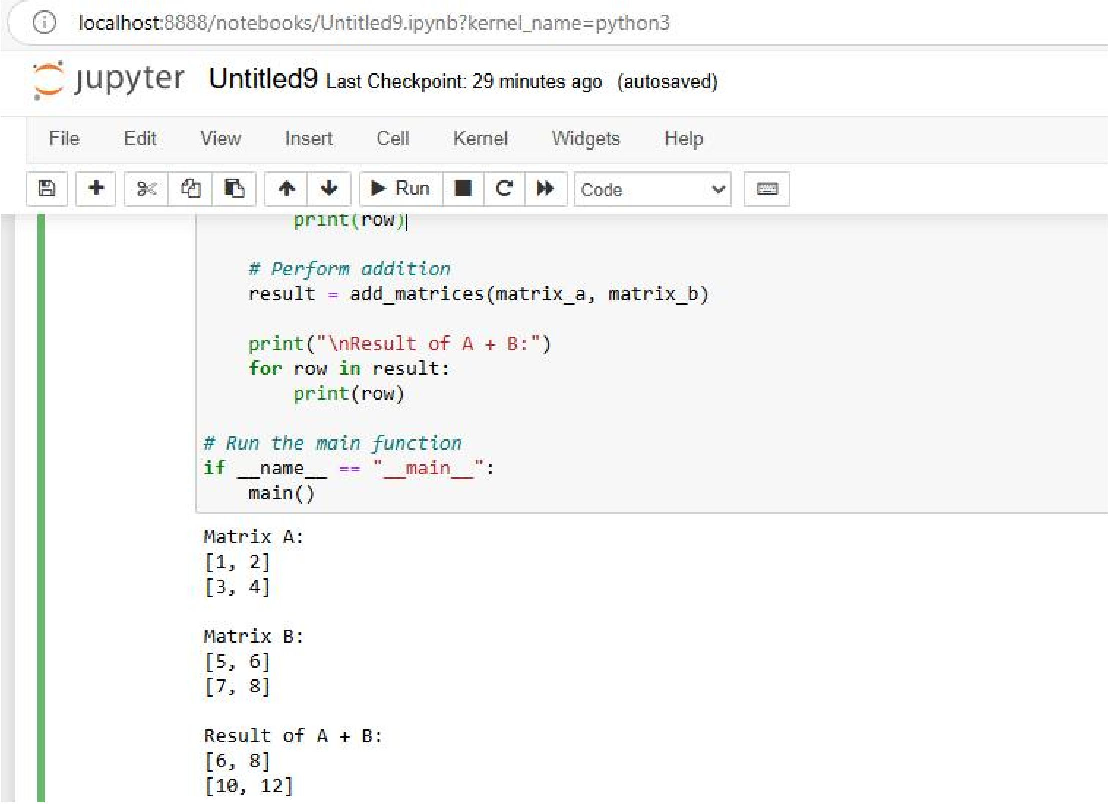

localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter
Untitled9 Last Checkpoint: 26 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Run
Code

```
In [7]: def read_matrices_from_file(filename):
            with open(filename, 'r') as file:
                lines = file.readlines()

            matrix_a = []
            matrix_b = []
            current_matrix = None

            for line in lines:
                line = line.strip()
                if line == "A:":
                    current_matrix = matrix_a
                    continue
                elif line == "B:":
                    current_matrix = matrix_b
                    continue

                if current_matrix is not None:
                    # Convert the line of numbers into a list of integers and add to the current matrix
                    current_matrix.append(list(map(int, line.split())))

            return matrix_a, matrix_b

        def add_matrices(matrix_a, matrix_b):
            # Element-wise addition of two 2x2 matrices
            return [
                [matrix_a[i][j] + matrix_b[i][j] for j in range(2)]
                for i in range(2)
            ]
```


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter
Untitled9 Last Checkpoint: 28 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
    ]

def main():
    filename = "matrices.txt"
    matrix_a, matrix_b = read_matrices_from_file(filename)

    print("Matrix A:")
    for row in matrix_a:
        print(row)

    print("\nMatrix B:")
    for row in matrix_b:
        print(row)

    # Perform addition
    result = add_matrices(matrix_a, matrix_b)

    print("\nResult of A + B:")
    for row in result:
        print(row)

# Run the main function
if __name__ == "__main__":
    main()
```

```
Matrix A:
[1, 2]
[3, 4]

Matrix B:
[5, 6]
[7, 8]
```


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter
Untitled9 Last Checkpoint: 29 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
        print(row)

    # Perform addition
    result = add_matrices(matrix_a, matrix_b)

    print("\nResult of A + B:")
    for row in result:
        print(row)

# Run the main function
if __name__ == "__main__":
    main()
```

```
Matrix A:
[1, 2]
[3, 4]

Matrix B:
[5, 6]
[7, 8]

Result of A + B:
[6, 8]
[10, 12]
```

Question 5:-

Write a program that overloads the + operator so that it can add two objects of the class Fraction.

Fraction can be considered of the for P/Q where P is the numerator and Q is the denominator

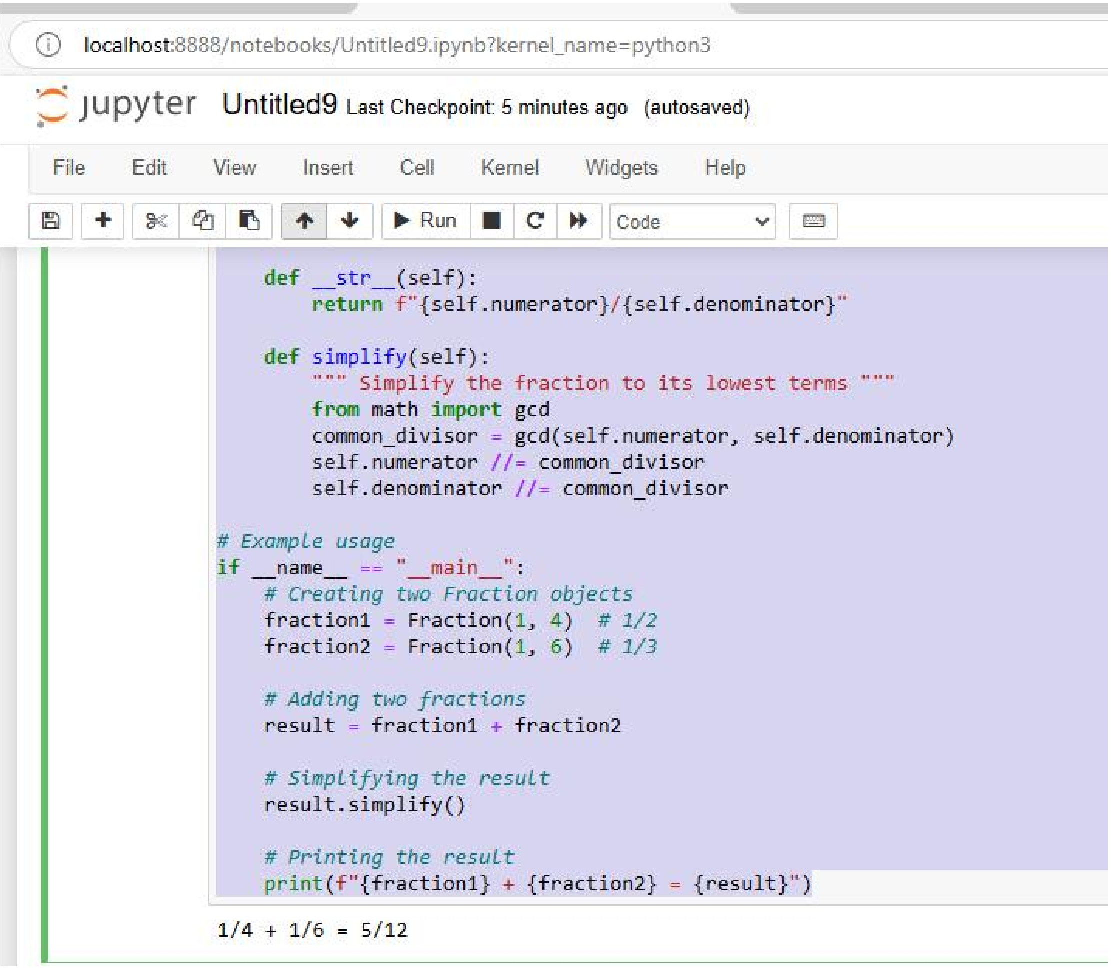
Answer:

```
class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero.")
        self.numerator = numerator
        self.denominator = denominator
    def __add__(self, other):
        if isinstance(other, Fraction):
            # Cross-multiply to find a common denominator
            new_numerator = (self.numerator * other.denominator) + (other.numerator * self.denominator)
            new_denominator = self.denominator * other.denominator
            return Fraction(new_numerator, new_denominator)
        return NotImplemented
    def __str__(self):
        return f"{self.numerator}/{self.denominator}"
    def simplify(self):
        """ Simplify the fraction to its lowest terms """
        from math import gcd
        common_divisor = gcd(self.numerator, self.denominator)
        self.numerator //= common_divisor
        self.denominator //= common_divisor
# Example usage
if __name__ == "__main__":
```

```
# Creating two Fraction objects
fraction1 = Fraction(1, 4)  # 1/2
fraction2 = Fraction(1, 6)  # 1/3

# Adding two fractions
result = fraction1 + fraction2

# Simplifying the result
result.simplify()

# Printing the result
print(f"{fraction1} + {fraction2} = {result}")
```

OUTPUT:

```
1/4 + 1/6 = 5/12
```


localhost:8888/notebooks/Untitled9.ipynb?kernel_name=python3
jupyter
Untitled9 Last Checkpoint: 5 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
    def __str__(self):
        return f"{self.numerator}/{self.denominator}"

    def simplify(self):
        """ Simplify the fraction to its lowest terms """
        from math import gcd
        common_divisor = gcd(self.numerator, self.denominator)
        self.numerator //= common_divisor
        self.denominator //= common_divisor

# Example usage
if __name__ == "__main__":
    # Creating two Fraction objects
    fraction1 = Fraction(1, 4)  # 1/2
    fraction2 = Fraction(1, 6)  # 1/3

    # Adding two fractions
    result = fraction1 + fraction2

    # Simplifying the result
    result.simplify()

    # Printing the result
    print(f"{fraction1} + {fraction2} = {result}")
```

```
1/4 + 1/6 = 5/12
```