????????APP

- CyberBiz
- MOMO
- ??
- ????
- ???
- ??

CyberBiz

? ??

????? **???? Excel ????????**??? SKU ????????????????????????
??? Python ?????????? Excel ?????????????

? ????

1. **??????**
   - ????????????????????????
2. **SKU ???????**
   - ?? `sku_cb.json` ?????????? SKU ?????????
3. **??????**
   - ????????C2C?????????????C2C??????
4. **???????**
   - ????????????????
   - ?????? × ????????
   - ??????????? = ???? - ??????
5. **????**
   - ?? `plist.json` ? SKU ????????????????
   - ????????????????????????????????
6. **????**
   - ????????????????????? `(□)` ?
   - ?????????????????????
7. **?? Excel ??**
   - ????????????????

```
□□□□_Sku□□_20250801_143000.xls
```

? ????

- **????**?
  - ???? Excel?.xls / .xlsx?
- **SKU ???**?
  - `\\nas-lianruey\office\sku\sku_cb.json`
- **????**?
  - `\\nas-lianruey\office\sku\plist.json`

? ????

??????????????????

- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ????
- ???
- ???

?? ????

1. ?????
2. ???????????? 1 ???
3. ?????? Excel ?????
4. ?????? SKU ??????????????????????
5. ??????????????????

? ????

- ??? SKU ????????????????????????
- ????????????????????????????

?? ?????

```
import os
import json
import re
import pyexcel as pe
import requests
from tkinter import Tk, filedialog, simpledialog
from datetime import datetime, timedelta
from collections import defaultdict


root = Tk()
```

```
root.withdraw()

days_to_add = simpledialog.askinteger("預交貨日設定", "請輸入要加幾天（預設為1）", initialvalue=1)
if not days_to_add:
    days_to_add = 1

file_path = filedialog.askopenfilename(
    title="選擇 Excel 檔案",
    filetypes=[("Excel 檔案", "*.xls *.xlsx")]
)
if not file_path:
    print("□ □□□□□□□□□□□□")
    exit()

sku_json_path = r"\\nas-lianruey\office\sku\sku_cb.json"
if not os.path.isfile(sku_json_path):
    print(f"□ □□□ SKU 對照表檔案: {sku_json_path}")
    exit()

with open(sku_json_path, "r", encoding="utf-8") as f:
    sku_map = json.load(f)

with open(file_path, "rb") as f:
    file_content = f.read()

file_ext = os.path.splitext(file_path)[1].lower()
file_type = "xlsx" if file_ext == ".xlsx" else "xls"
records = pe.get_array(file_type=file_type, file_content=file_content)

next_date = (datetime.today() + timedelta(days=days_to_add)).strftime("%Y%m%d")

header = records[0]
try:
    order_index = header.index("訂單編號")
    status_index = header.index("訂單狀態")
    item_index = header.index("SKU")
    qty_index = header.index("數量")
    price_index = header.index("商品售價")
    address_index = header.index("收件人地址")
    name_index = header.index("收件人名稱")
```

```
    if "轉單編號" not in header:
        header.append("轉單編號")
        for i in range(1, len(records)):
            records[i].append(str(records[i][order_index]))
    transfer_index = header.index("轉單編號")
except ValueError as e:
    print("□ □□□□□□□□", e)
    exit()

# □□替換付款方式
if "付款方式" in header:
    payment_index = header.index("付款方式")
    for i in range(1, len(records)):
        if str(records[i][payment_index]).strip() == "全家冷凍C2C貨到付款":
            records[i][payment_index] = "全冷C2C貨到付款"

total_amount_col_index = -1
try:
    total_amount_col_index = header.index("總額")
    print("□ □□□□□□□□□□□□")
except ValueError:
    print("□◌ 未找到原始「總額」欄位，將根據「數量」和「商品售價」計算。")

order_total_map = {}
if total_amount_col_index != -1:
    for row in records[1:]:
        order_no = str(row[order_index])
        try:
            total_value = float(row[total_amount_col_index])
            order_total_map[order_no] = total_value
        except (ValueError, TypeError):
            pass

original_total_amount_by_order = defaultdict(float)
unique_order_nos = {str(row[order_index]) for row in records[1:]}

for order_no in unique_order_nos:
    if order_no in order_total_map:
        original_total_amount_by_order[order_no] = order_total_map[order_no]
```

```
    else:
        order_total_calculated = 0
        for row in records[1:]:
            if str(row[order_index]) == order_no:
                try:
                    qty = int(row[qty_index])
                    price = float(row[price_index])
                    order_total_calculated += qty * price
                except (ValueError, TypeError):
                    pass
        original_total_amount_by_order[order_no] = order_total_calculated

total_amount_by_order = defaultdict(float)
for row in records[1:]:
    order_no = str(row[order_index])
    product_id = str(row[item_index])
    try:
        original_qty = int(row[qty_index])
    except:
        original_qty = 0

    if product_id in sku_map:
        for sku in sku_map[product_id]:
            try:
                mapped_qty = int(sku["□□□"])
            except:
                mapped_qty = 0
            final_qty = mapped_qty * original_qty
            try:
                new_price = float(sku["□□□"])
            except ValueError:
                new_price = 0.0
            item_total = final_qty * new_price
            total_amount_by_order[order_no] += item_total
    else:
        try:
            new_price = float(row[price_index])
        except ValueError:
            new_price = 0.0
        item_total = original_qty * new_price
```

```
        total_amount_by_order[order_no] += item_total

plist_url = r'\\nas-lianruey\office\sku\plist.json'



try:
    with open(plist_url, "r", encoding="utf-8") as f:
        temp_data = json.load(f)
except FileNotFoundError:
    print(f"□ □□□□□□□□□□□□{plist_url}")
    exit()
except json.JSONDecodeError as e:
    print(f"□ □□□□□□□□□□□□{e}")
    exit()

sku_temp_map = {}
for item in temp_data:
    sku = item.get("sku")
    temp = item.get("temp", "").strip() or "冷凍"
    if sku:
        sku_temp_map[sku] = temp

fill_value_columns = [
    "收件人名稱", "收件人電話", "收件人地址", "收貨超商代號",
    "收件人國家", "收件人省份", "收件人城市", "收件人區",
    "購買人名稱", "購買人電話", "購買人地址", "時間", "取消時間",
    "訂單編號", "訂單狀態", "訂單來源", "備註", "收款方", "會員名稱",
    "Email", "會員手機號碼", "會員資料備註", "會員標籤", "會員生日", "會員ID"
]

fill_value_index = {}
for col in fill_value_columns:
    if col in header:
        fill_value_index[col] = header.index(col)

first_row_map = {}
for row in records[1:]:
    order_no = str(row[order_index])
    if order_no not in first_row_map:
        first_row_map[order_no] = {}
```

```
        for col, idx in fill_value_index.items():
            first_row_map[order_no][col] = row[idx]

# 確保備註欄位存在
if "備註" not in header:
    header.append("備註")
    for i in range(1, len(records)):
        records[i].append("")
remark_index = header.index("備註")

# 確認額外資訊欄位，不存在就新增空欄
if "額外資訊" not in header:
    header.append("額外資訊")
    for i in range(1, len(records)):
        records[i].append("")
extra_info_index = header.index("額外資訊")

# 新增欄位
header += ["新SKU", "新數量", "新進價", "新預交貨日", "溫層", "分錄備註", "新總金額", "新折扣"]
header.insert(header.index("新折扣") + 1, "新備註")

new_total_amount_index = header.index("新總金額")
new_discount_index = header.index("新折扣")
new_remark_index = header.index("新備註")
temp_col_index = header.index("溫層")
remark_col_index = header.index("分錄備註")

new_records = [header]
temp_by_order = defaultdict(list)

for row in records[1:]:
    row = row + ["" for _ in range(len(header) - len(records[0]))]

    if not row[item_index] or str(row[item_index]).strip() == "":
        row[item_index] = "A"

    order_no = str(row[order_index])
    for col, idx in fill_value_index.items():
        if not row[idx] or str(row[idx]).strip() == "":
            row[idx] = first_row_map[order_no][col]
```

```
    product_id = str(row[item_index])
    try:
        original_qty = int(row[qty_index])
    except:
        original_qty = 0

    transfer_no = row[transfer_index]
    address = str(row[address_index])
    name = str(row[name_index])
    cleaned_addr = re.sub(r"[0-9\s]", "", address)[:3]
    split_remark = cleaned_addr + name

    buyer_name = str(row[fill_value_index.get("購買人名稱", "")] if "購買人名稱" in fill_value_index
else "")
    receiver_name = str(row[name_index])
    current_remark = str(row[remark_index]) if row[remark_index] else ""

    if buyer_name and receiver_name and buyer_name != receiver_name:
        row[remark_index] = current_remark + f"({buyer_name}寄)"
    else:
        row[remark_index] = current_remark

    original_remark = str(row[remark_index]) if row[remark_index] else ""
    extra_info = str(row[extra_info_index]) if row[extra_info_index] else ""
    new_remark = original_remark + extra_info
    new_remark = new_remark.replace("如有送禮需求請幫忙註明送禮", "").strip()

    if product_id in sku_map:
        for sku in sku_map[product_id]:
            try:
                mapped_qty = int(sku["新數量"])
            except:
                mapped_qty = 0
            final_qty = mapped_qty * original_qty
            new_sku = sku["新SKU"]
            try:
                new_price = float(sku["新進價"])
            except ValueError:
                new_price = 0.0
```

```
            temp = sku_temp_map.get(new_sku, "□□")
            temp_by_order[transfer_no].append(temp)
            new_row_data = [new_sku, str(final_qty), new_price, next_date, temp, split_remark,
"", "", new_remark]
            new_row = row[:len(records[0])] + new_row_data
            new_records.append(new_row)
    else:
        new_sku = row[item_index]
        try:
            new_price = float(row[price_index])
        except ValueError:
            new_price = 0.0
        temp = sku_temp_map.get(new_sku, "□□")
        temp_by_order[transfer_no].append(temp)
        new_row_data = [new_sku, str(original_qty), new_price, next_date, temp, split_remark,
"", "", new_remark]
        new_row = row[:len(records[0])] + new_row_data
        new_records.append(new_row)

final_records = [new_records[0]]
seen_transfers = set()

for row in new_records[1:]:
    transfer_no = row[transfer_index]
    order_no = str(row[order_index])
    new_total = total_amount_by_order.get(order_no, 0)
    original_total = original_total_amount_by_order.get(order_no, 0)

    row[new_total_amount_index] = new_total
    row[new_discount_index] = original_total - new_total

    print(f"Order: {order_no}, Original Total (□□): {original_total}, New Total (□□□□):
{new_total})")
    print(f"Calculated New Discount (□□□): {original_total - new_total})")

    temps = temp_by_order[transfer_no]
    row[temp_col_index] = "□□" if all(t == "□□" for t in temps) else "□□"
    if transfer_no in seen_transfers:
        row[remark_col_index] = ""
    else:
```

```
        seen_transfers.add(transfer_no)
    final_records.append(row)

base_dir = os.path.dirname(file_path)
name_part, ext_part = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)

pe.save_as(array=final_records, dest_file_name=new_path)
print(f"□ SKU轉換、預交貨日、溫層、分錄備註、總金額與新折扣加總完成，檔案儲存為：{new_path}")
```

MOMO

? ??

????????? **MOMO ??????? Excel ????**????? SKU ??????????????????????????? Excel ???

? ????

1. **?? Excel ?**
   - ?????? `3841566a` ?? MOMO ????
2. **SKU ????**
   - ?? `sku_momo.json` ????? MOMO ?????+???? ???? SKU?
   - ????????????
3. **????**
   - ?? `plist.json` ??????????????????
   - ????????? ? ????????????
4. **??????**
   - ??????????? + ???????????
   - ??????????????????
5. **??????**
   - ??????????? `MO+□□+□□□` ?
   - ?????? `sku_transfer_log.json` ?????????????
6. **?????**
   - ????????

```
□□□□_SKU□□_YYYYMMDD_HHMMSS.xls
```

? ????

- **????**?
  - MOMO ???? Excel ???.xls?
- **SKU ???**?
  - `\\nas-lianruey\office\sku\sku_momo.json`
- **????**?
  - `\\nas-lianruey\office\sku\plist.json`
- **??????**?
  - `\\nas-lianruey\office\sku\app\sku_transfer_log.json`

? ???????

????????????

- ??+?????????????
- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ?????

?? ????

1. ??????? MOMO ????? Excel ???
2. ?????? ? ???? ? ?????? SKU?
3. ??????????????????
4. ?????????????
5. ???????????????

? ????

- ??? **SKU ??????????????????????????**?
- ????????????????

?? ?????

```
import os
import io
import json
import msoffcrypto
import pyexcel as pe
import requests
import re
from tkinter import Tk, filedialog
from datetime import datetime, timedelta
from collections import defaultdict, OrderedDict

# □□ tkinter □□□
root = Tk()
```

```
root.withdraw()

# □□□□□ Excel □□
file_path = filedialog.askopenfilename(title="□□□□□ XLS □□", filetypes=[("Excel Files",
"*.xls")])
if not file_path:
    print("□ □□□□□□□□□□□□□")
    exit()

# SKU □□□
sku_json_path = r"\\nas-lianruey\office\sku\sku_momo.json"
if not os.path.isfile(sku_json_path):
    print(f"□ □□□ SKU □□□□□□{sku_json_path}")
    exit()

with open(sku_json_path, "r", encoding="utf-8") as f:
    sku_map = json.load(f)

# □□□□
plist_url = r'\\nas-lianruey\office\sku\plist.json'



try:
    with open(plist_url, "r", encoding="utf-8") as f:
        temp_data = json.load(f)
except FileNotFoundError:
    print(f"□ □□□□□□□□□□{plist_url}")
    exit()
except json.JSONDecodeError as e:
    print(f"□ □□□□□□□□□□□{e}")
    exit()

sku_temp_map = {}
for item in temp_data:
    sku = item.get("sku")
    temp = item.get("temp", "").strip() or "□□"
    if sku:
        sku_temp_map[sku] = temp

# □□ Excel
```

```
decrypted = io.BytesIO()
with open(file_path, "rb") as f:
    office_file = msoffcrypto.OfficeFile(f)
    office_file.load_key(password="3841566a")
    office_file.decrypt(decrypted)
decrypted.seek(0)

records = pe.get_array(file_type="xls", file_content=decrypted.read())
next_date = (datetime.today() + timedelta(days=1)).strftime("%Y%m%d")
header = records[0]

# 確保有必要欄位
try:
    order_index = header.index("訂單編號")
    box_index = header.index("併箱編號")

    # 產生轉單編號欄位
    if "轉單編號" not in header:
        header.append("轉單編號")
        for i in range(1, len(records)):
            order_id = str(records[i][order_index])[:14]
            box_id = str(records[i][box_index]).strip()
            mo_id = f"MO{order_id}{box_id}"
            records[i].append(mo_id)
        header = records[0]

    item_index = header.index("品號")
    single_name_index = header.index("單名編號")
    qty_index = header.index("數量")
    address_index = header.index("收件人地址")
    name_index = header.index("收件人姓名")

except ValueError as e:
    print("□ □□□□□□□", e)
    exit()

# 更新欄位
header[item_index] = "品號+單名編號"
header += ["新SKU", "新數量", "新進價", "新預交貨日", "溫層", "分錄備註", "新轉單編號"]
```

```
new_records = [header]
temp_by_order_id = defaultdict(list)
transfer_index = header.index("□□□□")
temp_index = header.index("□□")
remark_index = header.index("□□□□")

# log □□□□
log_file_path = r"\\nas-lianruey\office\sku\app\sku_transfer_log.json"
if os.path.exists(log_file_path):
    with open(log_file_path, "r", encoding="utf-8") as f:
        transfer_log = json.load(f)
else:
    transfer_log = {}

today_key = datetime.today().strftime("%Y%m%d")
today_prefix = f"MO{today_key}"
start_seq = transfer_log.get(today_key, 0) + 1
transfer_seq_counter = start_seq

transfer_id_map = OrderedDict()

for row in records[1:]:
    row = row + [""] * (len(header) - 7 - len(row))

    original_product_id = str(row[item_index]).strip()
    single_name = str(row[single_name_index]).strip()
    combined_product_id = "" if original_product_id == "001" else original_product_id +
single_name
    row[item_index] = combined_product_id

    try:
        original_qty = int(row[qty_index])
    except (ValueError, IndexError):
        original_qty = 1

    mo_id = row[transfer_index]

    if mo_id not in transfer_id_map:
        transfer_id_map[mo_id] = f"{today_prefix}{transfer_seq_counter:03d}"
        transfer_seq_counter += 1
```

```
    if combined_product_id in sku_map:
        for sku in sku_map[combined_product_id]:
            try:
                mapped_qty = int(sku.get("新數量", 1))
            except (ValueError, TypeError):
                mapped_qty = 1
            new_qty = mapped_qty * original_qty
            new_sku = sku.get("新SKU", "")
            temp = sku_temp_map.get(new_sku, "冷凍")
            temp_by_order_id[mo_id].append(temp)

            address = str(row[address_index])
            address_cleaned = re.sub(r"[0-9\s]", "", address)[:3]
            name = str(row[name_index])
            remark = address_cleaned + name

            new_row = row + [new_sku, new_qty, sku.get("新進價", ""), next_date, temp, remark,
transfer_id_map[mo_id]]
            new_records.append(new_row)
    else:
        temp_by_order_id[mo_id].append("冷凍")
        address = str(row[address_index])
        address_cleaned = re.sub(r"[0-9\s]", "", address)[:3]
        name = str(row[name_index])
        remark = address_cleaned + name
        new_records.append(row + ["", "", "", next_date, "冷凍", remark,
transfer_id_map[mo_id]])

# 清除重複備註、統一溫層
seen_transfer_ids = set()
final_records = [new_records[0]]
for row in new_records[1:]:
    mo_id = row[transfer_index]
    final_temp = "常溫" if all(t == "常溫" for t in temp_by_order_id[mo_id]) else "冷凍"
    row[temp_index] = final_temp

    if mo_id in seen_transfer_ids:
        row[remark_index] = ""
    else:
```

```
        seen_transfer_ids.add(mo_id)

    final_records.append(row)

# 儲存最新流水號回 log 檔
transfer_log[today_key] = transfer_seq_counter - 1
with open(log_file_path, "w", encoding="utf-8") as f:
    json.dump(transfer_log, f, ensure_ascii=False, indent=2)

# 儲存輸出 Excel
base_dir = os.path.dirname(file_path)
name_part, _ = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)

pe.save_as(array=final_records, dest_file_name=new_path)
print(f"□ SKU轉換、預交貨日、溫層與分錄備註處理完成，檔案儲存為: {new_path}")
```

??

? ?? ?? SKU Excel ???? ????

? ??

????????? **?? (Shopee) ?????????? Excel?**?
????? **SKU ??????????????????????** ? **????????**?
??????????????????

? ????

1. **?? Excel ??**
   - ?????? `365507` ???????????
2. **SKU ????**
   - ?? `sku_sp.json` ??????????????????? SKU?
   - ????????????
3. **????**
   - ?? `plist.json` ?????
     - ????????? ? ??????
     - ?? ? ????????
4. **??????**
   - ?????????? + ???????
   - ??????????????????????????
5. **???????**
   - ??? `SYYYYMMDD001`
   - ???????????? `sku_transfer_log.json` ??????
6. **????**
   - ????????

```
□□□□_SKU□□_YYYYMMDD_HHMMSS.xls
```

? ????

- **????**
  - ???? Excel ???.xls ? .xlsx?
- **SKU ???**
  - `\\nas-lianruey\office\sku\sku_sp.json`
- **????**
  - `\\nas-lianruey\office\sku\plist.json`

- **?????**
  - `\\nas-lianruey\office\sku\app\sku_transfer_log.json`

? ????

??????????

- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ?????

?? ????

1. ??????????? Excel ???
2. ???????????
3. ? SKU ??????????????????????
4. ??????????????
5. ???????????

? ????

- ??? **SKU ??????????????????** ? **????????**?
- ??????????????????

?? ?????

```
import os
import io
import json
import msoffcrypto
import pyexcel as pe
import requests
import re
from tkinter import Tk, filedialog
```

```
from datetime import datetime, timedelta
from collections import defaultdict, OrderedDict

# □□ tkinter □□□
root = Tk()
root.withdraw()

# □□□□ Excel
file_path = filedialog.askopenfilename(
    title="□□□□□ Excel □□",
    filetypes=[("Excel Files", "*.xls *.xlsx")]
)
if not file_path:
    print("□ □□□□□□□□□□□□□")
    exit()

# SKU □□□
sku_json_path = r"\\nas-lianruey\office\sku\sku_sp.json"
if not os.path.isfile(sku_json_path):
    print(f"□ □□□ SKU □□□□□□{sku_json_path}")
    exit()

with open(sku_json_path, "r", encoding="utf-8") as f:
    sku_map = json.load(f)

# □□ SKU □□□□□□
plist_url = r'\\nas-lianruey\office\sku\plist.json'



try:
    with open(plist_url, "r", encoding="utf-8") as f:
        temp_data = json.load(f)
except FileNotFoundError:
    print(f"□ □□□□□□□□□□{plist_url}")
    exit()
except json.JSONDecodeError as e:
    print(f"□ □□□□□□□□□□□{e}")
    exit()

sku_temp_map = {}
```

```
for item in temp_data:
    sku = item.get('sku')
    temp = item.get('temp', '').strip() or '冷凍'
    if sku:
        sku_temp_map[sku] = temp

# 解密 Excel
decrypted = io.BytesIO()
with open(file_path, "rb") as f:
    office_file = msoffcrypto.OfficeFile(f)
    office_file.load_key(password="365507")
    office_file.decrypt(decrypted)
decrypted.seek(0)

# 讀取為表格陣列
file_ext = os.path.splitext(file_path)[1].lower()
file_type = "xlsx" if file_ext == ".xlsx" else "xls"
records = pe.get_array(file_type=file_type, file_content=decrypted.read())

# 日期加一天
next_date = (datetime.today() + timedelta(days=1)).strftime("%Y%m%d")

# 新轉單編號 log 讀取
log_path = r"\\nas-lianruey\office\sku\app\sku_transfer_log.json"
if os.path.exists(log_path):
    with open(log_path, "r", encoding="utf-8") as f:
        transfer_log = json.load(f)
else:
    transfer_log = {}

today_key = datetime.today().strftime("%Y%m%d")
prefix = f"S{today_key}"
start_seq = transfer_log.get(today_key, 0) + 1
seq_counter = start_seq
transfer_id_map = OrderedDict()

# 找出欄位
header = records[0]
try:
    order_index = header.index("訂單編號")
```

```
    if "轉單編號" not in header:
        header.append("轉單編號")
        for i in range(1, len(records)):
            order_id = str(records[i][order_index])
            mo_id = "S" + order_id[:14] if len(order_id) >= 14 else "S" + order_id
            records[i].append(mo_id)
        header = records[0]
    transfer_index = header.index("轉單編號")
    item_index = header.index("商品選項貨號")
    qty_index = header.index("數量")
    address_index = header.index("收件地址")
    name_index = header.index("收件者姓名")
except ValueError as e:
    print("□ □□□□□□□□", e)
    exit()

# 加欄位
header += ["新SKU", "新數量", "新進價", "新預交貨日", "溫層", "分錄備註", "新轉單編號"]

# 展開資料列
new_records = [header]
temp_by_order_id = defaultdict(list)

for row in records[1:]:
    row = row + [""] * (len(header) - 7 - len(row))  # 填滿欄位
    product_id = str(row[item_index])
    mo_id = row[transfer_index]

    try:
        original_qty = int(row[qty_index])
    except (ValueError, TypeError):
        original_qty = 1

    # 建立分錄備註
    addr = str(row[address_index])
    addr_cleaned = re.sub(r"[0-9\s]", "", addr)[:3]
    name = str(row[name_index])
    remark = addr_cleaned + name

    # 建立新轉單編號
```

```
    if mo_id not in transfer_id_map:
        transfer_id_map[mo_id] = f"{prefix}{seq_counter:03d}"
        seq_counter += 1
    new_transfer_id = transfer_id_map[mo_id]

    if product_id in sku_map:
        for sku in sku_map[product_id]:
            try:
                new_qty = int(sku["新數量"]) * original_qty
            except:
                new_qty = ""
            new_sku = sku["新SKU"]
            temp = sku_temp_map.get(new_sku, "冷凍")
            temp_by_order_id[mo_id].append(temp)
            new_records.append(row + [new_sku, new_qty, sku["新進價"], next_date, temp, remark,
new_transfer_id])
    else:
        temp_by_order_id[mo_id].append("冷凍")
        new_records.append(row + ["", "", "", next_date, "冷凍", remark, new_transfer_id])

# 根據同轉單判斷是否全為常溫，且分錄備註只保留第一筆
final_records = [new_records[0]]
temp_index = header.index("溫層")
remark_index = header.index("分錄備註")
seen_transfer_ids = set()

for row in new_records[1:]:
    mo_id = row[transfer_index]
    row[temp_index] = "常溫" if all(t == "常溫" for t in temp_by_order_id[mo_id]) else "冷凍"

    if mo_id in seen_transfer_ids:
        row[remark_index] = ""
    else:
        seen_transfer_ids.add(mo_id)

    final_records.append(row)

# 更新 log 檔案
transfer_log[today_key] = seq_counter - 1
with open(log_path, "w", encoding="utf-8") as f:
```

```
    json.dump(transfer_log, f, ensure_ascii=False, indent=2)

# 輸出
base_dir = os.path.dirname(file_path)
name_part, ext_part = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)

pe.save_as(array=final_records, dest_file_name=new_path)
print(f"□ SKU轉換、預交貨日與新轉單編號處理完成，檔案儲存為: {new_path}")
```

????

? ??

????? **?? Excel ? CSV ??** ????????? **SKU ??????????????** ? **???????**????????

? ????

1. **??????**
   - ???????????????? 2????????????
2. **SKU ????**
   - ???SKU / ??? / ??????????????????????
3. **????**
   - ???????? `https://ec.zfun.com.tw/plist.json` ?? SKU ???
   - ???????????????? ????????????
4. **?????**
   - ??? `HYYYYMMDD001` ???????
   - ?????? `\\nas-lianruey\office\sku\app\sku_transfer_log.json` ??????
5. **??????**
   - ???????? + ??????
   - ????????????????????
6. **????**
   - ????????????????

```
□□□□_SKU□□_20250801_143000.xls
```

? ????

- **????**
  - Excel ? CSV ??????????????
- **??????**
  - `https://ec.zfun.com.tw/plist.json` ???? `wu` / ??? `wu2266228` ?
- **???????**
  - `\\nas-lianruey\office\sku\app\sku_transfer_log.json`

? ????

??????????????????

- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ?????

?? ????

1. ?????
2. ???????????
3. ?? Excel ? CSV ?????
4. ?????? SKU?????????????
5. ???????????

? ????

- ??? **SKU ??????????????** ? **????????**?
- ??????????????????

?? ?????

```
import os
import json
import pyexcel as pe
import requests
import re
from tkinter import Tk, filedialog, simpledialog
from datetime import datetime, timedelta
from collections import defaultdict, OrderedDict

# 關閉 tkinter 主視窗
root = Tk()
root.withdraw()

# 詢問加幾天，預設為2
days_to_add = simpledialog.askinteger("預交貨日設定", "請輸入要加幾天（預設為2）", initialvalue=2)
```

```
if not days_to_add:
    days_to_add = 2

# 選取檔案
file_path = filedialog.askopenfilename(
    title="選擇 Excel 或 CSV 檔案",
    filetypes=[("Excel/CSV Files", "*.xls *.xlsx *.csv")]
)
if not file_path:
    print("□ □□□□□□□□□□□□")
    exit()

# 讀取 SKU 溫層資料
plist_url = 'https://ec.zfun.com.tw/plist.json'
username = 'wu'
password = 'wu2266228'
try:
    response = requests.get(plist_url, auth=(username, password))
    response.raise_for_status()
    temp_data = response.json()
except requests.exceptions.RequestException as e:
    print("□ □□□□□□□□□", e)
    exit()

# 建立 SKU 對應溫層字典
sku_temp_map = {}
for item in temp_data:
    sku = str(item.get('sku', '')).strip()
    temp = item.get('temp', '').strip() or '冷凍'
    if sku:
        sku_temp_map[sku] = temp

# 根據副檔名讀取資料
ext = os.path.splitext(file_path)[1].lower()
if ext == ".csv":
    records = pe.get_array(file_name=file_path, encoding="utf-8-sig")
else:
    records = pe.get_array(file_name=file_path)

if not records or len(records) < 2:
```

```
    print("□ □□□□□□□□")
    exit()

# 預交貨日加天數
next_date = (datetime.today() + timedelta(days=days_to_add)).strftime("%Y%m%d")

# 找出必要欄位
header = records[0]
def idx(col):
    try:
        return header.index(col)
    except Exception:
        print(f"□ □□□□□□□{col}")
        exit()

order_index      = idx("訂單單號")
item_index       = idx("產品名稱")
qty_index        = idx("訂貨數量")
address_index    = idx("收貨地址")
name_index       = idx("收貨人姓名")
phone_index      = idx("收貨人電話")
vendor_sku_index = idx("廠商貨號")
unit_price_index = idx("單價(元)")
# 下方三個欄位若原本沒有也OK, 會補上
new_sku_index    = header.index("新SKU")     if "新SKU"     in header else -1
new_qty_index    = header.index("新數量")    if "新數量"    in header else -1
new_price_index  = header.index("新進價")    if "新進價"    in header else -1

# 新增必要欄位
if "新SKU" not in header:
    header.append("新SKU")
    new_sku_index = len(header) - 1
if "新數量" not in header:
    header.append("新數量")
    new_qty_index = len(header) - 1
if "新進價" not in header:
    header.append("新進價")
    new_price_index = len(header) - 1
if "新預交貨日" not in header:
    header.append("新預交貨日")
```

```
if "溫層" not in header:
    header.append("溫層")
if "分錄備註" not in header:
    header.append("分錄備註")
if "新轉單編號" not in header:
    header.append("新轉單編號")

# 準備 LOG
log_path = r"\\nas-lianruey\office\sku\app\sku_transfer_log.json"
today_key = datetime.today().strftime("%Y%m%d")
today_prefix = f"H{today_key}"

if os.path.exists(log_path):
    with open(log_path, "r", encoding="utf-8") as f:
        log_data = json.load(f)
else:
    log_data = {}

seq_start = log_data.get(today_key, 0) + 1
transfer_seq_counter = seq_start
transfer_id_map = OrderedDict()

# 主處理
temp_by_order_id = defaultdict(list)
rows_by_order_id = defaultdict(list)
final_records = [header]

for row in records[1:]:
    row = row + [""] * (len(header) - len(row))  # 補足欄位

    # 基本資料
    vendor_sku = str(row[vendor_sku_index]).strip()
    order_id = str(row[order_index]).strip()
    mo_id = "H" + order_id[:13] if len(order_id) >= 13 else "H" + order_id
    mo_id = mo_id.strip()

    if mo_id not in transfer_id_map:
        transfer_id_map[mo_id] = f"{today_prefix}{transfer_seq_counter:03d}"
        transfer_seq_counter += 1
```

```
# 新SKU、新數量、新進價的自動填入
order_qty = str(row[qty_index]).strip()
unit_price = str(row[unit_price_index]).strip()

new_sku = str(row[new_sku_index]).strip() if new_sku_index >= 0 else ""
if not new_sku:
    new_sku = vendor_sku
    row[new_sku_index] = new_sku

new_qty = str(row[new_qty_index]).strip() if new_qty_index >= 0 else ""
if not new_qty:
    new_qty = order_qty
    row[new_qty_index] = new_qty

new_price = str(row[new_price_index]).strip() if new_price_index >= 0 else ""
if not new_price:
    new_price = unit_price
    row[new_price_index] = new_price

# 電話分拆
phone_raw = str(row[phone_index]).strip()
if "/" in phone_raw:
    new_phone, new_mobile = map(str.strip, phone_raw.split("/", 1))
else:
    new_phone, new_mobile = phone_raw, ""

# 分錄備註
raw_address = str(row[address_index])
cleaned_address = re.sub(r"[0-9\s]", "", raw_address)[:3]
name = str(row[name_index])
remark = cleaned_address + name

# 溫層判斷
temp = sku_temp_map.get(new_sku, "冷凍")
print(f"[廠商貨號比對] 訂單號: {order_id}, 廠商貨號: {vendor_sku}, 新SKU: {new_sku}, 溫層: {temp}")

# 組成新 row
row_idx_map = {k: header.index(k) for k in header}
row_out = row.copy()
row_out[row_idx_map["新SKU"]]     = new_sku
```

```
    row_out[row_idx_map["新數量"]]    = new_qty
    row_out[row_idx_map["新進價"]]    = new_price
    row_out[row_idx_map["新預交貨日"]]= next_date
    row_out[row_idx_map["溫層"]]      = temp
    row_out[row_idx_map["分錄備註"]]  = remark
    row_out[row_idx_map["新轉單編號"]]= transfer_id_map[mo_id]
    # 也可以視情況加入 new_phone、new_mobile 欄位

    temp_by_order_id[mo_id].append(temp)
    rows_by_order_id[mo_id].append(row_out)

# 決定 unified_temp
temp_index = header.index("溫層")
remark_index = header.index("分錄備註")
for mo_id, rows in rows_by_order_id.items():
    print(f"\n[訂單統計] mo_id: {mo_id}, 此單所有SKU溫層: {temp_by_order_id[mo_id]}")
    unified_temp = "常溫" if all(t == "常溫" for t in temp_by_order_id[mo_id]) else "冷凍"
    print(f"[訂單統計] 統一判定此單 unified_temp = {unified_temp}")
    if unified_temp == "冷凍":
        print(f"[警告] 此訂單被判成冷凍, 明細如下: ")
        for t, row in zip(temp_by_order_id[mo_id], rows):
            print(f"    廠商貨號: {row[header.index('新SKU')]}, 溫層: {t}")
    for i, row in enumerate(rows):
        row[temp_index] = unified_temp
        if i > 0:
            row[remark_index] = ""
        final_records.append(row)

# 儲存 LOG
log_data[today_key] = transfer_seq_counter - 1
with open(log_path, "w", encoding="utf-8") as f:
    json.dump(log_data, f, ensure_ascii=False, indent=2)

# 儲存檔案
base_dir = os.path.dirname(file_path)
name_part, ext_part = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)
pe.save_as(array=final_records, dest_file_name=new_path)
```

```
print(f"□ □□□□□□□□□□□□□{new_path}")
```

???

? ??

??????? **??? (i) ????????Excel ? CSV?**????? SKU
??

? ????

1. **??????**
   - ???????????????? 1????????????
2. **SKU ????**
   - ?? `sku_i.json` ???????????????? SKU????????????
3. **????**
   - ?? `plist.json` ???????
     - ??????????? ? ???????
     - ?? ? ???????
   - ???? **P11-000-05** ? ????????
4. **??????**
   - ????????? + ????????????????
   - ??????????????????????
5. **???????**
   - ??? `IYYYYMMDD001`
   - ???????????? `sku_transfer_log.json` ??????
6. **????**
   - ????????

```
□□□□_SKU□□_YYYYMMDD_HHMMSS.xls
```

? ????

- **????**
  - ??? ?? Excel / CSV
- **SKU ???**
  - `\\nas-lianruey\office\sku\sku_i.json`
- **????**
  - `\\nas-lianruey\office\sku\plist.json`
- **?????**
  - `\\nas-lianruey\office\sku\app\sku_transfer_log.json`

? ????

??????????????

- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ?????

?? ????

1. ?????
2. ????????????????????? 1??
3. ??????????Excel ? CSV??
4. ?????? SKU ?????????????????????
5. ???????????????

? ????

- ??? **SKU ????????????????????????**?
- ??????????????

?? ?????

```
import os
import json
import pyexcel as pe
import requests
import re
from tkinter import Tk, filedialog, simpledialog
from datetime import datetime, timedelta
from collections import defaultdict, OrderedDict

# □□ tkinter □□□
root = Tk()
```

```
root.withdraw()

# 詢問加幾天，預設為1
days_to_add = simpledialog.askinteger("預交貨日設定", "請輸入要加幾天（預設為1）", initialvalue=1)
if not days_to_add:
    days_to_add = 1

# 選取檔案（支援 Excel 與 CSV）
file_path = filedialog.askopenfilename(
    title="選擇 Excel 或 CSV 檔案",
    filetypes=[("Excel/CSV Files", "*.xls *.xlsx *.csv")]
)
if not file_path:
    print("□ □□□□□□□□□□□□")
    exit()

# SKU 對照表路徑
sku_json_path = r"\\nas-lianruey\office\sku\sku_i.json"
if not os.path.isfile(sku_json_path):
    print(f"□ □□□ SKU 對照表檔案: {sku_json_path}")
    exit()

with open(sku_json_path, "r", encoding="utf-8") as f:
    sku_map = json.load(f)

# 取得溫層資料
plist_url = r"\\nas-lianruey\office\sku\plist.json"

try:
    with open(plist_url, "r", encoding="utf-8") as f:
        temp_data = json.load(f)
except FileNotFoundError:
    print(f"□ □□□□□□□□□□□{plist_url}")
    exit()
except json.JSONDecodeError as e:
    print(f"□ □□□□□□□□□□□{e}")
    exit()

sku_temp_map = {}
for item in temp_data:
```

```
    sku = item.get("sku")
    temp = item.get("temp", "").strip() or "冷凍"
    if sku:
        sku_temp_map[sku] = temp

# 讀取資料
ext = os.path.splitext(file_path)[1].lower()
if ext == ".csv":
    records = pe.get_array(file_name=file_path, encoding="utf-8-sig")
else:
    records = pe.get_array(file_name=file_path)

# 刪除前3列，只留表頭與資料
if len(records) > 3:
    records = records[3:]

# 預交貨日
next_date = (datetime.today() + timedelta(days=days_to_add)).strftime("%Y%m%d")

# 讀取 log 紀錄: 新轉單編號流水號
log_file_path = r"\\nas-lianruey\office\sku\app\sku_transfer_log.json"
if os.path.isfile(log_file_path):
    with open(log_file_path, "r", encoding="utf-8") as f:
        transfer_log = json.load(f)
else:
    transfer_log = {}

today_key = datetime.today().strftime("%Y%m%d")
prefix = f"I{today_key}"
start_seq = transfer_log.get(today_key, 0) + 1
transfer_seq_counter = start_seq
transfer_id_map = OrderedDict()

# 找出欄位位置
header = records[0]
try:
    order_index = header.index("訂單編號")
    item_index = header.index("原廠商品編號")
    qty_index = header.index("數量")
    price_index = header.index("單價(含稅)")
```

```
    address_index = header.index("收貨地址")
    name_index = header.index("收件人")

    # 若無"轉單編號"欄位，補上
    if "轉單編號" not in header:
        header.append("轉單編號")
        for i in range(1, len(records)):
            order_id = str(records[i][order_index])
            mo_id = "I" + order_id[:7] if len(order_id) >= 7 else "I" + order_id
            records[i].append(mo_id)
        header = records[0]

except ValueError as e:
    print("□ □□□□□□□□", e)
    exit()

# 加入新欄位
header += ["新SKU", "新數量", "新進價", "新預交貨日", "溫層", "分錄備註", "新轉單編號"]

# 處理資料列
new_records = [header]
temp_by_order = defaultdict(list)
transfer_index = header.index("轉單編號")

for row in records[1:]:
    row = row + [""] * (len(header) - 7 - len(row))
    product_id = str(row[item_index])
    try:
        order_qty = int(row[qty_index])
    except (ValueError, TypeError):
        order_qty = 1
    mo_id = row[transfer_index]

    # 建立新轉單編號
    if mo_id not in transfer_id_map:
        transfer_id_map[mo_id] = f"{prefix}{transfer_seq_counter:03d}"
        transfer_seq_counter += 1

    # 分錄備註
    address = str(row[address_index])
```

```
    address_clean = re.sub(r"[0-9\s]", "", address)[:3]
    recipient = str(row[name_index])
    raw_remark = address_clean + recipient
    remark = re.sub(r"[^\u4e00-\u9fffA-Za-z0-9]", "", raw_remark)

    if product_id in sku_map:
        for sku in sku_map[product_id]:
            try:
                new_qty = int(sku["新數量"]) * order_qty
            except (ValueError, TypeError):
                new_qty = sku["新數量"]
            new_sku = sku["新SKU"]
            new_temp = sku_temp_map.get(new_sku, "冷凍")
            if new_sku == "P11-000-05":
                new_temp = "常溫"
            temp_by_order[mo_id].append(new_temp)
            new_row = row + [new_sku, new_qty, sku["新進價"], next_date, new_temp, remark,
transfer_id_map[mo_id]]
            new_records.append(new_row)
    else:
        default_temp = "冷凍"
        temp_by_order[mo_id].append(default_temp)
        new_row = row + [
            product_id,
            order_qty,
            row[price_index] if price_index < len(row) else "",
            next_date,
            default_temp,
            remark,
            transfer_id_map[mo_id]
        ]
        new_records.append(new_row)

# 統一溫層與清除重複備註
final_records = [new_records[0]]
temp_col_index = header.index("溫層")
remark_col_index = header.index("分錄備註")
sku_col_index = header.index("新SKU")
seen_transfer_ids = set()
```

```
for row in new_records[1:]:
    mo_id = row[transfer_index]
    sku = row[sku_col_index]
    if sku == "P11-000-05":
        row[temp_col_index] = "常溫"
    else:
        row[temp_col_index] = "常溫" if all(t == "常溫" for t in temp_by_order[mo_id]) else "冷凍"

    if mo_id in seen_transfer_ids:
        row[remark_col_index] = ""
    else:
        seen_transfer_ids.add(mo_id)

    final_records.append(row)

# 更新 log 檔
transfer_log[today_key] = transfer_seq_counter - 1
with open(log_file_path, "w", encoding="utf-8") as f:
    json.dump(transfer_log, f, ensure_ascii=False, indent=2)

# 儲存檔案
base_dir = os.path.dirname(file_path)
name_part, ext_part = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)

pe.save_as(array=final_records, dest_file_name=new_path)
print(f"□ SKU轉換、預交貨日、溫層與分錄備註、新轉單編號處理完成，檔案儲存為：{new_path}")
```

??

? ??

??????? **?????????????Excel ? CSV?**????? SKU ??

? ????

1. **??????**
   - ????????????? 1?????????????
2. **SKU ????**
   - ?? `sku_ma.json` ????????????????????? SKU?
   - ?????????????
3. **????**
   - ?? `plist.json` ?????
     - ????????????? ? ????
     - ?? ? ?????
4. **????**
   - ????????????? + ???????
   - ?????????? ? ?????????????
5. **???????**
   - ??? `MYYYYMMDD001`
   - ??????????? `sku_transfer_log.json` ??????
6. **????**
   - ?????

```
□□□□_SKU□□_YYYYMMDD_HHMMSS.xls
```

? ????

- **????**
  - ???? Excel ? CSV
- **SKU ???**
  - `\\nas-lianruey\office\sku\sku_ma.json`
- **????**
  - `\\nas-lianruey\office\sku\plist.json`
- **?????**
  - `\\nas-lianruey\office\sku\app\sku_transfer_log.json`

? ????

???????????

- ?SKU
- ???
- ???
- ?????
- ??
- ????
- ?????

?? ????

1. ?????
2. ????????????????????? 1??
3. ???????Excel ? CSV??
4. ?????? SKU ??????????????????????
5. ?????????????

? ????

- ??? **SKU ??**????????**????**????**????**????????**?**
- ??????????????????

?? ?????

```
import os
import json
import pyexcel as pe
import requests
import re
from tkinter import Tk, filedialog, simpledialog
from datetime import datetime, timedelta
from collections import defaultdict, OrderedDict

# □□ tkinter □□□
root = Tk()
root.withdraw()
```

```
# 詢問加幾天，預設為1
days_to_add = simpledialog.askinteger("預交貨日設定", "請輸入要加幾天（預設為1）", initialvalue=1)
if not days_to_add:
    days_to_add = 1

# 選取檔案（支援 Excel 與 CSV）
file_path = filedialog.askopenfilename(
    title="選擇 Excel 或 CSV 檔案",
    filetypes=[("Excel/CSV Files", "*.xls *.xlsx *.csv")]
)
if not file_path:
    print("□ □□□□□□□□□□□□")
    exit()

# SKU 對照表
sku_json_path = r"\\nas-lianruey\office\sku\sku_ma.json"
if not os.path.isfile(sku_json_path):
    print(f"□ □□□ SKU 對照表檔案: {sku_json_path}")
    exit()

with open(sku_json_path, "r", encoding="utf-8") as f:
    sku_map = json.load(f)

# 溫層資料
plist_url = r'\\nas-lianruey\office\sku\plist.json'

try:
    with open(plist_url, "r", encoding="utf-8") as f:
        temp_data = json.load(f)
except FileNotFoundError:
    print(f"□ □□□□□□□□□□{plist_url}")
    exit()
except json.JSONDecodeError as e:
    print(f"□ □□□□□□□□□□□{e}")
    exit()

sku_temp_map = {}
for item in temp_data:
    sku = item.get("sku")
```

```
    temp = item.get("temp", "").strip() or "冷凍"
    if sku:
        sku_temp_map[sku] = temp

# 讀取資料
ext = os.path.splitext(file_path)[1].lower()
if ext == ".csv":
    records = pe.get_array(file_name=file_path, encoding="utf-8-sig")
else:
    records = pe.get_array(file_name=file_path)

next_date = (datetime.today() + timedelta(days=days_to_add)).strftime("%Y%m%d")
header = records[0]

# 欄位定位與轉單欄位建立
try:
    order_index = header.index("訂單編號")
    item_index = header.index("瑪黑家居新版商品編號")
    qty_index = header.index("數量")
    address_index = header.index("收件人地址")
    name_index = header.index("收件人姓名")

    if "轉單編號" not in header:
        header.append("轉單編號")
        for i in range(1, len(records)):
            order_id = str(records[i][order_index])
            mo_id = order_id[:12] if len(order_id) >= 12 else order_id
            records[i].append(mo_id)
        header = records[0]

except ValueError as e:
    print("□ □□□□□□□", e)
    exit()

# 新欄位
header += ["新SKU", "新數量", "新進價", "新預交貨日", "溫層", "分錄備註", "新轉單編號"]

# 新轉單編號 log 處理
log_file_path = r"\\nas-lianruey\office\sku\app\sku_transfer_log.json"
if os.path.exists(log_file_path):
```

```
    with open(log_file_path, "r", encoding="utf-8") as f:
        transfer_log = json.load(f)
else:
    transfer_log = {}

today_key = datetime.today().strftime("%Y%m%d")
today_prefix = f"M{today_key}"
start_seq = transfer_log.get(today_key, 0) + 1
transfer_seq_counter = start_seq
transfer_id_map = OrderedDict()

# □□□□□
new_records = [header]
temp_by_order = defaultdict(list)

for row in records[1:]:
    row = row + [""] * (len(header) - 7 - len(row))
    product_id = str(row[item_index])
    try:
        order_qty = int(row[qty_index])
    except (ValueError, TypeError):
        order_qty = 1

    transfer_id = row[header.index("□□□□")]
    if transfer_id not in transfer_id_map:
        transfer_id_map[transfer_id] = f"{today_prefix}{transfer_seq_counter:03d}"
        transfer_seq_counter += 1

    address = str(row[address_index])
    address_cleaned = re.sub(r"[0-9\s]", "", address)[:3]
    name = str(row[name_index])
    remark = address_cleaned + name

    if product_id in sku_map:
        for sku in sku_map[product_id]:
            try:
                new_qty = int(sku["□□□"]) * order_qty
            except (ValueError, TypeError):
                new_qty = sku["□□□"]
            new_sku = sku["□SKU"]
```

```
            temp_value = sku_temp_map.get(new_sku, "冷凍")
            temp_by_order[transfer_id].append(temp_value)
            new_row = row + [new_sku, new_qty, sku["新進價"], next_date, temp_value, remark, transfer_id_map[transfer_id]]
            new_records.append(new_row)
    else:
        temp_by_order[transfer_id].append("冷凍")
        new_records.append(row + ["", "", "", next_date, "冷凍", remark, transfer_id_map[transfer_id]])

# 清除重複備註、整合溫層
final_records = [new_records[0]]
temp_index = header.index("溫層")
transfer_index = header.index("轉單編號")
remark_index = header.index("分錄備註")
seen_transfer_ids = set()

for row in new_records[1:]:
    transfer_id = row[transfer_index]
    row[temp_index] = "常溫" if all(t == "常溫" for t in temp_by_order[transfer_id]) else "冷凍"
    if transfer_id in seen_transfer_ids:
        row[remark_index] = ""
    else:
        seen_transfer_ids.add(transfer_id)
    final_records.append(row)

# 更新 log
transfer_log[today_key] = transfer_seq_counter - 1
with open(log_file_path, "w", encoding="utf-8") as f:
    json.dump(transfer_log, f, ensure_ascii=False, indent=2)

# 儲存檔案
base_dir = os.path.dirname(file_path)
name_part, ext_part = os.path.splitext(os.path.basename(file_path))
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
new_filename = f"{name_part}_SKU轉換_{timestamp}.xls"
new_path = os.path.join(base_dir, new_filename)

pe.save_as(array=final_records, dest_file_name=new_path)
print(f"□ SKU轉換、預交貨日、溫層與分錄備註、新轉單編號處理完成，檔案儲存為：{new_path}")
```