Use case -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 card with values the objective is to find the best way to maximize cost from the remaining pile

Assumptions

- each player tries to max their score
- cards are represented by integers which indicates their values
- two players alternate turns & each player picks a card from either the beginning or end of the list

plan: we can solve problem dynamic programming by calculate the optimal score for every possible scenario

steps: define the game: Represent the pile of card as list of integers

2) Recursive strategy

3) Dynamic program

4) Basic case

Program

```
def find_optimal_strategy(cards):
    n=len(cards)
    # create a memorization table to store
      sub problem result
    dp=[[0] * n for _ in range(n)]
    # fill the table for sub problem
      of increasing size
    for length in range(1)
    j = i+ length
    else:
        # close test of two choic
```

Example walk through:

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

Optimizing strategy:

By using dynamic programming we ensure that the solution is computed efficiently avoiding redundant calculations. This approach ensures both players play optimally and the first player gets the highest score possible given the opponents best move.

```
# choose the best of two choices
# 1. Take the left coin, and the opponent
plays optimally on the remaining (i+1, j)
    # 2. Take the right coin, and the
opponent plays optimally on the remaining
(i, j-1)
fake_left = coins[i] - dp[i+1][j]
fake_right = coin[j] - dp[i][j-1]
dp[i][j] = max (fake_left, fake_right)
return (dp[0][n-1] + sum(coins))//2
player's maximum possible score
# example case
coin = [3, 9, 1, 2]
print("first player's optimal score:",
find_optimal_strategy(coins))
```

Example walkthrough

consider the array of coins [3, 9, 1, 2]

1. first player (you) can choose between:
   - Taking the leftmost (3), leaving the coins [9, 1, 2].
   - Taking the rightmost coin (2) leaving the coins [3, 9, 1].
2. The opponent will then make their turn, playing optimally to minimize the first player's score.

This program computes the best possible outcome for the first player.
first player's optimal score: 5
first player, if played optimally, can guarantee a score of 5 regardless of how the opponent plays.

optimizing strategy

By using dynamic programming, we ensure that the solution is computed efficiently avoiding optimally. & the first ensure first player gets the highest score possible give the opponent's best move

	13
	5
	3
...I AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	20
TOTAL (20)	5
SIGN WITH DATE	25/10