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

Example 1:

Player: 1 2

Computer played: [3, 4]

Player: 5 6 7

Computer played: [8, 9]

Player: 10

Computer played: [11, 12, 13]

Player: 14 15

Computer played: [16, 17, 18]

Player: 19 20

Player Wins!!!

Example 2:

Player: 1

Computer played: [2, 3]

Player: 4 5

Computer played: [6, 7, 8]

Player: 9 10

Computer played: [11]

Player: 12

Computer played: [13]

Player: 14 15

Computer played: [16]

Player: 17 18

Computer played: [19, 20]

Computer Wins!!!

Answer :----------

```
import random

def computer_turn(current_number):

    pick = random.randint(1, 3)
    next_number = current_number + pick
    if next_number > 20:
        next_number = 20
    return list(range(current_number + 1, next_number + 1))

def user_turn(current_number):

    while True:
        try:
            user_input = input(f"Enter 1, 2, or 3 next digits after {current_number}: ")
            user_numbers = list(map(int, user_input.split()))



            if len(user_numbers) not in [1, 2, 3]:
                print("You must enter 1, 2, or 3 numbers in sequence.")
```

```
            continue

        if user_numbers[0] != current_number + 1 or user_numbers != list(range(current_number + 1,
current_number + 1 + len(user_numbers))):
            print("You must enter the next numbers in sequence.")
            continue

        if user_numbers[-1] > 20:
            print("You cannot enter numbers greater than 20.")
            continue

        return user_numbers

    except ValueError:
        print("Invalid input. Please enter numbers in sequence.")

def number_game():
    current_number = 0
    print("Game Start! Reach 20 first to win!")

    while current_number < 20:

        user_numbers = user_turn(current_number)
        current_number = user_numbers[-1]
        print(f"You picked: {user_numbers}")
        if current_number == 20:
            print("Congratulations! You reached 20 first and won the game!")
            break
```

```
    comp_numbers = computer_turn(current_number)
    current_number = comp_numbers[-1]
    print(f"Computer picked: {comp_numbers}")
    if current_number == 20:
        print("Computer reached 20 first! You lost the game.")
        break

number_game()
```

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 factorial(num):
    if num == 0 or num == 1:
        return 1
    else:
        result = 1
        for i in range(2, num + 1):
            result *= i
        return result

def ncr(n, r):
    return factorial(n) // (factorial(r) * factorial(n - r))

def print_pascals_triangle(rows):
    for n in range(rows):
        print(' ' * (rows - n), end='')
        for r in range(n + 1):
            print(ncr(n, r), end=' ')
```

```
    print()

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

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.

Example :

Input:- [2,1,2,3,4,5,1,3,6,2,3,4]

Output:-

Element 2 has come 3 times

Element 1 has come 2 times

Element 3 has come 2 times

Element 4 has come 2 times

Element 1 has come 1 times

Element 6 has come 1 times

Answer :

```
from collections import Counter

def count_repeated_elements(numbers):

    frequency = Counter(numbers)

    repeated_elements = {key: value for key, value in frequency.items() if value > 1}

    if repeated_elements:
```

```
        print("Repeated elements with their frequencies:")
        for element, count in repeated_elements.items():
            print(f"Element {element}: {count} times")
    else:
        print("No repeated elements found.")


numbers = list(map(int, input("Enter the numbers separated by spaces: ").split()))


count_repeated_elements(numbers)
```

Question 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_matrix_from_file(filename):
    """Reads two 2x2 matrices from the given file."""
    with open(filename, 'r') as file:
        matrix_a = []
        matrix_b = []


        for i in range(2):
            row = list(map(int, file.readline().split()))
            matrix_a.append(row)
```

```
        for i in range(2):
            row = list(map(int, file.readline().split()))
            matrix_b.append(row)

    return matrix_a, matrix_b

def add_matrices(matrix_a, matrix_b):
    """Performs addition of two 2x2 matrices."""
    result_matrix = []

    for i in range(2):
        row = []
        for j in range(2):
            row.append(matrix_a[i][j] + matrix_b[i][j])
        result_matrix.append(row)

    return result_matrix

def print_matrix(matrix):
    """Prints the matrix."""
    for row in matrix:
        print(" ".join(map(str, row)))


filename = 'matrices.txt'  # Make sure the file exists with correct format
matrix_a, matrix_b = read_matrix_from_file(filename)
```

```
result_matrix = add_matrices(matrix_a, matrix_b)

print("Matrix A + Matrix B is:")
print_matrix(result_matrix)
```

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 :

```
from math import gcd

class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero.")
        self.numerator = numerator
        self.denominator = denominator
        self.simplify()

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

    def __add__(self, other):
        if not isinstance(other, Fraction):
            return NotImplemented

        new_numerator = (self.numerator * other.denominator) + (other.numerator * self.denominator)
```

```
        new_denominator = self.denominator * other.denominator

        return Fraction(new_numerator, new_denominator)

    def simplify(self):
        """Simplify the fraction to its simplest form."""
        common_divisor = gcd(self.numerator, self.denominator)
        self.numerator //= common_divisor
        self.denominator //= common_divisor


if __name__ == "__main__":

    fraction1 = Fraction(1, 2)  # Represents 1/2
    fraction2 = Fraction(1, 3)  # Represents 1/3

    result = fraction1 + fraction2

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