```
# Structure for an item which stores weight and
# corresponding value of Item
class Item:
    def __init__(self, profit, weight):
        self.profit = profit
        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.profit/x.weight), reverse=True)

    # 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.profit
```

```
        # If we can't add current Item,
        # add fractional part of it
        else:
            finalvalue += item.profit * W / item.weight
            break

    # Returning final value
    return finalvalue



# Driver Code
if __name__ == "__main__":
    W = 50
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]

    # Function call
    max_val = fractionalKnapsack(W, arr)
    print(max_val)
```