1) Number Game between User and Computer.

Source code :-

```
def number_game():
    max_num = 20
    current = 0

    while current < max_num:
        try:
            user_input = int(input("Enter 1, 2, or 3
                                to add to the Sequence :"))
            if user_input not in [1,2,3]:
                print("Invalid input. Please enter
                                1, 2, or 3.")
                continue
            current += user_input
            print(f"player: {current}")

            if current >= max_num:
                print("player wins!!!")
                return
```

```
computer_play = random.randint(1, min(3, max_num - current))
current += computer_play
print(f"Computer played: {list(range(current - computer_play + 1, current + 1))}")
if current >= max_num:
    print("Computer wins!!!")
    return
except ValueError:
    print("please enter a valid integer")
number_game()
```

Output:

Enter 1, 2, or 3 to add to the player's Sequence: 3

Computer played : [4, 5]

Sequence : 3

Player : 8

computer played : [9, 10, 11]

Sequence : 3

player : 14

computer played : [15, 16, 17]

Sequence : 3

player : 20

player Wins!!!

2) nCr Function and Pascal's Traingle.

```
def ncr (n, r):
    from math import factorial.
    return factorial (n) // (factorial (r) *
                             factorial (n-r)).
def print_pascal_triangle (rows):
    for i in rangle (rows):
```

```
        for j in range (i+1):
            print (ncr(i,j), end = " ")
        print()

rows = int(input("Enter the number of rows
                  for Pascal's Traingle :"))
print_pascal_traingle (rows).
```

Output :-

Enter the number of rows For
Pascal's Traingle : 4 .

```
1
1 1
1 2 1
1 3 3 1
```

3) Frequency Count of list elements.

```
freq_dict = {}
for item in lst:
    freq_dict[item] = freq_dict.get(item, 0)+1
for item, count in freq_dict.items():
    print(f"Element {item} has come {count}
                                times")
lst = [2, 1, 2, 3, 4, 5, 1, 3, 6, 2, 3, 4]
frequency_count(lst)
```

Output:-

Element 2 has come 3 times.
Element 1 has come 2 times
Element 3 has come 3 times.
Element 4 has come 2 times.
Element 5 has come 1 time.
Element 6 has come 1 time.

4) Matrix Addition from file.

```
def read_matricer_and_add(file_path):
    with open(file_path, 'r') as file:
        rows, cols = map(int, file.readline().split())
        matrix_a = [[int(num) for num in file.readline().
                     split()] for _ in range(rows)]
        matrix_b = [[int(num) for num in file.readline().
                     split()] for _ in range
                     (rows)]
        result = [[matrix_a[i][j] + matrix_b[i][j] for j in range
                   (cols)] for i in range(rows)]
        print("Result of Matrix Addition:")
        for row in result:
            print(row)

file_path = "matricer.txt".
read_matricer_and_add(file_path)
```

Output :-

Create the matrices.text file :

for ex :- 2 3

1 2 3

4 5 6

7 8 9

10 11 12

output of Matrix Addition :

[8, 10, 12]

[14, 16, 18]

5) Operator Overloading for Fraction Addition.

```
Class Fraction:
    def __intit__(Self, numerator, denominator):
        Sdf.numerator = numerator
        self.denominator = denominator

    def __add__(self, other):
        new_numerator = (self.numerator * other.
                         denominator)
        new_denominator = self.denominator * other.
                          denominator
        return Fraction(new_numerator, new_
                        denominator)

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

frac1 = Fraction(1, 6)
frac2 = Fraction(1, 4)
result = frac1 + frac2
print("Result of Fraction Addition:", result)
```

output:- Result of Fraction Addition: 5/24