assignment-1-1

November 3, 2024

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

[1]:
```
#Number game between user and computer
def computer_move(last_num):
    target = 20
    move_count = (target - last_num - 1) % 4
    if move_count == 0:
        move_count = 1
    return list(range(last_num + 1, last_num + move_count + 1))

def number_game():
    print("Number Game: Reach 20 to Win!")
    last_num = 0

    while last_num < 20:
        player_input = input("enter the next 1, 2, or 3 numbers in sequence: ")
        player_numbers = list(map(int, player_input.split()))

        # Check player's numbers are valid
        if any(num <= last_num or num > last_num + 3 for num in player_numbers)␣
 ↪or len(player_numbers) > 3:
            print("Invalid move! Please enter 1 to 3 numbers in sequence␣
 ↪starting from the last number.")
            continue
        last_num = player_numbers[-1]

        # Check if the player wins
        if last_num >= 20:
            print("Player Wins!!!")
            break

        # Computer's move
```

```
        computer_numbers = computer_move(last_num)
        print(f"Computer played: {computer_numbers}")
        last_num = computer_numbers[-1]

        # Check if the computer wins
        if last_num >= 20:
            print("Computer Wins!!!")
            break
number_game()
```

```
Number Game: Reach 20 to Win!

enter the next 1, 2, or 3 numbers in sequence:  1

Computer played: [2, 3]

enter the next 1, 2, or 3 numbers in sequence:  4

Computer played: [5, 6, 7]

enter the next 1, 2, or 3 numbers in sequence:  8

Computer played: [9, 10, 11]

enter the next 1, 2, or 3 numbers in sequence:  12

Computer played: [13, 14, 15]

enter the next 1, 2, or 3 numbers in sequence:  16

Computer played: [17, 18, 19]

enter the next 1, 2, or 3 numbers in sequence:  20

Player Wins!!!
```

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.

[2]:
```
import math

def ncr(n, r):
    """Function to calculate n choose r (C(n, r)) using the formula."""
    return math.factorial(n) // (math.factorial(r) * math.factorial(n - r))

def print_pascal_triangle(rows):
    """Function to print Pascal's Triangle with the given number of rows."""
    for n in range(rows):
        # Print spaces for formatting the triangle shape
        print(" " * (rows - n), end=" ")

        # Print each combination in row n
        for r in range(n + 1):
            print(ncr(n, r), end=" ")
```

```
        print()

# Print Pascal's Triangle with 5 rows
rows = int(input("Enter the number of rows for Pascal's Triangle: "))
print_pascal_triangle(rows)
```

```
Enter the number of rows for Pascal's Triangle:  6
```

```
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
1 5 10 10 5 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.

[3]:
```
numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
frequency_count = {}
for num in numbers:
    if num in frequency_count:
        frequency_count[num] += 1
    else:
        frequency_count[num] = 1
print("Output:")
for number, count in frequency_count.items():
    print(f"Element {number} has come {count} times")
```

```
Enter numbers separated by spaces:  1 2 3 4 5 6 7 2 4 2 5 8 9 8 3
```

```
Output:
Element 1 has come 1 times
Element 2 has come 3 times
Element 3 has come 2 times
Element 4 has come 2 times
Element 5 has come 2 times
Element 6 has come 1 times
Element 7 has come 1 times
Element 8 has come 2 times
Element 9 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.

[4]:
```
def read_matrices(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()
```

```
    # Extract lines for Matrix A and Matrix B
    matrix_a_lines = lines[:2]
    matrix_b_lines = lines[3:5]

    # Parse Matrix A
    matrix_a = []
    for line in matrix_a_lines:
        matrix_a.append([int(num) for num in line.split()])

    # Parse Matrix B
    matrix_b = []
    for line in matrix_b_lines:
        matrix_b.append([int(num) for num in line.split()])

    return matrix_a, matrix_b

def add_matrices(matrix_a, matrix_b):
    # Initialize the result matrix as a 2x2 matrix filled with 0s
    result_matrix = [[0, 0], [0, 0]]

    for i in range(2):
        for j in range(2):
            result_matrix[i][j] = matrix_a[i][j] + matrix_b[i][j]

    return result_matrix

# Specify the file path
filename = r"D:\matrices.txt"  # Update this path as needed

# Read matrices from file
matrix_a, matrix_b = read_matrices(filename)

# Perform addition
result_matrix = add_matrices(matrix_a, matrix_b)

# Print result
print("Matrix A:")
for row in matrix_a:
    print(row)

print("Matrix B:")
for row in matrix_b:
    print(row)

print("Result of A + B:")
for row in result_matrix:
    print(row)
```

```
Matrix A:
[1, 2]
[3, 4]
Matrix B:
[5, 6]
[7, 8]
Result of A + B:
[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

[6]:
```python
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 __add__(self, other):
        # Check if the other object is a Fraction
        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)

    def simplify(self):
        def find_common_factor(numerator, denominator):
            # Find common factors by trial division
            min_value = min(abs(numerator), abs(denominator))
            for i in range(min_value, 1, -1):
                if numerator % i == 0 and denominator % i == 0:
                    return i
            return 1

        common_factor = find_common_factor(self.numerator, self.denominator)
        self.numerator //= common_factor
        self.denominator //= common_factor

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

```python
try:
    # Taking input from the user for the first fraction
    num1 = int(input("Enter numerator for the first fraction: "))
    den1 = int(input("Enter denominator for the first fraction: "))
    fraction1 = Fraction(num1, den1)

    # Taking input from the user for the second fraction
    num2 = int(input("Enter numerator for the second fraction: "))
    den2 = int(input("Enter denominator for the second fraction: "))
    fraction2 = Fraction(num2, den2)

    # Adding the two fractions
    result = fraction1 + fraction2

    # Output
    print(f"{fraction1} + {fraction2} = {result}")

except ValueError as ve:
    print(f"Error: {ve}")
except ZeroDivisionError:
    print("Denominator cannot be zero.")
```

```
Enter numerator for the first fraction:  1
Enter denominator for the first fraction:  3
Enter numerator for the second fraction:  4
Enter denominator for the second fraction:  6

1/3 + 2/3 = 1/1
```

[ ]: