Chapter. 03

데이터 저장

I csv와 xlsx 저장하기

FASTCAMPUS

ONLINE

금융공학/퀀트I

강사. 서찬웅

이번 시간에 배울 내용

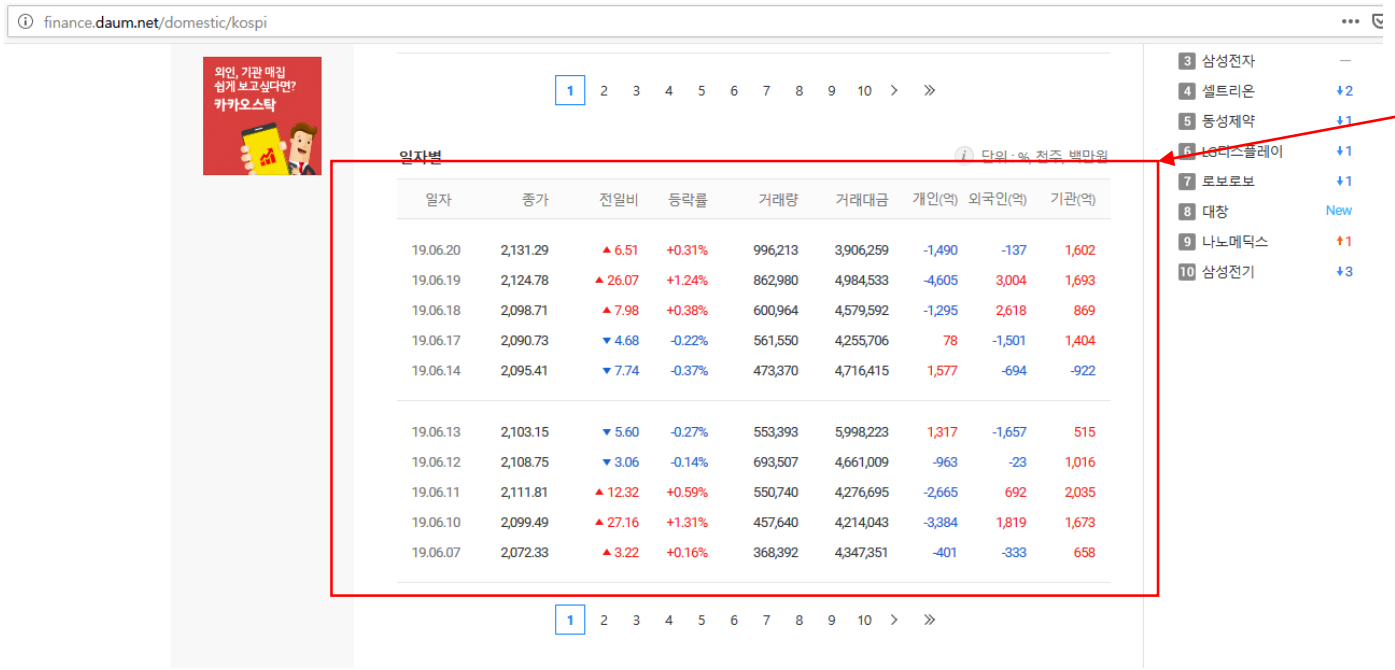
강의내용

1. csv로 저장하기
2. 엑셀 파일로 저장하기

FAST CAMPUS
ONLINE
서찬웅 강사.

I csv 파일에 대해서 알아보겠습니다.

- csv 파일이란 CSV(comma-separated values)는 데이터를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일이며, 확장자는 .csv를 사용합니다.
- 크롤링을 한 결과를 csv 파일로 쉽게 저정하는 방법에 대해서 알아보겠습니다.
  1. 단순하게 open() 함수를 사용하여 ,로 구분자로 하여 텍스트 파일로 저장하는 방법
  2. pandas의 DataFrame 형식으로 변환하여 to_csv() 메소드를 사용하는 방법
- 아래 예제를 통해서 csv 파일로 저장하는 방법에 대해서 알아보겠습니다.



kospi 일자별 데이터를 가져와서 csv 파일로 저장하겠습니다.

I csv 저장을 위해서 크롤링을 진행합니다.

- 다음의 코스피 정보 사이트는 kospi 정보를 get 방식을 사용하여 json 형식으로 데이터를 전송받습니다.
  - json형식은 java script data 형식으로 json, pickle 때 공부하겠습니다.
- Chrome의 개발자 모드에서 network tab 정보를 보시면 해당 데이터가 아래 url 주소로 전송되는 것을 확인 가능
  - 해당 주소로 응답을 받을 때 requests의 기본 헤더값으로 전송하면 접근이 되지 않기 때문에 헤더 설정

```
# get 방식에 사용될 url 주소
url = 'http://finance.daum.net/api/market_index/days?page=1&perPage=100&market=KOSPI&pagination=true'


# 헤더 정보 설정 수정
header = {'Accept' : 'application/json, text/javascript, */*; q=0.01',
        'Referer'         :'http://finance.daum.net/domestic/kospi',
        'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.100 Safari/537.36',
        'X-Requested-With' : 'XMLHttpRequest'}


r = requests.get(url, headers = header)
```

I csv 저장을 위해서 크롤링을 진행합니다.

- requests를 사용하여 text 객체를 추출하면 문자열 객체입니다. 이 객체를 json 라이브러리를 사용하여 dict 형태로 저장하겠습니다.

```
import json
# loads 함수를 사용하여  문자열 객체를 dict 형태로 변환합니다.
json_r = json.loads(r.text)

# dict의 keys값을 출력합니다.
json_r.keys()
```

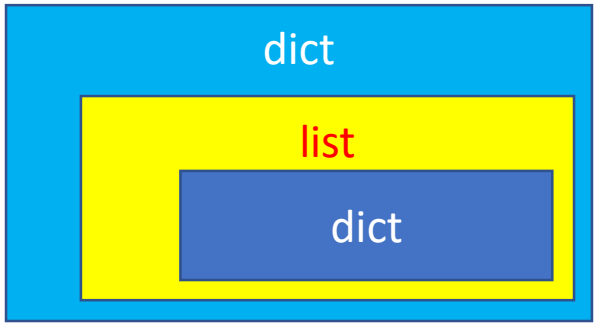
→ 결과 : Out[1]: dict_keys(['data', 'totalPages', 'currentPage', 'pageSize'])

json_r['data'] 결과 →

```
{'date': '2019-01-24 00:00:00',
 'tradePrice': 2145.03,
 'change': 'RISE',
 'changePrice': 17.25,
 'accTradeVolume': 413652,
 'accTradePrice': 6035836,
 'individualStraightPurchasePrice': -175206162984,
 'foreignStraightPurchasePrice': 495880482929,
 'institutionStraightPurchasePrice': -318736757652},
{'date': '2019-01-23 00:00:00',
 'tradePrice': 2127.78,
 'change': 'RISE',
 'changePrice': 10.01,
 'accTradeVolume': 408600,
 'accTradePrice': 5170561,
 'individualStraightPurchasePrice': -347266098123,
 'foreignStraightPurchasePrice': -100581804247,
 'institutionStraightPurchasePrice': 445760608478}]
```

I dict 데이터를 csv 형태로 저장하기 위한 작업

- dict 데이터를 csv 형식으로 저장해보겠습니다.
- json_r['data'] 내부에는 list 형식으로 저장되어 있습니다. 리스트의 원소는 다시 dict 형식으로 되어 있습니다.



- dict 객체의 values() 메소드를 사용하면 dict의 value의 값만 추출할 수 있습니다. 이 값을 list() 형식으로 변환하여 list comprehension 기법을 사용하여 간단하게 리스트안의 원소를 문자열로 변환하고, join 메소드를 사용하여 각각의 리스트를 ,를 붙여서 하나의 문자열 객체로 만들어보겠습니다.

```
for content in json_r3['data']:
    ",".join([str(x) for x in list(content.values())])
```

```
In [95]: for content in json_r3['data']:
    ...:     print(",".join([str(x) for x in list(content.values())]))
2019-06-20 00:00:00,2131.29,RISE,6.51,996213,3906259,-148834364156,-13807253680,160110061638
2019-06-19 00:00:00,2124.78,RISE,26.07,862980,4984533,-460489624085,300430954174,169276816225
2019-06-18 00:00:00,2098.71,RISE,7.98,600964,4579592,-129501534187,261816928364,86891545505
2019-06-17 00:00:00,2090.73,FALL,4.68,561550,4255706,7838261537,-150088594638,140374240557
2019-06-14 00:00:00,2095.41,FALL,7.74,473370,4716415,157690404429,-69368483603,-92172949183
2019-06-13 00:00:00,2103.15,FALL,5.6,553393,5998223,131707587548,-165712563303,51511213419
2019-06-12 00:00:00,2108.75,FALL,3.06,693507,4661009,-96265719945,-2329064115,101634418700
2019-06-11 00:00:00,2111.81,RISE,12.32,550740,4276695,-266518306801,69234514513,203477891603
2019-06-10 00:00:00,2099.49,RISE,27.16,457640,4214043,-338433585668,181865828210,167291422679
```

I 결과를 csv로 저장합니다.

- with 구문을 사용하여 open() 함수를 사용합니다.
  - with 구문을 사용하면 컨텍스트 매니저가 자동으로 __exit__ 호출되어 open의 close()가 자동실행
- dict의 keys 및 values의 값들을 문자열로 변경하고 콤마(,)를 붙여서 csv의 형태로 만들고 text 파일로 저장합니다.
- 텍스트 파일의 각 줄이 끝나면 개행문자(\n)를 붙여서 다음 줄로 만들어 줍니다.

```
# 윈도우 엑셀에서 한글 깨짐을 방지하기 위해서 utf-8-sig형태로 저장
with open("./daum_kospi.csv", 'w', encoding='utf-8-sig') as f:
    f.write(",".join(list(content.keys())) + "\n")
    for content in json_r3['data']:
        f.write(",".join([str(x) for x in list(content.values())]) + "\n")
```

	A	B	C	D	E	F	G	H	I
1	date	tradePrice	change	changePrice	accTradeVolume	accTradePrice	individualStraightPurchasePrice	foreignStraightPurchasePrice	institutionStraightPurchasePrice
2	2019-06-20 0:00	2131.29	RISE	6.51	996213	3906259	(148834364156)	(13807253680)	160110061638
3	2019-06-19 0:00	2124.78	RISE	26.07	862980	4984533	(460489624085)	300430954174	169276816225
4	2019-06-18 0:00	2098.71	RISE	7.98	600964	4579592	(129501534187)	261816928364	86891545505
5	2019-06-17 0:00	2090.73	FALL	4.68	561550	4255706	7838261537	(150088594638)	140374240557
6	2019-06-14 0:00	2095.41	FALL	7.74	473370	4716415	157690404429	(69368483603)	(92172949183)
7	2019-06-13 0:00	2103.15	FALL	5.6	553393	5998223	131707587548	(165712563303)	51511213419

I DataFrame으로 변환하여 쉽게 저장하는 방법

- dict 형태로 저장되어 있는 데이터를 pandas의 DataFrame 형태로 변환하면 to_csv() 메소드를 사용하여 보다 쉽게 csv 형태로 저장할 수 있음

```
import pandas as pd

pd.DataFrame(json_r['data']).to_csv("./daum_kospi.csv", index=False)
```

- DataFrame은 파이썬에서 데이터 처리할 때 매우 중요한 데이터 객체입니다.
  - 다른 챕터 시간에 DataFrame을 정리하는 시간이 있으니 우선은 조금씩 사용법만 익히겠씁니다.

I 수집한 금액 데이터에 콤마(,) 구분자가 포함되어 있는 경우

- 수집한 데이터에 (,)가 구분자로 되어 있는 금액 데이터의 경우 지금까지 학습한 open() 함수를 사용하여 저장하면 원하는 결과로 저장하기 힘듭니다.

```
성함, 금액
서찬웅, 10,000원
김철수, 2,180,500원
김영애, 5,600,000원
```

	A	B	C	D
1	성함	금액		
2	서찬웅	10	000원	
3	김철수	2	180	500원
4	김영애	5	600	000원
5				

통화명	매매기준율	현찰		
		사실 때	파실 때	보내실 때
미국 USD	1,163.50	1,183.86	1,143.14	1,174.90
유럽연합 EUR	1,315.57	1,341.74	1,289.40	1,328.72

- 네이버 환율은 금액에 콤마 구분자가 포함되어 있습니다.
- 아래는 데이터를 콤마를 사용하여 csv로 저장 한 예제

```
url = "https://finance.naver.com/marketindex/exchangeList.nhn"

r = requests.get(url)
bs = BeautifulSoup(r.text, 'html.parser')

with open("./exchange_rate.csv", "w", encoding='utf-8-sig') as f:
    for idx, cont in enumerate(bs.findAll("tr")):
        if(idx>1):
            bs2 = cont.findAll("td")
            for cont2 in bs2:
                f.write(cont2.text.strip() +",")
            f.write("₩n")
```

I 네이버 환율을 저장한 파일을 확인해보겠습니다.

- 데이터 자체에 콤마를 포함하고 있는데 구분자를 콤마로 사용하여 open()를 통해서 저장할 경우 아래처럼 엑셀에서 출력됩니다.

1	미국 USD	1	163.5	1	183.86	1	143.14	1	174.9	1	152.1	1
2	유럽연합 EUR	1	315.57	1	341.74	1	289.4	1	328.72	1	302.42	1.131
3	일본 JPY (100엔)	1	81.77	1	100.7	1	62.84	1	92.37	1	71.17	0.93
4	중국 CNY	169.27	177.73	160.81	170.96	167.58	0.146					
5	홍콩 HKD	148.95	151.88	146.02	150.43	147.47	0.128					
6	대만 TWD	37.5	42.41	34.88	0	0	0.032					
7	영국 GBP	1	472.99	1	502	1	443.98	1	487.71	1	458.27	1.266
8	오만 OMR	3	21.61	3	290.53	2	840.32	0	0	2.597		
9	캐나다 CAD	882.41	899.79	865.03	891.23	873.59	0.758					
10	스위스 CHF	1	183.86	1	207.18	1	160.54	1	195.69	1	172.03	1.018
11	스웨덴 SEK	123.9	126.93	120.87	125.13	122.67	0.107					

- 우리가 원하는 결과의 모습이 아니기 때문에 제대로 된 형식으로 저장하겠습니다.
- csv 모듈은 파이썬에 내장된 모듈이며, 데이터 값에 포함된 콤마 및 복잡한 패턴을 정확하게 처리할 수 있습니다.

I csv 모듈을 사용하여 데이터를 저장하겠습니다.

- csv 모듈의 읽고, 쓰기를 담당하는 함수는 아래와 같습니다.
  - 읽기 : reader()
  - 쓰기 : write(), writerow()
- csv 파일을 쓰기 위해서는 open()의 파일객체를 csv.write() 함수에 전달하고, writerow() 함수를 사용하여 값을 저장합니다. 파일 open() 할 때 newline 매개변수의 값을 ''으로 설정해야 합니다. 그렇지 않으면 각각의 row 값 사이에 한 줄씩 추가되어 있음을 확인할 수 있습니다.

```
with open("exchange_rate_csv_module.csv", "w", encoding='utf-8-sig', newline='') as f:
    filewrite = csv.writer(f, delimiter=",")
    for idx, cont in enumerate(bs.findAll("tr")):
        if(idx>1):
            bs2 = cont.findAll("td")
            total_txt = list()
            for cont2 in bs2:
                total_txt.append(cont2.text.strip())
            filewrite.writerow(total_txt)
```

	미국 USD	1,163.50	1,183.86	1,143.14	1,174.90	1,152.10	1
	유럽연합 EUR	1,315.57	1,341.74	1,289.40	1,328.72	1,302.42	1.131
	일본 JPY (100엔)	1,081.77	1,100.70	1,062.84	1,092.37	1,071.17	0.93
	중국 CNY	169.27	177.73	160.81	170.96	167.58	0.146
	홍콩 HKD	148.95	151.88	146.02	150.43	147.47	0.128
	대만 TWD	37.5	42.41	34.88	0	0	0.032
	영국 GBP	1,472.99	1,502.00	1,443.98	1,487.71	1,458.27	1.266
	오만 OMR	3,021.61	3,290.53	2,840.32	0	0	2.597
	캐나다 CAD	882.41	899.79	865.03	891.23	873.59	0.758
0	스위스 CHF	1,183.86	1,207.18	1,160.54	1,195.69	1,172.03	1.018
1	스웨덴 SEK	123.9	126.93	120.87	125.13	122.67	0.107

I csv 모듈을 사용하여 데이터를 읽어보겠습니다.

- 이번에는 저장된 csv 파일을 csv 모듈을 사용하여 읽어보겠습니다.
- reader() 함수를 사용하여 csv.reader 객체를 생성하고 파일을 읽는 예제입니다.

```
with open("./fastcampus/exchange_rate_csv_module.csv", "r", encoding='utf-8-sig') as f :
    filereader = csv.reader(f, delimiter=',')
    for content in filereader:
        print (content)
```

csv 모듈을 사용하여 csv 파일을 읽으면 한 행당 하나의 리스트로 출력된다.

```
['미국 USD', '1,163.50', '1,183.86', '1,143.14', '1,174.90', '1,152.10', '1.000']
['유럽연합 EUR', '1,315.57', '1,341.74', '1,289.40', '1,328.72', '1,302.42', '1.131']
['일본 JPY (100엔)', '1,081.77', '1,100.70', '1,062.84', '1,092.37', '1,071.17', '0.930']
['중국 CNY', '169.27', '177.73', '160.81', '170.96', '167.58', '0.146']
['홍콩 HKD', '148.95', '151.88', '146.02', '150.43', '147.47', '0.128']
['대만 TWD', '37.50', '42.41', '34.88', '0.00', '0.00', '0.032']
['영국 GBP', '1,472.99', '1,502.00', '1,443.98', '1,487.71', '1,458.27', '1.266']
['오만 OMR', '3,021.61', '3,290.53', '2,840.32', '0.00', '0.00', '2.597']
['캐나다 CAD', '882.41', '899.79', '865.03', '891.23', '873.59', '0.758']
['스위스 CHF', '1,183.86', '1,207.18', '1,160.54', '1,195.69', '1,172.03', '1.018']
['스웨덴 SEK', '123.90', '126.93', '120.87', '125.13', '122.67', '0.107']
```

I 파일을 엑셀에 저장하는 방법에 대해서 알아보겠습니다.

- 엑셀 파일의 확장자는 크게 xls, xlsx 2가지 존재합니다.
  - xls : 2003 버전 이하에서 사용한 파일
  - xlsx : 2007 버전 이상에서 사용하는 파일
- 엑셀 파일을 읽고 쓰기를 지원하는 라이브러리는 아래와 같이 존재합니다.
  - xlwt : 엑셀 95부터 엑셀 2003 버전을 지원하는(xls) 쓰기 라이브러리
  - xlrd : 엑셀 95부터 엑셀 2003 버전을 지원하는(xls) 읽기 라이브러리
  - xlsxwriter : 엑셀 2007 이상 버전을 지원하는 엑셀 라이브러리
  - openpyxl : 엑셀 2010 버전의 xlsx/xlsm/xltx/xltm 파일을 읽고 쓰기 할 수 있는 라이브러리
- xlsxwriter의 사용법을 알아보겠습니다.
  - Workbook() : excel 파일 생성하는 함수 , add_worksheet() : 새로운 sheet를 추가하는 함수
  - add_format() : 엑셀 서식을 설정하는 함수
  - write() : 엑셀에 데이터를 기록하는 함수

I 데이터를 엑셀로 저장하겠습니다.

- 환율 정보를 total이라는 식별자를 갖는 리스트에 저장하여 결과를 엑셀에 저장하겠습니다.

```
import os, csv

total = []
with open("./fastcampus/exchange_rate_csv_module.csv", "r", encoding='utf-8-sig') as f :
    filereader = csv.reader(f, delimiter=',')
    for content in filereader:
        total.append(content)
```

앞의 예제에서 total이라는 리스트를 생성하여 값을 append한다.

total의 데이터를 엑셀로 저장하는 예제

```
# Sheet 이름을 정한다.
worksheet = workbook.add_worksheet('환율정보')

# 컬럼의 넓이를 정할수 있다.
worksheet.set_column('A:F', 20)

# Cell의 스타일을 지정할 수 있다.
bold = workbook.add_format({'bold' : True})

for idx1, x1 in enumerate(total):
    for idx2, x2 in enumerate(x1):
        # write() 메소드는 실제 엑셀 파일에 기록하는 메소드이며,
        # idx2, idx2 번째 위치에 x2의 값을 bold형태로 저장한다.
        worksheet.write(idx1, idx2, x2, bold)

# 파일을 닫는다
workbook.close()
```

I 정리

- csv 파일에 대해서 알아보고 저장하는 방법
- csv 라이브러리를 활용하여 저장하는 방법 및 장점
- 파이썬에서 엑셀을 다루는 라이브러리
- xlsxwriter 라이브러리를 활용하여 저정하는 방법

FAST CAMPUS
ONLINE
서찬웅 강사.

감사합니다

FAST CAMPUS
ONLINE
서찬웅 강사.