

```
In [ ]: Geben Sie den gewünschten Batterietyp ein (A, B oder C): b
Nº Batterien: 50
Nettogewicht Batterie: 23
Geben Sie die Höhe der Box in mm an:400
Geben Sie die Breite der Box in mm an:300
Geben Sie die Länge der Box in mm an:400

Geben Sie den gewünschten Batterietyp ein (A, B oder C): b
Nº Batterien: 50
Nettogewicht Batterie: 23
Geben Sie die Höhe der Box in mm an:400
Geben Sie die Breite der Box in mm an:300
Geben Sie die Länge der Box in mm an:400
- este de arriba tambien da error en pyrobubbles - no se puded correr por primer
- si doy la condicionde de tipo b por debajo de 30 kg, la taba da Nan en los co
- Redondear hacia arriba la info de DF alternativo - redondear hacia arriba las
- Avisa si queda estosck y donde en la Behältermanagement compartida

- alternative boxes simepre sale aunque si haya superado el max ruladon a nivel
- type b en pyro esta anhadiendo las box de pyro -> hay q poner un condicional q
```

New target:

poner poner varias baterias de distintas medidas y kgs, y decir en que baterias entrarian

```
In [ ]: type b el vemriculit no da bien
iinpus: b,50,400 (kg),300,400,300

In [ ]: Alternative errors: b, 50 numero, 200 kg, 300, 250,300

In [ ]: # it doesnt contain the Herstekker Gelkoh
```

Geben Sie den gewünschten Batterietyp ein (A, B oder C): b N° Batterien: 50

Nettogewicht Batterie: 23 Geben Sie die Höhe der Box in mm an:400 Geben Sie die Breite der Box in mm an:300 Geben Sie die Länge der Box in mm an:400

Geben Sie den gewünschten Batterietyp ein (A, B oder C): b N° Batterien: 50

Nettogewicht Batterie: 23 Geben Sie die Höhe der Box in mm an:400 Geben Sie die Breite der Box in mm an:300 Geben Sie die Länge der Box in mm an:400

- este de arriba tambien da error - no se puded correr por primer vez tipo b menor de 30 kg - no existe una variable, solo s epuded despues de haber corrido el codigo con otras tipos de baterias
- si doy la condicione de tipo b por debajo de 30 kg, la taba da Nan en los correctos de vermiculi y problemas con el Pyro, sin embargo en los otros modos (B >30kg y C) no lo hace.
- Redondear hacia arriba la info de DF alternativo - redondear hacia arriba las cjas necesarias, si se supera el max
- Avisa si queda estosck y donde en la Behältermanagement compartida

```
In [14]: # FINAL - solo con aviso de pyrobubbles y sin calcular el DF de max_zuladung_Error

import pandas as pd
import math

# cargar el archivo de datos
Behälterzuordnung = pd.read_excel('Behälterzuordnung(reduced).xlsx', sheet_name='S'
Behälterzuordnung['Zuladung'] = Behälterzuordnung['Zuladung'].fillna(0)

# convertir columnas a tipo int
Behälterzuordnung['B'] = Behälterzuordnung['B'].astype('int64')
Behälterzuordnung['Zuladung'] = Behälterzuordnung['Zuladung'].astype('int64')

# obtener los inputs del usuario

# Frage den Benutzer nach dem Batterietyp
battery_type = input("Geben Sie den gewünschten Batterietyp ein (A, B oder C): "

# Überprüfe den eingegebenen Batterietyp
if battery_type == "A":
    # Berechnungen und Operationen für Batterietyp A
    # ...
    pass
elif battery_type == "B":
    # Berechnungen und Operationen für Batterietyp B
    # ...
    pass
elif battery_type == "C":
    # Berechnungen und Operationen für Batterietyp C
    # ...
    pass
else:
    print("Ungültiger Batterietyp. Bitte geben Sie A, B oder C ein.")

num_batteries = int(input('Nº Batterien: '))
battery_weight = float(input('Nettogewicht Batterie: '))
box_height = float(input('Geben Sie die Höhe der Box in mm an:'))
box_width = float(input('Geben Sie die Breite der Box in mm an:'))
box_length = float(input('Geben Sie die Länge der Box in mm an:'))

# calcular el número de baterías que caben en cada caja
Behälterzuordnung['Batteries per box'] = Behälterzuordnung['Zuladung'] // battery_
#
# calcular numero de cajas necesarias
#Behälterzuordnung['Boxes needed']= 15 / Behälterzuordnung['Batteries per box']

# obtener las cajas que cumplen con las condiciones requeridas y el tipo de bate
boxes = Behälterzuordnung.loc[(Behälterzuordnung['L'] >= box_height) &
                               (Behälterzuordnung['B'] >= box_width) &
                               (Behälterzuordnung['H'] >= box_length) &
                               (Behälterzuordnung['Batterietyp'].str.contains(batte

#Vermiculite, inspirado de excel

#sizes
medida_a=boxes["L"]/int(box_height)
medida_b=boxes["B"]/int(box_width)
medida_c=boxes["H"]/int(box_length)
```

```

#redondeos
medida_a = medida_a.apply(lambda x: math.floor(x))
medida_b = medida_b.apply(lambda x: math.floor(x))
medida_c = medida_c.apply(lambda x: math.floor(x))

max_Anzahl_Batterien_Behälter= medida_a*medida_b*medida_c
#tipo A y B
Behälter_bei_Max_Auss = num_batteries / max_Anzahl_Batterien_Behälter
Behälter_bei_Max_Auss = Behälter_bei_Max_Auss.apply(lambda x: math.ceil(x))

boxes["Vermiculite_Ausschöpfung"] = ((boxes["L"]*boxes["B"]*boxes["H"])*Behälter_bei_Max_Auss)

boxes["Anzahl_100l_Säcke_Vermiculite"] = boxes["Vermiculite_Ausschöpfung"] / 100
boxes["Anzahl_100l_Säcke_Vermiculite"] = boxes["Anzahl_100l_Säcke_Vermiculite"].

# sorting Battery Type and its policy; Type C == 1 Box per batterey

if battery_type == "A":
    boxes["Anzahl_der_benötigen_Behälter"] = Behälter_bei_Max_Auss

#Calculating Zuladung for type A
#making formula for Zuladung
boxes["Vermiculite_Ausschöpfung_Zuladung"] = ((boxes["L"]*boxes["B"]*boxes["H"])*Behälter_bei_Max_Auss)
Zuladung_KG_je_Behälter=((battery_weight*num_batteries)+(boxes["Vermiculite_Ausschöpfung"]))

#turning zuladung into DF
Zuladung_KG_je_Behälter = pd.DataFrame(Zuladung_KG_je_Behälter)
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={"0": "Nuevo peso en KG"})

#renaming column
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={0: 'Zuladung'})

# Redondear la columna "Valores" a 2 decimales
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.round(2)

# Perform an inner merge on the index - Both DF must be in the same DF
merged_df = pd.merge(Zuladung_KG_je_Behälter, boxes, left_index=True, right_index=True)

# Suponiendo que ya tienes un DataFrame llamado df con las columnas 'Zuladung_KG_je_Behälter' y 'Nuevo peso en KG'
# Crear una nueva columna llamada 'Resultado' y establecerla inicialmente como 'No bueno'
merged_df['Resultado'] = 'Gut aber Check'

# Aplicar la condición y asignar 'no bueno' si se cumple
merged_df.loc[(merged_df['Zuladung_KG_je_Behälter'] >= merged_df['Zuladung']) & (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladung']) & (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladung'])]

#Reordenar las columnas

```

```
# Extraer la columna a mover
columna = merged_df.pop('Ergebnis')

# Insertar la columna en la posición deseada
posición = 9 # Posición a la que deseas mover la columna
merged_df.insert(posición, 'Ergebnis', columna)

# Extraer la columna a mover
columna = merged_df.pop('Anzahl_der_benötigen_Behälter')

# Insertar la columna en la posición deseada
posición = 10 # Posición a la que deseas mover la columna
merged_df.insert(posición, 'Anzahl_der_benötigen_Behälter', columna)

# Extraer la columna a mover
columna = merged_df.pop('Zuladung_KG_je_Behälter')

# Insertar la columna en la posición deseada
posición = 6 # Posición a la que deseas mover la columna
merged_df.insert(posición, 'Zuladung_KG_je_Behälter', columna)

boxes = merged_df.sort_values(by='Batteries per box', ascending=True)

elif battery_type == "B":

    # filtering weight of battery type B
    if battery_weight >=30:

        boxes["Anzahl_der_benötigen_Behälter"] = num_batteries

        # Calcular vermiculita para el tipo C

        max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c

        boxes["Vermiculite Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"]

        boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Vermiculite Gesamt in l"]

        boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Anzahl 100l Säcke Vermiculite"]

        boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Anzahl 100l Säcke Vermiculite"]
        boxes["Batteries per box"] = num_batteries # Type C = box per battery

        boxes.drop('Vermiculite_Ausschöpfung', axis=1, inplace=True) # keep only
        boxes.drop('Anzahl_100l_Säcke_Vermiculite', axis=1, inplace=True)

        # Ordenar las cajas filtradas
        boxes = boxes.sort_values(by='Batteries per box', ascending=True)

        # Calculating Zuladung for type C

        Zuladung_KG_je_Behälter = ((battery_weight * num_batteries) + (boxes["Ve

        # Turning Zuladung into a DataFrame
```

```
Zuladung_KG_je_Behälter = pd.DataFrame(Zuladung_KG_je_Behälter)
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={0: 'Zu

# Rounding the "Zuladung_KG_je_Behälter" column to 2 decimals
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.round(2)

# Perform an inner merge on the index - Both DF must be in the same DF
merged_df = pd.merge(Zuladung_KG_je_Behälter, boxes, left_index=True, ri

# Create a new column called 'Ergebnis' and initially set it as 'Gut aber Check'
merged_df['Ergebnis'] = 'Gut aber Check'

# Apply the condition and assign 'max. Zuladung des Behälters wird übers
merged_df.loc[(merged_df['Zuladung_KG_je_Behälter'] >= merged_df['Zuladu
    (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladu
    (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladu

# Reorder the columns
# Extract the column to move
columna = merged_df.pop('Ergebnis')

# Insert the column at the desired position
posicion = 9 # Position where you want to move the column
merged_df.insert(posicion, 'Ergebnis', columna)

# Extract the column to move
columna = merged_df.pop('Anzahl_der_benötigen_Behälter')

# Insert the column at the desired position
posicion = 10 # Position where you want to move the column
merged_df.insert(posicion, 'Anzahl_der_benötigen_Behälter', columna)

# Extract the column to move
columna = merged_df.pop('Zuladung_KG_je_Behälter')

# Insert the column at the desired position
posicion = 6 # Position where you want to move the column
merged_df.insert(posicion, 'Zuladung_KG_je_Behälter', columna)

# batteries per box in type B > 30kg and type C -> 1 battery = 1 box
merged_df['Batteries_per_box'] = 1

# Sort the boxes based on 'Batteries_per_box'
boxes = merged_df.sort_values(by='Batteries_per_box', ascending=True)

elif battery_weight <30:
    boxes["Anzahl_der_benötigen_Behälter"] = Behälter_bei_Max_Auss

    #Calculating Zuladung for type A
    #making formula for Zuladung

    boxes["Vermiculite_Ausschöpfung_Zuladung"] = ((boxes["L"]*boxes["B"]*box
```

```

Zuladung_KG_je_Behälter=((battery_weight*num_batteries)+(boxes["Vermicul
#turning zuladung into DF

Zuladung_KG_je_Behälter = pd.DataFrame(Zuladung_KG_je_Behälter)
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={"0": " 

#renaming column
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={0: 'Zu

# Redondear la columna "Valores" a 2 decimales
Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.round(2)

# Perform an inner merge on the index -Both DF must be in the same DF
merged_df = pd.merge(Zuladung_KG_je_Behälter,boxes , left_index=True, ri

# Suponiendo que ya tienes un DataFrame llamado df con las columnas 'Zul
# Crear una nueva columna llamada 'Resultado' y establecerla inicialment
merged_df['Ergebnis'] = 'Gut aber Check '

# Aplicar la condición y asignar 'no bueno' si se cumple
merged_df.loc[(merged_df['Zuladung_KG_je_Behälter'] >= merged_df['Zuladu
(merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladu
(merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladu

#Reordenar las columnas
# Extraer la columna a mover
columna = merged_df.pop('Ergebnis')

# Insertar la columna en la posición deseada
posicion = 9 # Posición a la que deseas mover la columna
merged_df.insert(posicion, 'Ergebnis', columna)

# Extraer la columna a mover
columna = merged_df.pop('Anzahl_der_benötigen_Behälter')

# Insertar la columna en la posición deseada
posicion = 10 # Posición a la que deseas mover la columna
merged_df.insert(posicion, 'Anzahl_der_benötigen_Behälter', columna)

# Extraer la columna a mover
columna = merged_df.pop('Zuladung_KG_je_Behälter')

# Insertar la columna en la posición deseada
posicion = 6 # Posición a la que deseas mover la columna
merged_df.insert(posicion, 'Zuladung_KG_je_Behälter', columna)

boxes = merged_df.sort_values(by='Batteries per box', ascending=True)

else:

```

```
print('Check the code line if battery_Type == "B" battery_weight >=30')

elif battery_type == "C":
    boxes = boxes.loc[boxes["Batterietyp"].str.contains('C')] # filtering Boxey
    boxes["Anzahl_der_benötigen_Behälter"] = num_batteries

    # Calcular vermiculita para el tipo C
    max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c

    boxes["Vermiculite Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"]) *
    boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Vermiculite Gesamt in l"] # t
    boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Anzahl 100l Säcke Vermiculit"
    boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Anzahl 100l Säcke Vermiculit"
    boxes["Batteries per box"] = num_batteries # Type C = box per battery

    boxes.drop('Vermiculite_Ausschöpfung', axis=1, inplace=True) # keep only Ver
    boxes.drop('Anzahl_100l_Säcke_Vermiculite', axis=1, inplace=True)

    # Ordenar las cajas filtradas
    boxes = boxes.sort_values(by='Batteries per box', ascending=True)

    # Calculating Zuladung for type C

    Zuladung_KG_je_Behälter = ((battery_weight * num_batteries) + (boxes["Vermic
    # Turning Zuladung into a DataFrame
    Zuladung_KG_je_Behälter = pd.DataFrame(Zuladung_KG_je_Behälter)
    Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.rename(columns={0: 'Zuladu

    # Rounding the "Zuladung_KG_je_Behälter" column to 2 decimals
    Zuladung_KG_je_Behälter = Zuladung_KG_je_Behälter.round(2)

    # Perform an inner merge on the index - Both DF must be in the same DF
    merged_df = pd.merge(Zuladung_KG_je_Behälter, boxes, left_index=True, right_
    # Create a new column called 'Ergebnis' and initially set it as 'Gut aber Ch
    merged_df['Ergebnis'] = 'Gut aber Check'

    # Apply the condition and assign 'max. Zuladung des Behälters wird überschri
    merged_df.loc[(merged_df['Zuladung_KG_je_Behälter'] >= merged_df['Zuladung'] [
        (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladung'] - 1
        (merged_df['Zuladung_KG_je_Behälter'] == merged_df['Zuladung'] + 1

    # Reorder the columns
    # Extract the column to move
    columna = merged_df.pop('Ergebnis')

    # Insert the column at the desired position
    posición = 9 # Position where you want to move the column
    merged_df.insert(9, 'Ergebnis', columna)
```

```
mergea_at.insert(posicion, 'Ergebnis', columnas)

# Extract the column to move
columnas = merged_df.pop('Anzahl_der_benötigen_Behälter')

# Insert the column at the desired position
posicion = 10 # Position where you want to move the column
merged_df.insert(posicion, 'Anzahl_der_benötigen_Behälter', columnas)

# Extract the column to move
columnas = merged_df.pop('Zuladung_KG_je_Behälter')

# Insert the column at the desired position

posicion = 6 # Position where you want to move the column
merged_df.insert(posicion, 'Zuladung_KG_je_Behälter', columnas)

# batteries per box in type B > 30kg and type C -> 1 battery = 1 box
merged_df['Batteries_per_box'] = 1

# Sort the boxes based on 'Batteries per box'
boxes = merged_df.sort_values(by='Batteries_per_box', ascending=True)

#boxes.drop('Anzahl_100L_Säcke_Vermiculite', axis=1, inplace=True)

#-----Pyrobubbles-----

#-----CALCULA EN CASO DE MAX ZULADUNG REACHED-----
# si no hay max zuladung reached, a dar error - ACHTUNG

if boxes["Ergebnis"].str.contains('max. Zuladung des Behälters wird überschritten'):

    alternative_boxes = boxes

    alternative_Anzahl_Batterien = num_batteries #BATTEREINE TYPE C

    alternative_boxes["alternative_Anzahl_Batterien"] = alternative_Anzahl_Batterien

    # new DF for alternatives - ACHTUNG BATTERIEN TYPE C

    # sort daqta, showing only infos reagrding max Zualdung overpassed

    alternative_boxes = alternative_boxes.loc[alternative_boxes["Ergebnis"] == "max. Zuladung des Behälters wird überschritten"]

    alternative_Anzahl_Batterien = alternative_boxes["Zuladung"] / battery_weight

    alternative_boxes["alternative_Anzahl_Batterien"] = alternative_Anzahl_Batterien

    # alternative Zuladung in kg bei max. Ausschöpfung (wenn max. Zuladung überschritten)
    # alternative boxes["alternative_Zuladung_Auss_KG"] = alternative_Anzahl_Batterien * battery_weight
```

```
alternative_boxes[ alternative_Zuladung_max_kg ] = alternative_max_kg

#alternative Anzahl der benötigten Behälter bei max. Ausschöpfung

# mat ceil solo se aplica a numero finitos y no infinitos

alternative_Anzahl_benötigten_Behälter_max_Ausschöpfung = num_batteries
alternative_boxes[ "alternative_Anzahl_benötigten_Behälter_max_Ausschöpfung" ] = alternative_Anzahl_benötigten_Behälter_max_Ausschöpfung

# eliminar filas que contienen un 0 en alternative_Zuladung_Auss_KG (si
alternative_boxes = alternative_boxes.drop(alternative_boxes[alternative_Zuladung_Auss_KG].eq(0))

# alternative Anzahl der benötigen Behälter para tipo A y B en el c = n°
alternative_Anzahl_benötigen_Behälter = alternative_Anzahl_benötigten_Behälter
alternative_boxes[ "alternative_Anzahl_benötigen_Behälter" ] = alternative_Anzahl_benötigen_Behälter

# Vermiculite Gesamt in l
alternative_max_Vermiculite_Gesamt_1=((Behälterzuordnung['L'])* Behälterzuordnung['V'])
alternative_boxes[ "alternative_max_Vermiculite_Gesamt_1" ] = alternative_max_Vermiculite_Gesamt_1

#alternative Zuladung in kg je Behälter (wenn max. Zuladung überschritte
alternative_Zuladung_kg_je_Behälter=((battery_weight*num_batteries)+(alternative_max_kg*alternative_max_Vermiculite_Gesamt_1))
alternative_boxes[ "alternative_Zuladung_kg_je_Behälter" ] = alternative_Zuladung_kg_je_Behälter
alternative_boxes[ "alternative_Zuladung_kg_je_Behälter" ] = alternative_Zuladung_kg_je_Behälter

#Anzahl 100l Säcke Vermiculite
alternative_Anzahl_100l_Säcke_Vermiculite = alternative_max_Vermiculite_Gesamt_1/(100)
alternative_boxes[ "alternative_Anzahl_100l_Säcke_Vermiculite" ] = alternative_Anzahl_100l_Säcke_Vermiculite
alternative_boxes[ "alternative_Anzahl_100l_Säcke_Vermiculite" ] = alternative_Anzahl_100l_Säcke_Vermiculite

#Menge Vermiculite je Behälter
alternative_Menge_Vermiculite_Behälter = alternative_max_Vermiculite_Gesamt_1/(alternative_max_kg)
alternative_boxes[ "alternative_Menge_Vermiculite_Behälter" ] = alternative_Menge_Vermiculite_Behälter

# Lista de columnas a eliminar
updated_alternative_boxes = alternative_boxes[ [ 'Hersteller', "Bezeichnung" ] ]
for column in updated_alternative_boxes.columns:
    if column not in columns_to_keep:
        updated_alternative_boxes = updated_alternative_boxes.drop(column, axis=1)

pass
else:
    pass # No hacer nada

else:
    print("Something went wrong at variable Anzahl_der_benötigen_Behälter")

----- FORMULA AFTER MAX ZULADUNG REACHED FOR TYPE A -----
```



```
#if (battery_type == "B" or battery_type == "A") and boxes[ "Ergebnis" ].str.contains("Max Zuladung übersteigt die Kapazität der Batterie"):
```



```
# new DF
#     alternative_boxes= boxes
# sort daata. showina only infos reaardina max Zualduna overpassed
```

```
# alternative_boxes = alternative_boxes.loc[alternative_boxes["Ergebnis"].str

# alternative_Anzahl_Batterien = alternative_boxes["Zuladung"]/battery_weight

# alternative_boxes["alternative_Anzahl_Batterien"] = alternative_Anzahl_Batt

#If there is a 0 in alternative_boxes["alternative_Anzahl_Batterien"] its be
# Check if any row contains the number 0
# mask = alternative_boxes.eq(0).any(axis=1)

# Replace the entire row with the message "Nettogewicht Batterie too high"
# alternative_boxes[mask] = 'Nettogewicht Batterie too high'

# alternative_Zuladung in kg bei max. Ausschöpfung (wenn max. Zuladung übers
# alternative_boxes["alternative_Zuladung_Auss_KG"] = alternative_Anzahl_Bat

#alternative Anzahl der benötigten Behälter bei max. Ausschöpfung

# mat ceil solo se aplica a numero finitos y no infinitos
# alternative_Anzahl_benötigten_Behälter_max_Ausschöpfung = num_batteries / a
# alternative_boxes ["alternative_Anzahl_benötigten_Behälter_max_Ausschöpfung"]

# eliminar filas que contienen un 0 en alternative_Zuladung_Auss_KG (si la b
# alternative_boxes = alternative_boxes.drop(alternative_boxes[alternative_box

# alternative Anzahl der benötigen Behälter para tipo A y B en el c = nº bat
# alternative_Anzahl_benötigten_Behälter = alternative_Anzahl_benötigten_Behäl
# alternative_boxes["alternative_Anzahl_benötigten_Behälter"] = alternative_An

# Vermiculite Gesamt in L

# alternative_max_Vermiculite_Gesamt_1=((Behälterzuordnung['L']* Behälterzuord
# alternative_boxes["alternative_max_Vermiculite_Gesamt_1"] = alternative_max_V

#alternative Zuladung in kg je Behälter (wenn max. Zuladung überschritten)
# alternative_Zuladung_kg_je_Behälter=((battery_weight*num_batteries)+(altern
# alternative_boxes["alternative_Zuladung_kg_je_Behälter"] = alternative_Zula
# alternative_boxes["alternative_Zuladung_kg_je_Behälter"] = alternative_boxe

#Anzahl 100L Säcke Vermiculite

# alternative_Anzahl_100L_Säcke_Vermiculite = alternative_max_Vermiculite_Ges
# alternative_boxes["alternative_Anzahl_100L_Säcke_Vermiculite"] = alternativ
# alternative_boxes["alternative_Anzahl_100L_Säcke_Vermiculite"] = alternativ

#Menge Vermiculite je Behälter

# alternative_Menge_Vermiculite_Behälter = alternative_max_Vermiculite_Gesam
```

```
#     alternative_boxes["alternative_Menge_Vermiculite_Behälter"] = alternative_M  
  
# Lista de columnas a eliminar  
# updated_alternative_boxes = alternative_boxes[['Hersteller','Bezeichnung','  
  
#else:  
#    pass  
  
#----- BESCHEID PYROBUBBLES-----  
  
# NOTA A VECES LAS COLUMNAS SE CREAN SIN ESPACIO O CON ESPACIO:  
# Anzahl_100L_Säcke_Vermiculite vs 'Anzahl 100L Säcke Vermiculite'  
# La columna Anzahl_100L_Säcke_Vermiculite va con esta columna: Vermiculite_Au  
# Pero La columna 'Anzahl 100L Säcke Vermiculite' (no guion) va con Vermiculite  
  
if any(boxes["Bezeichnung"].str.contains('Gen')):  
  
    if 'Anzahl 100l Säcke Vermiculite' in boxes.columns:  
        # La columna existe, realiza operaciones condicionales aquí  
  
        # Define the replacement value  
        palabra_similar = 'Pyrobubbles'  
        # Function to apply the replacement  
  
        def reemplazar_palabra(row):  
            if 'Gen' in row['Bezeichnung']:  
                return palabra_similar  
            return row['Anzahl 100l Säcke Vermiculite']  
        # Apply the replacement only when 'Bezeichnung' contains 'Gen'  
        boxes['Anzahl 100l Säcke Vermiculite'] = boxes.apply(reemplazar_palabra,  
        boxes['Vermiculite Gesamt in l'] = boxes.apply(reemplazar_palabra, axis=  
  
        pass  
  
    elif 'Anzahl_1001_Säcke_Vermiculite' in boxes.columns:  
        # La columna existe, realiza operaciones condicionales aquí  
  
        # Define the replacement value  
        palabra_similar = 'Pyrobubbles'  
        # Function to apply the replacement  
  
        def reemplazar_palabra(row):  
            if 'Gen' in row['Bezeichnung']:  
                return palabra_similar  
            return row['Anzahl_1001_Säcke_Vermiculite']  
        # Apply the replacement only when 'Bezeichnung' contains 'Gen'  
        boxes['Anzahl_1001_Säcke_Vermiculite'] = boxes.apply(reemplazar_palabra,  
        boxes['Vermiculite_Ausschöpfung_Zuladung'] = boxes.apply(reemplazar_pala  
  
        pass  
  
    else: # La columna no existe, realiza acciones alternativas aquí  
        pass
```

```
else:  
    pass  
  
#-----  
  
# PASAR INFORMACION SEGUN TIPO DE BATERIA Y EN FORMATO df  
  
if boxes.empty:  
    print("-----")  
    print("\033[1mSorry, none of our boxes fit your dimensions or can contain th  
  
else :  
    print("-----")  
  
    #boxes = merged_df.sort_values(by='Batteries per box', ascending=True)  
  
        # calcular el número de cajas necesarias  
    batteries_per_box = merged_df.iloc[0]['Batteries per box']  
    num_boxes = math.ceil(num_batteries / batteries_per_box)  
  
#----- EXPORTING FILES -----  
# Frage den Benutzer nach dem Exporting normal file  
question = input("Remember not to have any file called Boxes in Downloads be  
                "Do you want to export the data as an Excel file? Yes or No: ")  
  
# Überprüfe den eingegebenen Batterietyp  
if question == "YES" or question == "Y" or question == "Yes" or question ==  
    # Specify the file path and name for the Excel file  
    file_path = "~/Downloads/boxes.xlsx"  
  
    # Export the DataFrame to Excel  
    boxes.to_excel(file_path, index=False)  
    print("Excel file exported successfully.")  
  
#-----  
palabra_objetivo = "max. Zuladung des Behälters wird überschritten"  
  
# NO HAY ARCHIVO (DF)DE MAX; POR ENDE NO SE PUEDE EXPORTAR  
  
#     if any(boxes['Ergebnis'] == palabra_objetivo):  
#         # Frage den Benutzer nach dem exporting alternative file  
#         question = input("Do you want to export the 'max. Zuladung des Behä  
#             # Überprüfe den eingegebenen Batterietyp  
  
#             if question == "YES" or question == "Y" or question == "Yes" or que  
#                 # Specify the file path and name for the Excel file  
#                 file_path = "~/Downloads/alternative_boxes.xlsx"  
#                 # Export the DataFrame to Excel  
#                 updated_alternative_boxes.to_excel(file_path, index=False)  
#                 print("Excel file exported successfully.")  
  
#             elif question == "NO" or question == "N" or question == "no" or que
```

```
#             print("No export")

#
#         else:
#             print("Something went wrong. Please try again.")
#     else:
#         pass

elif question == "NO" or question == "N" or question == "no" or question ==
    print("No export")
#
#-----#
#     palabra_objetivo = "max. Zuladung des Behälters wird überschritten"
#
#     if any(boxes['Ergebnis'] == palabra_objetivo):
#         # Frage den Benutzer nach dem exporting alternative file
#         question = input("Do you want to export the 'max. Zuladung des Behä
#         # Überprüfe den eingegebenen Batterietyp
#
#     if question == "YES" or question == "Y" or question == "Yes" or que
#
#         # Specify the file path and name for the Excel file
#         file_path = "~/Downloads/alternative_boxes.xlsx"
#
#         # Export the DataFrame to Excel
#         updated_alternative_boxes.to_excel(file_path, index=False)
#         print("Excel file exported successfully.")
#
#     elif question == "NO" or question == "N" or question == "no" or que
#         print("No export")
#
#     else:
#         print("Something went wrong. Please try again.")

else:
    print("Something went wrong. Please try again.")
```

boxes

```
Geben Sie den gewünschten Batterietyp ein (A, B oder C): b
Nº Batterien: 28
Nettogewicht Batterie: 28
Geben Sie die Höhe der Box in mm an:360
Geben Sie die Breite der Box in mm an:400
Geben Sie die Länge der Box in mm an:380
```

```
C:\Users\llorman\AppData\Local\Temp\ipykernel_25516\2779278845.py:76: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
    boxes["Vermiculite_Ausschöpfung"] = ((boxes["L"]*boxes["B"]*boxes["H"]*Behälte
r_bei_Max_Auss)-(int(box_height)*int(box_width)*int(box_length)*num_batteries))/
1000000
C:\Users\llorman\AppData\Local\Temp\ipykernel_25516\2779278845.py:78: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
    boxes["Anzahl_100l_Säcke_Vermiculite"]=boxes["Vermiculite_Ausschöpfung"]/100
C:\Users\llorman\AppData\Local\Temp\ipykernel_25516\2779278845.py:79: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
    boxes["Anzahl_100l_Säcke_Vermiculite"] = boxes["Anzahl_100l_Säcke_Vermiculit
e"].apply(lambda x: math.ceil(x))
C:\Users\llorman\AppData\Local\Temp\ipykernel_25516\2779278845.py:245: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
    boxes["Anzahl_der_benötigen_Behälter"]=Behälter_bei_Max_Auss
C:\Users\llorman\AppData\Local\Temp\ipykernel_25516\2779278845.py:251: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
    boxes["Vermiculite_Ausschöpfung_Zuladung"] = ((boxes["L"]*boxes["B"]*boxes["
H"]*Behälter_bei_Max_Auss)-(int(box_height)*int(box_width)*int(box_length)*boxes
["Anzahl_der_benötigen_Behälter"]))/1000000
-----
Remember not to have any file called Boxes in Downloads before you select Yes.,
Do you want to export the data as an Excel file? Yes or No: no
No export
```

Out[14]:

		Hersteller	Bezeichnung	L	B	H	Zuladung	Zuladung_KG_je_Behälter	Leergewicht
0		Holzkiste	X114	770	570	540	114	74.23	40.0
14		Karton	GG-Karton 770	770	570	550	150	74.67	150.0
1		Holzkiste	Y390	1170	770	470	390	115.27	60.0
7		Europlast	Paloxe 760	1120	920	600	400	213.15	44.0
5		Gelkoh	XL+	2505	1665	430	950	565.87	1370.0
11		Nefab	Nefab	2354	1754	531	1850	605.77	203.0
4		Genius	Gen LP1	2699	2000	570	2024	1086.21	764.0

In []: boxes

PARA PYROBUBBLES TYP A UND TYP B < 30

In [17]: #PARA PYROBUBLES TYP A UND TYP B < 30

```

boxes["Anzahl_der_benötigen_Behälter"] = Behälter_beim_Max_Auss
    # Calcular vermiculita para el tipo C
max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"]) * boxe
#boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"] # type
boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(lambda x
    #boxes["Batteries per box"] = num_batteries # Type C = box per battery
    #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep only Ve
#boxes.drop('Anzahl 100l Säcke Vermiculite', axis=1, inplace=True)

#####
# AHORA HAY QUE SUBSTITUIR TEN LAS COLUMNAS DE PYRO LA PALABRA DE VERMICULI ----

# CODIGO ABAJO; SIGUIENTE CELDA

#if any(boxes["Anzahl_100l_Säcke_Vermiculite"].str.contains('Pyrobubbles')):
```

PARA PYROBUBLES TYP A UND TYP B > 30 and C

In []: #PARA PYROBUBLES TYP A UND TYP B > 30 and C

```

boxes["Anzahl_der_benötigen_Behälter"] = num_batteries
    # Calcular vermiculita para el tipo C
max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"]) * boxe
#boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"] # type
boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(lambda x
    #boxes["Batteries per box"] = num_batteries # Type C = box per battery
    #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep only Ve
#boxes.drop('Anzahl 100l Säcke Vermiculite', axis=1, inplace=True)
```

In [18]: boxes

Out[18]:

		Hersteller	Bezeichnung	L	B	H	Zuladung	Zuladung_KG_je_Behälter	Leergewicht
0		Holzkiste	X114	770	570	540	114	74.23	40.0
14		Karton	GG-Karton 770	770	570	550	150	74.67	150.0
1		Holzkiste	Y390	1170	770	470	390	115.27	60.0
7		Europlast	Paloxe 760	1120	920	600	400	213.15	44.0
5		Gelkoh	XL+	2505	1665	430	950	565.87	1370.0
11		Nefab	Nefab	2354	1754	531	1850	605.77	203.0
4		Genius	Gen LP1	2699	2000	570	2024	1086.21	764.0

Union final de pyrobubbles, ahora solo falta substituir los dastos de las rows q solo tienne vermiculi

```
In [8]: # ACTUNG LAS BATERIAS POR DEBAJO DE 30=KG Y TIPO B LAS CALULCA COMO TIPO C OR B

if 'Anzahl 100l Säcke Vermiculite' in boxes.columns:

    if (battery_type == "B" and battery_weight >= 30) or battery_type == "C":
        #PARA PYROBUBLES TYP A UND TYP B < 30 and C
        boxes["Anzahl der benötigen Behälter"] = num_batteries
        # Calcular vermiculita para el tipo C
        max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
        boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"])
        #boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"]
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(
        #boxes["Batteries per box"] = num_batteries # Type C = box per batte
        #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep
        #boxes.drop('Anzahl 100l Säcke Vermiculite', axis=1, inplace=True)

    elif battery_type == B and battery_weight < 30:

        max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
        boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"])
        #boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"]
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(
        #boxes["Batteries per box"] = num_batteries # Type C = box per batte
        #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep
        #boxes.drop('Anzahl 100l Säcke Vermiculite', axis=1, inplace=True)

elif 'Anzahl_100l_Säcke_Vermiculite' in boxes.columns:

    if (battery_type == "B" and battery_weight >= 30) or battery_type == "C":
        #PARA PYROBUBLES TYP A UND TYP B < 30 and C
        boxes["Anzahl der benötigen Behälter"] = num_batteries
        # Calcular vermiculita para el tipo C
        max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
        boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"])
        #boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"]
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(
        #boxes["Batteries per box"] = num_batteries # Type C = box per batte
        #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep
        #boxes.drop('Anzahl 100l Säcke Vermiculite', axis=1, inplace=True)

    elif battery_type == 'B' and battery_weight < 30:

        boxes["Anzahl der benötigen Behälter"] = Behälter_bei_Max_Auss
        # Calcular vermiculita para el tipo C
        max_Anzahl_Batterien_Behälter = medida_a * medida_b * medida_c
        boxes["Pyrobubbles Gesamt in l"] = ((boxes["L"] * boxes["B"] * boxes["H"])
        #boxes["Anzahl 100l Säcke Vermiculite"] = boxes["Pyrobubbles Gesamt in l"]
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Pyrobubbles Gesamt in l"] / 50
        boxes["Anzahl 50l Pyrobubbles"] = boxes["Anzahl 50l Pyrobubbles"].apply(
        #boxes["Batteries per box"] = num_batteries # Type C = box per batte
        #boxes.drop('Vermiculite Ausschöpfung', axis=1, inplace=True) # keep
```

```
#boxes.drop('Vermiculite Ausschüttung', axis=1, inplace=True) # keep
#boxes.drop('Anzahl 100L Säcke Vermiculite', axis=1, inplace=True)
```

```
else:
```

```
    print("Check the Pyrobubbles code & Anzahl_100l VS Anzahl 100l")
```

In [9]: boxes

Out[9]:

	Hersteller	Bezeichnung	L	B	H	Zuladung	Zuladung_KG_je_Behälter	Leergewicht
0	Holzkiste	X114	770	570	540	114	73.69	40.0
1	Holzkiste	Y390	1170	770	470	390	92.33	60.0
4	Genius	Gen LP1	2699	2000	570	2024	357.67	764.0
5	Gelkoh	XL+	2505	1665	430	950	229.33	1370.0
6	Europlast	Paloxe 400	1115	715	245	300	69.52	26.0
7	Europlast	Paloxe 760	1120	920	600	400	111.81	44.0
8	Zarges	AkkuSafe	430	540	130	51	53.01	11.5
9	Zarges	groß	550	350	310	70	55.96	5.0
10	Zarges	klein	350	250	310	58	52.70	3.1
11	Nefab	Nefab	2354	1754	531	1850	269.23	203.0
12	EU Karton	GG-Karton 420	430	310	300	40	53.99	40.0
13	BB Verp	GG-Karton 570	570	370	430	60	59.06	60.0
14	Karton	GG-Karton 770	770	570	550	150	74.13	150.0

In [12]: # ahora hay que poner una version para las columnas con _ o sin y todos los tipos

```
boxes.loc[boxes['Anzahl 1001 Säcke Vermiculite'] != 'Pyrobubbles', 'Anzahl 501 P
boxes.loc[boxes['Anzahl 1001 Säcke Vermiculite'] != 'Pyrobubbles', 'Pyrobubbles
boxes
```

Out[12]:

	Hersteller	Bezeichnung	L	B	H	Zuladung	Zuladung_KG_je_Behälter	Leergewicht
0	Holzkiste	X114	770	570	540	114	73.69	40.0
1	Holzkiste	Y390	1170	770	470	390	92.33	60.0
4	Genius	Gen LP1	2699	2000	570	2024	357.67	764.0
5	Gelkoh	XL+	2505	1665	430	950	229.33	1370.0
6	Europlast	Paloxe 400	1115	715	245	300	69.52	26.0
7	Europlast	Paloxe 760	1120	920	600	400	111.81	44.0
8	Zarges	AkkuSafe	430	540	130	51	53.01	11.5
9	Zarges	groß	550	350	310	70	55.96	5.0
10	Zarges	klein	350	250	310	58	52.70	3.1
11	Nefab	Nefab	2354	1754	531	1850	269.23	203.0
12	EU Karton	GG-Karton 420	430	310	300	40	53.99	40.0
13	BB Verp	GG-Karton 570	570	370	430	60	59.06	60.0
14	Karton	GG-Karton 770	770	570	550	150	74.13	150.0

In []: