

파일 처리

일정한 간격 텍스트 파일 읽기

❖ 클립보드의 내용을 읽어오기

`read_clipboard()`

❖ 일정한 간격을 갖는 텍스트 파일 읽기

- ✓ 일정한 간격 또는 고정된 폭 구조의 외부데이터를 불러와서 DataFrame을 리턴하는 함수

```
read.fwf('파일경로',  
widths = (글자 사이의 간격),  
names = (컬럼이름))
```

- ✓ 매개변수
 - encoding: 인코딩 방식

일정한 간격 텍스트 파일 읽기

```
import pandas as pd
from pandas import DataFrame, Series
import numpy as np
frame = pd.read_fwf('data_fwf.txt', widths=(10, 2, 5),
names=('date','name','price'),encoding='ms949')
print(frame)
```

date	name	price
2017-04-10	다음	32000
2017-04-11	다음	34000
2017-04-12	다음	33000

❖ data_fwf.txt

```
2017-04-10다음32000
2017-04-11다음34000
2017-04-12다음33000
```

CSV 파일 읽기

❖ CSV 파일 읽기

- ✓ `pandas.read_csv('파일경로')`, `pandas.read_table('파일경로')`: 파일경로에 있는 파일을 가지고 DataFrame 객체를 생성해서 리턴해줍니다.
- ✓ `read_csv`는 기본 구분자로 ,를 사용하고 `table`은 탭을 사용합니다.
- ✓ 옵션 설정이 없으면 첫 행의 데이터가 컬럼의 이름이 됩니다.

CSV 파일 읽기

❖ item.csv

code,manufacture,name,price

1,korea,apple,1500

2,korea,watermelon,15000

3,korea,oriental melon,1000

4,philippines,banana,500

5,korea,lemon,1500

6,korea,mango,700

CSV 파일 읽기

```
import pandas as pd
from pandas import DataFrame, Series
import numpy as np

items = pd.read_csv('item.csv')
print(items)
```

	code	manufacture	name	price
0	1	korea	apple	1500
1	2	korea	watermelon	15000
2	3	korea	oriental melon	1000
3	4	philippines	banana	500
4	5	korea	lemon	1500
5	6	korea	mango	700



CSV 파일 읽기

❖ CSV 옵션

- ✓ sep: 구분자 설정
- ✓ header: 컬럼이름의 행 번호로 기본은 0이며 없을 때는 None
- ✓ index_col: 인덱스로 사용할 컬럼 번호나 이름 또는 리스트
- ✓ names: 컬럼 이름으로 사용할 리스트, header=None과 함께 사용
- ✓ skiprows : 파일의 시작부터 읽지 않을 로우의 개수 또는 무시할 로우 번호가 담긴 리스트
- ✓ na_values : NA값으로 처리할 값들의 나열
- ✓ comment : 주석으로 분류되어 파싱하지 않을 문자 혹은 문자열
- ✓ converters : 변환 시 칼럼에 적용할 함수를 지정

CSV 파일 읽기

- ✓ `nrows` : 파일의 첫 일부만 읽어올 때 처음 몇 줄을 읽을 것인지 지정
- ✓ `skip_footer` : 무시할 파일의 마지막 줄 수
- ✓ `encoding` : 인코딩 종류를 지정 - UTF-8로 인코딩 된 텍스트일 경우 'utf-8'로 지정
- ✓ `squeeze` : 로우가 하나뿐이라면 Series객체를 반환 기본값은 False
- ✓ `thousands` : 숫자를 천 단위로 끊을 때 ',' 나 '.' 같은 구분자

CSV 파일 읽기

❖ good.csv

apple,10,1500

banana,5,15000

melon,7,1000

kiwi,20,500

mango,30,1500

orange,4,700



CSV 파일 읽기

```
import pandas as pd
from pandas import DataFrame, Series
import numpy as np

items = pd.read_csv('good.csv',
                    header=None,
                    names=['제품명', '개수', '가격'],
                    thousands=',')
print(items['개수'].cumsum()) # 개수 누적 합
print('=====')
print(items.describe()) #데이터 요약
```

```
0    10
1    15
2    22
3    42
4    72
5    76
```

Name: 개수, dtype: int64

```
=====
```

	개수	가격
count	6.000000	6.000000
mean	12.666667	3366.666667
std	10.269697	5713.726163
min	4.000000	500.000000
25%	5.500000	775.000000
50%	8.500000	1250.000000
75%	17.500000	1500.000000
max	30.000000	15000.000000

CSV 파일 읽기

❖데이터의 양이 많을 때

- ✓ nrows 속성을 이용해서 시작위치에서 일부분의 데이터만 읽어오고 skiprows를 이용해서 읽을 수 있습니다.
- ✓ chunksize를 이용해서 한번에 읽을 데이터의 개수를 설정하면 TextParser 객체가 리턴되는데 이 객체를 순회하면 데이터를 읽을 수 있습니다.

CSV 파일 읽기

```
import pandas as pd
from pandas import DataFrame, Series
import numpy as np
#데이터를 2개 단위로 읽을 수 있는 parser 객체 생성
parser = pd.read_csv('good.csv', header=None, chunksize=2)
#parser객체를 순회하기
for piece in parser:
    print(piece.sort_values(by=2, ascending=False))#2번째 컬럼의 내림차순 정렬
    print("=====")
0  1  2
1 banana 5 15000
0 apple 10 1500
=====
0  1  2
2 melon 7 1000
3 kiwi 20 500
```

csv.reader

❖ csv.reader

- ✓ 문자열이 '"', '"' 로 감싸져 있거나 구분 문자가 2개 이상의 문자열인 경우 pandas의 라이브러리로는 읽을 수가 없습니다.
- ✓ 이 경우에는 csv.reader를 이용해서 줄 단위로 데이터를 읽은 후 frame으로 저장해야 합니다.
- ✓ csv.reader의 매개변수가 파일 객체라서 open 함수를 이용해서 파일을 객체를 만든 후 매개변수로 대입해 주어야 합니다.
- ✓ 이 때 옵션으로 delimiter로 구분 문자열을 대입할 수 있습니다.

csv.reader

fruit.csv

"apple"|10|1500

"banana"|5|15000

"orange"|20|500

"pear"|30|1500

"kiwi"|7|1000

"mango"|4|700



CSV 파일 읽기

```
import pandas as pd
import csv
from pandas import DataFrame, Series
import numpy as np

items = pd.read_csv("fruit.csv", header=None)
print(items)

print("=====")
lines = list(csv.reader(open("fruit.csv"), delimiter='|'));
print(lines)

print("=====")
frame = DataFrame(lines, columns=['name', 'count', 'price'])
print(frame)
```

CSV 파일 읽기

```
0
0 apple|10|1500
1 banana|5|15000
2 orange|20|500
3 pear|30|1500
4 kiwi|7|1000
5 mango|4|700
=====
[['apple', '10', '1500'], ['banana', '5', '15000'], ['orange', '20',
'500'], ['pear', '30', '1500'], ['kiwi', '7', '1000'], ['mango', '4',
'700']]
    name count price
0 apple    10  1500
1 banana     5 15000
2 orange    20   500
3 pear     30  1500
4 kiwi      7  1000
5 mango     4   700
```


CSV 파일 저장

❖ csv 파일 저장

- ✓ Series 나 DataFrame객체가 to_csv()를 호출하고 매개변수로 파일 경로를 대입하면 됩니다.
- ✓ sep 옵션을 이용해서 구분자를 설정할 수 있습니다.
- ✓ na_rep 옵션을 이용해서 NaN 값을 원하는 형식으로 출력할 수 있습니다.
- ✓ index 나 header에 False를 대입하면 인덱스와 컬럼이름은 출력되지 않습니다.
- ✓ cols 옵션을 이용해서 필요한 컬럼만 저장할 수 있습니다.

CSV 파일 저장

```
import pandas as pd
from pandas import DataFrame, Series
import numpy as np
```

```
items = {'apple':{'count':10,'price':1500},
         'banana': {'count':5, 'price': 15000},
         'melon': { 'count':7,'price': 1000},
         'kiwi': {'count':20,'price': 500},
         'mango': {'count':30,'price': 1500},
         'orange': { 'count':4,'price': 700}}
```

```
data = DataFrame(items)
```

```
data = data.T
```

```
data.to_csv("data.csv", index=False, header=False)
```

10,1500
5,15000
20,500
30,1500
7,1000
4,700

Excel 파일

❖엑셀 파일 열기

- ✓ `xl = pandas.ExcelFile("엑셀 파일 경로")`
- ✓ `DataFrame변수 = pandas.parse("시트 이름")`
- ✓ `DataFrame변수 = pandas.io.excel.read_excel("엑셀 파일 경로, sheetname=시트이름")`

❖엑셀 파일 저장

```
writer = pandas.ExcelWriter( ' simple.xlsx ' , engine= ' xlsxwriter ' )  
DataFrame.to_excel(writer, sheet_name='Sheet1')  
writer.save()
```

Excel 파일

번호	이름	1과목	2과목	3과목
1	이효준	40		45
2	김성희	42	55	50
3	이연숙		60	55
4	노화현	50	70	60
5	최희순	55	75	
6	정원주	60	80	70

Excel 파일

```
import pandas as pd
import csv
from pandas import DataFrame, Series
import numpy as np

f = pd.ExcelFile('excel.xlsx')
frame = f.parse("Sheet1",index_col=0)
print(frame)

print("=====")
df = pd.io.excel.read_excel("excel.xlsx",
    sheetname='Sheet1',index_col=0)
print(df)
```

이름 1과목 2과목 3과목
번호

1	이효준	40.0	NaN	45.0
2	김성희	42.0	55.0	50.0
3	이연숙	NaN	60.0	55.0
4	노화현	50.0	70.0	60.0
5	최희순	55.0	75.0	NaN
6	정원주	60.0	80.0	70.0

=====

이름 1과목 2과목 3과목
번호

1	이효준	40.0	NaN	45.0
2	김성희	42.0	55.0	50.0
3	이연숙	NaN	60.0	55.0
4	노화현	50.0	70.0	60.0
5	최희순	55.0	75.0	NaN
6	정원주	60.0	80.0	70.0

MySQL

- <https://downloads.mariadb.org>



MySQL 연동

❖MySQL 모듈 설치

❖<https://downloads.mariadb.org>

C:\Users\Wkitcoop\Anaconda3\Scripts>pip install pyMySQL

pip install pyMySQL

❖데이터베이스 접속 확인

```
import pymysql
```

```
#데이터베이스 연결
```

```
con = pymysql.connect(host='localhost', port=3306,  
                      user='root', passwd='11111',  
                      db='test', charset = 'utf8')
```

```
print(con)
```

```
con.close()
```


MySQL 연동

❖데이터베이스 접속이 되지 않는 경우 아래와 같은 예외 발생

Traceback (most recent call last):

```
File "C:\Users\Administrator\python\test\__main__.py", line 12, in <module>
exception: (<class 'pymysql.err.OperationalError'>, OperationalError(2003, "Can't
connect to MySQL server on '211.183.2.253' ([WinError 10060] 연결된 구성원으로
부터 응답이 없어 연결하지 못했거나, 호스트로부터 응답이 없어 연결이 끊어졌습니
다)"), <traceback object at 0x00C839E0>)
```

```
if con != None:
```

```
NameError: name 'con' is not defined
```

maria db 기본 명령어

❖ 기본명령어..

- show databases;
- use test; 데이터베이스 선택
- show tables;



MySQL 연동

❖MySQL에 접속해서 테이블 생성

```
create table contact(  
    num int primary key auto_increment,  
    name varchar(100) not null,  
    phone varchar(20));
```

❖데이터 삽입 및 확인

```
insert into contact(name, phone) values('park', '01037901997')  
commit;  
select * from contact;
```

MySQL 연동

❖파이썬에서 삽입 및 삭제 또는 갱신

1. 연결 객체의 `cursor()` 메소드를 호출해서 sql 실행 객체를 가져옵니다.
2. `execute(실행 할 sql문장)`
3. 연결 객체의 `commit()` 을 호출하면 작업 내용이 반영되고 `rollback()`을 호출하면 작업 취소

MySQL 연동

❖ 데이터 삽입

```
import sys, pymysql
try:
    con = pymysql.connect(host='211.183.2.253',
                           port=3306, user='root', passwd='wnddkd', db='mysql', charset
                           ='utf8')
    cursor = con.cursor()
    cursor.execute("insert into contact(name, phone) values('박문석', '01037901997') ")
    con.commit()
    print("삽입 성공")
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```

MySQL 연동

❖ **execute** 안의 **sql** 문장을 아래처럼 수정하고 데이터베이스 확인

```
cursor.execute("update contact set phone='01031391997' where name = '박문석'")
```

```
cursor.execute("delete from contact where name = '박문석'")
```



MySQL 연동


❖파이썬에서 데이터 검색

1. 연결 객체의 cursor() 메소드를 호출해서 sql 실행 객체를 가져옵니다.
2. execute(실행 할 sql문장)
3. cursor 객체를 가지고 fetchall 메소드를 호출하면 튜플들의 튜플로 결과가 리턴되며 fetchone 메소드를 호출하면 첫번째 데이터 1개만 튜플로 리턴됩니다.

MySQL 연동

❖ 데이터 1개 검색

```
import sys, pymysql
try:
    con = pymysql.connect(host='211.183.2.253',
                           port=3306, user='root', passwd='wnddkd', db='mysql',
                           charset='utf8')
    cursor = con.cursor()
    cursor.execute("select * from contact")
    data = cursor.fetchone()
    for imsi in data:
        print(imsi)
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```




MySQL 연동

❖ 데이터 여러 개 검색

```
import sys, pymysql
try:
    con = pymysql.connect(host='211.183.2.253',
                           port=3306, user='root', passwd='wnddkd', db='mysql', charset
                           ='utf8')
    cursor = con.cursor()

    cursor.execute("select * from contact")
    data = cursor.fetchall()
    for imsi in data:
        print(imsi)
except:
    print('exception:', sys.exc_info())
finally:
    con.close()
```



MySQL 연동

❖ `select * from contact;`

num	name	phone
1	park	01037901997
2	박문석	01037901997

MySQL 연동

❖ pandas의 DataFrame으로 만들기

```
import pymysql, sys
from pandas import Series, DataFrame
import pandas as pd
import numpy as np

con = pymysql.connect(host='211.183.2.253', port=3306,
                      user='root', passwd='wnddkd',
                      db='user30', charset='utf8')

list = []
```

MySQL 연동

```
list = []  
try:  
    cursor = con.cursor()  
    cursor.execute("select * from contact")  
    data = cursor.fetchall()  
    for imsi in data:  
        list.append(imsi)  
except:  
    print('exception:', sys.exc_info())  
finally:  
    con.close()  
frame = DataFrame(list, columns=['번호', '이름', '전화번호'])  
print(frame)
```

MongoDB

MongoDB 연동

❖ MongoDB 다운로드

<https://www.mongodb.com/download-center#community>

❖ MySQL 모듈 설치

```
pip install pymongo
```

❖ 데이터베이스 접속 확인

```
import pymongo  
con = pymongo.MongoClient("211.183.2.253", 27017)  
print(con)
```

MongoDB 연동

	name	count	price
0	apple	10	1500
1	banana	4	500
2	orange	13	1000
3	kiwi	22	1300
4	melon	16	2500
5	mango	4	550



MongoDB 연동

```
import pymongo
import pandas as pd
from pandas import Series, DataFrame
con = pymongo.MongoClient("211.183.2.253", 27017)
db = con.sample
#데이터 삽입
#db.my_collection.insert_one({"name": "apple", "count":10, "price":1500})
#db.my_collection.insert_one({"name": "banana", "count":4, "price":500})
#db.my_collection.insert_one({"name": "orange", "count":13, "price":1000})
#db.my_collection.insert_one({"name": "kiwi", "count":22, "price":1300})
#db.my_collection.insert_one({"name": "melon", "count":16, "price":2500})
#db.my_collection.insert_one({"name": "mango", "count":4, "price":550})
```


MongoDB 연동

#데이터 가져오기

```
#print(db.my_collection.find({}))
```

#데이터를 list로 변환하기

```
#print(list(db.my_collection.find({})))
```

```
li = list(db.my_collection.find({}))
```

#프레임으로 만들기

```
df = pd.DataFrame(li, columns=['name', 'count', 'price'])
```

```
print(df)
```

웹에서 가져오기

웹에서 가져오기

❖웹에서 get 방식 요청

```
params = {'param1': 'value1', 'param2': 'value'}
```

```
res = requests.get(URL, params=params)
```

❖웹에서 post 방식 요청은 get 메서드를 post로 변경

웹에서 가져오기

❖ 웹에서 문자열 읽어오기

```
import requests
```

```
url = 'http://www.naver.com'
```

```
data = requests.get(url)
```

```
print(data.text)
```



HTML

❖HTML

- ✓ 웹에서 가져온 문자열을 태그 형식의 트리 형식으로 변환:
`lxml.html.parse(io.StringIO(문자열))`
- ✓ 루트 객체 가져오기: `트리객체.getRoot()`
- ✓ 태그에 해당하는 데이터를 Element의 list로 가져오기: `루트객체.findall('.//태그명')`
- ✓ `루트객체.find("from")`은 루트객체 태그 하위에 from과 일치하는 첫 번째 태그를 찾아서 리턴하고, 없으면 None을 리턴하며 `루트객체.findall(" from ")`은 루트객체 태그 하위에 from과 일치하는 모든 태그를 리스트로 리턴하고 `루트객체.findtext("from")`은 루트객체 태그 하위에 from과 일치하는 첫 번째 태그의 텍스트 값을 리턴한다.
- ✓ Element의 `get('속성')`: 속성 값 가져오기
- ✓ Element의 `text_content()`: 태그 안의 내용 가져오기

HTML

❖ http://finance.daum.net/quote/kospi_yyyymmdd.daum

일자별								
단위: %, 천주, 백만원								
일자	종가	전일비	등락률	거래량	거래대금	개인(억)	외국인(억)	기관(억)
17.04.20	2,148.35	▲ 9.95	+0.47%	190,903	3,317,208	-1,856	+859	+1,152
17.04.19	2,138.40	▼ 10.06	-0.47%	344,221	4,892,976	+1,159	-2,262	+1,063
17.04.18	2,148.46	▲ 2.70	+0.13%	291,399	3,971,074	-482	+97	+344
17.04.17	2,145.76	▲ 10.88	+0.51%	259,303	3,750,130	-1,609	-1,145	+2,602
17.04.14	2,134.88	▼ 13.73	-0.64%	383,688	3,408,417	+444	-755	+203
17.04.13	2,148.61	▲ 19.70	+0.93%	357,582	4,396,072	-1,137	-18	+1,139
17.04.12	2,128.91	▲ 5.06	+0.24%	427,358	3,970,631	+247	+741	-1,088
17.04.11	2,123.85	▼ 9.47	-0.44%	489,309	4,209,403	+313	-1,275	+985
17.04.10	2,133.32	▼ 18.41	-0.86%	512,916	4,050,058	+64	-541	-166
17.04.07	2,151.73	▼ 1.02	-0.05%	408,456	4,143,350	+104	-790	+849

1 2 3 4 5 6 7 8 9 10 11~20 >

HTML

일자	종가	전일비	등락률	거래량	거래대금	개인(억)	외국인
(억) ₩							
0 17.04.20	2,148.50	▲10.10	+0.47%	189,122	3,281,223	-1,840	+831
1 17.04.19	2,138.40	▼10.06	-0.47%	344,221	4,892,976	+1,159	-2,262
2 17.04.18	2,148.46	▲2.70	+0.13%	291,399	3,971,074	-482	+97
3 17.04.17	2,145.76	▲10.88	+0.51%	259,303	3,750,130	-1,609	-1,145
4 17.04.14	2,134.88	▼13.73	-0.64%	383,688	3,408,417	+444	-755
5 17.04.13	2,148.61	▲19.70	+0.93%	357,582	4,396,072	-1,137	-18
6 17.04.12	2,128.91	▲5.06	+0.24%	427,358	3,970,631	+247	+741
7 17.04.11	2,123.85	▼9.47	-0.44%	489,309	4,209,403	+313	-1,275
8 17.04.10	2,133.32	▼18.41	-0.86%	512,916	4,050,058	+64	-541
9 17.04.07	2,151.73	▼1.02	-0.05%	408,456	4,143,350	+104	-790

HTML


```
import pandas as pd
from pandas import Series, DataFrame
import numpy as np
import requests
from lxml.html import parse
from io import StringIO

url = 'http://finance.daum.net/quote/kospi_yyyymmdd.daum?page=1'
data = requests.get(url) #응답받을 response 객체 만들기
doc = parse(StringIO(data.text))

root = doc.getroot()
```


HTML

```
tables = root.findall('.//table')
titles = tables[0].findall(".//th")
cols = []
for title in titles:
    cols.append(title.text_content())
contents = tables[0].findall(".//td")
values = []
for title in contents:
    if title.text_content() != "":
        values.append(title.text_content())
ar = np.array(values)
ar = ar.reshape(10,9)
df = DataFrame(ar, columns=cols)
print(df)
```

A close-up, slightly blurred photograph of a silver fountain pen lying diagonally across a document. The document features a table with several columns and rows of text, though the text is not legible. The pen's nib is visible in the lower-left corner of the image area.

XML

❖ XML 파싱

- ✓ xml 파일 읽기 – local file

```
import xml.etree.ElementTree as ET
```

```
doc = ET.parse(file_name)
```

```
root = doc.getroot()
```

- ✓ url 읽기

```
url = "
```

```
request = urllib.request.Request(url)
```

```
response = urllib.request.urlopen(request)
```

```
tree = ET.parse(response)
```

```
Root = tree.getroot()
```

XML

❖ <https://apis.daum.net/local/v1/search/category.xml?apikey=465b06fae32febacbc59502598dd7685&code=AT4&location=37.514322572335935,127.06283102249932&radius=20000>

```
<channel>
<info>
<count>15</count>
<totalCount>45</totalCount>
<page>1</page>
</info>
<item>
<direction>북</direction>
<title>서울어린이대공원</title>
<addressBCode>1121510200</addressBCode>
<placeUrl>http://place.map.daum.net/7996800</placeUrl>
<id>7996800</id>
<distance>4288</distance>
<imageUrl>http://t1.daumcdn.net/cfile/1154FD3A4EBA26E02D</imageUrl>
<newAddress/>
```

XML


title	latitude	longitude
0	서울어린이대공원	37.55013887884033 127.08104900377838
1	신사동가로수길	37.52115586945625 127.02283697718428
2	서울어린이대공원 동물원	37.54823869581357 127.08231889202375
3	롯데월드	37.51132165580067 127.09806802140692
4	압구정로데오거리	37.52675582301717 127.03915202952251
5	응봉산	37.54825280891864 127.02983415604075
6	세빛섬	37.512068671570425 126.99561141627026
7	어린이회관 눈썰매장	37.54595445177209 127.08132510752537
8	아차산	37.570580882750875 127.1038190615064
9	청계천	37.56914696867927 126.97864706815112
10	덕수궁	37.565553059115096 126.97489475861694
11	창경궁	37.579416538449586 126.9951999117328
12	창덕궁	37.58179135743142 126.9916086357791
13	서울랜드	37.43379106268724 127.01897326939677
14	석촌호수 서호	37.5076807262772 127.09911283700599

XML

```
import pandas as pd
from pandas import Series, DataFrame
import numpy as np
import requests
import xml.etree.ElementTree as ET
import urllib.request
url =
'https://apis.daum.net/local/v1/search/category.xml?apikey=465b06fae32febacbc5
9502598dd7685&code=AT4&location=37.514322572335935,127.06283102249932
&radius=20000'
request = urllib.request.Request(url)
response = urllib.request.urlopen(request)
```

XML

```
tree = ET.parse(response)
root = tree.getroot()
items = root.findall('item')
ar = []
for imsi in items:
    item = {}
    item['title'] = imsi.find('title').text
    item['latitude'] = imsi.find('latitude').text
    item['longitude'] = imsi.find('longitude').text
    ar.append(item)
df = DataFrame(ar, columns= ['title', 'latitude', 'longitude'])
print(df)
```



JSON

❖JSON

- ✓ 속성-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 텍스트를 사용하는 개방형 표준 포맷
- ✓ 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법 중의 하나
- ✓ 본래는 자바스크립트 언어로부터 파생되어 자바스크립트의 구문 형식을 따르지만 언어 독립형 데이터 포맷
- ✓ 프로그래밍 언어나 플랫폼에 독립적이므로, 구문 분석 및 JSON 데이터 생성을 위한 코드는 C, C++, C#, 자바, 자바스크립트, 펄, 파이썬 등 수많은 프로그래밍 언어에서 쉽게 이용
- ✓ 파이썬에서는 `json.loads(문자열)`을 이용하면 파싱된 결과가 리턴됩니다.
- ✓ 데이터 표현 방법

배열 : [], 객체:{속성:값, 속성:값....}, 배열+객체:[{속성:값},{속성:값}]

JSON

❖ <https://apis.daum.net/local/v1/search/category.json?apikey=465b06fae32febacbc59502598dd7685&code=AT4&location=37.514322572335935,127.06283102249932&radius=20000>

```
{"channel":{"item":[{"phone":"02-450-9311","newAddress":"","imageUrl":"http://t1.daumcdn.net/cfile/1154FD3A4EBA26E02D","direction":"북","zipcode":"","placeUrl":"http://place.map.daum.net/7996800","id":"7996800","title":"서울 어린이대공원","distance":"4288","category":"여행 > 관광,명소 > 테마파크","address":"서울 광진구 능동 18 어린이대공원내 놀이동산","longitude":"127.08104900377838","latitude":"37.55013887884033","addressBCode":"1121510200"}, {"phone":"","newAddress":"서울 강남구 압구정로 120","imageUrl":"http://t1.daumcdn.net/cfile/03062A0379C4077F94","direction":"서","zipcode":"135888","placeUrl":"http://place.map.daum.net/7984226","id":"7984226","title":"신사동가로수길","distance":"3616","category":"여행 > 관광,명소 > 테마거리","address":"서울 강남구 신사동 530","longitude":"127.02283697718428","latitude":"37.52115586945625","addressBCode":"1168010700"}}
```


JSON

	title	phone	address
0	서울어린이대공원	02-450-9311	서울 광진구 능동 18 어린이대공원내 놀이동산
1	신사동가로수길		서울 강남구 신사동 530
2	서울어린이대공원 동물원		서울 광진구 능동 18 어린이대공원 중앙부
3	롯데월드	02-411-2000	서울 송파구 잠실동 40-1
4	압구정로데오거리	02-3445-6402	서울 강남구 신사동 668-33
5	응봉산	02-2286-6061	서울 성동구 응봉동
6	세빛섬	1566-3433	서울 서초구 반포동 650
7	어린이회관 눈썰매장	02-444-6377	서울 광진구 능동 18-11
8	아차산	02-450-1395	서울 광진구 중곡동
9	청계천		서울 종로구 서린동 148
10	덕수궁	02-771-9955	서울 중구 정동 5-1
11	창경궁	02-762-9514	서울 종로구 와룡동 2-1
12	창덕궁		서울 종로구 와룡동 2-71
13	서울랜드	02-509-6000	경기 과천시 막계동 33
14	석촌호수 서호	02-415-3595	서울 송파구 잠실동 47

JSON

```
import requests
import json
import pandas as pd
import numpy as np
from pandas import DataFrame, Series
url =
'https://apis.daum.net/local/v1/search/category.json?apikey=465b06fae32febabc5
9502598dd7685&code=AT4&location=37.514322572335935,127.06283102249932
&radius=20000'
data = requests.get(url)
result = json.loads(data.text) #파싱한 결과
channel = result['channel']
item = channel['item']
df = DataFrame(item, columns=['title', 'phone', 'address'])
print(df)
```