USECASE : 1

use case - Finding the winning strategy in a card game in python

Problem Description:-

Imagine a Card game where each player receives a hand of cards with values. The objective is to find the best way to maximize the score for a player, assuming the players take turn drawing cards.

Assumptions:-

* Each player tries to maximize their score.
* Cards are represented by integers, which indicate their values.
* Two players alternate turns, and each player picks a card from either the beginning or the end of the list.

You need to design an algorithm that helps a player find the optimal strategy to guarantee the highest possible score given that the opponent is also playing optimally.

plan:-

we can solve this problem using dynamix programming by calculating the optimal score for every possible scene optimally.

steps:-

1) Define the Game.
2) Recursive strategy.
3) Dynamic programming.
4) Base capes.

Program:-

```
def find-optined-strategy (cards):
    n = len[cards]
# create a memoization table to store sabproblem results
    dp = [[0]*n for _ in range[n]]
# fill the table for subproblems of increasing sizes
    for length in range [1,n+1]:
        for i in range [n length+1):
            j = i + length - 1
# if only one card is left, the player taks it
            if i = i:
                dp[i] [j] = cards[i]
            else:
```

Choose the best of two choices
1. Take the left card, and opponent plays optimally on remaining
2. Take the right card, and oppenent plays optimally remaining

take_left = cards[i] - dp[i+1][j]
take_right = cards[j] - dp[i][j-1]
dp[i][j] = max[take_left, take_right]

# dp[0][n-1] will have optimal score for first player
return (dp[0][n-1] + sum(cards)) //2 # first player's max
passive score

# Example age
cards = [3,9,1,2]
print ["first player's optimal score:", find_optimal_strategy(cards)]

Explanation:-

* Dynamic programming table (dp): Each cell dp[i][j] represents the difference in score b/w the first player and opponent if game is p
b/w cards from index to index j.
* Two choices: for each move, the player can either to play optimal
remaining cards.
* pick the right most cards[j], leaving the opponents the res
cards.
* Recursive Relation: the value of each subproblem is determined
minimizing the score difference b/w current player and opponent

Example walk through:-

Consider the array of cards: [3,9,1,2]

1.) first player (you) can choose between:
* Taking the leftmost card (3), leaving the cards [9,1,2]
* Taking the right most card (2), leaving the cards [3,9,1]

2.) the opponent will then take their turn, playing optimally to minimize of player's first sum.

this program computes the best possible outcome for first player.

first player's optimal score, 5

Minimizing strategy:-

By using Dynamic programming; we ensure that solution is computed efficients avoiding redundant calculations. This approach ensures both players play optimally, and first player gets the highest score possible given the opponent's best move.

Completed

VEL TECH	
EX No.	13
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	15/10