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!!!

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

```
def ncr(n, r):
   Compute r-combinations of n-distinct objects.
   n (int): Total number of distinct objects.
   r (int): Number of objects to choose.

   Returns:
   int: Number of combinations.
   if r > n:
      return 0
def print_pascal_triangle(rows):
  rows (int): Number of rows in the triangle.
   for i in range(rows):
      for j in range(rows - i - 1):
         print(" ", end=" ")
      for k in range(i + 1):
```

```
        print(ncr(i, k), end=" ")
    print()
```

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

```
def print_repeated_elements(nums):
    nums (list): List of numbers.
  freq_dict = {}

  for num in nums:
    if num in freq_dict:
      freq_dict[num] += 1
    else:
      freq_dict[num] = 1
  for num, freq in freq_dict.items():
    if freq > 1:
      print(f"Element {num} has come {freq} times")
```

```
        else:
            print(f"Element {num} has come {freq} time")
n = int(input("Enter the number of elements: "))
nums = list(map(int, input("Enter the elements separated by space: ").split()))

print_repeated_elements(nums)
```

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.

```
def read_matrix_from_file(filename, matrix_name):

    filename (str): Name of the file.
    matrix_name (str): Name of the matrix.
    with open(filename, 'r') as file:
        lines = file.readlines()
        for line in lines:
            if line.startswith(matrix_name):
                matrix_rows = lines[lines.index(line)+1:lines.index(line)+3]
                matrix = [list(map(float, row.split())) for row in matrix_rows]
                return np.array(matrix)

def add_matrices(A, B):
    A (numpy.ndarray): Matrix A.
    B (numpy.ndarray): Matrix B.

    numpy.ndarray: Sum of matrices A and B.
```

```
    return A + B


def main():
    filename = input("Enter the filename: ")

    # Read matrices A and B from file
    A = read_matrix_from_file(filename, 'Matrix A:')
    B = read_matrix_from_file(filename, 'Matrix B:')

    # Add matrices A and B
    result = add_matrices(A, B)

    # Print matrices A, B, and the result
    print("Matrix A:")
    print(A)
    print("\nMatrix B:")
    print(B)
    print("\nMatrix A + Matrix B:")
    print(result)

if __name__ == "__main__":
    main()
```

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.

```
  def __init__(self, numerator, denominator):
      numerator (int): Numerator of the fraction.
      denominator (int): Denominator of the fraction.
      if denominator == 0:
      self.numerator = numerator
      self.denominator = denominator
      self.simplify()

  def simplify(self):
      Simplify the fraction by dividing both numerator and denominator by their GCD.
      gcd = self.gcd(self.numerator, self.denominator)
      self.numerator //= gcd
      self.denominator //= gcd



def gcd(a, b):
      a (int): First number.
      b (int): Second number.

      int: GCD of a and b.
      while b:
         a, b = b, a % b
      return a

  def __add__(self, other):
```

```
    other (Fraction): Fraction to add.


    numerator = self.numerator * other.denominator + other.numerator * self.denominator
    denominator = self.denominator * other.denominator
    return Fraction(numerator, denominator)


def __str__(self):
    str: Fraction as a string.
    return f"{self.numerator}/{self.denominator}"
```