

할리스커피매장 크롤링 프로그램 분석 & 정리

REPORT

2017301009 김민지

목차

[코드]

hollysCrawler.py -2p

[코드 분석]

Import – 3p

def hollys_store(result): - 4p

def main(): - 11p

if __name__ == '__main__': - 13p

[코드]

[illegible]

[코드 분석 - Import]

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import datetime
```

import는 이미 만들어진 파이썬 프로그램 파일, 라이브러리에 있는 파일등을 사용할 수 있게 해주는 명령어입니다.

¹**Beautiful Soup 모듈**은 HTML 및 XML 파일에서 데이터를 가져오기 위한 Python 라이브러리입니다. 사용자가 선호하는 해석기(파서)와 함께 사용하여 일반적인 방식으로 해석 트리를 항해, 검색 및 변경하는 기능을 제공합니다.

이때 새로운 import 방식이 나오는데 `from A import B` 라는 구문입니다.

이는 A 모듈에 있는 B 함수를 import 한다는 뜻입니다.

이런 방식은 BeautifulSoup 라는 함수가 여러 모듈에 동시에 존재할 수 있기 때문에 bs4 라는 모듈을 특정해서 지칭해주는 것입니다.

urllib.request 모듈은 URL(대부분 HTTP)을 여는 데 도움이 되는 함수와 클래스를 정의합니다.

Pandas 모듈은 관계형 또는 레이블이 된 데이터로 쉽고 직관적으로 작업할 수 있도록 설계되었고 빠르고, 유연한 데이터 구조를 제공하는 Python 패키지입니다.

외부 데이터를 입력 받아 Pandas 자료구조로 저장 및 출력하는 기능을 제공합니다.

이번 프로젝트에서는 Pandas 의 DataFrame을 사용했으며 DataFrame이란 가로축과 세로축이 있는 2차원 데이터를 저장하는 자료구조를 의미합니다.

datetime 모듈은 날짜와 시간을 조작하는 클래스를 제공합니다.

¹ Beautiful Soup Documentation (<https://beautiful-soup-4.readthedocs.io/en/latest/>)

[코드 분석 - def hollys_store(result):]

```
def hollys_store(result):
    for page in range(1,57):
        Hollys_url =
        'https://www.hollys.co.kr/store/korea/korStore.do?pageNo=%d&sid0=&gugun=&store='
        %page
        print(Hollys_url)
        html = urllib.request.urlopen(Hollys_url)
        soupHollys = BeautifulSoup(html, 'html.parser')
        tag_tbody = soupHollys.find('tbody')

        for store in tag_tbody.find_all('tr'):
            if len(store) <= 3:
                break
            store_td = store.find_all('td')
            store_name = store_td[1].string
            store_sido = store_td[0].string
            store_address = store_td[3].string
            store_phone = store_td[5].string

            result.append([store_name]+[store_sido]+[store_address]
+[store_phone])

        return
```

```
def hollys_store(result):
```

hollys_store 에선 result 를 매개변수로 받는 것을 확인할 수 있습니다. 이 result 는 비어있는 list 로 이후 나올 반복문에서 사용됩니다.

```
for page in range(1,57):
```

반복문 구절입니다.

반복문에서 시작 숫자와 끝 숫자를 지정하려면 range(시작 숫자, 끝 숫자) 형태를 사용하는데, 이 때 끝 숫자는 포함되지 않습니다. 때문에 (1,57) 은 1부터 56까지 반복하겠다는 것입니다.

이때 page 는 1부터 56까지의 숫자까지 올라가면서 반복문 내부에서 사용됩니다.

아래의 코드들은 모두 현재 반복문 내부 블록에서 실행됩니다.

```
Hollys_url =
'https://www.hollys.co.kr/store/korea/korStore.do?pageNo=%d&sid=&gugun=&store='
' %page
print(Hollys_url)
html = urllib.request.urlopen(Hollys_url)
soupHollys = BeautifulSoup(html, 'html.parser')
tag_tbody = soupHollys.find('tbody')
```

```
Hollys_url =
'https://www.hollys.co.kr/store/korea/korStore.do?pageNo=%d&sid=&gugun=&store='
' %page
```

Hollys_url 은 실제 할리스커피의 한국 매장 지점을 보여주는 사이트의 URL 을 적은 것입니다.

이때 pageNo=%d 를 이용하여 for 문의 page 즉 크롤링할 페이지를 계속해서 변경할 수 있도록 초기화한 것을 알 수 있습니다.

아래 사진은 실제 pageNo 에 들어갈 번호를 키보드로 변경하여 접속 시 보이는 페이지입니다.

hollys.co.kr/store/korea/korStore.do?pageNo=35&sid=&gugun=&store=

* 할리스 전 매장에서 무선인터넷 서비스 이용이 가능합니다 (단, 휴게소 및 특수매장 제외)

지역	매점명	현황	주소	매점 서비스	전화번호
충남 공주시	이인휴게소점		충남 공주시 이인면 이인리 3-8	 	010-5406-5747
서울 강남구	도심공항2점	영업중	서울시 강남구 테헤란로 87길 22 (도심공항터미널 2층 203-1호)		02-2016-5400
서울 종로구	경복궁역점	영업중	서울특별시 종로구 자하문로 14-1		02-720-5589
충남 당진시	행당도휴게소 (상)점		충청남도 당진군 신평면 서해안고속도로 275	 	
경기 안양시 동안구	안양동편마점	영업중	경기도 안양시 동안구 동편로 6	 	031-422-1880
서울 서초구	구반포역점	영업중	서울특별시 서초구 신반포로 15 구반포상가 G동 1층		02-537-9386
대구 달서구	대구상인점	영업중	대구광역시 달서구 상화북로 194, 2층	  	053-641-9522
경기 수원시 팔달구	수원경인일보점	영업중	경기도 수원시 팔달구 효원로 299	 	031-226-0530
경기 수원시 영통구	수원영통점	영업중	경기도 수원시 영통구 청명남로 10		031-202-3356
경기 수원시 팔달구	수원역점	영업중	경기도 수원시 팔달구 향교로 6(매산로1가,1-3층)	  	031-257-3313

< 31 32 33 34 35 36 37 38 39 40 >

```
print(Hollys_url)
```

해당 Hollys_url 을 콘솔에서 확인할 수 있도록 출력문을 사용한 것으로 보입니다.

실제로 코드를 실행하면 아래와 같이 실제로 사용가능한 URL 이 출력되는 것을 확인할 수 있습니다.

[illegible]

```
html = urllib.request.urlopen(Hollys_url)
```

`urllib.request.urlopen` 기능은 문자열 또는 객체일 수 있는 URL `url` 을 열도록 하는 기능입니다.

`urllib.request.urlopen()` 메서드로 `url` 을 열면 `HTTPResponse` 객체가 만들어집니다.

이 Request 객체로 urlopen을 호출하면 요청된 URL에 대한 응답 객체를 반환합니다.

따라서 해당 구문은 Hollys_url 을 인자로 받아 HTTPResponse 객체를 생성하여, 해당 URL 에 대한 응답 객체를 html 에 저장한다는 것으로 해석할 수 있습니다.

```
soupHollys = BeautifulSoup(html, 'html.parser')
```

대상 웹 페이지의 HTML을 얻은 후에는 BeautifulSoup()생성자를 사용하여 구문 분석 BeautifulSoup하고 문서 트리를 탐색하고 필요한 데이터를 추출하는 데 사용할 수 있는 개체를 가져와야 합니다.

BeautifulSoup 함수는 두개의 매개변수를 갖습니다. 각각 아래와 같은 목적으로 사용됩니다.

```
soup = BeautifulSoup(markup_string, parser)
```

markup_string : 웹 페이지의 문자열

parser : 사용할 파서의 이름으로 구성된 문자열.

특히 BeautifulSoup 라는 함수는 다양한 parser 를 지원합니다..

파서	예시	장점	단점
파이썬의 html.parser	BeautifulSoup(markup, "html.parser")	설치할 필요 없음 적당한 속도	lxml만큼 빠르지 않다.
lxml의 HTML 파서	BeautifulSoup(markup, "lxml")	매우 빠르다.	외부 C 종속성 (lxml 추가 설치 필요)
lxml의 XML 파서	BeautifulSoup(markup, "lxml- xml") BeautifulSoup(markup, "xml")	매우 빠르다. xml을 지원하는 유일 한 parser	외부 C 종속성 (lxml 추가 설치 필요)
HTML5lib	BeautifulSoup(markup, "html5lib")	웹 브라우저와 동일한 방식으로 페이지를 구 문을 분석한다.	외부 Python 종속성 (html5lib 추가 설치 필요) 매우 느리다.

따라서 따로 설치하지 않기 위해 Python 내장 html.parser 를 이용한 것으로 분석됩니다.

이를 통해 soupHollys 에 HTML 파서를 통해 만든 BeautifulSoup객체를 저장하였습니다.

```
tag_tbody = soupHollys.find('tbody')
```

find 함수를 통해 tbody 태그를 찾아와 tag_tbody 에 저장하는 구문입니다.

```
for store in tag_tbody.find_all('tr'):
```

tag_tbody 에서 tr 이란 태그의 개수만큼 반복문을 실행시킵니다.

위의 두 구문을 같이 분석하겠습니다.

먼저 find 함수의 인자를 알아보시다.

```
find(name, attrs, recursive, string, **kwargs)
```

find 함수는 해당 조건에 맞는 하나의 태그를 가져오는 기능을 수행합니다.

find 함수와 비슷한 기능을 하는 함수가 존재하는데 이는 find_all() 함수입니다.

```
find_all(name, attrs, recursive, string, limit, **kwargs)
```

find_all 함수는 해당 조건에 맞는 모든 태그들을 가져오는 기능을 수행합니다.

find 함수와 find_all 함수의 가장 큰 차이점은 중복 개체를 어떻게 처리하는가입니다.

find 함수는 중복이면 가장 첫 번째 태그를 가져옵니다. 그러나 find_all 함수는 all 이라는 이름답게 첫번째 이외의 다른 중복된 태그들까지 가져옵니다.

따라서 find_all 함수는 limit 을 정하여 가져올 수 있는 개수를 제한하는 옵션을 가집니다.

아래의 예시를 통해 find 와 find_all 의 차이점을 확인할 수 있습니다.

```
soup.find_all('title', limit=1)
# [<title>The Dormouse's story</title>]

soup.find('title')
# <title>The Dormouse's story</title>
```



```
if len(store) <= 3:
    break
```

전체의 마지막 tr 인 경우 매장정보가 없으므로 break 를 통해 반복문을 중단하는 구문입니다.

```
store_td = store.find_all('td')
store_name = store_td[1].string
store_sido = store_td[0].string
store_address = store_td[3].string
store_phone = store_td[5].string
```

tr 태그 하위의 td 태그 중에서 필요한 항목만 추출하여 result 리스트에 추가 저장하는 구문입니다.

각각 아래의 목적으로 사용됩니다. 이를 아래의 사진으로 쉽게 파악할 수 있습니다.

store_name : 매장명 -> td[1]

store_sido : 지역 -> td[0]

store_address : 주소 -> td[3]

store_phone : 전화번호 -> td[5]

지역	매장명	현황	주소	매장 서비스	전화번호
경기 성남시 분당구	성남터미널점	영업중	경기도 성남시 분당구 성남대로925번길 16, 성남종합버스터미널 1층		031-725-1004
부산 사상구	부산백양대로점	영업중	괘법동 141		051-311-5399
서울 성동구	왕십리역점	영업중	성동구 왕십리광장로 17 (행당동 168-151), 비트를팩스들 3층		02-2200-1303
서울 마포구	홍대출판사거리점	영업중	서울시 마포구 독막로7길 40, 1층		02-332-3567
인천 남동구	간석오거리점	영업중	인천광역시 남동구 간석동 남동대로 931 씨앤케이 웨딩홀 할리스커피 간석오거리점		032-425-0915
경기 성남시 수정구	위례중앙점	영업중	경기 성남시 수정구 위례광장로 104 위례한화오벨리스크센트럴스퀘어 1층 할리스커피 위례중앙점		031-755-5064
서울 강북구	수유사거리점	영업중	서울시 강북구 도봉로 82길 11 (2~4층)		070-5226-1999
경기 하남시	하남덕풍점	영업중	덕풍북로 2		031-792-9389
서울 영등포구	전경련회관점	영업중	서울시 영등포구 여의도동 28-1 전경련신속회관 8109-11		02-786-3485
대전 서구	대전만년점	영업중	대전 서구 둔산대로117번길 95(만년동, 리더스타운)		

매장명이 td[1] 인 이유는 지역이 더 앞에 존재하기 때문으로 실제 지역이 td[0]으로 설정된 것을 확인할 수 있습니다.

```
result.append([store_name]+[store_sido]+[store_address] +[store_phone])
```

여기서 result 는 def hollys_store 가 받은 매개변수 result 를 의미합니다.

list 에서 append 함수란 리스트의 맨 마지막에 값을 추가하는 기능을 수행합니다.

리스트 안에는 어떤 자료형도 추가할 수 있습니다.

따라서 실제로 result 를 출력할 시

```
[['성남터미널점', '경기 성남시 분당구', '경기도 성남시 분당구 성남대로925번길 16, 성남종합버스터미널 1층', '031-725-1004']]
```

와 같이 저장되는 것을 확인할 수 있습니다.

```
def hollys_store(result):  
    for page in range(1,57):  
        ...  
  
    return
```

모든 반복문이 종료되면 return 합니다.

이 때 사용되는 return 은 어떤 것을 반환하는 용도로 사용되는 것이 아닌 공식적으로 함수가 종료된 것으로 사용한 것이라 분석됩니다.

return 의 방식에는 대략 4 가지가 존재합니다.

```
1. return x  
2. return None  
3. return  
4.
```

1 번은 실제로 어떤 값을 반환하는 용도

2 번은 1 번과 함께 사용되며 반환되는 값이 존재하지 말아야 하는 경우 사용됩니다.

3 번은 break 와 같은 용도로 사용되는데 함수가 종료됨을 의미합니다.

4 번은 함수 내부에 return 하는 것이 없는 것을 의미하며 None 이 반환되기는 하지만 단순히 함수가 종료됨을 나타냅니다.

[코드 분석 - def main():]

```
def main():  
    result = []  
    print('Hollys store crawling >>>>>>>>>>>>>>>>>>>>')  
    holllys_store(result) #[CODE 1] 호출  
    holllys_tbl = pd.DataFrame(result, columns = ('store', 'side-gu',  
'address', 'phone'))  
    holllys_tbl.to_csv('D:/Documents/Jupyter_Workspace/holllys1.csv', encoding =  
'cp949', mode = 'w', index = True)  
    del result[:]
```

```
result = []
```

비어있는 리스트를 선언합니다. 이 result 는 이후에 있을 hollys_store 의 인자로 사용될 것입니다.

```
print('Hollys store crawling >>>>>>>>>>>>>>>>>>')
```

프로젝트에서 크롤링의 시작을 출력하는 구문입니다.

Main 함수에서 바로 아래에 있는 함수로 들어가기 직전에 사용된 것을 보아 Main 함수와 hollys_store 함수와의 구별을 나타내기 위해 출력한 것으로 분석됩니다.

```
hollys_store(result)
```

비어있는 result 를 인수로 주어 hollys_store 함수를 호출합니다.

[['성남시민열점', '경기 성남시 분당구', '경기도 성남시 분당구 성남대로925번길 16, 성남종합버스터미널 1층', '031-725-1004']]


이를 통해 [매장명,지역,주소,전화번호] 값들이 들어간 리스트가 됩니다.

```
hollys_tbl = pd.DataFrame(result, columns = ('store', 'sido-gu', 'address', 'phone'))
```

hollys_tbl 은 매장명,지역,주소,전화번호를 Columns 로 가진 딕셔너리가 됩니다.

```
hollys_tbl.to_csv('D:/Documents/Jupyter_Workspace/hollys1.csv', encoding = 'cp949', mode = 'w', index = True)
```

hollys_tbl 을 to_csv() 함수를 이용하여 아래의 사진과 같이 csv 파일로 저장합니다.

 hollys1.csv 2021-09-30 오후 2:49 Microsoft Excel ... 44KB

해당 csv 파일을 열면 아래와 같은 테이블로 출력됩니다.

	store	sido-gu	address	phone
0	성남터미널	경기 성남	경기도 성남시 분당구	031-725-1004
1	부산백양대	부산 사상	과법동 14	051-311-5399
2	왕십리역점	서울 성동	성동구 왕십리	02-2200-1303
3	홍대출판사	서울 마포	서울시 마포구	02-332-3567
4	간석오거리	인천 남동구		032-425-0915
5	위례중앙점	경기 성남시	수정구	031-755-5064

```
del result[:]
```

Main 함수의 가장 마지막 코드입니다.

result 리스트 뒤에 [:] 가 존재합니다.

여기서 [] 란 시퀀스 슬라이스로, 말 그대로 시퀀스 객체의 일부를 잘라내는 기능을 합니다.

[:] 와 같이 시작 인덱스와 끝 인덱스를 둘다 생략하면 리스트 전체를 가져옵니다.

del 함수는 슬라이스를 삭제하는 기능을 합니다. 따라서 해당 코드는 result 에 들어있는 전체 요소를 삭제하는 것으로 분석할 수 있습니다.

[코드 분석 - if __name__ == '__main__':]

```
if __name__ == '__main__':  
    main()
```

메인 함수의 선언, 시작을 의미합니다.

`__name__`은 모듈의 이름이 저장되는 변수이며 `import` 로 모듈을 가져왔을 때 모듈의 이름이 들어갑니다.

파이썬은 최초로 시작하는 스크립트 파일과 모듈의 차이가 없습니다. 어떤 스크립트 파일이든 시작점도 될 수 있고, 모듈도 될 수 있습니다. 그래서 `__name__` 변수를 통해 현재 스크립트 파일이 시작점인지 모듈인지 판단합니다.

`if __name__ == '__main__':`처럼 `__name__` 변수의 값이 `'__main__'`인지 확인하는 코드는 현재 스크립트 파일이 프로그램의 시작점이 맞는지 판단하는 작업입니다. 즉, 스크립트 파일이 메인 프로그램으로 사용될 때와 모듈로 사용될 때를 구분하기 위한 용도입니다.

따라서 현재 위의 코드는 프로그램의 시작점일 때만 `main()` 함수를 실행한다는 의미를 갖습니다.