

Dr. C. Syamsundar

Assistant Professor and Dean R&D,
Department of Mechanical Engineering,
CMR Engineering College (UGC Autonomous),
Hyderabad 501 401, INDIA.

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!!!

Algorithm

1. Initialize Game State:

- Set a counter `current_number` to keep track of the last number played.
- Set `MAX_NUMBER` to 20, which is the target number to win.

2. Define Functions for Turns:

- Define a function for the **user turn** that accepts a sequence input and validates it.
- Define a function for the **computer turn** that generates a sequence based on a simple strategy.

3. Game Loop:

- Alternate turns between user and computer, updating `current_number` each time.
- Check after each turn if the last number reached is 20. If so, end the game and declare the winner.

Program

```
import random

def user_turn(current_number):

    while True:
        try:
            user_input = input(f"Enter 1, 2, or 3 numbers starting from {current_number + 1}: ")
            # Split input into a list of integers
            user_sequence = list(map(int, user_input.split()))
            # Validate the sequence length and the sequence continuity

            if 1 <= len(user_sequence) <= 3 and user_sequence[0] == current_number + 1 and all(
                user_sequence[i] == user_sequence[i - 1] + 1 for i in range(1, len(user_sequence))):
                return user_sequence[-1] # Return the last number in the sequence

        else:
            print("Invalid sequence, please try again.")

    except ValueError:
        print("Invalid input, enter numbers separated by spaces.")

def computer_turn(current_number):

    # Computer strategy: pick 1 to 3 numbers to get closer to 20 but avoid letting user win
    max_pick = min(3, 20 - current_number)
    computer_sequence = list(range(current_number + 1, current_number + 1 + max_pick))
    print(f"Computer picks: {' '.join(map(str, computer_sequence))}")
    return computer_sequence[-1] # Return the last number in the sequence

def play_game():
    current_number = 0
    MAX_NUMBER = 20

    while current_number < MAX_NUMBER:
        # User's turn
        print("\nYour turn:")
        current_number = user_turn(current_number)
        if current_number >= MAX_NUMBER:
            print("Congratulations! You reached 20 and won the game!")
            return

        # Computer's turn
        print("\nComputer's turn:")
        current_number = computer_turn(current_number)
        if current_number >= MAX_NUMBER:
            print("Computer reached 20 and won the game. Better luck next time!")
            return

# Start the game
play_game()
```

Output -1:

```
IDLE Shell 3.12.6
File Edit Shell Debug Options Window Help
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\pc\AppData\Local\Programs\Python\Python312\1.py

Your turn:
Enter 1, 2, or 3 numbers starting from 1: 1 2

Computer's turn:
Computer picks: 3 4 5

Your turn:
Enter 1, 2, or 3 numbers starting from 6: 6 7 8

Computer's turn:
Computer picks: 9 10 11

Your turn:
Enter 1, 2, or 3 numbers starting from 12: 12 13 14

Computer's turn:
Computer picks: 15 16 17

Your turn:
Enter 1, 2, or 3 numbers starting from 18: 18 19 20
Congratulations! You reached 20 and won the game!
>>> |
```

Output -2:

```
IDLE Shell 3.12.6
File Edit Shell Debug Options Window Help
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\pc\AppData\Local\Programs\Python\Python312\1.py

Your turn:
Enter 1, 2, or 3 numbers starting from 1: 1 2 3

Computer's turn:
Computer picks: 4 5 6

Your turn:
Enter 1, 2, or 3 numbers starting from 7: 7 8 9

Computer's turn:
Computer picks: 10 11 12

Your turn:
Enter 1, 2, or 3 numbers starting from 13: 13 14 15

Computer's turn:
Computer picks: 16 17 18

Your turn:
Enter 1, 2, or 3 numbers starting from 19: 19

Computer's turn:
Computer picks: 20
Computer reached 20 and won the game. Better luck next time!
>>> |
```

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 compute nCr

```
def ncr(n, r):

    if r > n:
        return 0
    if r == 0 or r == n:
        return 1

    # Calculate factorial
    numerator = 1
    denominator = 1
    for i in range(r):
        numerator *= (n - i)
        denominator *= (i + 1)
    return numerator // denominator # Integer division
```

Function to print Pascal's Triangle using nCr

```
def print_pascals_triangle(rows):
    for n in range(rows):

        # Print leading spaces for triangle shape
        print(" " * (rows - n), end="")

        # Print each number in the row using nCr
        for r in range(n + 1):
            print(ncr(n, r), end=" ")
        print() # Move to the next line
```

Input for number of rows

```
rows = int(input("Enter the number of rows for Pascal's Triangle: "))
print_pascals_triangle(rows)
```

Output:

```
IDLE Shell 3.12.6
File Edit Shell Debug Options Window Help
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Python/JNTUH-Data Science/Assignment 1/P2.py
Enter the number of rows for Pascal's Triangle: 4
 1
 1 1
1 2 1
1 3 3 1
>>>
===== RESTART: D:/Python/JNTUH-Data Science/Assignment 1/P2.py =====
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.

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

Program:

Function to count frequencies of elements in the list

```
def count_frequencies(lst):
```

```
    frequency = { }
```

Count occurrences of each element

```
    for num in lst:
```

```
        if num in frequency:
```

```
            frequency[num] += 1
```

```
        else:
```

```
            frequency[num] = 1
```

Print each element with its frequency count

```
    for num, count in frequency.items():
```

```
        print(f"Element {num} has come {count} times")
```

Input: list of numbers from user

```
numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
```

Call function to count and print frequencies

```
count_frequencies(numbers)
```

Output:

```
IDLE Shell 3.12.6
File Edit Shell Debug Options Window Help
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Python/JNTUH-Data Science/Assignment 1/P3.py
Enter numbers separated by spaces: 2 1 2 3 4 5 1 3 6 2 3 4
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 matrix 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:

Function to read a 2x2 matrix from the file

```
def read_matrix(file, label):
    matrix = [ ]
    for line in file:
        if line.strip() == label:
            for _ in range(2):
                row = list(map(int, file.readline().split()))
                matrix.append(row)
            break
    return matrix
```

Function to add two 2x2 matrices

```
def add_matrices(A, B):
    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
```

Reading matrices from file

```
with open("matrices.txt", "r") as file:
    matrix_A = read_matrix(file, "Matrix A:")
    matrix_B = read_matrix(file, "Matrix B:")
```

Perform matrix addition

```
result_matrix = add_matrices(matrix_A, matrix_B)
```

Print the result

```
print("Resultant Matrix after Addition:")
for row in result_matrix:
    print(" ".join(map(str, row)))
```

Output:

Matrix A:

1 2
3 4

Matrix B:

5 6
7 8

Resultant Matrix after Addition:

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 form 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() # Simplify upon creation

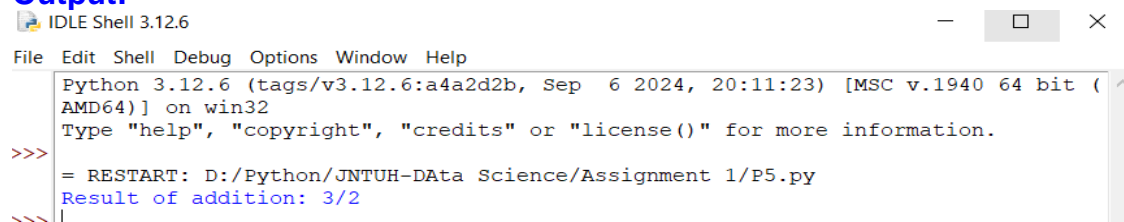
    def _simplify(self):
        """Simplify the fraction by dividing both numerator and denominator by their
        GCD."""
        gcd = math.gcd(self.numerator, self.denominator)
        self.numerator //= gcd
        self.denominator //= gcd

    def __add__(self, other):
        """Overload the + operator to add two Fraction objects."""
        if isinstance(other, Fraction):
            # Calculate the numerator and denominator for the result
            result_numerator = (self.numerator * other.denominator) + (other.numerator *
self.denominator)
            result_denominator = self.denominator * other.denominator
            return Fraction(result_numerator, result_denominator) # Return a new simplified
Fraction
        else:
            raise TypeError("Operands must be instances of Fraction class")

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

# Example usage
f1 = Fraction(1, 4)
f2 = Fraction(5, 4)
result = f1 + f2
print("Result of addition:", result)
```

Output:



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Python/JNTUH-Data Science/Assignment 1/P5.py
Result of addition: 3/2
>>>
```