**Code:-**

def knapSack(W, wt, val, n):

# Base Case

if n == 0 or W == 0:

return 0

# If weight of the nth item is more than Knapsack of capacity W, then this item cannot be included

# in the optimal solution

if (wt[n-1] > W):

return knapSack(W, wt, val, n-1)

# return the maximum of two cases: (1) nth item included (2) not included

else:

return max(

val[n-1] + knapSack(

W-wt[n-1], wt, val, n-1),

knapSack(W, wt, val, n-1))

if '\_\_main\_\_' == \_\_name\_\_:

val = input('Enter the values of the item(s) in order: ').split()

val = [int(v) for v in val] # 60 100 120

wt = input('Enter the positive weights of the item(s) in order: ').split()

wt = [int(w) for w in wt] # 10 20 30

W = int(input('Enter the maximum capacity of the knapsack: ')) # 50

n = len(val)

print(knapSack(W, wt, val, n))

**Output:-**

C:\Users\asus\PycharmProjectsCommunity\LP3\venv\Scripts\python.exe "F:\7th Sem\LP3 Practical\DAA\_FInal\4\_0-1\_Knapsack\0-1 Knapsack Problem.py"

Enter number of items: 4

Enter the values of the 4 item(s) in order: 30 25 2 6

Enter the positive weights of the 4 item(s) in order: 15 10 2 4

Enter maximum weight: 37

The maximum value of items that can be carried: 63

Process finished with exit code 0

Enter number of items: 5

Enter the values of the 5 item(s) in order: 15 10 2 4 3

Enter the positive weights of the 5 item(s) in order: 30 36 20 10 5

Enter maximum weight: 40

The maximum value of items that can be carried: 19

Process finished with exit code 0