#Costco推薦排行榜專案最終版

from tkinter import \*

import tkinter as tk

from tkinter.messagebox import \*

from tkinter import messagebox

import webbrowser

import pandas as pd

#背景

window = Tk()

window.geometry("1447x728")

window.configure(bg = "#FFFFFF")

canvas = Canvas(

window,

bg = "#FFFFFF",

height = 728,

width = 1447,

bd = 0,

highlightthickness = 0,

relief = "ridge")

canvas.place(x = 0, y = 0)

background\_img = PhotoImage(file = f"background.png")

background = canvas.create\_image(

723.5, 364.0,

image=background\_img)

pd.set\_option('display.unicode.ambiguous\_as\_wide', True)

pd.set\_option('display.unicode.east\_asian\_width', True)

#點擊按鈕觸發FB爬蟲、Costco爬蟲

def btn\_clicked():

print("Button Clicked")

#FB爬蟲匯入mongoDB

import pandas as pd

import os

import time

import requests

from bs4 import BeautifulSoup

from selenium import webdriver

from chromedriver\_py import binary\_path

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

import random

import pandas as pd

import pymongo

from pymongo import MongoClient

import tkinter as tk

from tkinter import messagebox

global df\_extract

global df

global df2

client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

db = client.costco

col = db.FB\_pythons

x = col.delete\_many({})

print(x.deleted\_count, "個已删除")

#wb=openpyxl.Workbook()

#ws=wb.active

# 你的資訊

url = "https://www.facebook.com/"

email = "linden710127@yahoo.com.tw"

password = "61252483"

#-----------------------------------------------------------------------------

# # 防止跳出通知

chrome\_options = webdriver.ChromeOptions()

prefs = {

"profile.default\_content\_setting\_values":{"notifications": 2}

}

chrome\_options.add\_experimental\_option("prefs", prefs)

# 使用ChromeDriverManager自動下載chromedriver

driver = webdriver.Chrome(executable\_path = binary\_path)

#-----------------------------------------------------------------------------

# 最大化視窗

driver.maximize\_window()

# 進入Facebook登入畫面

driver.get(url)

# 填入帳號密碼，並送出

driver.find\_element\_by\_id("email").send\_keys(email)

driver.find\_element\_by\_id("pass").send\_keys(password)

driver.find\_element\_by\_name("login").click()

time.sleep(5)

#-----------------------------------------------------------------------------

# # # 防止跳出通知

# chrome\_options = webdriver.ChromeOptions()

# prefs = {

# "profile.default\_content\_setting\_values":{"notifications": 2}

# }

# chrome\_options.add\_experimental\_option("prefs", prefs)

# # 使用ChromeDriverManager自動下載chromedriver

# driver = webdriver.Chrome(executable\_path = binary\_path)

# # ChromeDriverManager().install(), chrome\_options=chrome\_options)

# # 最大化視窗

# driver.maximize\_window()

#-----------------------------------------------------------------------------

# 進入Costco好市多 商品經驗老實說

driver.get("https://www.facebook.com/groups/1260448967306807?sorting\_setting=CHRONOLOGICAL")

time.sleep(5)

# 往下滑10次，讓Facebook載入文章內容

for x in range(20):

driver.execute\_script("window.scrollTo(0,document.body.scrollHeight)")

print("scroll",x)

time.sleep(random.randint(3,5))

root = BeautifulSoup(driver.page\_source, "html.parser")

# 定位文章標題

titles = root.find\_all(

"div", class\_="ecm0bbzt hv4rvrfc ihqw7lf3 dati1w0a")

for title in titles:

# 定位每一行標題

posts = title.find\_all("div", class\_="kvgmc6g5 cxmmr5t8 oygrvhab hcukyx3x c1et5uql ii04i59q")

# 如果有文章標題才印出

if len(posts) != 0:

for post in posts:

#print(post.text)

#ws.append([post.text])

#新增單筆資料

st={"FB內文":post.text

}

result=col.insert\_one(st)

os.system('taskkill /im chromedriver.exe /F')

os.system('taskkill /im chrome.exe /F')

#----------------------------------------

#FB爬蟲做成文字雲Dataframe

from pymongo import MongoClient

import pandas as pd

#連接MongoDB

client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

#指定資料庫

db = client.costco

#指定資料表

col=db.FB\_pythons

#直接從MongoDB查詢

post=col.distinct("FB內文")

print(post)

#將FB內文進行斷詞

import jieba

word=[]

for i in post:

a=jieba.cut(i)

word+=a

print(word)

#統計每個字串出現次數 #把每個字串當成key去計算出現的次數(當成值)

dic\_w={}

#判斷KEY(ele)是否在dic\_w{}裡面

for ele in word:

#如果KEY(ele)不在dic\_w{}裡面,如果KEY沒出現過，就新增一筆KEY並讓值從1開始

if ele not in dic\_w:

dic\_w[ele]=1

#如果KEY重複出現，值(AKA次數)就+1

else:

dic\_w[ele]+=1

print(dic\_w)

#調整顯示方式 #因為dic是無序的，所以要用.items來排序呼叫

for ele in dic\_w.items():

#兩個字以上的，才會出現

if len(ele[0])>=2:

num=list()

word=list()

#因為經過.items的關係，變成(0,1)=(前面,後面)

#要排序次數

import operator

#itemgetter是指出現該函數內第幾個的意思(從0開始數)

#.itemgetter()從0開始會變成依筆畫排列

sort\_w=sorted(dic\_w.items(),key=operator.itemgetter(1),reverse=True)

num=[]

word=[]

for ele in sort\_w:

if len(ele[0])>=2 and ele[1]>=2:

word.append(ele[0])

num.append(ele[1])

df\_extract=pd.DataFrame({'num':num,'word':word})

print(df\_extract)

#----------------------------------------

#Costco爬蟲食品清單1

# import requests #請求

# from bs4 import BeautifulSoup #BS要大寫

# import time

# import random

# import pymongo

# import openpyxl #匯入EXCEL格式

#至少要有一筆資料，資料庫才能存在(沒辦法建立空資料庫)

#建立資料庫

client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

db = client.costco

#建立collection

col=db.commodity

x = col.delete\_many({})

print(x.deleted\_count, "筆舊資料已删除")

res=requests.get('https://www.costco.com.tw/c/CL8?') #網址是字串

soup=BeautifulSoup(res.text)

# print(soup.html.body.h1.string)

# print(soup.prettify()) #輸出排版後的HTML內容

#開啟一個工作簿

# wb=openpyxl.Workbook() #只有W要大寫

# ws=wb.active #在工作簿底下開一張新的工作表

# ws["A1"]="商品名稱"

# ws["B1"]="商品價格"

# ws["C1"]="網址"

# print(soup.find\_all("ul",class\_="product-listing product-grid")[0].find\_all("li",class\_="product-list-item product-list-item--grid vline ng-star-inserted")[0].prettify())

for commodity in soup.find\_all("ul",class\_="product-listing product-grid")[0].find\_all("li",class\_="product-list-item product-list-item--grid vline ng-star-inserted"):

#商品名稱

a=commodity.find\_all("a",class\_="lister-name js-lister-name")[0].text

#商品價格

b=commodity.span.text

#網址

c=commodity.a["href"]

#新增單筆資料

st={"商品名稱":a,

"商品價格":b,

"網址":c

}

result=col.insert\_one(st)

# print(a)

# print(b)

# print(c)

# ws.append([a,b,c]) #()內要放list

# wb.save("20220425\_COSTCO商品清單.xlsx") #放在哪個位置決定多久存檔一次

time.sleep(random.randint(1,3))

#連接MongoDB

client = MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

#指定資料庫

db = client.costco

#指定資料表

col=db.commodity

#直接從MongoDB查詢

name=col.distinct("商品名稱")

price=col.distinct("商品價格")

web=col.distinct("網址")

df = pd.DataFrame(list(db.commodity.find({},{ "\_id": 0, "商品名稱": 1, "商品價格": 1, "網址": 1 })))

print(df)

#----------------------------------------

#Costco爬蟲賣場限定清單2

#至少要有一筆資料，資料庫才能存在(沒辦法建立空資料庫)

#建立資料庫

client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

db = client.costco

#建立collection

col=db.commodity2

x = col.delete\_many({})

print(x.deleted\_count, "筆舊資料已删除")

#隱藏瀏覽視窗不跳出

chrome\_options = Options()

chrome\_options.add\_argument("--headless")

driver = webdriver.Chrome(executable\_path = 'C:\\Users\\user\\chromedriver.exe',options=chrome\_options)

driver.get("https://www.costco.com.tw/c/CLWH?q=:relevance:category:WH1205:category:WH1201:category:WH0806:category:WH1202:category:WH0811:category:WH1203:category:WH0808:category:WH1204:category:WH0810:category:WH0809:category:WH0807")

#讓頁面自動往下滑

driver.execute\_script("window,scrollTo(0,document.body.scrollHeight)")

time.sleep(5)

soup = BeautifulSoup(driver.page\_source,"html.parser")

for commodity2 in soup.find\_all("ul",class\_="product-listing product-grid")[0].find\_all("li",class\_="product-list-item product-list-item--grid vline ng-star-inserted"):

#商品名稱

a=commodity2.find\_all("a",class\_="lister-name js-lister-name")[0].text

#網址

c="https://www.costco.com.tw/"+commodity2.a["href"]

# print(a)

# print(c)

# print('-------------------')

#新增單筆資料

st={"商品名稱":a,

"網址":c

}

result=col.insert\_one(st)

time.sleep(random.randint(1,3))

#連接MongoDB

client = MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")

#指定資料庫

db = client.costco

#指定資料表

col=db.commodity2

#直接從MongoDB查詢

name=col.distinct("商品名稱")

price=col.distinct("商品價格")

web=col.distinct("網址")

df2 = pd.DataFrame(list(db.commodity2.find({},{ "\_id": 0, "商品名稱": 1, "商品價格": 1, "網址": 1 })))

print(df2)

root = tk.Tk()

root.withdraw()

messagebox.showinfo('系統提示', '資料已蒐集完畢，點擊「買什麼好呢」獲取推薦清單')

return df\_extract,df,df2

#背景爬蟲按鈕

img0 = PhotoImage(file = f"img0.png")

b0 = Button(

image = img0,

borderwidth = 0,

highlightthickness = 0,

command = btn\_clicked,

relief = "flat")

b0.place(

x = 80, y = 193,

width = 304,

height = 133)

#介面def\*2

#Costco清單1模糊比對結果

def shoplist1(): #定義按鈕使用功能

import pandas as pd

from fuzzywuzzy import fuzz

from fuzzywuzzy import process

global dfx

global final

global complete

#FB貼文的斷詞

chart=df\_extract

#Costco商品列表

chart2=df

def fuzzy\_merge(chart, chart2, key1, key2, threshold=40, limit=1):

"""

:param df\_1: the left table to join

:param df\_2: the right table to join

:param key1: key column of the left table

:param key2: key column of the right table

:param threshold: how close the matches should be to return a match, based on Levenshtein distance

:param limit: the amount of matches that will get returned, these are sorted high to low

:return: dataframe with boths keys and matches

"""

s = chart2[key2].tolist()

m = chart[key1].apply(lambda x: process.extract(x, s, limit=limit))

chart['matches'] = m

m2 = chart['matches'].apply(lambda x: ', '.join([i[0] for i in x if i[1] >= threshold]))

chart['matches'] = m2

chart['merge\_key'] = chart['matches']

chart2['merge\_key'] = chart2[key2]

dfx = pd.merge(chart, chart2, how='left',on='merge\_key')

return dfx

dfx = fuzzy\_merge(chart,chart2,'word','商品名稱',40)

final=dfx.drop(['matches','merge\_key'],axis=1,inplace=True)

print(final)

#資料清洗

# complete=dfx.dropna().sort\_values(["num"],ascending=0,ignore\_index=True).drop(["num","word"],axis=1).drop\_duplicates()[0:10]

final=dfx.dropna().sort\_values(["num"],ascending=0,ignore\_index=True).drop(["num","word"],axis=1).drop\_duplicates()[0:3]

final.index=[1,2,3]

print(dfx)

# l1.config(text=final,justify="left",font=("微軟正黑體"))

# return complete

# 插入資料

global ans

final\_dict=final.to\_dict('list')

for num in range(0,10):

ans=[]

a=final\_dict["商品名稱"][num],final\_dict["商品價格"][num],final\_dict["網址"][num]

ans.append(a)

for index, data in enumerate(ans):

table.insert('', END, values=data) # 新增資料到末尾

#網址連結

def click\_link(event):

select = table.selection()

values = table.item(select, 'values')

link = values[2]

print('link:', link)

webbrowser.open(link)

#Costco清單2模糊比對結果

def shoplist2():

#資料模糊比對流程整合

import pandas as pd

from fuzzywuzzy import fuzz

from fuzzywuzzy import process

global dfy

global final2

global complete2

#FB貼文的斷詞

chart=df\_extract

#Costco商品列表

chart2=df2

def fuzzy\_merge(chart, chart2, key1, key2, threshold=40, limit=1):

"""

:param df\_1: the left table to join

:param df\_2: the right table to join

:param key1: key column of the left table

:param key2: key column of the right table

:param threshold: how close the matches should be to return a match, based on Levenshtein distance

:param limit: the amount of matches that will get returned, these are sorted high to low

:return: dataframe with boths keys and matches

"""

s = chart2[key2].tolist()

m = chart[key1].apply(lambda x: process.extract(x, s, limit=limit))

chart['matches'] = m

m2 = chart['matches'].apply(lambda x: ', '.join([i[0] for i in x if i[1] >= threshold]))

chart['matches'] = m2

chart['merge\_key'] = chart['matches']

chart2['merge\_key'] = chart2[key2]

dfy = pd.merge(chart, chart2, how='left',on='merge\_key')

return dfy

dfy = fuzzy\_merge(chart,chart2,'word','商品名稱',40)

final2=dfy.drop(['matches','merge\_key'],axis=1,inplace=True)

print(dfy)

#資料清洗

# complete2=dfy.dropna().sort\_values(["num"],ascending=0,ignore\_index=True).drop(["num","word"],axis=1).drop\_duplicates()[0:10]

final2=dfy.dropna().sort\_values(["num"],ascending=0,ignore\_index=True).drop(["num","word"],axis=1).drop\_duplicates()[0:3]

final2.index=[1,2,3]

print(final2)

# l2.config(text=final2,justify="left",font=("微軟正黑體"))

# return complete2

# 插入資料

global ans2

final\_dict2=final2.to\_dict('list')

for num in range(0,10):

ans2=[]

b=final\_dict2["商品名稱"][num],final\_dict2["網址"][num]

ans2.append(b)

for index, data in enumerate(ans2):

table2.insert('', END, values=data) # 新增資料到末尾

#網址連結2

def click\_link2(event):

select = table2.selection()

values = table2.item(select, 'values')

link = values[1]

print('link:', link)

webbrowser.open(link)

#背景清單1按鈕

img1 = PhotoImage(file = f"img1.png")

b1 = Button(

image = img1,

borderwidth = 0,

highlightthickness = 0,

command = shoplist1,

relief = "flat")

b1.place(

x = 1131, y = 13,

width = 295,

height = 63)

#背景清單2按鈕

img2 = PhotoImage(file = f"img2.png")

b2 = Button(

image = img2,

borderwidth = 0,

highlightthickness = 0,

command = shoplist2,

relief = "flat")

b2.place(

x = 1123, y = 364,

width = 276,

height = 53)

#treeview插入資料1

import tkinter

from tkinter import \*

from tkinter import ttk

columns = ['商品名稱', '商品價格', '商品連結']

table = ttk.Treeview(

master=window, # 父容器

height=10, # 表格显示的行数,height行

columns=columns, # 显示的列

show='headings', # 隐藏首列

)

table.bind("<Double-Button-1>", click\_link)

table.heading(column='商品名稱', text='商品名稱') # 定义表头

# table.heading(column='商品名稱', text='商品名稱', anchor='w',

# command=lambda: print('商品名稱')) # 定义表头

table.heading('商品價格', text='商品價格', ) # 定义表头

table.heading('商品連結', text='商品連結', ) # 定义表头

table.column('商品名稱', width=100, minwidth=100, anchor=S, ) # 定义列

table.column('商品價格', width=50, minwidth=50, anchor=S) # 定义列

table.column('商品連結', width=150, minwidth=150, anchor=S) # 定义列

table.place(

x = 471, y = 77,

width = 961,

height = 272)

#treeview插入資料2

columns2 = ['商品名稱','商品連結']

table2 = ttk.Treeview(

master=window, # 父容器

height=10, # 表格显示的行数,height行

columns=columns2, # 显示的列

show='headings', # 隐藏首列

)

table2.bind("<Double-Button-1>", click\_link2)

table2.heading(column='商品名稱', text='商品名稱') # 定义表头

# table.heading(column='商品名稱', text='商品名稱', anchor='w',

# command=lambda: print('商品名稱')) # 定义表头

table2.heading('商品連結', text='商品連結', ) # 定义表头

table2.column('商品名稱', width=100, minwidth=100, anchor=S, ) # 定义列

table2.column('商品連結', width=200, minwidth=200, anchor=S) # 定义列

table2.place(

x = 471, y = 425,

width = 961,

height = 272)

window.resizable(False, False)

window.mainloop()