**NAME: UDUTHA RAJENDER**

# ASSIGNMENT - 1

## 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 get_user_input(max_number):
  while True:
    user_input = input(f"Enter 1, 2, or 3 sequential numbers starting from {max_number + 1}: ")
    numbers = user_input.split()
    # Validate input
    if len(numbers) < 1 or len(numbers) > 3:
      print("You must enter 1 to 3 numbers.")
      continue
    try:
      numbers = [int(num) for num in numbers]
    except ValueError:
      print("Invalid input! Please enter valid numbers.")
      continue

    # Check if the numbers are in sequence and within the allowed range
    if numbers[0] != max_number + 1 or any(numbers[i] != numbers[i - 1] + 1 for i in range(1,
len(numbers))):
      print("Numbers must be in sequence starting from", max_number + 1)
      continue
```

```
        if len(numbers) > 3 or len(numbers) < 1:
            print("You can only enter 1 to 3 numbers.")
            continue
        return numbers
def computer_turn(max_number):
    count = random.randint(1, 3)  # Computer picks 1 to 3 numbers
    return list(range(max_number + 1, max_number + 1 + count))
def main():
    max_number = 0
    while max_number < 20:
        # User's turn
        user_numbers = get_user_input(max_number)
        max_number = user_numbers[-1]  # Update max number
        print(f"You picked: {user_numbers}. Current max number: {max_number}")
        if max_number >= 20:
            print("Congratulations! You reached 20 and won!")
            break
        # Computer's turn
        computer_numbers = computer_turn(max_number)
        max_number = computer_numbers[-1]  # Update max number
        print(f"Computer picked: {computer_numbers}. Current max number: {max_number}")
        if max_number >= 20:
            print("Computer reached 20. You lose!")
if __name__ == '__main__':
    main()
```

**RESULT:**

```
Enter 1, 2, or 3 sequential numbers starting from 1: 1 2
You picked: [1, 2]. Current max number: 2
```

```
Computer picked: [3, 4, 5]. Current max number: 5

Enter 1, 2, or 3 sequential numbers starting from 6: 6 7

You picked: [6, 7]. Current max number: 7

Computer picked: [8, 9]. Current max number: 9

Enter 1, 2, or 3 sequential numbers starting from 10: 10 11 12

You picked: [10, 11, 12]. Current max number: 12

Computer picked: [13]. Current max number: 13

Enter 1, 2, or 3 sequential numbers starting from 14: 14 15 16

You picked: [14, 15, 16]. Current max number: 16

Computer picked: [17, 18, 19]. Current max number: 19

Enter 1, 2, or 3 sequential numbers starting from 20: 20

You picked: [20]. Current max number: 20

Congratulations! You reached 20
```

**<u>Question 2:</u>**

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

**<u>ANSWER:</u>**

```
import math

def ncr(n, r):

    """Calculate the number of combinations of n items taken r at a time."""

    if r < 0 or r > n:

        return 0

    return math.factorial(n) // (math.factorial(r) * math.factorial(n - r))

def print_pascals_triangle(rows):

    """Print Pascal's Triangle with the specified number of rows."""

    for i in range(rows):

        # Print leading spaces for formatting

        print(' ' * (rows - i), end='')
        for j in range(i + 1):
            print(ncr(i, j), end=' ')
        print()  # Move to the next line after each row
def main():
    num_rows = int(input("Enter the number of rows for Pascal's Triangle: "))
    print_pascals_triangle(num_rows)
if __name__ == '__main__':
    main()
```

**RESULT:**

```
Enter the number of rows for Pascal's Triangle: 5
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 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.

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:**

```
def count_elements(nums):
    frequency = {}
    # Count the frequency of each element
    for num in nums:
        if num in frequency:
            frequency[num] += 1
        else:
            frequency[num] = 1
    # Print elements with their frequency
    for num, count in frequency.items():
        print(f"Element {num} has come {count} times")
def main():
    # Read input from the user
    input_string = input("Enter a list of numbers separated by commas: ")
    # Convert the input string to a list of integers
    nums = list(map(int, input_string.split(',')))
    count_elements(nums)
if __name__ == '__main__':
    main()
```

**RESULT**

**Input:**

```
Enter a list of numbers separated by commas: 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 3 times

Element 4 has come 2 times

Element 5 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.

## Create a File with Matrices

First, create a file named matrices.txt with the following content (each row of the matrix on a new line):

```
3 4

2 1

8 5

7 6
```

This represents:

Matrix A:

```
3 4

2  1
```

Matrix B:

```
8  5

7  6
```

**ANSWER:**

```
def read_matrices_from_file(filename):

    """Read two 2x2 matrices from a file."""

    with open(filename, 'r') as file:
        lines = file.readlines()
      # Parse Matrix A
        A = []
        for i in range(2):
            A.append(list(map(int, lines[i].strip().split())))
        # Parse Matrix B
        B = []
        for i in range(2, 4):
            B.append(list(map(int, lines[i].strip().split())))
    return A, B
def add_matrices(A, B):
    """Add two 2x2 matrices."""
    result = [[0, 0], [0, 0]]
    for i in range(2):
        for j in range(2):
            result[i][j] = A[i][j] + B[i][j]
    return result
def print_matrix(matrix):
    """Print a matrix."""
    for row in matrix:
        print(' '.join(map(str, row)))
def main():
    # Read matrices from the file
    A, B = read_matrices_from_file('matrices.txt')
    # Perform addition
    result = add_matrices(A, B)

    # Print the matrices and the result
    print("Matrix A:")
    print_matrix(A)
    print("\nMatrix B:")
    print_matrix(B)
    print("\nResultant Matrix (A + B):")
    print_matrix(result)
if __name__ == '__main__':
    main()
```

**RESULT:**

```

Matrix A:
3 4
2 1

Matrix B:
8 5
7 6

Data read from the file

Resultant Matrix (A + B):
11 9
9  7
```

**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 simplify(self):
        """Simplify the fraction to its lowest terms."""
        common_divisor = gcd(self.numerator, self.denominator)
        self.numerator //= common_divisor
        self.denominator //= common_divisor
        # Handle negative denominator
        if self.denominator < 0:
            self.numerator = -self.numerator
            self.denominator = -self.denominator
    def __add__(self, other):
        """Overload the + operator to add two Fraction objects."""
        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 __str__(self):
        """Return the string representation of the fraction."""
        return f"{self.numerator}/{self.denominator}"


    def __repr__(self):
        """Return a string that can be used to recreate the fraction."""
        return f"Fraction({self.numerator}, {self.denominator})"
# Example usage
if __name__ == '__main__':
    f1 = Fraction(5, 6)  # Represents  5/6
    f2 = Fraction(2, 7)  # Represents  2/7
    result = f1 + f2  # Should add the fractions
    print(f"{f1} + {f2} = {result}")  #
```

**RESULT:**

```
5/6 + 2/7 = 47/42
```