??????

SKU APP ??? ????

? ??

?????????? **GUI ??**??????????????????????????? SKU ????
????? **Python (Tkinter)**???????????????

? ????

- Windows 10 / 11
- ??? **Python 3.10 ????**
- ???????????tkinter?pyexcel ???????????

? ????

```
import tkinter as tk
from tkinter import messagebox
import subprocess
import os

# 設定腳本所在資料夾（轉換功能存放目錄）
APP_DIR = r"\\nas-lianruey\office\sku\app"

# 設定 SKU 編輯器路徑（注意有空白的檔名）
SKU_EDITOR_PATH = r"\\nas-lianruey\office\sku\sku_editor v1.1.py"

# 轉換功能的腳本
scripts = {
    "轉換 CyberBiz 檔": "cb_newpath.py",
    "轉換 MOMO 檔": "momo_newpath.py",
    "轉換 瑪黑 檔": "ma_newpath.py",
    "轉換 蝦皮 檔": "sp_newpath.py",
    "轉換 好物市集 檔": "good_newpath.py",
    "轉換 愛合購 檔": "i_newpath.py",
```

```
}

def run_script(script_path):
    if os.path.isfile(script_path):
        subprocess.Popen(["python", script_path], shell=True)
    else:
        messagebox.showerror("錯誤", f"找不到檔案: {script_path}")

def run_app_script(script_name):
    full_path = os.path.join(APP_DIR, script_name)
    run_script(full_path)

def run_editor():
    run_script(SKU_EDITOR_PATH)

# GUI 主畫面
root = tk.Tk()
root.title("□□SKU APP 啟動器")

# 轉換功能區域
frame_converter = tk.LabelFrame(root, text="轉換功能", padx=10, pady=10)
frame_converter.pack(padx=10, pady=5, fill="both", expand=True)

for label, filename in scripts.items():
    btn = tk.Button(frame_converter, text=label, width=30, command=lambda f=filename:
run_app_script(f))
    btn.pack(padx=5, pady=3)

# 分隔線
separator = tk.Frame(root, height=2, bd=1, relief="sunken")
separator.pack(fill="x", padx=5, pady=10)

# 編輯 SKU 區域
frame_editor = tk.LabelFrame(root, text="其他管理功能", padx=10, pady=10)
frame_editor.pack(padx=10, pady=5, fill="both", expand=True)

btn_editor = tk.Button(frame_editor, text="編輯 SKU", width=30, command=run_editor)
btn_editor.pack(padx=5, pady=3)

root.mainloop()
```

- ?? `subprocess.Popen` ??????????????
- GUI ?? **Tkinter**??????????????????????

? ????

1. ????? `\\nas-lianruey\office\sku\app` ??????
2. ????????????????????????????
3. ????????????????? **Python 3.10+** ????????
4. ????????????????????????????????

? SKU Map ??? ????

? ??

SKU Map ??? ????? **Python Tkinter** ??????????
??????? **SKU ??? (SKU Map)**??? **JSON ??** ? **Excel ??** ??????
????????????????????????

???????????????? SKU ????????????

? ????

- Windows 10 / 11
- Python 3.10 ????
- ??????
  - `tkinter`
  - `requests`
  - `openpyxl`

????????

```
pip install requests openpyxl
```

? ????

```
sku_editor_v1.1.py        # □□□□□
```

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

```
https://wu:wu2266228@ec.zfun.com.tw/plist.json
```

? ????

1. ????

- **?? JSON ??**?????? SKU Map?
- **?? JSON**???????????? JSON ??
- **?? JSON**?? SKU Map ????? JSON ??
- **?? Excel**??? Excel ?????? SKU Map?

- **?? Excel**?? SKU Map ? Excel ???????????????? SKU?
- **????????**?????????????

---

2. SKU ????

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

---

3. ?????

???? SKU ????????????????

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

??????

- **????**
- **??????**
- **????**

---

4. ?????

- ???????? `(□□ × □□)` ????
- ??????????????????

? Excel ??/????

????

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

????

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

? ??????

JSON ??

```
{
    "A123": [
        {
            "新SKU": "B456",
            "新數量": "10",
            "新進價": "150"
        },
        {
            "新SKU": "C789",
            "新數量": "5",
            "新進價": "200"
        }
    ]
}
```

Excel ??

??SKU	?SKU	????	???	???
A123	B456	?????(??)	10	150
A123	C789	?????(??)	5	200

?? ???????

```
import json
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import requests
```

```
from openpyxl import Workbook, load_workbook
from openpyxl.styles import PatternFill

class SKUEditor:
    def __init__(self, root):
        self.root = root
        self.root.title("SKU Map 編輯器")

        self.file_path = None
        self.sku_map = {}
        self.product_list = []
        self.product_lookup = {}

        self.selected_sku = None
        self.selected_item_index = None

        self.setup_ui()
        self.load_product_list_from_url()  # 程式啟動時自動載入產品列表

    def load_product_list_from_url(self):
        url = "https://wu:wu2266228@ec.zfun.com.tw/plist.json"
        try:
            response = requests.get(url)
            response.raise_for_status()  # 檢查 HTTP 狀態碼
            data = response.json()

            self.product_list = []
            self.product_lookup = {}
            for item in data:
                sku = str(item.get("sku", "")).strip()
                name = str(item.get("name", "")).strip()
                price = str(item.get("price", "")).strip()
                temp = str(item.get("temp", "")).strip()  # 取得溫層資訊
                # 如果 temp 不為空，則在產品名稱後加上 (溫層)
                if temp:
                    display = f"{name}({temp})"
                else:
                    display = name
                self.product_list.append((sku, display, price))
```

```
                self.product_lookup[sku] = {"name": name, "price": price, "temp": temp}
            self.update_sku_combobox("")
            messagebox.showinfo("成功", "成功從線上載入產品清單！")
        except Exception as e:
            messagebox.showerror("讀取產品清單失敗", f"無法從 {url} 載入資料: {str(e)}")
            self.product_list = []
            self.product_lookup = {}

    def update_sku_combobox(self, keyword):
        keywords = keyword.lower().split()
        filtered = []
        for sku, display, price in self.product_list:
            product_name = display.lower()
            if all(k in product_name for k in keywords):
                filtered.append(display)
        self.sku_combo["values"] = filtered

    def load_json(self, file_path):
        try:
            with open(file_path, "r", encoding="utf-8") as f:
                self.sku_map = json.load(f)
                self.file_path = file_path
                self.refresh_sku_list()
                self.tree.delete(*self.tree.get_children())
                messagebox.showinfo("成功", f"成功讀取: {file_path}")
        except Exception as e:
            messagebox.showerror("錯誤", f"讀取 JSON 檔案失敗: {e}")

    def save_json(self):
        if not self.file_path:
            messagebox.showwarning("未指定檔案", "請先讀取一個 JSON 檔案")
            return
        try:
            with open(self.file_path, "w", encoding="utf-8") as f:
                json.dump(self.sku_map, f, ensure_ascii=False, indent=4)
            messagebox.showinfo("成功", f"資料已儲存至 {self.file_path}")
        except Exception as e:
            messagebox.showerror("儲存失敗", str(e))
```

```
def save_as_json(self):
    file_path = filedialog.asksaveasfilename(
        defaultextension=".json",
        filetypes=[("JSON Files", "*.json")],
        title="另存 JSON 檔案"
    )
    if not file_path:
        return
    try:
        with open(file_path, "w", encoding="utf-8") as f:
            json.dump(self.sku_map, f, ensure_ascii=False, indent=4)
        self.file_path = file_path  # 更新當前檔案路徑
        messagebox.showinfo("成功", f"JSON 檔案已儲存至 {file_path}")
    except Exception as e:
        messagebox.showerror("儲存失敗", f"無法儲存 JSON 檔案: {str(e)}")

def import_from_xlsx(self):
    file_path = filedialog.askopenfilename(
        filetypes=[("Excel Files", "*.xlsx")],
        title="選擇 Excel 檔案"
    )
    if not file_path:
        return
    try:
        wb = load_workbook(file_path)
        ws = wb.active
        headers = [cell.value for cell in ws[1]]
        expected_headers = ["原始SKU", "新SKU", "產品名稱", "新數量", "新進價"]
        if not all(h in headers for h in expected_headers):
            messagebox.showerror("格式錯誤", "Excel 檔案缺少必要的欄位: 原始SKU, 新SKU, 產品名稱, 新數量, 新進價")
            return
        self.sku_map = {}
        for row in ws.iter_rows(min_row=2, values_only=True):
            row_dict = dict(zip(headers, row))
            original_sku = str(row_dict["原始SKU"]).strip()
            new_sku = str(row_dict["新SKU"]).strip()
            quantity = str(row_dict["新數量"]).strip()
            price = str(row_dict["新進價"]).strip()
            # 驗證新SKU是否存在於 product_lookup
```

```
                if new_sku not in self.product_lookup:
                    messagebox.showwarning("警告", f"SKU {new_sku} 不存在於產品清單中，將被忽略")
                    continue
                # 建立資料結構，不包含產品名稱
                item = {
                    "新SKU": new_sku,
                    "新數量": quantity,
                    "新進價": price
                }
                if original_sku not in self.sku_map:
                    self.sku_map[original_sku] = []
                self.sku_map[original_sku].append(item)
            self.file_path = None  # 重置檔案路徑，因為這是新匯入的資料
            self.refresh_sku_list()
            self.tree.delete(*self.tree.get_children())
            messagebox.showinfo("成功", f"成功從 {file_path} 匯入資料")
        except Exception as e:
            messagebox.showerror("匯入失敗", f"無法匯入 Excel 檔案: {str(e)}")

    def export_to_xlsx(self):
        file_path = filedialog.asksaveasfilename(
            defaultextension=".xlsx",
            filetypes=[("Excel Files", "*.xlsx")],
            title="儲存 Excel 檔案"
        )
        if not file_path:
            return
        try:
            wb = Workbook()
            ws = wb.active
            ws.title = "SKU Map"
            # 寫入標頭
            headers = ["原始SKU", "新SKU", "產品名稱", "新數量", "新進價"]
            ws.append(headers)
            # 定義顏色列表 (淺色調，16進位格式，無需前綴"0x")
            colors = [
                "FFFACD",  # 淺檸檬色
                "E0FFFF",  # 淺青色
                "FFE4E1",  # 淺玫瑰色
```

```
                "E6E6FA",  # 淺薰衣草色
                "F0FFF0",  # 蜂蜜露色
            ]
            # 收集所有資料並按原始SKU排序
            data_rows = []
            for original_sku, items in sorted(self.sku_map.items()):
                for item in items:
                    new_sku = item["新SKU"]
                    product_info = self.product_lookup.get(new_sku, {})
                    product_name = product_info.get("name", "未知產品")
                    temp = product_info.get("temp", "")
                    display_name = f"{product_name}({temp})" if temp else product_name
                    quantity = item["新數量"]
                    price = item["新進價"]
                    data_rows.append([original_sku, new_sku, display_name, quantity, price])
            # 追蹤當前原始SKU和顏色索引
            current_sku = None
            color_index = 0
            for row_idx, row_data in enumerate(data_rows, start=2):  # 從第2行開始 (第1行是標頭)
                original_sku = row_data[0]
                # 如果原始SKU改變，切換到下一個顏色
                if original_sku != current_sku:
                    current_sku = original_sku
                    color_index = (color_index + 1) % len(colors)  # 循環使用顏色
                # 寫入資料行
                ws.append(row_data)
                # 應用顏色到整行
                fill = PatternFill(start_color=colors[color_index],
end_color=colors[color_index], fill_type="solid")
                for col_idx in range(1, len(headers) + 1):
                    ws.cell(row=row_idx, column=col_idx).fill = fill
            # 儲存檔案
            wb.save(file_path)
            messagebox.showinfo("成功", f"Excel 檔案已匯出至 {file_path}")
        except Exception as e:
            messagebox.showerror("匯出失敗", f"無法匯出 Excel 檔案: {str(e)}")

    def setup_ui(self):
        top_frame = tk.Frame(self.root)
```

```
        top_frame.pack(fill=tk.X, pady=5)
        tk.Button(top_frame, text="讀取 JSON 檔案",
command=self.select_json_file).pack(side=tk.LEFT, padx=5)
        tk.Button(top_frame, text="儲存 JSON", command=self.save_json).pack(side=tk.LEFT,
padx=5)
        tk.Button(top_frame, text="另存 JSON", command=self.save_as_json).pack(side=tk.LEFT,
padx=5)
        tk.Button(top_frame, text="匯入 Excel",
command=self.import_from_xlsx).pack(side=tk.LEFT, padx=5)
        tk.Button(top_frame, text="匯出 Excel", command=self.export_to_xlsx).pack(side=tk.LEFT,
padx=5)
        tk.Button(top_frame, text="重新載入產品清單",
command=self.load_product_list_from_url).pack(side=tk.LEFT, padx=5)

        # 使用 PanedWindow 來讓左邊框架可以左右拖曳調整大小
        self.paned_window = tk.PanedWindow(self.root, orient=tk.HORIZONTAL)
        self.paned_window.pack(fill=tk.BOTH, expand=True)

        self.left_frame = tk.Frame(self.paned_window)
        self.center_frame = tk.Frame(self.paned_window)
        self.right_frame = tk.Frame(self.paned_window)

        self.paned_window.add(self.left_frame, minsize=150)
        self.paned_window.add(self.center_frame, minsize=300)
        self.paned_window.add(self.right_frame, minsize=150)

        tk.Label(self.left_frame, text="搜尋原始 SKU").pack()
        self.sku_search_var = tk.StringVar()
        self.sku_search_entry = tk.Entry(self.left_frame, textvariable=self.sku_search_var)
        self.sku_search_entry.pack()
        self.sku_search_var.trace("w", lambda *args: self.search_sku())

        tk.Label(self.left_frame, text="原始 SKU").pack()
        self.sku_listbox = tk.Listbox(self.left_frame)
        self.sku_listbox.pack(fill=tk.BOTH, expand=True)
        self.sku_listbox.bind("<<ListboxSelect>>", self.on_sku_select)

        tk.Label(self.left_frame, text="□ □□□□ SKU").pack(pady=2)
        self.new_sku_entry = tk.Entry(self.left_frame)
```

```
        self.new_sku_entry.pack()
        tk.Button(self.left_frame, text="新增 SKU", command=self.add_new_sku).pack(pady=2)

        tk.Label(self.left_frame, text="✏◌ 修改選取 SKU").pack(pady=2)
        self.rename_sku_entry = tk.Entry(self.left_frame)
        self.rename_sku_entry.pack()
        tk.Button(self.left_frame, text="修改 SKU 名稱", command=self.rename_sku).pack(pady=2)
        tk.Button(self.left_frame, text="□ □□□□ SKU", command=self.delete_sku).pack(pady=5)

        tk.Label(self.center_frame, text="子項目").pack()
        self.tree = ttk.Treeview(self.center_frame, columns=("新SKU", "產品名稱", "新數量", "新進價"),
show="headings")
        for col in ("新SKU", "產品名稱", "新數量", "新進價"):
            self.tree.heading(col, text=col)
            self.tree.column(col, width=100)  # 設定欄位寬度
        self.tree.pack(fill=tk.BOTH, expand=True)
        self.tree.bind("<<TreeviewSelect>>", self.on_item_select)

        # 總金額總和標籤
        self.total_label = tk.Label(self.center_frame, text="總金額總和: 0")
        self.total_label.pack(pady=5)

        tk.Label(self.right_frame, text="編輯子項目").pack(pady=5)
        self.entries = {}

        tk.Label(self.right_frame, text="搜尋 SKU").pack()
        self.search_var = tk.StringVar()
        self.search_entry = tk.Entry(self.right_frame, textvariable=self.search_var)
        self.search_entry.pack()
        self.search_var.trace("w", lambda *args:
self.update_sku_combobox(self.search_var.get()))

        self.sku_var = tk.StringVar()
        self.sku_combo = ttk.Combobox(self.right_frame, textvariable=self.sku_var)
        self.sku_combo.bind("<<ComboboxSelected>>", self.on_sku_selected)
        tk.Label(self.right_frame, text="新SKU").pack()
        self.sku_combo.pack()
        self.entries["新SKU"] = self.sku_combo
```

```
        for field in ("新數量", "新進價"):
            tk.Label(self.right_frame, text=field).pack()
            entry = tk.Entry(self.right_frame)
            entry.pack()
            self.entries[field] = entry

        tk.Button(self.right_frame, text="儲存子項修改", command=self.save_item).pack(pady=5)
        tk.Button(self.right_frame, text="新增子項", command=self.add_item).pack(pady=5)
        tk.Button(self.right_frame, text="刪除子項", command=self.delete_item).pack(pady=5)

    def search_sku(self):
        keyword = self.sku_search_var.get().lower()
        self.sku_listbox.delete(0, tk.END)
        for sku in self.sku_map.keys():
            if keyword in sku.lower():
                self.sku_listbox.insert(tk.END, sku)

    def select_json_file(self):
        file_path = filedialog.askopenfilename(
            filetypes=[("JSON Files", "*.json")],
            title="選擇 JSON 檔案"
        )
        if file_path:
            self.load_json(file_path)

    def refresh_sku_list(self):
        self.search_sku()  # Use search_sku to populate list with current search term

    def refresh_items(self):
        self.tree.delete(*self.tree.get_children())
        total_sum = 0.0  # 用浮點數計算總金額總和
        if self.selected_sku:
            for item in self.sku_map[self.selected_sku]:
                new_sku = item["新SKU"]
                # 從 product_lookup 中獲取產品名稱和溫層
                product_info = self.product_lookup.get(new_sku, {})
                product_name = product_info.get("name", "未知產品")
                temp = product_info.get("temp", "")
                # 如果 temp 不為空，則在產品名稱後加上 (溫層)
```

```
            if temp:
                display_name = f"{product_name}({temp})"
            else:
                display_name = product_name
            quantity = item["新數量"]
            price = item["新進價"]
            # 計算總金額，使用浮點數
            try:
                total = float(quantity) * float(price)
                total_sum += total
            except (ValueError, TypeError):
                total = 0.0
            self.tree.insert("", tk.END, values=(new_sku, display_name, quantity, price))
    # 四捨五入總金額總和
    total_sum = round(total_sum)
    self.total_label.config(text=f"總金額總和: {total_sum}")

def on_sku_select(self, event):
    try:
        index = self.sku_listbox.curselection()[0]
        self.selected_sku = self.sku_listbox.get(index)
        self.refresh_items()
    except IndexError:
        return

def on_item_select(self, event):
    selected = self.tree.selection()
    if selected:
        self.selected_item_index = self.tree.index(selected[0])
        values = self.tree.item(selected[0], "values")
        self.sku_var.set(values[0])
        self.entries["新數量"].delete(0, tk.END)
        self.entries["新數量"].insert(0, values[2])
        self.entries["新進價"].delete(0, tk.END)
        self.entries["新進價"].insert(0, values[3])

def on_sku_selected(self, event):
    selected = self.sku_var.get()
    # 從顯示格式中提取 SKU
```

```
        for sku, display, price in self.product_list:
            if display == selected:
                if sku in self.product_lookup:
                    price = self.product_lookup[sku]["price"]
                    self.entries["□□□"].delete(0, tk.END)
                    self.entries["□□□"].insert(0, price)
                break

    def get_clean_entry(self, key):
        if key == "□SKU":
            selected = self.entries[key].get()
            for sku, display, _ in self.product_list:
                if display == selected:
                    return sku
            return selected
        else:
            return self.entries[key].get()

    def save_item(self):
        if self.selected_sku is None or self.selected_item_index is None:
            return
        new_data = {key: self.get_clean_entry(key) for key in self.entries}
        self.sku_map[self.selected_sku][self.selected_item_index] = new_data
        self.refresh_items()

    def add_item(self):
        if self.selected_sku is None:
            return
        new_data = {key: self.get_clean_entry(key) for key in self.entries}
        self.sku_map[self.selected_sku].append(new_data)
        self.refresh_items()

    def delete_item(self):
        if self.selected_sku is None or self.selected_item_index is None:
            return
        del self.sku_map[self.selected_sku][self.selected_item_index]
        self.refresh_items()

    def add_new_sku(self):
```

```
        new_sku = self.new_sku_entry.get().strip()
        if not new_sku:
            return
        if new_sku in self.sku_map:
            messagebox.showwarning("□□□", "□ SKU □□□")
            return
        self.sku_map[new_sku] = []
        self.refresh_sku_list()
        self.new_sku_entry.delete(0, tk.END)

    def rename_sku(self):
        new_name = self.rename_sku_entry.get().strip()
        if not self.selected_sku or not new_name:
            return
        if new_name in self.sku_map:
            messagebox.showwarning("□□□□", "□□ SKU □□□□□")
            return
        self.sku_map[new_name] = self.sku_map.pop(self.selected_sku)
        self.selected_sku = new_name
        self.refresh_sku_list()
        self.rename_sku_entry.delete(0, tk.END)

    def delete_sku(self):
        if not self.selected_sku:
            return
        del self.sku_map[self.selected_sku]
        self.selected_sku = None
        self.refresh_sku_list()
        self.tree.delete(*self.tree.get_children())


if __name__ == "__main__":
    root = tk.Tk()
    app = SKUEditor(root)
    root.mainloop()
```

? ????

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

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

?? ????

1. **??????????**???????SKU ?????????????
2. **Excel ??????**???????????????
3. ??????? **???? JSON ??? Excel**????????
4. ??????????????????????????