

## Part 2. 데이터 입출력

---

1. 외부파일 읽기
  - 1-1. CSV 파일
  - 1-2. Excel 파일
  - 1-3. JSON 파일
2. 웹(web)에서 가져오기
  - 2-1. HTML 웹 페이지에서 표 속성 가져오기
  - 2-2. 웹 스크래핑
3. API 활용하여 데이터 수집하기
4. 데이터 저장하기
  - 4-1. CSV 파일로 저장
  - 4-2. JSON 파일로 저장
  - 4-3. Excel 파일로 저장
  - 4-4. 여러 개의 데이터프레임을 하나의 Excel 파일로 저장

## Part 2. 데이터 입출력

### 1. 외부파일 읽기

#### • 판다스 데이터 입출력 도구

판다스는 다양한 형태의 외부 파일을 읽어와서 데이터프레임으로 변환하는 함수를 제공한다. 어떤 파일이든 판다스 객체인 데이터프레임으로 변환되고 나면, 판다스의 모든 함수와 기능을 자유롭게 사용할 수 있다.

반대로, 데이터프레임을 다양한 유형의 파일로 저장할 수도 있다. [표 2-1]은 판다스 공식 사이트에서 제공하는 입출력 도구에 관한 자료를 요약한 것이다.

File Format	Reader	Writer
CSV	read_csv	to_csv
JSON	read_json	to_json
HTML	read_html	to_html
Local clipboard	read_clipboard	to_clipboard
MS Excel	read_excel	to_excel
HDF5 Format	read_hdf	to_hdf
SQL	read_sql	to_sql

[표 2-1] 판다스 데이터 입출력 도구(출처: <http://pandas.pydata.org>)

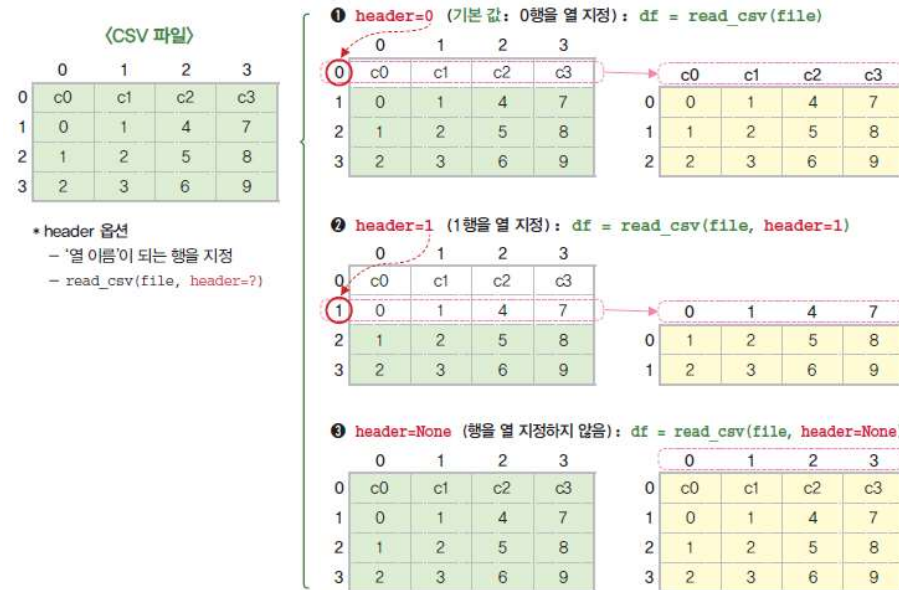
# Part 2. 데이터 입출력

## 1-1. CSV 파일

데이터 값을 쉼표(,)로 구분하고 있다는 의미로 CSV(comma-separated values)라고 부르는 텍스트 파일이다. 쉼표(,)로 열을 구분하고 줄바꿈으로 행을 구분한다.

read\_csv() 함수에 확장자(.csv)를 포함하여 파일경로(파일명)을 입력하면, CSV 파일을 읽어와서 데이터프레임으로 변환한다.

CSV 파일 → 데이터프레임: `pandas.read_csv("파일 경로(이름)")`



[그림 2-1] CSV 파일 읽기 - header 옵션 비교



[그림 2-2] CSV 파일 읽기 - index\_col 옵션 비교

옵션	설명
path	파일의 위치(파일명 포함), URL
sep(또는 delimiter)	텍스트 데이터를 필드별로 구분하는 문자
header	열 이름으로 사용될 행의 번호(기본값은 0) header가 없고 첫 행부터 데이터가 있는 경우 None으로 지정 가능
index_col	행 인덱스로 사용할 열의 번호 또는 열 이름
names	열 이름으로 사용할 문자열의 리스트
skiprows	처음 몇 줄을 skip할 것인지 설정(숫자 입력) skip하려는 행의 번호를 담은 리스트로 설정 가능(예: [1, 3, 5])
parse_dates	날짜 텍스트를 datetime64로 변환할 것인지 설정(기본값은 False)
skip_footer	마지막 몇 줄을 skip할 것인지 설정(숫자 입력)
encoding	텍스트 인코딩 종류를 지정(예: 'utf-8')

[표 2-2] read\_csv() 함수의 옵션

## Part 2. 데이터 입출력

### 1-1. CSV 파일

#### ① CSV 파일 미리보기

예제에서 불러올 CSV 파일의 내용을 확인하면, 데이터가 쉼표(,)와 행으로 구분된 것을 확인할 수 있다.

〈CSV 파일〉 미리보기

(File: example/part2/read\_csv\_sample.csv)

```
1 c0,c1,c2,c3
2 0,1,4,7
3 1,2,5,8
4 2,3,6,9
```

#### ② CSV 파일 읽어오기

〈예제 2-1〉 CSV 파일 읽기

(File: example/part2/2.1\_csv\_to\_df.py)

```
1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5
6 # 파일 경로(파이션 파일과 같은 폴더)를 찾고, 변수 file_path에 저장
7 file_path = './read_csv_sample.csv'
8
9 # read_csv() 함수로 데이터프레임 변환. 변수 df1에 저장
10 df1 = pd.read_csv(file_path)
11 print(df1)
12 print('\n')
13
14 # read_csv() 함수로 데이터프레임 변환. 변수 df2에 저장. header=None 옵션
15 df2 = pd.read_csv(file_path, header=None)
16 print(df2)
17 print('\n')
18
19 # read_csv() 함수로 데이터프레임 변환. 변수 df3에 저장. index_col=None 옵션
20 df3 = pd.read_csv(file_path, index_col=None)
21 print(df3)
22 print('\n')
23
24 # read_csv() 함수로 데이터프레임 변환. 변수 df4에 저장. index_col='c0' 옵션
25 df4 = pd.read_csv(file_path, index_col='c0')
26 print(df4)
```

〈실행 결과〉 코드 전부 실행

```
   c0  c1  c2  c3
0  0   1   4   7
1  1   2   5   8
2  2   3   6   9
```

```
   0   1   2   3
0 c0  c1  c2  c3
1  0   1   4   7
2  1   2   5   8
3  2   3   6   9
```

```
   c0  c1  c2  c3
0  0   1   4   7
1  1   2   5   8
2  2   3   6   9
```

```
   c1  c2  c3
c0
0   1   4   7
1   2   5   8
2   3   6   9
```

header 옵션이 없으면 CSV 파일의 첫 행의 데이터(c0,c1,c2,c3)가 열 이름이 된다.

한편, index\_col 옵션을 지정하지 않으면, 행 인덱스는 정수 0, 1, 2가 자동으로 지정된다.

데이터프레임 df4의 경우, index\_col='c0' 옵션을 사용하여 'c0' 열이 행 인덱스가 되는 것을 볼 수 있다.

# Part 2. 데이터 입출력

## 1-2. Excel 파일

Excel 파일(확장자: .xlsx)의 행과 열은 데이터프레임의 행, 열로 일대일 대응된다. read\_excel() 함수의 사용법은 앞에서 살펴본 read\_csv() 함수와 거의 비슷하다.

header, index\_col 등 대부분의 옵션을 그대로 사용할 수 있다.

Excel 파일 → 데이터프레임: `pandas.read_excel("파일 경로(이름) ")`

### ① Excel 파일 미리보기

(Excel 파일) 미리보기 (File: example/part2/남북한발전전력량.xlsx)

	A	B	C	D	E	F	AA	AB	AC	AD
	전력량 (kWh)	발전 전력량	1990	1991	1992	1993	2014	2015	2016	
1	남한	합계	1,077	1,186	1,310	1,444	5,220	5,171	5,220	5,404
2		수력	64	51	49	60	78	58	66	
3		화력	484	573	696	803	3,402	3,402	3,523	
4		원자력	529	563	565	581	1,564	1,648	1,620	
5		신재생	-	-	-	-	151	173	195	
6	북한	합계	277	263	247	221	216	190	239	
7		수력	156	150	142	133	130	100	128	
8		화력	121	113	105	88	86	90	111	
9		원자력	-	-	-	-	-	-	-	

### ② Excel 파일 읽어오기

(예제 2-2) Excel 파일 읽기 (File: example/part2/2.2\_read\_excel.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # read_excel() 함수로 데이터프레임 변환
6 df1 = pd.read_excel('./남북한발전전력량.xlsx') # header=0(default 옵션)
7 df2 = pd.read_excel('./남북한발전전력량.xlsx', header=None) # header = None 옵션
```

```
9 # 데이터프레임 출력
10 print(df1)
11 print('\n')
12 print(df2)
```

<실행 결과> 코드 전부 실행

전력량 (kWh)	발전	전력별	1990	1991	1992	...	2012	2013	2014	2015	2016
0	남한	합계	1077	1186	1310	...	5096	5171	5220	5281	5404
1	NaN	수력	64	51	49	...	77	84	78	58	66
2	NaN	화력	484	573	696	...	3430	3581	3427	3402	3523
3	NaN	원자력	529	563	565	...	1503	1388	1564	1648	1620
4	NaN	신재생	-	-	-	...	86	118	151	173	195
5	북한	합계	277	263	247	...	215	221	216	190	239
6	NaN	수력	156	150	142	...	135	139	130	100	128
7	NaN	화력	121	113	105	...	80	82	86	90	111
8	NaN	원자력	-	-	-	...	-	-	-	-	-

[9 rows x 29 columns]

	0	1	2	3	4	...	24	25	26	27	28	
0	전력량 (kWh)	발전	전력별	1990	1991	1992	...	2012	2013	2014	2015	2016
1	남한	합계		1077	1186	1310	...	5096	5171	5220	5281	5404
2	NaN	수력		64	51	49	...	77	84	78	58	66
3	NaN	화력		484	573	696	...	3430	3581	3427	3402	3523
4	NaN	원자력		529	563	565	...	1503	1388	1564	1648	1620
5	NaN	신재생		-	-	-	...	86	118	151	173	195
6	북한	합계		277	263	247	...	215	221	216	190	239
7	NaN	수력		156	150	142	...	135	139	130	100	128
8	NaN	화력		121	113	105	...	80	82	86	90	111
9	NaN	원자력		-	-	-	...	-	-	-	-	-

[10 rows x 29 columns]

header 옵션을 추가하지 않은 경우에는 Excel 파일의 첫 행이 열 이름을 구성한다. 한편, header=None 옵션을 사용하면, 정수형 인덱스(0, 1, 2, ...)를 열 이름으로 자동 할당한다.

## Part 2. 데이터 입출력

### 1-3. JSON 파일

JSON 파일(확장자: .json)은 JavaScript에서 유래한 데이터 공유를 목적으로 개발된 특수한 파일형식이다.

파이썬 딕셔너리와 비슷하게 'key : value' 구조를 갖는다.

read\_json() 함수를 사용하여, JSON 파일을 데이터프레임으로 변환한다.

JSON 파일 → 데이터프레임: `pandas.read_json("파일 경로(이름) ")`

#### ① JSON 파일 미리보기

〈JSON 파일〉 미리보기 (File: example/part2/read\_json\_sample.json)

```
1 {
2   "name": {"pandas": "",
3            "NumPy": "",
4            "matplotlib": ""},
5
6   "year": {"pandas": 2008,
7            "NumPy": 2006,
8            "matplotlib": 2003},
9
10  "developer": {"pandas": "Wes McKinney",
11               "NumPy": "Travis Oliphant",
12               "matplotlib": "John D. Hunter"},
13
14  "opensource": {"pandas": "True",
15                 "NumPy": "True",
16                 "matplotlib": "True"}
17 }
```

JSON 파일에는 주요 파이썬 패키지의 출시년도, 개발자, 오픈소스 정보가 들어있다.

#### ② JSON 파일 읽어오기

〈예제 2-3〉 JSON 파일 읽기

(File: example/part2/2.3\_read\_json.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # read_json() 함수로 데이터프레임 변환
6 df = pd.read_json('./read_json_sample.json')
7 print(df)
8 print('\n')
9 print(df.index)
```

〈실행 결과〉 코드 전부 실행

name	year	developer	opensource
NumPy	2006	Travis Oliphant	True
matplotlib	2003	John D. Hunter	True
pandas	2008	Wes McKinney	True

Index(['NumPy', 'matplotlib', 'pandas'], dtype='object')

JSON 파일의 "name" 데이터("pandas", "NumPy", "matplotlib")가 인덱스로 지정된다.



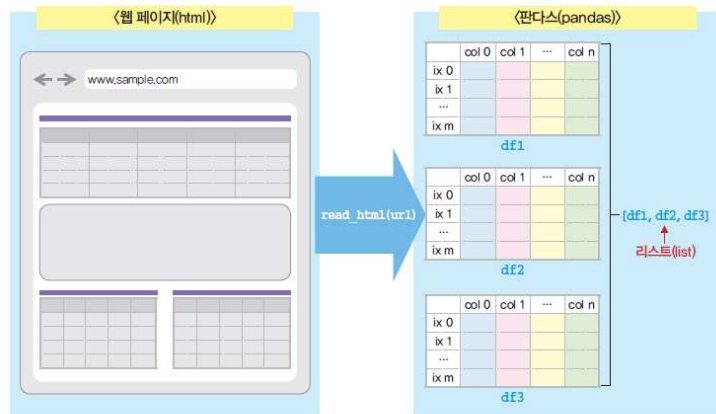
## Part 2. 데이터 입출력

### 2. 외부파일 읽기

#### 2-1. HTML 웹 페이지에서 표 속성 가져오기

read\_html() 함수는 HTML 웹 페이지에 있는 <table> 태그에서 표 형식의 데이터를 모두 찾아서 데이터프레임으로 변환한다.

HTML 표 속성 읽기 : `pandas.read_html( "웹 주소(URL)" 또는 "HTML 파일 경로(이름)" )`



[그림 2-3] HTML 페이지의 표 가져오기

<예제 2-4> 웹에서 표 정보 읽기

(File: example/part2/2.4\_read\_html.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # HTML 파일 경로 or 웹 페이지 주소를 url 변수에 저장
6 url = './sample.html'
7
8 # HTML 웹페이지의 표(table)를 가져와서 데이터프레임으로 변환
9 tables = pd.read_html(url)
10
11 # 표(table)의 개수 확인
12 print(len(tables))
```

```
15 # tables 리스트의 원소를 iteration하면서 각각 화면 출력
16 for i in range(len(tables)):
17     print("tables[%s]" % i)
18     print(tables[i])
19     print('\n')
20
21 # 파이썬 패키지 정보가 들어 있는 두 번째 데이터프레임을 선택하여 df 변수에 저장
22 df = tables[1]
23
24 # 'name' 열을 인덱스로 지정
25 df.set_index(['name'], inplace=True)
26 print(df)
```

<실행 결과> 코드 전부 실행

```
2

tables[0]
   Unnamed: 0  c0  c1  c2  c3
0           0   0   1   4   7
1           1   1   2   5   8
2           2   2   3   6   9

tables[1]
   name  year  developer  opensource
0  NumPy  2006  Travis Oliphant      True
1  matplotlib  2003  John D. Hunter      True
2   pandas  2008   Wes Mckinney      True

name  year  developer  opensource
NumPy  2006  Travis Oliphant      True
matplotlib  2003  John D. Hunter      True
pandas  2008   Wes Mckinney      True
```

표 데이터들은 각각 별도의 데이터프레임으로 변환되기 때문에, 여러 개의 데이터프레임(표)을 원소로 갖는 리스트가 반환된다.

## Part 2. 데이터 입출력

### 2. 외부파일 읽기

#### 2-2. 웹 스크래핑

BeautifulSoup 등 웹 스크래핑(scraping) 도구로 수집한 내용을 파이썬 리스트, 딕셔너리 등으로 정리하고, DataFrame() 함수에 리스트/딕셔너리 형태로 전달하여 데이터프레임으로 변환한다.

다음은 위키피디아에서 미국 ETF 리스트 데이터를 가져와서 데이터프레임으로 변환하는 예제이다.

〈예제 2-5〉 미국 ETF 리스트 가져오기

(File: example/part2/2.5\_us\_etf\_list.py)

```
1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 from bs4 import BeautifulSoup
5 import requests
6 import re
7 import pandas as pd
8
9 # 위키피디아 미국 ETF 웹 페이지에서 필요한 정보를 스크래핑하여 딕셔너리 형태로 변수 etfs에 저장
10 url = "https://en.wikipedia.org/wiki/List_of_American_exchange-traded_funds"
11 resp = requests.get(url)
12 soup = BeautifulSoup(resp.text, 'lxml')
13 rows = soup.select('div > ul > li')
14
15 etfs = {}
16 for row in rows:
17
18     try:
19         etf_name = re.findall('^(.*) \\\(NYSE', row.text)
20         etf_market = re.findall('\\((.*)\\)', row.text)
21         etf_ticker = re.findall('NYSE Arca\\(.*\\)', row.text)
22
23         if (len(etf_ticker) > 0) & (len(etf_market) > 0):
24             etfs[etf_ticker[0]] = [etf_market[0], etf_name[0]]
25
26     except AttributeError as err:
27         pass
```

```
29 # etfs 딕셔너리 출력
30 print(etfs)
31 print('\\n')
32
33 # etfs 딕셔너리를 데이터프레임으로 변환
34 df = pd.DataFrame(etfs)
35 print(df)
```

〈실행 결과〉 코드 전부 실행

```
{ 'ITOT': ['NYSE Arca', 'iShares Core S&P Total US Stock Mkt'], 'IWV': ['NYSE Arca', 'iShares Russell 3000 Index'], 'SCHB': ['NYSE Arca', 'Schwab US Broad Market ETF'], 'FNDB': ['NYSE Arca', 'Schwab Fundamental U.S. Broad Market Index ETF'], 'VTI': ['NYSE Arca', 'Vanguard Total World Stock'], 'VTI': ['NYSE Arca', 'Vanguard Total Stock Market'], 'VXUS': ['NYSE Arca', 'Vanguard Total International Stock'], 'VTHR': ['NYSE Arca', 'Vanguard Russell 3000'], 'DIA': ['NYSE Arca', 'DIAMONDS Trust, Series 1'], 'RSP': ['NYSE Arca', 'Guggenheim S&P 500 Equal Weight'], 'IOO': ['NYSE Arca', 'iShares S&P Global 100 Index'],
```

... 중략 ...

```
, 'EMCB': ['NYSE Arca', 'WisdomTree Emerging Markets Corporate Bond Fund'], 'EU': ['NYSE Arca', 'WisdomTree Euro Debt Fund'], 'ICB': ['NYSE Arca', 'WisdomTree Dreyfus Indian Rupee'], 'RRF': ['NYSE Arca', 'WisdomTree Global Real Return'], 'USDU': ['NYSE Arca', 'WisdomTree Bloomberg U.S. Dollar Bullish Fund'], 'WDTI': ['NYSE Arca', 'WisdomTree Managed Futures Strategy Fund'] }
```

	ITOT	...	WDTI
0	NYSE Arca	...	NYSE Arca
1	iShares Core S&P Total US Stock Mkt	...	WisdomTree Managed Futures Strategy Fund

[2 rows x 378 columns]

24번 라인의 `etfs[etf_ticker[0]] = [etf_market[0], etf_name[0]]` 와 같이 리스트를 원소로 갖는 딕셔너리를 정의하는 방법을 반드시 기억한다. 왼쪽의 딕셔너리 키는 열 이름이 되고, 오른쪽 리스트는 열 데이터가 된다. 예제에서는, ETF 거래코드(etf\_ticker)가 데이터프레임의 열 이름이 된다.



## Part 2. 데이터 입출력

### 3. API 활용하여 데이터 수집하기

인터넷 업체가 제공하는 API를 통해서 수집한 데이터를, 판다스 자료구조로 변환하는 방법이다. API를 통해 가져온 데이터를 판다스 데이터프레임으로 손쉽게 변환할 수 있다.

#### • Google 지오코딩 API

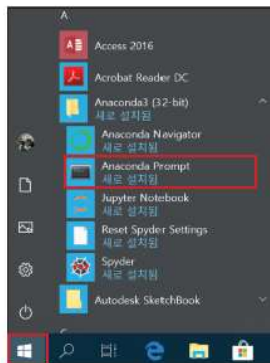
구글 지오코딩이란 장소 이름 또는 주소를 입력하면, 위도와 경도 좌표 정보를 변환해 주는 서비스이다. 서비스를 이용하려면, 사용자 인증 후에 API 키를 발급받아야 한다.

##### 구글 지오코딩 API 발급 절차

1. 구글 지도 서비스(<https://cloud.google.com/maps-platform/places/?hl=ko>) 접속
2. 새 프로젝트 만들기(무료 평가판은 사용 가능하지만 신용카드 정보 등록 필요)
3. API 설정
4. 사용자 인증
5. API 키 발급

#### • googlemaps 라이브러리 설치

- ① 아나콘다 프롬프트(Anaconda Prompt)를 실행한다.



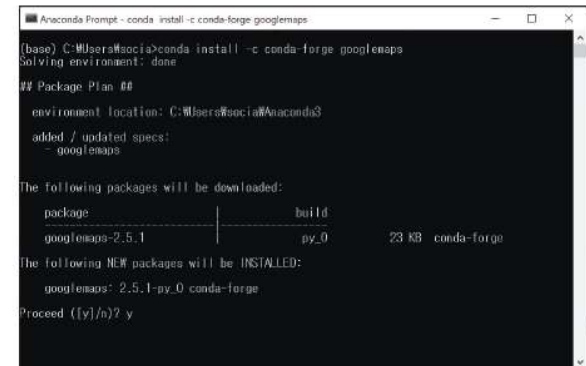
[그림 2-4] 아나콘다 프롬프트 실행

- ② 콘다 설치명령 `conda install -c conda-forge googlemaps`를 입력하고 `Enter`를 누른다.



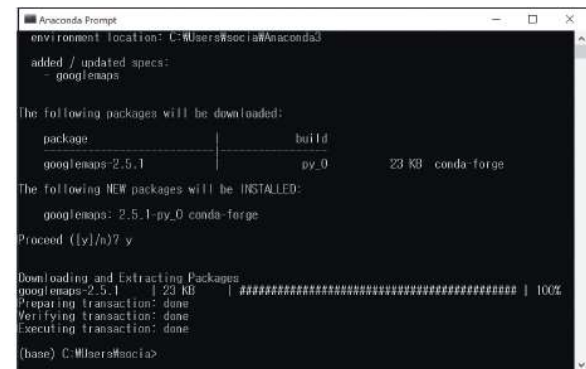
[그림 2-5] 콘다 설치명령 입력

- ③ 설치 계속 여부 확인: 'y' 입력하고 `Enter`를 누른다.



[그림 2-6] 설치 계속 여부 확인

- ④ 설치가 완료된다.



[그림 2-7] 설치 완료

## Part 2. 데이터 입출력

### 3. API 활용하여 데이터 수집하기 (계속)

#### ① 지오코딩 API 호출 결과 미리보기

구글 지오코딩 API를 사용하여 "해운대해수욕장" 위치 정보를 확인하면, 딕셔너리와 비슷한 형태를 갖는다.

예를 들면, 'location'을 키로 하는 데이터 중에서 'lat'와 매칭되는 숫자가 "해운대해수욕장"의 위도를 나타낸다. 경도는 'lng' 값과 매칭되는 숫자다.

〈참고〉 지오코딩 API 호출 결과 미리보기(해운대 해수욕장)

```
1 {'location': {'lat': 35.1586975, 'lng': 129.1603842},
2  'location_type': 'APPROXIMATE',
3  'viewport': {'northeast': {'lat': 35.1678193, 'lng': 129.1763916},
4  'southwest': {'lat': 35.1495747, 'lng': 129.1443768}}}
```

#### ② 지오코딩으로 위도, 경도 정보 가져오기

〈예제 2-6〉 구글 지오코딩 위치 정보

(File: example/part2/2.6\_geocoding\_api.py)

```
1 # -*- coding: utf-8 -*-
2
3 ## google 지오코딩 API를 통해 위도, 경도 데이터 가져오기
4
5 # 라이브러리 가져오기
6 import googlemaps
7 import pandas as pd
8
9 my_key = "-----발급받은 API 키 입력-----"
10
11 # 구글맵스 객체 생성하기
12 maps = googlemaps.Client(key=my_key) # my key값 입력
13
14 lat = [] # 위도
15 lng = [] #경도
16
17 # 장소(또는 주소) 리스트
18 places = ["서울시청", "국립국악원", "해운대해수욕장"]
```

```
20 i=0
21 for place in places:
22     i = i + 1
23     try:
24         print(i, place)
25         # 지오코딩 API 결과값 호출하여 geo_location 변수에 저장
26         geo_location = maps.geocode(place)[0].get('geometry')
27         lat.append(geo_location['location']['lat'])
28         lng.append(geo_location['location']['lng'])
29
30     except:
31         lat.append('')
32         lng.append('')
33         print(i)
34
35 # 데이터프레임으로 변환하기
36 df = pd.DataFrame({'위도':lat, '경도':lng}, index=places)
37 print('\n')
38 print(df)
```

〈실행 결과〉 코드 전부 실행

```
1 서울시청
2 국립국악원
3 해운대해수욕장
```

	위도	경도
서울시청	37.566295	126.977945
국립국악원	37.477759	127.008304
해운대해수욕장	35.158698	129.160384

예제코드 9번 라인에 직접 발급받은 API 키를 입력한다. 3개의 장소("서울시청", "국립국악원", "해운대해수욕장")에 대한 GPS(위도, 경도) 데이터를 2개의 리스트(lat, lng)에 저장한다.

DataFrame() 메소드로 데이터프레임을 만들 때, '위도' 열에 lat 리스트를 매칭하고 '경도' 열에는 lng 리스트를 매칭한다. 장소명이 들어 있는 리스트(places)를 행 인덱스로 설정한다.

## Part 2. 데이터 입출력

### 4. 데이터 저장하기

#### 4-1 CSV 파일로 저장

판다스 데이터프레임은 2차원 배열로 구조화된 데이터이기 때문에 2차원 구조를 갖는 CSV 파일로 변환할 수 있다. 데이터프레임을 CSV 파일로 저장하려면 `to_csv()` 메소드를 적용한다. CSV 파일을 저장할 파일 경로와 파일명(확장자 포함)을 따옴표(" 또는 ') 안에 입력한다.

CSV 파일로 저장: `DataFrame` 객체.`to_csv("파일 이름 (경로)")`

〈예제 2-7〉 CSV 파일로 저장

(File: example/part2/2.7\_to\_csv.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # 판다스 DataFrame() 함수로 데이터프레임 변환. 변수 df에 저장
6 data = {'name' : [ 'Jerry', 'Riah', 'Paul'],
7         'algol' : [ "A", "A+", "B"],
8         'basic' : [ "C", "B", "B+"],
9         'c++' : [ "B+", "C", "C+"],
10        }
11
12 df = pd.DataFrame(data)
13 df.set_index('name', inplace=True) # name 열을 인덱스로 지정
14 print(df)
15
16 # to_csv() 메소드를 사용하여 CSV 파일로 내보내기. 파일명은 df_sample.csv로 저장
17 df.to_csv("./df_sample.csv")
```

〈실행 결과〉 코드 전부 실행 ① - IPython 콘솔에 출력되는 화면

	algol	basic	c++
name			
Jerry	A	C	B+
Riah	A+	B	C
Paul	B	B+	C+

14라인의 `print(df)` 명령에 의해 데이터프레임의 내용이 IPython 콘솔에 출력된다. 한편 17라인은 `to_csv()` 메소드를 적용하여 파이썬 실행 파일이 위치하고 있는 현재 디렉터리(./)에 `"df_sample.csv"`라는 파일명으로 저장하는 명령이다. 저장된 CSV 파일을 메모장 등 편집기로 열어보면 다음과 같이 쉼표와 줄바꿈으로 구분되는 2차원 구조가 확인된다.

〈실행 결과〉 코드 전부 실행 ② - CSV 파일 내용 보기(File: example/part2/df\_sample.csv)

```
name,algol,basic,c++
Jerry,A,C,B+
Riah,A+,B,C
Paul,B,B+,C+
```

#### 4-2 JSON 파일로 저장

데이터프레임을 JSON 파일로 저장하려면 `to_json()` 메소드를 이용한다. JSON 파일의 이름(확장자 포함)을 저장하려는 파일 경로와 함께 따옴표(" 또는 ') 안에 입력한다.

JSON 파일로 저장: `DataFrame` 객체.`to_json("파일 이름 (경로)")`

〈예제 2-8〉 JSON 파일로 저장

(File: example/part2/2.8\_to\_json.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # 판다스 DataFrame() 함수로 데이터프레임 변환. 변수 df에 저장
6 data = {'name' : [ 'Jerry', 'Riah', 'Paul'],
7         'algol' : [ "A", "A+", "B"],
8         'basic' : [ "C", "B", "B+"],
9         'c++' : [ "B+", "C", "C+"],
10        }
11
12 df = pd.DataFrame(data)
13 df.set_index('name', inplace=True) # name 열을 인덱스로 지정
14 print(df)
15
16 # to_json() 메소드를 사용하여 JSON 파일로 내보내기. 파일명은 df_sample.json로 저장
17 df.to_json("./df_sample.json")
```

## Part 2. 데이터 입출력

### 4. 데이터 저장하기

〈실행 결과〉 코드 전부 실행 ① - IPython 콘솔에 출력되는 화면

```
      algol  basic  c++
name
Jerry      A      C      B+
Riah       A+     B      C
Paul       B      B+     C+
```

IPython 콘솔에 `print(df)` 명령에 의해 데이터프레임의 내용이 출력된다. 동시에 `to_json()` 메소드를 이용하여 데이터프레임을 현재 디렉터리(`./`)에 JSON 파일로 변환하여 저장한다. JSON 파일을 열어보면 다음과 같이 데이터프레임의 행, 열이 JSON 파일의 형식에 맞춰 정리된다.

〈실행 결과〉 코드 전부 실행 ② - JSON 파일 내용 보기 (File: example/part2/df\_sample.json)

```
{
  "algol": {"Jerry": "A", "Riah": "A+", "Paul": "B"},
  "basic": {"Jerry": "C", "Riah": "B", "Paul": "B+"},
  "c++": {"Jerry": "B+", "Riah": "C", "Paul": "C+"}
}
```

#### 4-3 Excel 파일로 저장

데이터프레임은 Excel 파일과 아주 유사한 구조를 갖는다. 데이터프레임의 행과 열은 Excel 파일의 행과 열로 일대일로 대응된다. 데이터프레임을 Excel 파일로 저장할 때는 `to_excel()` 메소드를 적용한다. 단, `to_excel()` 메소드를 사용하려면 `openpyxl` 라이브러리를 사전에 설치해야 한다. 아나콘다 배포판에는 `openpyxl` 라이브러리가 기본 제공되므로 따로 설치하지 않아도 된다.

Excel 파일로 저장: `DataFrame` 객체.`to_excel("파일 이름 (경로)")`

〈예제 2-9〉 Excel 파일로 저장

(File: example/part2/2.9\_to\_excel.py)

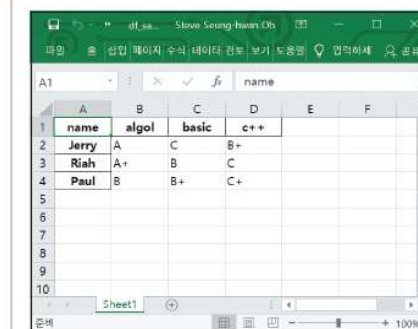
```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # 판다스 DataFrame() 함수로 데이터프레임 변환. 변수 df에 저장
6 data = {'name' : [ 'Jerry', 'Riah', 'Paul'],
7         'algol' : [ "A", "A+", "B"],
8         'basic' : [ "C", "B", "B+"],
9         'c++' : [ "B+", "C", "C+"],
10        }
11
12 df = pd.DataFrame(data)
13 df.set_index('name', inplace=True) # name 열을 인덱스로 지정
14 print(df)
15
16 # to_excel() 메소드를 사용하여 Excel 파일로 내보내기. 파일명은 df_sample.xlsx로 저장
17 df.to_excel("./df_sample.xlsx")
```

〈실행 결과〉 코드 전부 실행 ① - IPython 콘솔에 출력되는 화면

```
      algol  basic  c++
name
Jerry      A      C      B+
Riah       A+     B      C
Paul       B      B+     C+
```

〈실행 결과〉 코드 전부 실행 ② - Excel 파일 내용 보기

(File: example/part2/df\_sample.xlsx)



	name	algol	basic	c++
1	Jerry	A	C	B+
2	Riah	A+	B	C
3	Paul	B	B+	C+



## Part 2. 데이터 입출력

### 4. 데이터 저장하기

#### 4-4 여러 개의 데이터프레임을 하나의 Excel 파일로 저장

데이터프레임 여러 개를 Excel 파일로 저장: `pandas.ExcelWriter( "파일 이름 (경로)" )`

〈예제 2-10〉 ExcelWriter() 활용

(File: example/part2/2.10\_excewriter.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # 판다스 DataFrame() 함수로 데이터프레임 변환. 변수 df1, df2에 저장
6 data1 = {'name': ['Jerry', 'Riah', 'Paul'],
7          'algol': ['A', 'A+', 'B'],
8          'basic': ['C', 'B', 'B+'],
9          'c++': ['B+', 'C', 'C+']}
10
11 data2 = {'c0': [1,2,3],
12          'c1': [4,5,6],
13          'c2': [7,8,9],
14          'c3': [10,11,12],
15          'c4': [13,14,15]}
16
17 df1 = pd.DataFrame(data1)
18 df1.set_index('name', inplace=True) # name 열을 인덱스로 지정
19 print(df1)
20 print('\n')
21
22 df2 = pd.DataFrame(data2)
23 df2.set_index('c0', inplace=True) # c0 열을 인덱스로 지정
24 print(df2)
25
26 # df1을 'sheet1'으로, df2를 'sheet2'로 저장(Excel 파일명은 "df_excelwriter.xlsx")
27 writer = pd.ExcelWriter("./df_excelwriter.xlsx")
28 df1.to_excel(writer, sheet_name="sheet1")
29 df2.to_excel(writer, sheet_name="sheet2")
30 writer.save()
```

〈실행 결과〉 코드 전부 실행 ① - IPython 콘솔에 출력되는 화면

```
algol basic c++
name
Jerry    A    C    B+
Riah     A+   B    C
Paul     B    B+   C+

c1 c2 c3 c4
c0
1   4   7  10 13
2   5   8  11 14
3   6   9  12 15
```

〈실행 결과〉 코드 전부 실행 ② - Excel 파일 내용 보기

(File: example/part2/df\_excelwriter.xlsx)

	A	B	C	D	E	F
1		algol	basic	c++		
2	Jerry	A	C	B+		
3	Riah	A+	B	C		
4	Paul	B	B+	C+		

	A	B	C	D	E	F
1	c0	c1	c2	c3	c4	
2	1	4	7	10	13	
3	2	5	8	11	14	
4	3	6	9	12	15	

판다스 ExcelWriter() 함수는 Excel 워크북 객체를 생성한다. 이 워크북 객체는 우리가 알고 있는 Excel 파일이라고 생각한다. 데이터프레임에 `to_excel()` 메소드를 적용할 때, 삽입하려는 워크북 객체(Excel 파일)를 인자로 전달한다.

`sheet_name` 옵션에 Excel 파일의 시트 이름을 입력하여 삽입되는 시트 위치를 지정할 수 있다. 한편, 데이터프레임을 삽입하는 시트 이름을 다르게 설정하면, 같은 Excel 파일의 서로 다른 시트에 여러 데이터프레임을 구분하여 저장한다.