

? 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
- ????
- ???
- ???


```

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 表格")

        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() # 初始化数据

    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() # 检查 HTTP 状态
            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() # 温度
                # 温度 格式为 10.000000000000 (10)
                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("", " ")
except Exception as e:
    messagebox.showerror("", f" {url} {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("", f" {file_path}")
    except Exception as e:
        messagebox.showerror("", f" JSON {e}")

def save_json(self):
    if not self.file_path:
        messagebox.showwarning("", " JSON ")
        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("", f" {self.file_path}")
    except Exception as e:

```

```

        messagebox.showerror("Error", str(e))

def save_as_json(self):
    file_path = filedialog.asksaveasfilename(
        defaultextension=".json",
        filetypes=[("JSON Files", "*.json")],
        title="Save JSON File"
    )
    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 # Save file path
        messagebox.showinfo("Success", f"JSON file saved at {file_path}")
    except Exception as e:
        messagebox.showerror("Error", f"Failed to save JSON file: {str(e)}")

def import_from_xlsx(self):
    file_path = filedialog.askopenfilename(
        filetypes=[("Excel Files", "*.xlsx")],
        title="Open Excel File"
    )
    if not file_path:
        return
    try:
        wb = load_workbook(file_path)
        ws = wb.active
        headers = [cell.value for cell in ws[1]]
        expected_headers = ["SKU", "SKU", "ID", "Name", "Price"]
        if not all(h in headers for h in expected_headers):
            messagebox.showerror("Error", "Excel file does not contain required headers: SKU, SKU, ID, Name, Price")
            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["SKU"]).strip()
            new_sku = str(row_dict["SKU"]).strip()
            quantity = str(row_dict["ID"]).strip()

```

```

        price = str(row_dict["price"]).strip()
        # sku product_lookup
        if new_sku not in self.product_lookup:
            messagebox.showwarning("", f"SKU {new_sku} 不存在")
            continue

        # sku
        item = {
            "sku": new_sku,
            "quantity": quantity,
            "price": 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 # sku
        self.refresh_sku_list()
        self.tree.delete(*self.tree.get_children())
        messagebox.showinfo("", f"sku {file_path} ")
    except Exception as e:
        messagebox.showerror("", f"Excel {str(e)}")

```

```

def export_to_xlsx(self):
    file_path = filedialog.asksaveasfilename(
        defaultextension=".xlsx",
        filetypes=[("Excel Files", "*.xlsx")],
        title="Excel "
    )
    if not file_path:
        return
    try:
        wb = Workbook()
        ws = wb.active
        ws.title = "SKU Map"
        # sku
        headers = ["sku", "sku", "quantity", "price", "price"]
        ws.append(headers)
        # sku (sku16)
        colors = [
            "FFFACD", # sku

```

```

        "E0FFFF", # 000
        "FFE4E1", # 0000
        "E6E6FA", # 00000
        "F0FFF0", # 0000
    ]
    # 0000000000SKU00
    data_rows = []
    for original_sku, items in sorted(self.sku_map.items()):
        for item in items:
            new_sku = item["SKU"]
            product_info = self.product_lookup.get(new_sku, {})
            product_name = product_info.get("name", "0000")
            temp = product_info.get("temp", "")
            display_name = f"{product_name}({temp})" if temp else product_name
            quantity = item["000"]
            price = item["000"]
            data_rows.append([original_sku, new_sku, display_name, quantity, price])
    # 00000SKU00000
    current_sku = None
    color_index = 0
    for row_idx, row_data in enumerate(data_rows, start=2): # 002000 (010000)
        original_sku = row_data[0]
        # 0000SKU000000000000
        if original_sku != current_sku:
            current_sku = original_sku
            color_index = (color_index + 1) % len(colors) # 000000
        # 00000
        ws.append(row_data)
        # 00000000
        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
    # 0000
    wb.save(file_path)
    messagebox.showinfo("00", f"Excel 000000 {file_path}")
except Exception as e:
    messagebox.showerror("0000", f"00000 Excel 0000{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="Open JSON File",
command=self.select_json_file).pack(side=tk.LEFT, padx=5)
    tk.Button(top_frame, text="Save JSON", command=self.save_json).pack(side=tk.LEFT,
padx=5)
    tk.Button(top_frame, text="Save as JSON", command=self.save_as_json).pack(side=tk.LEFT,
padx=5)
    tk.Button(top_frame, text="Import Excel",
command=self.import_from_xlsx).pack(side=tk.LEFT, padx=5)
    tk.Button(top_frame, text="Export Excel", command=self.export_to_xlsx).pack(side=tk.LEFT,
padx=5)
    tk.Button(top_frame, text="Load Product List from URL",
command=self.load_product_list_from_url).pack(side=tk.LEFT, padx=5)

# Create PanedWindow
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="SKU Search").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="SKU List").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="□□ SKU", command=self.add_new_sku).pack(pady=2)

tk.Label(self.left_frame, text="□□ □□□□ SKU").pack(pady=2)
self.rename_sku_entry = tk.Entry(self.left_frame)
self.rename_sku_entry.pack()
tk.Button(self.left_frame, text="□□ SKU □□", 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="□□□□").pack()
self.tree = ttk.Treeview(self.center_frame, columns=("□SKU", "□□□□", "□□□", "□□□"),
show="headings")
for col in ("□SKU", "□□□□", "□□□", "□□□"):
    self.tree.heading(col, text=col)
    self.tree.column(col, width=100) # □□□□□□
self.tree.pack(fill=tk.BOTH, expand=True)
self.tree.bind("<<TreeviewSelect>>", self.on_item_select)

# □□□□□□□□
self.total_label = tk.Label(self.center_frame, text="□□□□□□□0")
self.total_label.pack(pady=5)

tk.Label(self.right_frame, text="□□□□□□").pack(pady=5)
self.entries = {}

tk.Label(self.right_frame, text="□□ 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="□SKU").pack()
self.sku_combo.pack()

```

```

self.entries["SKU"] = self.sku_combo

for field in ("ID", "NAME"):
    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="SAVE", command=self.save_item).pack(pady=5)
tk.Button(self.right_frame, text="ADD", command=self.add_item).pack(pady=5)
tk.Button(self.right_frame, text="DELETE", 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="Select JSON File"
    )
    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 # Initialize total sum
    if self.selected_sku:
        for item in self.sku_map[self.selected_sku]:
            new_sku = item["SKU"]
            # product_lookup function call
            product_info = self.product_lookup.get(new_sku, {})
            product_name = product_info.get("name", "Unknown")

```

```

        temp = product_info.get("temp", "")
        # 温度 0表示不开启 (0)
        if temp:
            display_name = f"{product_name}({temp})"
        else:
            display_name = product_name
        quantity = item["数量"]
        price = item["价格"]
        # 计算总价
        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))
# 更新总价
total_sum = round(total_sum)
self.total_label.config(text=f"总价{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["数量"].delete(0, tk.END)
        self.entries["数量"].insert(0, values[2])
        self.entries["价格"].delete(0, tk.END)
        self.entries["价格"].insert(0, values[3])

def on_sku_selected(self, event):

```

```

selected = self.sku_var.get()
# 00000000 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["000"].delete(0, tk.END)
            self.entries["000"].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("警告", "SKU 已存在")
        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("警告", "SKU 已存在")
        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. ?????????????????
-

Revision #3

Created 1 August 2025 02:38:56 by Wayne

Updated 1 August 2025 03:18:24 by Wayne