Python Lab 11A & 11B - Loading Cargo

Cheyenne Mitchell

CS 162

6/8/2025

**Lab 11A: Loading cargo**

**A. Starting Heuristic Code**

Use the provided code...

# Starting code:

cargo = [9, 7, 3, 4, 5]

**B. Modified Code for Two-Truck Check**

“If the crate cannot fit in truck1 check to see if it can fit in truck2.  If it can, add it to truck2.”

cargo = [9, 7, 3, 4, 5]

truck1 = []

truck2 = []

for crate in cargo:

if sum(truck1) + crate <= 14:

truck1.append(crate)

elif sum(truck2) + crate <= 14:

truck2.append(crate)

**C. Try different cargo lists manually**

I tried cargo lists like [10, 10, 4, 4] and [7, 7, 7, 7].  
The algorithm worked well when the weights of both trucks were even, but performed not so well when large crates came first and the remaining space wasn’t enough to fit smaller ones.

**D. & E. Sorting Results - Did sorting help?**  
Yes, sorting with:

cargo.sort(reverse=True)

helped my algorithm pack large crates first, which often led to more balanced truck loads.

**F. New Algorithm**

**Different solution (code):**

cargo = [9, 7, 3, 4, 5]

cargo.sort(reverse=True) # Sort large to small for better efficiency

truck1 = []

truck2 = []

for crate in cargo:

# Put crate on the lighter truck, if it fits

if sum(truck1) <= sum(truck2):

if sum(truck1) + crate <= 14:

truck1.append(crate)

else:

truck2.append(crate)

else:

if sum(truck2) + crate <= 14:

truck2.append(crate)

else:

truck1.append(crate)

print("Truck 1:", truck1, "Sum:", sum(truck1))

print("Truck 2:", truck2, "Sum:", sum(truck2))

**G. Classmate-breaking list**

Cargo list with optimal Solution:

cargo = [8, 8, 6, 6]

* Truck 1: [8, 6] → sum = 14
* Truck 2: [8, 6] → sum = 14  
  Perfectly balanced.

Cargo list that breaks the algorithm:

Cargo list: [10, 10, 4, 4]

* Truck 1: [10, 10] → sum = 20 (too much)
* Truck 2: [4, 4] → sum = 8

**Lab 11B: Loading Cargo with Randomized Lists**

**A. Import random module**

cargo = []  
while sum(cargo) < total:  
 crate = random.randint(1, total - sum(cargo))  
 cargo.append(crate)

**B. Generate random cargo list up to a total of 28**

cargo = []

while sum(cargo) < 28:

crate = random.randint(1, 28)

cargo.append(crate)

print(cargo, sum(cargo))

**C. Fix the issue of going over 28**  
To make sure the list always adds up to exactly 28.

crate = random.randint(1, 28 - sum(cargo)) #this line will limit the last crate to only what is needed.

**D. User Input for Total Weight**  
total = int(input("Enter total cargo weight: "))  
truck\_capacity = total // 2

**E. Handling Odd Totals**

In this step I made the code check to see if the total weight is odd and then assigned the extra 1ton to one of the trucks. This helps avoid imbalance.

truck\_capacity = total // 2 + (total % 2) *# one truck gets the extra if needed*