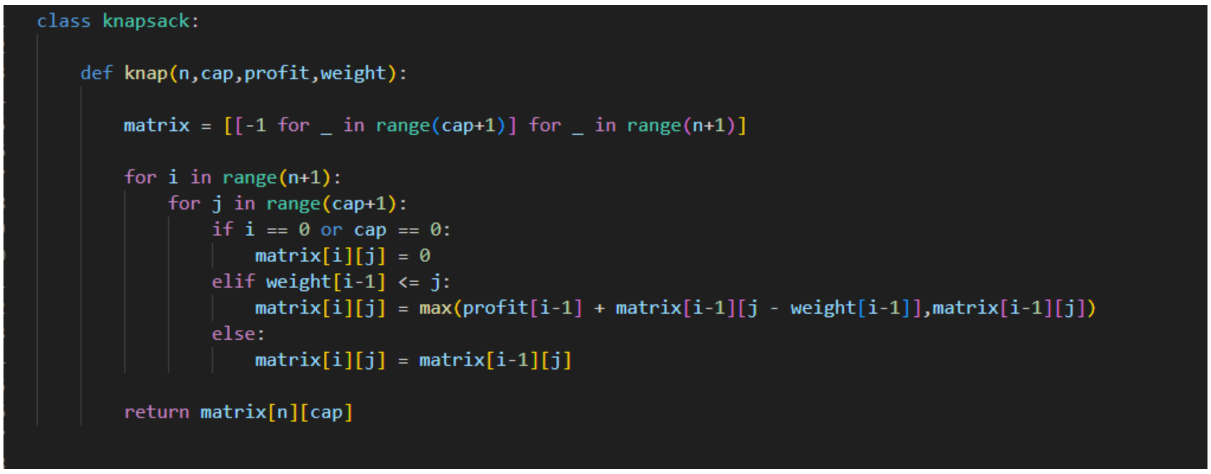
DAA Lab Assignment Week 8

Shashank K

SE21UCSE198

CSE-3

Logic snippet:



```
class knapsack:

    def knap(n,cap,profit,weight):

        matrix = [[-1 for _ in range(cap+1)] for _ in range(n+1)]

        for i in range(n+1):
            for j in range(cap+1):
                if i == 0 or cap == 0:
                    matrix[i][j] = 0
                elif weight[i-1] <= j:
                    matrix[i][j] = max(profit[i-1] + matrix[i-1][j - weight[i-1]],matrix[i-1][j])
                else:
                    matrix[i][j] = matrix[i-1][j]

        return matrix[n][cap]
```

Class knapsack has a method called knap which takes in number of jobs, max capacity of the bag, List of profits and list weights as arguments.

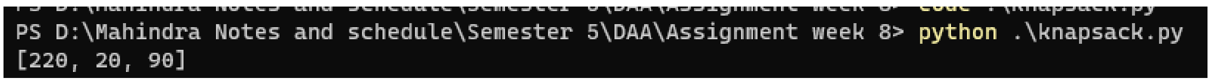
The method generates a matrix of dimensions (capacity+1)*(count+1) full of -1.

Then we traverse the entire matrix and check certain conditions which help us find the ideal jobs to take so that we don’t fill up our hypothetical basket.

The question asks us to take a dynamic programming approach rather than a recursive approach.

The time and space complexity of the approach is O(count*capacity).

Result snippets



```
PS D:\Mahindra Notes and schedule\Semester 5\DAA\Assignment week 8> python .\knapsack.py
[220, 20, 90]
```