El objetivo de mi código python es lograr calcular la mejor manera de armar un cajón grande relleno de cajas de diferentes tamaños con diferentes productos, todo esto a raíz de un archivo que indica los productos de ese packing list. Es una cuestión de cálculo espacial optimizado y automático. Quiero que el código tenga únicamente como dato de ingreso por el usuario el packing list (un archivo excel en este caso). El código buscará en el archivo "DIMENSIONES CAJAS-NORMALIZADO.xlxs" toda la información necesaria sobre tamaños de cajas, cajones y cual usar con cada producto. Además, hay un archivo “PESO\_P.T.xlxs” el cual contiene dos columnas: “CODIGO” y “PESO(kg)”, este archivo contiene el peso de los productos el cuál será un factor clave, junto al volumen de cada caja, para organizar de manera óptima el acomodo de cajas en las dos variantes de cajones grandes que hay. Las tablas de los archivos excel son las siguientes:

En el archivo **"DIMENSIONES CAJAS-NORMALIZADO.xlxs"** hay 3 hojas y cada hoja contiene una tabla:

**HOJAS:**

* "TAMAÑO-CAJAS": contiene una tabla con las medidas externas de las cajas usadas para embalar los productos
* "CAJA-PRODUCTO": contiene una tabla relaciona el modelo de caja asociada a un producto
* "TAMAÑO-CAJONES": contiene una pequeña tabla con los cajones grandes disponibles en los cuales se meten las cajas individuales

**Columnas de la tabla "CAJAS" en la hoja "TAMAÑO DE CAJAS":**

* "CODIGO": código ID de la caja
* "DESCRIPCION": descripción de la caja
* "VOLUMEN\_m³": volumen en metros cúbicos que ocupa la caja
* "LARGO\_(mm)": medida de largo externo de la caja
* "ALTO\_(mm)": medida de alto externo de la caja
* "ANCHO\_(mm)": medida de ancho externo de la caja

**Columnas de la tabla "RELACIONES" en la hoja "CAJA-PRODUCTO":**

* "CODIGO": código ID del producto
* "DESCRIPCION": descripción del producto
* "CAJA\_OP\_1": descripción de la opción de caja 1, debe coincidir con un valor único de la columna "DESCRIPCION" de la tabla "CAJAS"
* "CANTIDAD x CAJA\_OP\_1": Cantidad de unidades del producto que entran en la opción de caja 1
* "CAJA\_OP\_2": descripción de la opción de caja 2, debe coincidir con un valor único de la columna "DESCRIPCION" de la tabla "CAJAS"
* "CANTIDAD\_x\_CAJA\_OP\_2": Cantidad de unidades del producto que entran en la opción de caja 2
* "CAJA\_OP\_3": descripción de la opción de caja 3, debe coincidir con un valor único de la columna "DESCRIPCION" de la tabla "CAJAS"
* "CANTIDAD\_x\_CAJA\_OP\_3": Cantidad de unidades del producto que entran en la opción de caja 3

**Columnas de la tabla "CAJONES" en la hoja "TAMAÑO CAJONES":**

* "CODIGO": código ID del cajón
* "DESCRIPCION": descripción del cajón
* "VOLUMEN\_m³": volumen en metros cúbicos que ocupa la caja
* "LARGO\_(mm)": medida de largo interno del cajón
* "ALTO\_(mm)": medida de alto interno del cajón
* "ANCHO\_(mm)": medida de ancho interno del cajón

------------------------------------------------------------

En el archivo **"PACKING LIST.xlsx"** hay una única hoja llamada "P.L" con una tabla

**Columnas de la tabla "P.List" en la hoja "P.L.":**

* "CODIGO": código del producto
* "DESCRIPCIÓN": descripción del producto
* "CANTIDAD": cantidad de unidades del producto

Los cajones mencionados en la tabla "TAMAÑO CAJONES" son los contenedores grandes donde irán acomodadas las cajas más pequeñas con los productos del packing list (PACKING LIST.xlsx).  
  
-----------------------------------------------------------  
  
En el archivo **"** **PESO\_P.T.xlsx"** hay una única hoja llamada "PRODUCTO-PESO" con una tabla:  
  
**Columnas de la tabla "TABLA.L" en la hoja "** **PRODUCTO-PESO ":**

* "CODIGO": código del producto
* "PESO(kg)": peso en kilogramos del producto

**LÓGICA PRINCIPAL**

1.) Crear un dataframe llamado **“ASIGNACION\_CAJAS”** y que contenga las siguientes columnas:

* “ITEM”: sería la numeración identificadora de cajas armadas para empaquetar en el contenedor, comienza por 1. Esto corresponde los objetos “ítems” en la biblioteca “py3dbp”.
* “CAJA”: descripción del modelo de caja asignada, este valor corresponde a alguno de la columna “DESCRIPCION” de la tabla “CAJAS”
* “CODIGO”: código del producto que hay en dicha caja, este valor lo debe obtener de la columna “CODIGO” de la tabla “P.List”
* “DESCRIPCION”: descripción del producto, este valor lo debe obtener de la columna “DESCRIPCION” la tabla “P.List”
* “CANTIDAD”: cantidad de unidades del producto que hay en la caja
* “PESO”: indicar el peso del paquete (se desprecia el peso del cartón) realizando el cálculo siguiente: “CANTIDAD” \*“PESO”. Donde “CANTIDAD” se refiere al valor del registro en cuestión dentro del dataframe y “PESO” lo debe obtener del archivo “PESO\_P.T.csv”
* “VOLUMNEN”: volumen del modelo de caja seleccionado. Este dato se obtiene de la columna “VOLUMNEN\_m3” en la tabla “CAJAS”

2.) Utilizando la biblioteca “py3dbp”, determinar cuál/cuales modelos de caja utilizará cada producto del packing list, teniendo en cuenta las cantidades que indica el mismo para seleccionar las opciones de caja/s óptima ocupando el menor volumen posible (volumen de cajas). Por ejemplo: si tengo 60 unidades de un producto X que utiliza la caja modelo 9 (opción 1) y en ella entran 50 unidades, pero también tengo disponible (en mi tabla "RELACIONES") una caja modelo 7 (opción 2) cuya capacidad para el producto X es 20, las 10 unidades que me faltan de empaquetar las meto en esta caja modelo 7 para optimizar espacio, no vuelvo a usar otra caja modelo 9 ya que es más grande y solo meteré 10 unidades. Cada producto en la tabla "RELACIONES" tiene como mínimo 1 opción de caja disponible y como máximo 3 opciones de caja disponible.

3.) Calcular el peso de cada caja para tener parámetros de ordenamiento en el cajón y no poner cajas muy pesadas encima de cajas livianas, cada producto tiene un peso asociado en el archivo "PESO\_P.T.csv". Si hay un producto en el packing list que no tiene un peso asociado, es decir no existe un registro en el archivo “PESO\_P.T.csv” con el código de producto en cuestión, entonces el código debe solicitar al usuario que ingrese el peso del producto manualmente.

4.) Una vez asignada la caja y obtenido el peso de cada producto que figura en el packing list, el código debe completar el dataframe con la información de todos los paquetes que se armaron para colocar en el/los contenedores.

5.) El código debe imprimir el dataframe completado que se comenzó a elaborar en el paso 1), sería una tabla que tendrá como cantidad de registros la cantidad de cajas armadas para empaquetar todos los productos del packing list, es decir que si se armaron 9 cajas habrá 9 registros en este dataframe.

6.) Ahora sí, con el dataframe completo el código debe acomodar las cajas (utilizando py3dbp) según los cajones disponibles en la tabla “CAJONES” de manera que respete la optimización de volumen y acate la siguiente lógica de peso:  
  
PESO DE CAJA > 7,5 kg : Estas cajas deben ser apiladas primero en el contenedor y siempre las más pesadas primero en orden descendente de peso a medida que se van apilando. Es decir que las cajas más pesadas van a ir abajo y arriba las   
  
Cabe resaltar nuevamente que estas dos variables son las únicas que importan a la hora de la optimización del armado del cajón: el peso y el volumen. Ambas variables están asociadas a cada caja y cada cajón en las tablas “CAJAS” y “CAJONES” respectivamente.

7.) Una vez que el código tenga los datos de cómo se armará el cajón, me gustaría poder representar esto en un dibujo 3D para que el operador sepa precisamente como armar el cajón de esta manera. Utiliza cualquier biblioteca que creas necesaria para la representación 3D como plotly o entre otras.