#爬取华南日报网站新闻

# Load modules

import requests

from bs4 import BeautifulSoup as soup

from datetime import datetime, date

# Receiving source code from the South China Morning Post website

scmp\_url = 'https://www.scmp.com/knowledge/topics/china-economy/news'

url\_request = requests.get(scmp\_url)

# Returns the content of the response

url\_content = url\_request.content

# Using BeautifulSoup to parse webpage source code

parsed\_content = soup(url\_content, 'html.parser')

# Find all news sections

filtered\_parts = parsed\_content.find\_all('div', class\_="sc-1yocfo6-0")

page\_info = []

# For loop iterates over every line in text

for section in filtered\_parts:

unit\_info = {}

# (1) Filter title, link, and text content

filtered\_part1 = section.find\_all('a', class\_="sc-1ij6sn6-0")

if len(filtered\_part1) < 2:

continue

# Extract the title and link from the section

news\_title = filtered\_part1[1].text.strip() if len(filtered\_part1) > 1 else ''

news\_link = filtered\_part1[1].get('href').strip() if len(filtered\_part1) > 1 else ''

news\_link = f"https://www.scmp.com{news\_link}" # adjust the relative link

# Filter the description text (optional if needed)

news\_text = filtered\_part1[0].text.strip() if len(filtered\_part1) > 0 else ''

# (2) Filter date

filtered\_part2 = section.find\_all('time', datetime=True)

if filtered\_part2:

try:

# Parse the date format (example format: 2 Aug 2024 - 10:15PM)

news\_date = datetime.strptime(filtered\_part2[0].text.strip(), '%d %b %Y - %I:%M%p')

news\_date = news\_date.date() # only keep the date part

except ValueError:

# If parsing fails, fallback to today's date

news\_date = date.today()

else:

news\_date = date.today()

# Add all info into the dictionary

unit\_info['news\_title'] = news\_title

unit\_info['news\_link'] = news\_link

unit\_info['news\_text'] = news\_text

unit\_info['news\_date'] = news\_date

page\_info.append(unit\_info)

# Print the collected information

for info in page\_info:

print(info)

# Load modules

import pandas as pd

import os

direct = os.getcwd()

# Calling DataFrame constructor on our list

df = pd.DataFrame(page\_info, columns=['news\_title', 'news\_link', 'news\_time'])

print(df)

# Exporting to .csv file

df.to\_csv(direct + '/CSMP\_Scraped\_News.csv')

**爬取中国天气预报数据：**

**"""**

**Application: On Scraping Daily Weather Report of China Cities**

**"""**

**"""**

This is a preliminary tutorial for scraping web pages

With a lot of comments, one can easily get touch web scraping with Python

Python Version: 3.6

@Author: Junjie Hu, jeremy.junjie.hu@gmail.com

"""

# Import all the packages you need, always remember that you can find 99% packages you need in python

import requests # take the website source code back to you

import urllib # some useful functions to deal with website URLs

from bs4 import BeautifulSoup as soup # a package to parse website source code

import numpy as np # all the numerical calculation related methods

import re # regular expression package

import itertools # a package to do iteration works

import pickle # a package to save your file temporarily

import pandas as pd # process structured data

save\_path = 'output/' # the path you save your files

base\_link = 'http://www.tianqihoubao.com/lishi/' # This link can represent the domain of a series of websites

def city\_collection():

request\_result = requests.get(base\_link) # get source code

parsed = soup(request\_result.content) # parse source code

dt\_items = parsed.find\_all('dt') # find the items with tag named 'dt'

for item in dt\_items:

# iterate within all the items

province\_name = item.text.strip() # get name of the province

province\_link2cities = item.find('a')['href'] # get link to all the cities in the province

province = {'province\_link': province\_link2cities}

provinces[province\_name] = province # save dict in the dict

for province in provinces.keys():

# iterate with the province link to find all the cities

cities = {}

print(provinces[province]['province\_link'])

request\_province = requests.get(urllib.parse.urljoin(base\_link, provinces[province]['province\_link']))

# use the urllib package to join relative links in the proper way

parsed\_province = soup(request\_province.content)

dd\_items = parsed\_province.find\_all('dd')

for dd\_item in dd\_items:

print(dd\_item)

cities\_items = dd\_item.find\_all('a')

for city\_item in cities\_items:

city\_name = city\_item.text.strip()

city\_link = city\_item.get('href').split('.')[0]

cities[city\_name] = city\_link

provinces[province]['cities'] = cities

return provinces

def weather\_collection(link):

"""

use the link to collect the weather data

:param link: url link

:return: dict, weather of a city everyday

"""

weather\_page\_request = requests.get(link)

parsed\_page = soup(weather\_page\_request.content)

tr\_items = parsed\_page.find\_all('tr')

month\_weather = dict()

for tr\_item in tr\_items[1:]:

# print(tr\_item)

# daily\_weather = dict()

td\_items = tr\_item.find\_all('td')

date = td\_items[0].text.strip()

split\_pattern = r'[\n\r\s]\s\*'

weather\_states = ''.join(re.split(split\_pattern, td\_items[1].text.strip()))

temperature = ''.join(re.split(split\_pattern, td\_items[2].text.strip()))

wind = ''.join(re.split(split\_pattern, td\_items[3].text.strip()))

month\_weather[date] = {

'weather': weather\_states,

'tempe': temperature,

'wind': wind

}

# month\_weather.append(daily\_weather)

return month\_weather

import datetime

start\_year = 2023

end\_year = 2024 # This is exclusive, so it will stop at 2020

dates = [

(start\_year + i // 12, i % 12 + 1) # Calculate year and month

for i in range((end\_year - start\_year) \* 12)

]

date = [

f"{year}{month:02d}" # Format the date as 'YYYYMM'

for year, month in dates

]

# ==== We have already download the links to all the cities=====

# ==== Otherwise, uncomment the function below to retrieve provinces information ======

provinces = dict() # initialize a dictionary to hold provinces information

# This dictionary includes 'province\_link' which is the links to find the cities for each province and the 'cities' contains city names and links

# provinces\_info = city\_collection() # Use this function to retrieve links to all the cities

# This is called context management, with open can close the document automatically when the

with open('output\_cities\_link.pkl', 'rb') as cities\_file: # write, change 'rb' -> 'wb'

provinces\_info = pickle.load(cities\_file)

print(provinces\_info)

# pickle.dump(provinces\_info, cities\_file) # write

weather\_record = dict()

# The structure is dict in dict

# first layer keyword is province name

# In each province you can find the cities

# In each city, you can find the date, in the date, you can find weather record

for key in provinces\_info.keys():

# Iterate over different provinces

print(key)

for city\_name, city\_link in provinces\_info[key]['cities'].items():

# Iterate cities within each provinces

print(city\_name)

for month\_date in date:

# Iterate over different months

print(city\_name)

print(month\_date)

print(provinces\_info[key]['cities'][city\_name])

print("On Scraping...")

month\_weather = weather\_collection(

urllib.parse.urljoin(base\_link, city\_link) + '/month/' + month\_date + '.html')

weather\_record[key] = {city\_name: {month\_date: month\_weather}}

print('Finished Scraping.')