**Name:-Amol Sakharam Gaikwad**

**Roll No:-COBA018**

**CODE**:-

# Structure for an item which stores weight and

# corresponding value of Item

class Item:

def \_\_init\_\_(self, value, weight):

self.value = value

self.weight = weight

# Main greedy function to solve problem

def fractionalKnapsack(W, arr):

# sorting Item on basis of ratio

arr.sort(key=lambda x: (x.value/x.weight), reverse=True)

# Uncomment to see new order of Items with their

# ratio

# for item in arr:

# print(item.value, item.weight, item.value/item.weight)

# Result(value in Knapsack)

finalvalue = 0.0

# Looping through all Items

for item in arr:

# If adding Item won't overflow, add it completely

if item.weight <= W:

W -= item.weight

finalvalue += item.value

# If we can't add current Item, add fractional part

# of it

else:

finalvalue += item.value \* W / item.weight

break

# Returning final value

return finalvalue

# Driver's Code

if \_\_name\_\_ == "\_\_main\_\_":

# Weight of Knapsack

W = 50

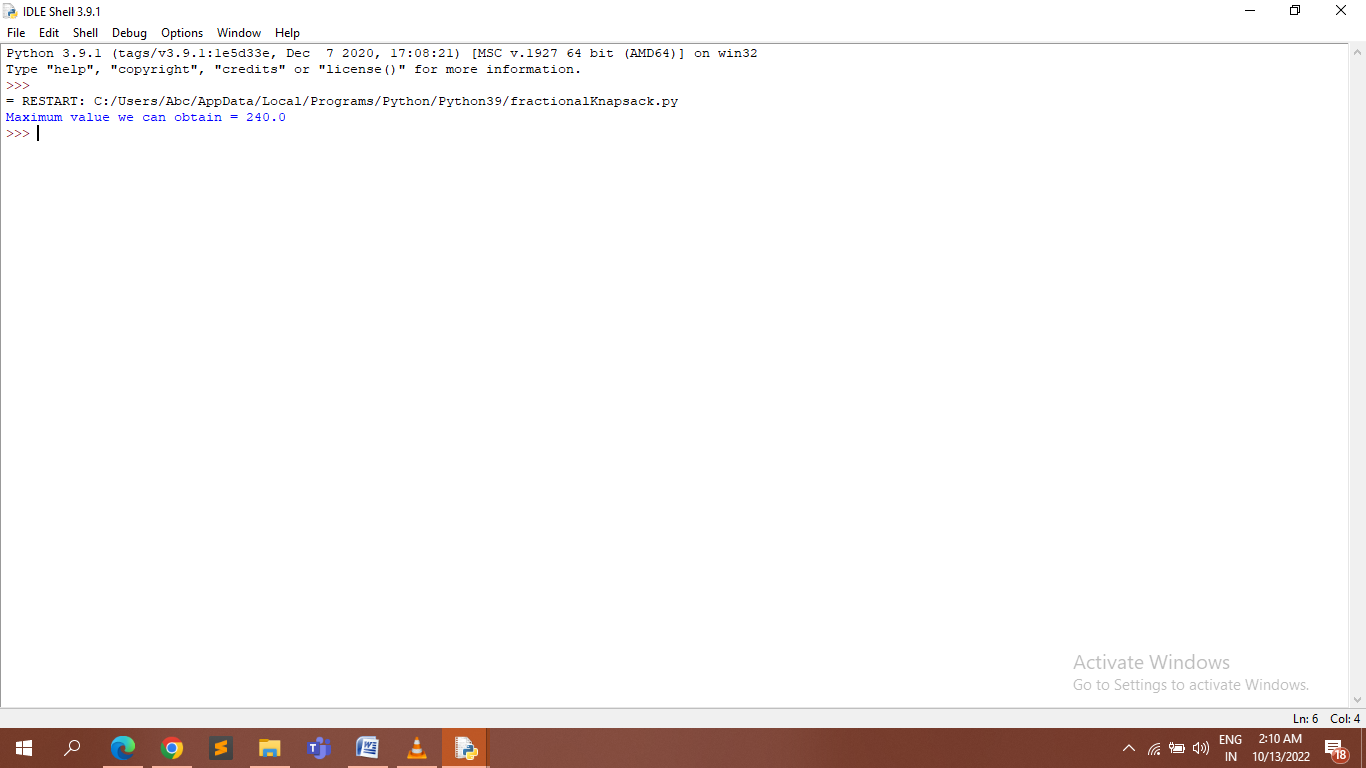
arr = [Item(60, 10), Item(100, 20), Item(120, 30)]

# Function call

max\_val = fractionalKnapsack(W, arr)

print ('Maximum value we can obtain = {}'.format(max\_val))

**OUTPUT:-**

****