Use case: 1

Use Case - Finding the winning strategy in a card game in python

Problem Description: Imagine a card game where each player recevies a hand of cards with values. The objective is to find the best way to maximize the score for a player, assuming Cards. Each player can either pick the first or last card from remanning file.

Assumptions:

- Each Player tries to maximize this score.
- Cards are represented by integer.

Algorithms:

1. Define the game: Repreat the file of cards as int
2. Recusive Strategy: A function will recursively determine the best score.
3. Score immedrate to avoid recalulating them
4. Base cases: when only one card is left, current Player takes it

Program:

```
def find_optional_strategy(cards):
    n=len(cards)
    dp=[[0]*n for _ in range(n)]
    for length in range(1,n+1):
        for i in range in(n-length+1):
```

Output:

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

y. First player can choose

. Taking left most card (3), leaving cards [9, 1, 2]

. Taking right most card (2), leaving cards [3, 9, 1]

First player optimal score: 5

```
        j = i + length - 1
        if i == j:
            dp[i][j] = cards[i]
        else:
            take_left = cards[i] - dp[i+1][j]
            take_right = cards[j] - dp[i][j-1]
            dp[i][j] = max(take_left, take_right)
    return (dp[0][n-1] + sum(cards))
cards = [3, 9, 1, 2]
Print("First Players optional score:", find_optional_strategy
(cards))
```

Explanation:

. DP: each cell dp[i][j] represents difference in score difference the each two players played between cards from i to j.

. Two choices for each move.

1. Pick the left most card [i]
2. Pick the right most card [j]

. Recursive relation: The value of each subproblem is maximizing the score difference btw players.

Result:

Thus, the program for use case is executed successfully and output is verified.

VELTECH	
EX.NO.	13
PERFORMANCE(5)	5
RESULT AND ANALYSIS(5)	5
VIVA VOCE(5)	5
RECORD(5)	
TOTAL (15)	20
SIGN WITH DATE	15/10