Assignment : 1

Name : A Gopi

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

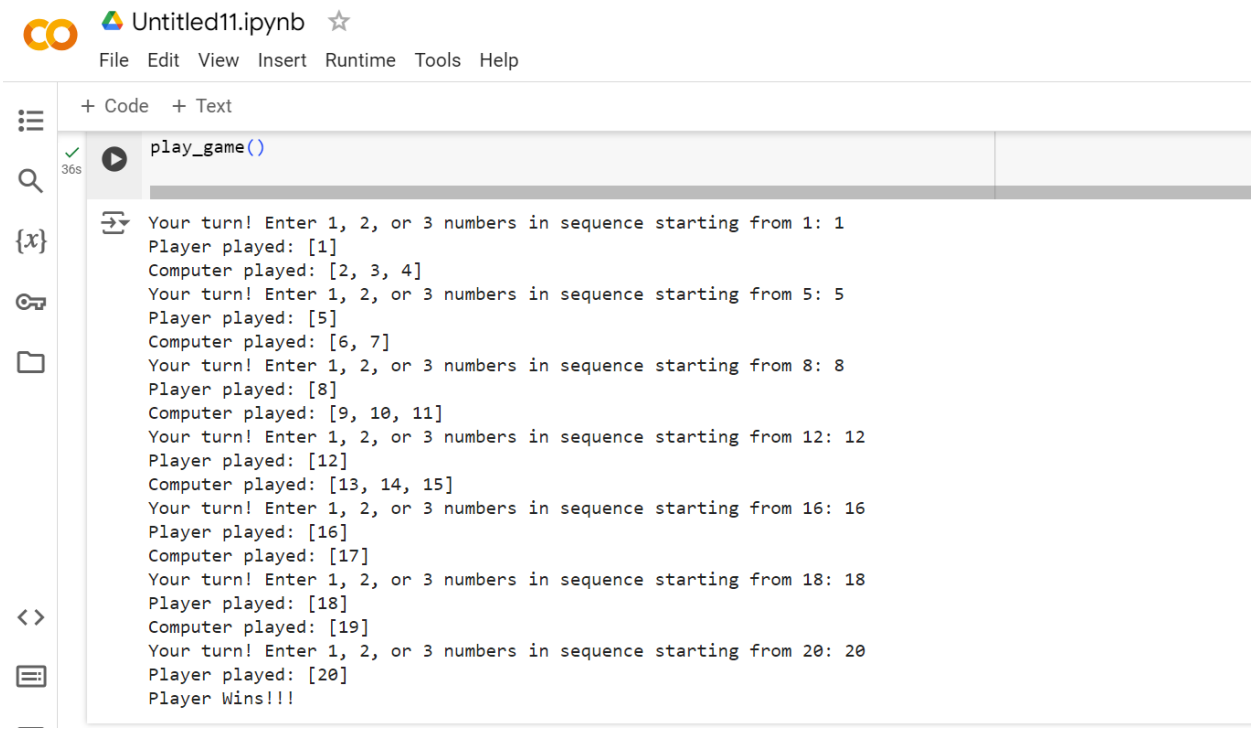
Player Wins!!!

Program:

```
import random

def computer_play(max_played):
    # Computer randomly picks between 1 and 3 digits in sequence
    pick = random.randint(1, 3)
    return list(range(max_played + 1, max_played + pick + 1))

def player_play(max_played):
    while True:
        try:
            # Player enters 1, 2, or 3 numbers in sequence starting from the max_played + 1
            player_input = input(f"Your turn! Enter 1, 2, or 3 numbers in sequence starting from {max_played + 1}: ")
            player_numbers = list(map(int, player_input.split()))
            if len(player_numbers) in [1, 2, 3] and player_numbers[0] == max_played + 1 and all(player_numbers[i] == player_numbers[i-1] + 1 for i in range(1, len(player_numbers))):
                return player_numbers
            else:
                print("Invalid input. Please enter numbers in sequence starting from the correct number.")
        except ValueError:
            print("Invalid input. Please enter numbers separated by spaces.")

def play_game():
    max_played = 0
    winning_number = 20
    while max_played < winning_number:
        # Player's turn
```

```
        player_numbers = player_play(max_played)
        max_played = player_numbers[-1]
        print(f"Player played: {player_numbers}")
        if max_played >= winning_number:
            print("Player Wins!!!")
            break
        # Computer's turn
        computer_numbers = computer_play(max_played)
        max_played = computer_numbers[-1]
        print(f"Computer played: {computer_numbers}")
        if max_played >= winning_number:
            print("Computer Wins!!!")
            break
# Start the game
play_game()
```

Output:



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]

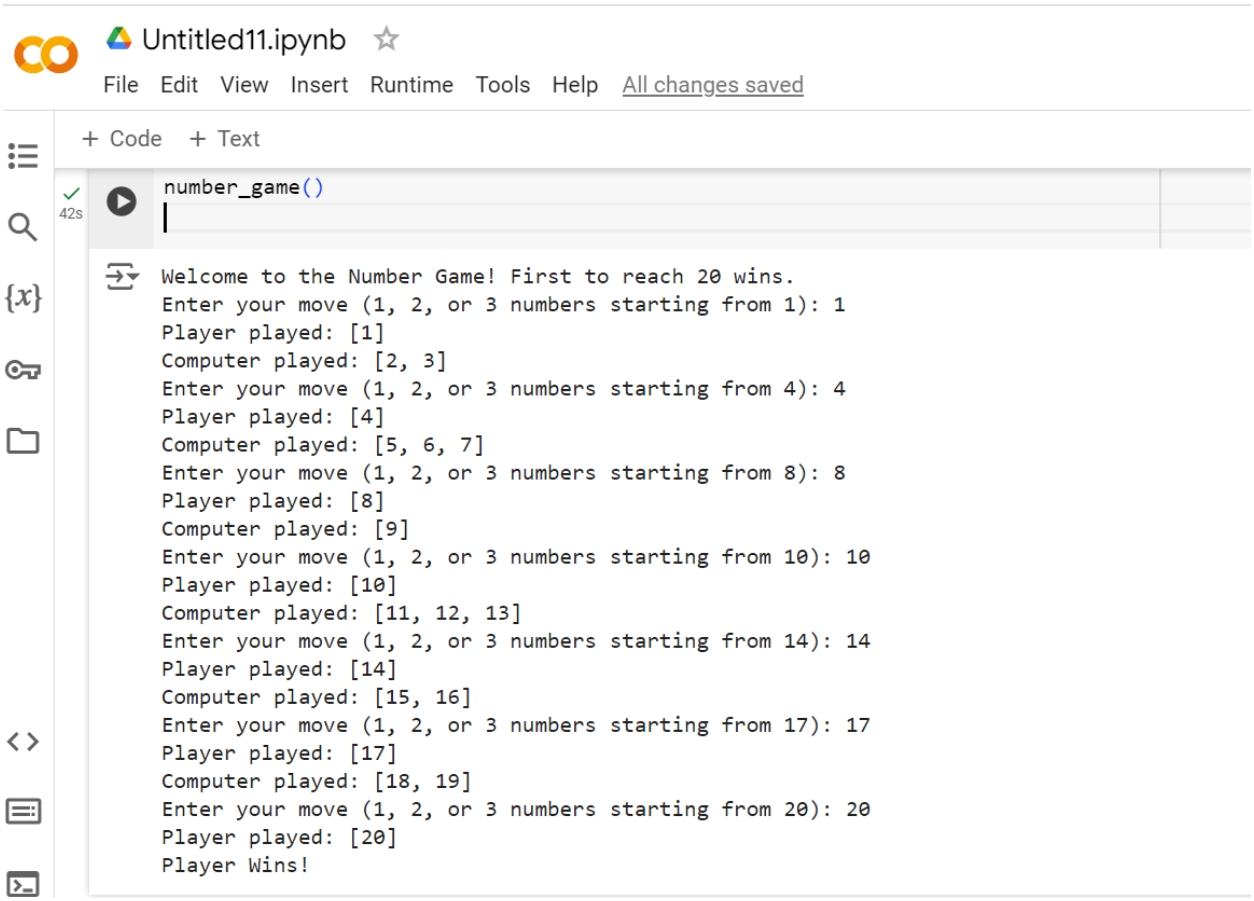
Computer Wins!!!

Program :

```
import random
def computer_move(current_number):
    # Computer can pick between 1 and 3 numbers in sequence
    move_count = random.randint(1, 3)
    move = list(range(current_number + 1, current_number + 1 + move_count))
    return move
def user_move(current_number):
    while True:
        user_input = input(f"Enter your move (1, 2, or 3 numbers starting from {current_number + 1}): ").strip()
        try:
            # Split the input into individual numbers and convert to integers
            user_numbers = list(map(int, user_input.split()))
                    # Check if numbers are sequential and start correctly from current_number + 1
            if len(user_numbers) in [1, 2, 3] and all(user_numbers[i] == current_number + i + 1 for i in range(len(user_numbers))):
                return user_numbers
            else:
                print(f"Invalid move. Please enter a sequence starting from {current_number + 1}.")
        except ValueError:
            print("Please enter valid numbers.")

def number_game():
    print("Welcome to the Number Game! First to reach 20 wins.")
    current_number = 0
    while current_number < 20:
        # User's turn
        user_numbers = user_move(current_number)
```

```
        current_number = user_numbers[-1]
        print(f"Player played: {user_numbers}")
        if current_number >= 20:
            print("Player Wins!")
            break
        # Computer's turn
        computer_numbers = computer_move(current_number)
        current_number = computer_numbers[-1]
        print(f"Computer played: {computer_numbers}")
        if current_number >= 20:
            print("Computer Wins!")
            break
# Start the game
number_game()
```

Output:



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

Program:

```
# Function to calculate nCr (binomial coefficient)
def ncr(n, r):
    if r > n:
        return 0
    # Use the factorial formula for combinations nCr = n! / (r! * (n-r)!)
    num = 1
    den = 1
    for i in range(r):
        num *= (n - i)
        den *= (i + 1)
    return num // den
```

```
# Function to print Pascal's Triangle
def print_pascals_triangle(rows):
    for n in range(rows):
        # Print spaces for formatting
        print(" " * (rows - n), end="")

        # Print the values in each row
        for r in range(n + 1):
            print(ncr(n, r), end=" ")
        print()  # Newline after each row

# Input: Number of rows in 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: 4
    1
   1 1
  1 2 1
 1 3 3 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.

Program :

```
# Function to count the frequency of elements in the list
def print_frequency_count(lst):
    # Create an empty dictionary to store frequency of elements
    frequency = {}

    # Iterate through the list and count frequencies
    for item in lst:
        if item in frequency:
```

```
            frequency[item] += 5
        else:
            frequency[item] = 1

    # Print each element with its frequency count
    for key, value in frequency.items():
        print(f"Element {key} has come {value} times")

# Input: Reading list of numbers at runtime
input_list = list(map(int, input("Enter the list of numbers separated by space: ").split()))

# Call the function to print frequency count
print_frequency_count(input_list)
```

Example :/ Output:

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

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

Program :

```
def read_matrix(file):
    with open(file, 'r') as f:
        matrix = [list(map(int, line.split())) for line in f.readlines()]
    return matrix
def add_matrices(A, B):
    return [[A[i][j] + B[i][j] for j in range(2)] for i in range(2)]
def print_matrix(matrix):
    for row in matrix:
        print(" ".join(map(str, row)))
if __name__ == "__main__":
    A = read_matrix('matrices.txt')[:2]  # Read first 2 lines for Matrix A
    B = read_matrix('matrices.txt')[2:]  # Read last 2 lines for Matrix B
    result = add_matrices(A, B)
    print("Result of A + B:")
    print_matrix(result)
```

Input:

Matrix – A

1 2

3 4

Matrix – B

5 6

7 8

Output

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

Program:

```
import math

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

    def __add__(self, other):
        """Overloads the + operator to add two fractions."""
        if isinstance(other, Fraction):
            # Calculate the new numerator and denominator
            new_numerator = self.numerator * other.denominator +
other.numerator * self.denominator
            new_denominator = self.denominator * other.denominator
            return Fraction(new_numerator, new_denominator)
        else:
            raise TypeError("Can only add Fraction objects.")

    def __str__(self):
        """String representation of the fraction."""
        return f"{self.numerator}/{self.denominator}"

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

# Adding the fractions
result = fraction1 + fraction2
print(f"The result of {fraction1} + {fraction2} is {result}")
```

Output :

The result of 1/2 + 1/3 is 5/6