assignment-1

October 27, 2024

[11]:
```
#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 5
Computer played: [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:  11 12 13
Invalid move! Please enter 1 to 3 numbers in sequence starting from the last
number.
enter the next 1, 2, or 3 numbers in sequence:  12 13
Computer played: [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!!!
```

[26]:
```
#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.
import math

def ncr(n, r):
    return math.comb(n, r)  # Alternatively: math.factorial(n) // (math.
 ↪factorial(r) * math.factorial(n - r))

def print_pascals_triangle(rows):
    for n in range(rows):
        row = []
        for r in range(n + 1):
            row.append(ncr(n, r))
        print(" " * (rows - n), " ".join(map(str, row)))

# Example usage:
num_rows = int(input("Enter the number of rows: "))
print_pascals_triangle(num_rows)
```

```
Enter the number of rows:  5
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
```

[28]:
```python
#Read a list of n numbers during runtime. Write a Python program to print the repeated elements with frequency count in a list.
# Read a list of numbers from the user (example input: "2 1 2 3 4 5 1 3 6 2 3 4")
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 2 3 2 4 5 6 4 5 7 2 3 5 7 9
Output:
Element 1 has come 1 times
Element 2 has come 4 times
Element 3 has come 3 times
Element 4 has come 2 times
Element 5 has come 3 times
Element 6 has come 1 times
Element 7 has come 2 times
Element 9 has come 1 times
```

[9]:
```python
#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.
def read_matrix_from_file(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        matrix = []
        for line in lines:
            # Convert each line into a list of integers
            row = list(map(int, line.strip().split()))
            matrix.append(row)
        return matrix

def add_matrices(A, B):
```

```
    result = []
    for i in range(2):
        row = []
        for j in range(2):
            row.append(A[i][j] + B[i][j])
        result.append(row)
    return result

def print_matrix(matrix):
    for row in matrix:
        print(' '.join(map(str, row)))

# Main program
file_path = r'C:\Users\I SAIJAYASREE\Downloads\Matrices.txt'
A = []
B = []

with open(file_path, 'r') as file:
    lines = file.readlines()

    for i in range(2):
        row = list(map(int, lines[i].strip().split()))
        A.append(row)
    for i in range(2, 4):
        row = list(map(int, lines[i].strip().split()))
        B.append(row)
result = add_matrices(A, B)
print("matrix A", A)
print("matrix B", B)
print("Result of Matrix A + Matrix B:")
print_matrix(result)
```

```
matrix A [[1, 2], [3, 4]]
matrix B [[5, 6], [7, 8]]
Result of Matrix A + Matrix B:
6 8
10 12
```

[17]:
```
#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
class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero.")
        self.numerator = numerator
```

```
        self.denominator = denominator

    def __add__(self, other):
        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).simplify()

    def simplify(self):
        def gcd(a, b):
            while b:
                a, b = b, a % b
            return a
        common_divisor = gcd(abs(self.numerator), abs(self.denominator))
        self.numerator //= common_divisor
        self.denominator //= common_divisor
        if self.denominator < 0:
            self.numerator = -self.numerator
            self.denominator = -self.denominator
        return self

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

    def __repr__(self):
        return f"Fraction({self.numerator}, {self.denominator})"

numerator1 = int(input("Enter the numerator for the first fraction: "))
denominator1 = int(input("Enter the denominator for the first fraction: "))
numerator2 = int(input("Enter the numerator for the second fraction: "))
denominator2 = int(input("Enter the denominator for the second fraction: "))

fraction1 = Fraction(numerator1, denominator1)
fraction2 = Fraction(numerator2, denominator2)
result = fraction1 + fraction2
print(f"{fraction1} + {fraction2} = {result}")
```

```
Enter the numerator for the first fraction:  1
Enter the denominator for the first fraction:  2
Enter the numerator for the second fraction:  3
Enter the denominator for the second fraction:  4

1/2 + 3/4 = 5/4
```

[ ]: