**import odoorpc**: Importa el módulo odoorpc, que es una librería de Python utilizada para interactuar con servidores Odoo a través de su API.

**import sys**: Importa el módulo sys, que proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete y a funciones que interactúan con el intérprete de Python.

**import re**: Importa el módulo re, que proporciona operaciones de coincidencia de expresiones regulares.

**HOST, PORT, DB, USER, PASSWORD**: Establecen los datos de conexión a la instancia de Odoo.

**odoo = odoorpc.ODOO(HOST, port=PORT):** Crea una instancia de conexión a la instancia de Odoo.

**odoo.login(DB, USER, PASSWORD)**: Inicia sesión en Odoo utilizando los datos de conexión proporcionados.

**sanitize\_product\_name(name)**: Una función que toma un nombre de producto y lo "limpia", eliminando caracteres especiales que podrían causar problemas en la generación del archivo de ventas.

sanitize\_product\_name(name): Esta es una función que toma un nombre de producto como entrada y devuelve una versión "limpia" de ese nombre. Utiliza una expresión regular para eliminar cualquier carácter que no sea alfanumérico, espacio, la letra "ñ" o letras con tilde. Esto se hace para asegurar que el nombre del producto no contenga caracteres especiales que puedan causar problemas en la generación del archivo de ventas.-->me daba error y por eso lo puse-->LO UNICO QUE VI DE INTERNET

**generar\_archivo\_ventas que toma dos argumentos, year (año) y month (mes), y realiza varias operaciones para generar un archivo de ventas para el mes y año especificados.**

**next\_month = month + 1 if month < 12 else 1:** Esta línea maneja el siguiente mes. Si month es menor que 12, next\_month será el mes siguiente. Si month es igual a 12 (diciembre), next\_month será 1 (enero del próximo año).

**next\_year = year + 1 if month == 12 else year:** Esta línea maneja el siguiente año. Si month es igual a 12 (diciembre), next\_year será el próximo año. De lo contrario, next\_year será igual al año actual.

El uso de next\_month y next\_year en lugar de month y year directamente se debe a que se está filtrando registros de ventas dentro de un rango de fechas específico. Para filtrar correctamente las ventas para un mes dado, es necesario tener en cuenta el siguiente mes y, en algunos casos, el siguiente año.

Por ejemplo, consideremos el caso en el que se desea filtrar las ventas para diciembre de un año determinado. Si simplemente se utiliza month como el mes en la condición de filtro, no se obtendrían las ventas de diciembre porque la condición de filtro se aplicaría para el mes de diciembre solamente. Sin embargo, lo que se necesita es obtener las ventas desde el 1 de diciembre hasta el 1 de enero del siguiente año.

Por lo tanto, se utiliza next\_month para representar el mes siguiente al que se está filtrando. Si el mes actual es diciembre (mes 12), entonces next\_month sería enero (mes 1) del siguiente año. De esta manera, se garantiza que las ventas incluyan el mes actual completo y no se pierdan ventas debido a la fecha límite.

**sales = odoo.env['sale.order'].search\_read([...], ['order\_line'])**: Esta línea busca las órdenes de venta (sales orders) para el año y mes especificados utilizando la conexión odoo establecida previamente. La función search\_read busca registros que cumplan ciertos criterios (en este caso, la fecha de la orden de venta) y devuelve los campos especificados ('order\_line') para esos registros.

**line = odoo.env['sale.order.line'].browse(line).read(['product\_id', 'product\_uom\_qty'])[0]**

Realiza lo siguiente:

odoo.env['sale.order.line']: Accede al modelo de Odoo llamado 'sale.order.line', que representa las líneas de pedidos de venta en Odoo.

.browse(line): Utiliza el método browse() para buscar una línea de pedido de venta específica en base a su ID (line). Este método devuelve un objeto que representa la línea de pedido de venta encontrada.

.read(['product\_id', 'product\_uom\_qty']): Utiliza el método read() para leer los valores de los campos especificados para la línea de pedido de venta encontrada. En este caso, se están leyendo los valores de los campos 'product\_id' (ID del producto) y 'product\_uom\_qty' (cantidad de unidad de medida del producto). Esto devuelve una lista de diccionarios que contienen los valores de los campos especificados para la línea de pedido de venta.

[0]: Indexa la lista de diccionarios devuelta por .read() para obtener el primer diccionario de la lista. En este caso, como solo se espera una línea de pedido de venta, se toma el primer elemento de la lista.

**if isinstance(product\_ids, list):**

**for product\_id in product\_ids:**

**if product\_id and isinstance(product\_id, int):**

**try:**

**product\_sales[product\_id] = product\_sales.get(product\_id, 0) + product\_qty**

**except Exception as e:**

**print(f"Error al procesar producto con ID {product\_id}: {e}")**

**print(f"Contenido de la variable 'line': {line}")**

if isinstance(product\_ids, list):: Verifica si el valor de product\_ids es una lista. Esto es importante porque en algunos casos, el campo product\_ids puede contener múltiples valores, mientras que en otros casos solo contendrá un único valor.

if product\_id and isinstance(product\_id, int):: Verifica si product\_id existe (es decir, si no es None) y si es una instancia de int

product\_sales[product\_id] = product\_sales.get(product\_id, 0) + product\_qty: Agrega la cantidad vendida (product\_qty) del producto (product\_id) al diccionario product\_sales. Si el producto ya existe en el diccionario product\_sales, se suma la cantidad vendida anterior a la nueva cantidad vendida. Si el producto aún no está en el diccionario, se agrega con la cantidad vendida como su valor inicial.

# Escribir los resultados en el archivo

file\_name = f'{year}{mes\_nombre(month)}.lst'

print(f"Escribiendo resultados en {file\_name}")

try:

with open(file\_name, 'w', encoding='utf-8') as f: # Utilizando UTF-8

for product\_id, qty\_sold in sorted\_products:

product = odoo.env['product.product'].browse(product\_id)

if product:

product\_name = sanitize\_product\_name(product.name)

f.write(f'{str(product\_name)}: {qty\_sold}\n')

print("Archivo generado exitosamente.")

except Exception as e:

print(f"Error al escribir en el archivo {file\_name}: {e}")

raise # Levantar la excepción para mostrarla completamente

else:

print(f"No hay ventas encontradas para {mes\_nombre(month)}")

file\_name = f'{year}{mes\_nombre(month)}.lst': Construye el nombre del archivo utilizando el año (year), el nombre del mes obtenido mediante la función mes\_nombre(month), y añadiendo la extensión ".lst". Por ejemplo, si year es 2023 y month es 1 (enero), el nombre del archivo será "2023enero.lst".

print(f"Escribiendo resultados en {file\_name}"): Imprime un mensaje indicando que se están escribiendo los resultados en el archivo con el nombre file\_name.

with open(file\_name, 'w', encoding='utf-8') as f:: Abre el archivo file\_name en modo de escritura ('w') utilizando UTF-8 como codificación de caracteres

for product\_id, qty\_sold in sorted\_products:: Itera sobre la lista sorted\_products, que contiene tuplas de pares (product\_id, qty\_sold) donde product\_id es el ID del producto y qty\_sold es la cantidad vendida de ese producto.

**product = odoo.env['product.product'].browse(product\_id): Utiliza el ID del producto para buscar y obtener el objeto del producto correspondiente desde la base de datos de Odoo.IMP**

**if product:: Verifica si se pudo encontrar el objeto del producto. Si el producto existe en la base de datos de Odoo, procede a escribir en el archivo.IMP**

product\_name = sanitize\_product\_name(product.name): Limpia el nombre del producto utilizando la función sanitize\_product\_name(). Esto asegura que el nombre del producto esté en un formato adecuado para ser escrito en el archivo.

f.write(f'{str(product\_name)}: {qty\_sold}\n'): Escribe una línea en el archivo que contiene el nombre del producto y la cantidad vendida, separados por : y seguidos por un salto de línea (\n).