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

```
#Ans
import random

def is_sequential(numbers):
    for i in range(len(numbers) - 1):
        if numbers[i] + 1 != numbers[i + 1]:
            return False
    return True

p=0
c=0
while True:
    if p>=20:
        print('Playher Wins!!!')
        break

    if c>=20:
        print('Computer Wins!!!')
        break

    u1 = input('Player: ')
    user_input = [int(x) for x in u1.split()]

    start = user_input[0]

    if (start == c+1) and len(user_input)<=3 and is_sequential(user_input)==True:

        #p = num_validation(user_input)
        p = user_input[-1]

        move_length = random.randint(1,3)
        computer_move = [p + i + 1 for i in range(move_length)]

        print('computer:', computer_move)

        c = computer_move[-1]

    else:
        print('Invalid')
```

```
Start coding or generate with AI.
```

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

```
#Ans:

def pascal_triangle(n):
    triangle = []
    for i in range(n):
        row = [1] * (i + 1)
        for j in range(1, i):
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j]
        triangle.append(row)
    return triangle



num= int(input('Enter no. of rows to compute Pascal triangle: '))

#num = 8
```

```
print('no. of rows: ', num)
rows = pascal_triangle(num)
for r in rows:
    print(r)
```

```
no. of rows:  8
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
```

Start coding or generate with AI.

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.

```
import ast

input_list = input('Enter list of numbers:')    # [1,1,2,2,3,3,5,5,5,7]

x = ast.literal_eval(input_list)

#x = [1,1,2,2,3,3,5,5,5,7]

print('input list: ', x)

frequency = {}
for n in x:
    frequency[n] = frequency.get(n, 0)+1

for k, v in frequency.items():
    print('Element', k, 'has come', v, 'times')
```

```
input list:  [1, 1, 2, 2, 3, 3, 5, 5, 5, 7]
Element 1 has come 2 times
Element 2 has come 2 times
Element 3 has come 2 times
Element 5 has come 3 times
Element 7 has come 1 times
```

Start coding or generate with AI.

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.

```
#Ans:

# 'file.txt'

# 1 3
# 6 8
# 7 2
# 9 12


import numpy as np

filename = 'file.txt'

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

matrix_A = [list(map(int, line.split())) for line in lines[:2]]

matrix_B = [list(map(int, line.split())) for line in lines[2:4]]

print('Matrix A:')
print(np.array(matrix_A), '\n')

print('Matrix B:')
print(np.array(matrix_B), '\n')
```

```
print('Addition of Matrix A and Matrix B:')
print(np.add(matrix_A, matrix_B))
```

```
Matrix A:
[[1 3]
 [6 8]]

Matrix B:
[[ 7  2]
 [ 9 12]]

Addition of Matrix A and Matrix B:
[[ 8  5]
 [15 20]]
```

Start coding or generate with AI.

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

```
class Fraction:

    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def __add__(self, other):
        # Calculate the numerator and denominator of the result
        numerator = (self.numerator * other.denominator) + (other.numerator * self.denominator)
        denominator = self.denominator * other.denominator
        return float(numerator / denominator)

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


frac1 = Fraction(2,3)
frac2 = Fraction(4,8)

print('Fraction 1: ', frac1)
print('Fraction 2: ', frac2)
print('Addition of 2 fractions: ', frac1+frac2)
```

```
Fraction 1:  2/3
Fraction 2:  4/8
Addition of 2 fractions:  1.1666666666666667
```