Assignment

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

Program

import random

def user\_turn(last\_num):

 while True:

 try:

 user\_input = input(f"Your turn! Enter 1, 2, or 3 consecutive numbers starting from {last\_num + 1}: ").split()

 user\_numbers = [int(num) for num in user\_input]

 if len(user\_numbers) > 3 or len(user\_numbers) < 1:

 print("You must enter 1, 2, or 3 numbers.")

 continue

 if user\_numbers[0] != last\_num + 1 or not all(user\_numbers[i] == user\_numbers[i - 1] + 1 for i in range(1, len(user\_numbers))):

 print(f"The numbers must start from {last\_num + 1} and be consecutive.")

 continue

 return user\_numbers[-1]

 except ValueError:

 print("Invalid input, please enter numbers only.")

def computer\_turn(last\_num):

 next\_count = random.randint(1, 3) # Computer picks between 1 and 3 numbers

 computer\_numbers = list(range(last\_num + 1, last\_num + 1 + next\_count))

 print(f"Computer's turn! Computer picks: {computer\_numbers}")

 return computer\_numbers[-1]

def play\_game():

 last\_num = 0

 while last\_num < 20:

 # User's turn

 last\_num = user\_turn(last\_num)

 if last\_num >= 20:

 print("Congratulations! You reached 20 and won the game!")

 break

 # Computer's turn

 last\_num = computer\_turn(last\_num)

 if last\_num >= 20:

 print("Computer reached 20. Computer wins!")

 break

# Uncomment below to play the game interactively

play\_game()

Output

**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!!!

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

def factorial(n):

 if n == 0 or n == 1:

 return 1

 result = 1

 for i in range(2, n + 1):

 result \*= i

 return result

def ncr(n, r):

 if r > n or r < 0:

 return 0

 return factorial(n) // (factorial(r) \* factorial(n - r))

def print\_pascal\_triangle(rows):

 for i in range(rows):

 print(" " \* (rows - i), end="")

 for j in range(i + 1):

 print(ncr(i, j), end=" ")

 print()

num\_rows = int(input("Enter the number of rows for Pascal's triangle: "))

if num\_rows >= 0:

 print\_pascal\_triangle(num\_rows)

else:

 print("Number of rows cannot be negative.")

Output

Enter the number of rows for Pascal's triangle: 7

 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

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

from collections import Counter

# Function to get input list from the user

def get\_input\_list():

 n = int(input("Enter the number of elements in the list: "))

 print("Enter the numbers:")

 input\_list = []

 for \_ in range(n):

 number = int(input())

 input\_list.append(number)

 return input\_list

# Function to print repeated elements with their frequency

def print\_repeated\_elements\_with\_frequency(input\_list):

 frequency = Counter(input\_list)

 print("\nRepeated elements with frequency count:")

 for num, count in frequency.items():

 if count > 1:

 print(f"{num} appears {count} times")

# Main code

if \_\_name\_\_ == "\_\_main\_\_":

 input\_list = get\_input\_list()

 print\_repeated\_elements\_with\_frequency(input\_list)

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

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):

 self.numerator = numerator

 self.denominator = denominator

 self.simplify()

 # Function to simplify the fraction

 def simplify(self):

 gcd = math.gcd(self.numerator, self.denominator)

 self.numerator //= gcd

 self.denominator //= gcd

 # Overloading the '+' operator

 def \_\_add\_\_(self, other):

 if isinstance(other, Fraction):

 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 two Fraction objects")

 # Function to represent the fraction as a string

 def \_\_str\_\_(self):

 return f"{self.numerator}/{self.denominator}"

# Main code

if \_\_name\_\_ == "\_\_main\_\_":

 # Creating two fractions

 frac1 = Fraction(1, 2)

 frac2 = Fraction(3, 4)

 # Adding the two fractions

 result = frac1 + frac2

 # Printing the result

 print(f"The result of adding {frac1} and {frac2} is {result}")

Output

The result of adding 1/2 and 3/4 is 5/4