CHAPTER6.1
Reading and Writing Data in Text Format

2021010702 진서영

목차

read_csy: 데이터를 콤마(,)로 구분, 작성된 파일을 읽어 올 때 사용

read_excel: Excel XLS 또는 XLSX 파일에서 표 형식 데이터 읽기

```
In [8]: !cat examples/ex1.csv
a,b,c,d,message
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

	A	B	C	D	E	F
1	a	b	c	d	message	
2	1	2	3	4	hello	
3	5	6	7	8	world	
4	9	10	11	12	foo	
5						

```
In [3]: import pandas as pd
        a=pd.read_excel('C:/Users/run07/OneDrive/data1/ex1.cell.xlsx')

In [4]: a

Out[4]:
```

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

header=None : 칼럼 값이 없는 데이터프레임에 대해 저장하거나 불러올 때 사용
names=[] :데이터프레임을 읽어올 때 칼럼 값을 지정해주고 싶을 때 사용

	A	B	C	D	E
1	0	1	2	3	4
2	1	2	3	4	hello
3	5	6	7	8	world
4	9	10	11	12	foo

```
In [5]: import pandas as pd
        a=pd.read_excel('C:/Users/run07/OneDrive/data1/ex2.cell.xlsx',header=None)

In [6]: a
```

Out[6]:

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	hello
2	5	6	7	8	world
3	9	10	11	12	foo

```
In [7]: a=pd.read_excel('C:/Users/run07/OneDrive/data1/ex2.cell.xlsx',names=['a','b','c','d','message'])

In [8]: a
```

Out[8]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

index_col=[] : 테이블 내의 특정한 열을 행 인덱스로 지정하고 싶을 때 사용

In [4]:
```
import pandas as pd
parsed = pd.read_csv('C:/Users/run07/Downloads/pydata-book-2nd-edition/pydata-book-2nd-edition/examples/csv_mindex.csv',
                     index_col=['key1', 'key2'])
parsed
```

Out[4]:

		value1	value2
key1	key2		
one	a	1	2
	b	3	4
	c	5	6
	d	7	8
two	a	9	10
	b	11	12
	c	13	14
	d	15	16

read_table:데이터를 '\s+'로 구분

ex3 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
a b c
aaa 1 2 3
bbb 4 5 6
ccc 7 8 9
ddd 10 11 12
```

```
In [16]: list(open('C:/Users/run07/OneDrive/data1/ex3.txt'))
Out[16]: ['a b c#n', 'aaa 1 2 3#n', 'bbb 4 5 6#n', 'ccc 7 8 9#n', 'ddd 10 11 12']

In [17]: import pandas as pd
         result=pd.read_table('C:/Users/run07/OneDrive/data1/ex3.txt',sep='#s+')

In [18]: result
Out[18]:
```

	a	b	c
aaa	1	2	3
bbb	4	5	6
ccc	7	8	9
ddd	10	11	12

skiprows=[] :특정한 행만 불러오고 싶을 때 사용/원하는 곳에 있는 데이터만 취득 가능
>불러오고 싶지 않은 행을 입력하면 됨

	A	B	C	D	E
1	a	b	c	d	message
2	1	2	3	4	hello
3	5	6	7	8	world
4	9	10	11	12	foo

```
In [2]: import pandas as pd
        a=pd.read_excel('C:/Users/run07/OneDrive/data1/ex4.cell.xlsx')

In [3]: a
```

Out[3]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

```
In [4]: a=pd.read_excel('C:/Users/run07/OneDrive/data1/ex4.cell.xlsx',skiprows=[0,2,3])

In [5]: a
```

Out[5]:

1	2	3	4	hello

.isnull(): 값이 있으면 False, 값이 없으면 True라고 표기

	A	B	C	D	E	F	G
1	something	a	b	c	d	message	
2	one	1	2	3	4	NA	
3	two	5	6		8	world	
4	three	9	10	11	12	foo	
5							

```
In [27]: import pandas as pd
         result=pd.read_excel('C:/Users/run07/OneDrive/data1/ex5.xlsx')

In [28]: result
```

Out[28]:

	something	a	b	c	d	message
0	one	1	2	3.0	4	NaN
1	two	5	6	NaN	8	world
2	three	9	10	11.0	12	foo

```
In [29]: pd.isnull(result)
```

Out[29]:

	something	a	b	c	d	message
0	False	False	False	False	False	True
1	False	False	False	True	False	False
2	False	False	False	False	False	False

Reading Text Files in Pieces

pd.options.display.max_rows = 10: 표를 출력할 때 최대 행 수
> 최대 10행 까지 출력 가능

In [6]:
```
import pandas as pd
pd.options.display.max_rows = 10
```

In [7]:
```
result = pd.read_csv('C:/Users/run07/Downloads/ex6.csv')
result
```

Out[7]:

	one	two	three	four	key
0	0.467976	-0.038649	-0.295344	-1.824726	L
1	-0.358893	1.404453	0.704965	-0.200638	B
2	-0.501840	0.659254	-0.421691	-0.057688	G
3	0.204886	1.074134	1.388361	-0.982404	R
4	0.354628	-0.133116	0.283763	-0.837063	Q
...	...	...	...	...	...
9995	2.311896	-0.417070	-1.409599	-0.515821	L
9996	-0.479893	-0.650419	0.745152	-0.646038	E
9997	0.523331	0.787112	0.486066	1.093156	K
9998	-0.362559	0.598894	-1.843201	0.887292	G
9999	-0.096376	-1.012999	-0.657431	-0.573315	0

10000 rows × 5 columns

nrows : 파일 전체를 읽는 대신 처음 몇 줄만 읽어보고 싶을 때 사용

```
In [5]: a=pd.read_csv('C:/Users/run07/Downloads/ex6.csv', nrows=5)
```

```
In [6]: a
```

Out[6]:

	one	two	three	four	key
0	0.467976	-0.038649	-0.295344	-1.824726	L
1	-0.358893	1.404453	0.704965	-0.200638	B
2	-0.501840	0.659254	-0.421691	-0.057688	G
3	0.204886	1.074134	1.388361	-0.982404	R
4	0.354628	-0.133116	0.283763	-0.837063	Q

chunksize: 한번에 가져올 데이터 양
> 대용량 csv 데이터 다룰 때 사용

```
In [8]: chunker = pd.read_csv('C:/Users/run07/Downloads/ex6.csv', chunksize=1000)
        chunker

Out[8]: <pandas.io.parsers.TextFileReader at 0x2a11c12beb0>
```

```
In [12]: chunker = pd.read_csv('C:/Users/run07/Downloads/ex6.csv', chunksize=1000)

         tot = pd.Series([])
         for piece in chunker:
             tot = tot.add(piece['key'].value_counts(), fill_value=0)

         tot = tot.sort_values(ascending=False)

<ipython-input-12-dca2b3e9ddf5>:3: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in
a future version. Specify a dtype explicitly to silence this warning.
  tot = pd.Series([])
```

Writing Data to Text Format

sep=‘,’ : 콤마(,)로 데이터 구분
sep=‘|’ : ‘|’로 데이터 구분
sys.stdout : sys.stdout 에 기록해서 텍스트 결과를 콘솔에 출력

In [13]:
```
import pandas as pd
data = pd.read_csv('C:/Users/run07/Downloads/ex5.csv')
data
```

Out[13]:

	something	a	b	c	d	message
0	one	1	2	3.0	4	NaN
1	two	5	6	NaN	8	world
2	three	9	10	11.0	12	foo

In [25]:
```
data.to_csv(sep=',')
```

Out[25]:
```
',something,a,b,c,d,message₩r₩n0,one,1,2,3.0,4,₩r₩n1,two,5,6,,8,world₩r₩n2,three,9,10,11.0,12,foo₩r₩n'
```

In [26]:
```
import sys
data.to_csv(sys.stdout, sep='|')
```

```
|something|a|b|c|d|message
0|one|1|2|3.0|4|
1|two|5|6||8|world
2|three|9|10|11.0|12|foo
```

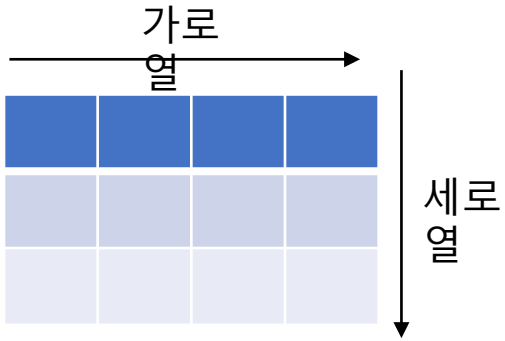
na_rep : 결과에서 누락된 값은 기본적으로 비어있는 문자열로 나타냄
[na_rep='NULL']과 같이 원하는 값으로 지정 가능

```
In [27]: data.to_csv(sys.stdout, na_rep='NULL')

,something,a,b,c,d,message
0,one,1,2,3.0,4,NULL
1,two,5,6,NULL,8,world
2,three,9,10,11.0,12,foo
```

header : True 면 header 출력
Index : True 면 Index 출력

```
In [29]: data.to_csv(sys.stdout, index=False, header=False)

something,a,b,c,d,message
one,1,2,3.0,4,
two,5,6,,8,world
three,9,10,11.0,12,foo
```

columns : 세로 열 이름
Index: 가로 열 이름




```
In [32]: data.to_csv(sys.stdout, index=False, columns=['a', 'b', 'c'])

a,b,c
1,2,3.0
5,6,
9,10,11.0
```

date_range(시작 날짜, period)
period= : 시작 날짜부터 원하는 개수만큼 날짜 생성

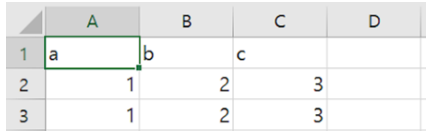
In [39]:
```
dates = pd.date_range('1/1/2000', periods=7)
ts = pd.Series(np.arange(7), index=dates)
ts.to_csv('C:/Users/run07/Downloads/tseries.csv')
a=pd.read_csv('C:/Users/run07/Downloads/tseries.csv')
a
```

Out[39]:

	Unnamed: 0	0
0	2000-01-01	0
1	2000-01-02	1
2	2000-01-03	2
3	2000-01-04	3
4	2000-01-05	4
5	2000-01-06	5
6	2000-01-07	6

Working with Delimited Formats

import csv
:단일 문자 구분 기호가 있는 모든 파일은 사용가능
사용하려면 열려 있는 파일이나 파일 같은 개체를 csv.reader 로 전달



	A	B	C	D
1	a	b	c	
2	1	2	3	
3	1	2	3	

In [18]:
```
import csv
f = open('C:/Users/run07/Downloads/ex7.csv')
reader = csv.reader(f)
```

In [19]:
```
for line in reader:
    print(line)
```

```
['a', 'b', 'c']
['1', '2', '3']
['1', '2', '3']
```

In [42]:
```
with open('C:/Users/run07/Downloads/ex7.csv') as f:
    lines = list(csv.reader(f))
```

In [43]:
```
header, values = lines[0], lines[1:]
```
#header line과 data lines로 분할

In [44]:
```
data_dict = {h: v for h, v in zip(header, zip(*values))}
data_dict
```

Out[44]:
```
{'a': ('1', '1'), 'b': ('2', '2'), 'c': ('3', '3')}
```

JSON Data

Import json
JSON: ‘Java Script Object Notification’의 줄임말
> 효율적으로 데이터를 저장하고 교환 (데이터 교환)하는데 사용하는 텍스트 데이터 포맷 중 하나

json.load : str 전처리 필요없이 바로 데이터에 접근해서 사용가능
> JSON 문자열을 파이썬 객체로 바꾸어 줌

json.dumps : 파이썬 객체를 JSON 문자열로 바꾸어 줌

In [3]:
```
obj = """
{"name": "Wes",
 "places_lived": ["United States", "Spain", "Germany"],
 "pet": null,
 "siblings": [{"name": "Scott", "age": 30, "pets": ["Zeus", "Zuko"]},
              {"name": "Katie", "age": 38,
               "pets": ["Sixes", "Stache", "Cisco"]}]
}
"""
```

In [4]:
```
import json
result = json.loads(obj)
result
```

Out[4]:
```
{'name': 'Wes',
 'places_lived': ['United States', 'Spain', 'Germany'],
 'pet': None,
 'siblings': [{'name': 'Scott', 'age': 30, 'pets': ['Zeus', 'Zuko']},
  {'name': 'Katie', 'age': 38, 'pets': ['Sixes', 'Stache', 'Cisco']}]}
```

In [5]:
```
asjson = json.dumps(result)
```

In [7]:
```
import pandas as pd
siblings = pd.DataFrame(result['siblings'], columns=['name', 'age'])
siblings
```

Out[7]:

	name	age
0	Scott	30
1	Katie	38

to_json() : pandas에서 JSON으로 데이터를 내보내야 하는 경우 Series 및 Data에서 사용

In [8]:
```
import pandas as pd
data = pd.read_json('C:/Users/run07/Downloads/example.json')
data
```

Out[8]:

	a	b	c
0	1	2	3
1	4	5	6
2	7	8	9

In [9]:
```
print(data.to_json())
print(data.to_json(orient='records'))
```

```
{"a":{"0":1,"1":4,"2":7},"b":{"0":2,"1":5,"2":8},"c":{"0":3,"1":6,"2":9}}
[{"a":1,"b":2,"c":3},{"a":4,"b":5,"c":6},{"a":7,"b":8,"c":9}]
```