-Pandas는label이 부여된 data를 쉽고 직관적으로 취급할 수 있도록 설계된 Python third-party package이다.
-Pandas의 2가지 주요 data 구조인 Series(1차원 data)와 DataFrame(2차원 data)은 금융, 통계, 사회과학 등 많은 분야의 data 처리에 적합하다.
1. Pandas의 특징
  1)pandas의 주요 기능을 설명
    -쉬운 결손값(missing data)처리
    -Label 위치를 자동적/명시적으로 정리한 data 작성
    -Data 집약
    -고도의 label base의 slicing, 추출, 큰 dataset의 subset화
    -직감적인 dataset 결합
    -Dataset의 유연한 변환 및 변형
    -축의 계층적 label 붙임
    -여러가지 data 형식에 대응한 강력한 I/O
    -시계열 data 고유의 처리

  2)pandas의 package 정보
    -Version : 0.23.0
    -공식 site : http://pandas.pydata.org
    -Repository : https://github.com/pandas-dev/pandas
    -PyPI : https://pypi.python.org/pypi/pandas

  3)설치 여부 확인
    !conda list | grep pandas   #Windows에서는 grep명령어를 사용할 수 없음.

    -Windows에서는
      !conda list   #목록 중에서 찾아야 함.

2. Series
  1)Series는 index라고 불리는 label을 가진 동일한 data형을 가지는 1차원 data이다.
  2)다음의 특징이 있다.
    -Index(label)를 가지는 1차원 data
    -Index는 중복 가능
    -Label 또는 data의 위치를 지정한 추출가능. Index에 대한 slice가 가능
    -산술 연산이 가능. 통계량을 산출하는 merit를 가지고 있음.

  3)Python 표준 list나 tuple 등에서 사용되는 index라는 언어와의 혼동을 피하기 위해 series의 index를 label이라고 한다.

  4)Series 작성하기
    -Series의 작성에는 pandas.series class를 사용한다.
    -제1인수에는 다음과 같은 1차원의 data를 넘겨준다.
      --List
      --Tuple
      --Dirctionary
      --numpy.ndarray
    -아래와 같이 keyword 인수 index에 label이 되는 값을 넘기는 것으로 data를 표시한다.

      import pandas as pd

      ser = pd.Series([1,2,3], index=['a', 'b', 'c'])

```
      ser
      ------------------------------------------------
      a    1
      b    2
      c    3
      dtype: int64

  -index를 생략한 경우
    --index를 생략한 경우에는 0부터 차례대로 정수가 할당된다.

      ser = pd.Series([1,2,3])
      ser
      ----------------------------------------
      0    1
      1    2
      2    3
      dtype: int64

5)Label을 사용해서 data를 선택하기
  -Series.loc를 사용해서 label에서 data를 선택한다.

    ser.loc['b']
    ---------------
    2

  -loc를 사용하지 않는 서식
    --loc를 사용하지 않는 다음과 같은 서식도 있다.

      ser['b']
      -----------
      2

  -Label의 범위 지정
    --Label의 범위를 지정해서 slice를 할 수 있다.

      ser.loc['b' : 'c']
      ----------------------------
      b    2
      c    3
      dtype: int64

    --Lable에 따른 slice는 label의 시작 위치와 종료 위치를 포함한다.
    --Python의 list나 tuple에 대한 slice와의 동작이 다른 경우에 주의한다.

  -복수의 요소 지정
    --복수의 요소를 list로 지정할 수 있다.

      ser.loc[['a', 'c']]
      ----------------------------
      a    1
      c    3
      dtype: int64
```

6)위치를 지정해서 data 선택하기
-Series.iloc를 사용해서 data의 위치를 정수값으로 지정하고 data를 선택할 수 있다.

```
ser.iloc[1]
--------------------
2
```

-iloc의 slice는 Python 표준 list나 tuple에 대한 slice와 같이 동작한다.
-위치를 slice로 지정

```
ser.iloc[1:3]
----------------------------
b    2
c    3
dtype: int64
```

7)논리값을 사용해서 data 선택하기
-loc와 iloc에는 논리값의 list를 넘길 수 있다.

```
ser.loc[[True, False, True]]
---------------------------------
a    1
c    3
dtype: int64
```

-인수에 부여된 논리값 list는 Series의 index 위치에 대하여 True에 지정된 위치만 되돌아간다.
-Series에 대한 비교 연산을 통해 논리값을 되돌려준다.

```
ser != 2
----------------------------
a  True
b False
c  True
dtype: bool
```

-이것을 이용해서 data를 추출할 수 있다.

```
ser.loc[ser != 2]
-------------------------
a    1
c    3
dtype: int64
```

3. DataFrame
1)DataFrame은 행과 열에 label을 가진 2차원 data이다.
2)Data형은 열마다 다른 형을 가질 수 있다.
3)1차원 data인 Series의 집합으로 인식하는 것도 가능하다.
4)Series의 특징을 포함해서 DataFrame에는 다음과 같은 특징이 있다.
-행과 열에 label을 가진 2차원 data
-열마다 다른 형태를 가질 수 있음.

-Table형 data에 대해 불러오기. data 쓰기가 가능
-DataFrame끼리 여러가지 조건을 사용한 결합 처리가 가능
-Cross 집계가 가능

5)Python 표준 list나 tuple 등에서 사용되는 index라는 언어와의 혼동을 피하기 위해서 DataFrame의 index를 label이라고 한다.

6)DataFrame 작성하기
-DataFrame의 작성에는 pandas.DataFrame class를 사용한다.
-제1인수에는 1차원 또는 2차원 data를 넘긴다.
-Keyword 인수 index(행) 및 columns(열)에 label이 되는 값을 넘기는 것으로 data를 표시한다.

```
import pandas as pd

df = pd.DataFrame(
  [[1, 10, 100], [2, 20, 200], [3, 30, 300]],
  index=['r1', 'r2', 'r3'],
  columns=['c1','c2','c3'])
df
-------------------------------------------------
    c1  c2   c3
r1   1  10  100
r2   2  20  200
r3   3  30  300
```

7)Label을 사용해서 data 선택하기
-Series와 같이 DataFrame.loc를 사용해서 data를 추출한다.
-DataFrame의 경우에는 행과 열의 label을 각각 지정한다.

```
df.loc['r2', 'c2']
--------------------------
20
```

-모든 행(열)을 지정하는 경우
--모든 열을 지정하는 경우 요소에 [:]를 넘긴다.

```
df.loc['r2', :]
-------------------------
c1    2
c2   20
c3    200
Name: r2, dtype: int64
```

--모든 행을 지정하는 경우도 같다.

```
df.loc[:, 'c2']
-------------------------
r1   10
r2   20
r3   30
Name: c2, dtype: int64
```

-행의 data의 수가 1이고, 열의 data의 수가 복수 또는 행의 data의 수가 복수이고, 열의 data의 수가 1의 경우. 되돌아오는 data형은 Series가 된다.
-Slice나 list를 넘겨주는 방법
--Slice난 list를 넘기는 방법은 Series와 같다.

```
# 행 label을 list로 지정, 열 label을 slice로 지정
df.loc[['r1', 'r3'], 'c2' : 'c3']
---------------------------------
    c2   c3
r1  10  100
r3  30  300
```

-행의 data의 수와 열의 data의 수가 복수가 될 경우, 되돌아오는 data형은 DataFrame이다.

8)iloc를 사용해서 data를 선택하기
-Series와 같이 DataFrame.iloc를 사용해서 data의 위치에 의한 data 추출이 가능하다.
-DataFrame의 경우에는 행과 열의 위치를 각각 지정한다.

```
df.iloc[1:3, [0, 2]]
--------------------------------
   c1   c3
r2  2  200
r3  3  300
```

9)열 이름을 지정해서 data 선택하기
-loc나 iloc를 지정하지 않고 DataFrame에 대해 아래와 같이 지정한 경우는 열 지정이 되고 되돌아 오는 Data형은 Series가 된다.

```
df['c2']
-------------
r1  10
r2  20
r3  30
Name: c2, dtype: int64
```

10)논리값을 사용해서 data 선택하기
-Series와 같이 비교 연산을 하면 논리값이 되돌아온다.

```
df > 10
---------------------------
    c1     c2     c3
r1  False  False  True
r2  False  True   True
r3  False  True   True
```

-DataFrame에서 Series를 사용해 비교 연산을 하여 data를 추출할 수 있다.

```
# c2열의 값이10보다 큰 data
df.loc[df['c2'] > 10]
---------------------------------
    c1   c2   c3
r2  2   20  200
```

```
254       r3     3    30   300
255
256     -2개 이상의 조건을 조합하는 경우
257       --다음의 연산자를 사용한다.
258       --& : and 조건
259       --| : or 조건
260
261     -2개 이상의 조건을 조합하는 경우, 괄호()로 묶어 조건을 분류한다.
262
263       # c1열이 1보다 큰 동시에 c3열이 300보다 작은 data
264       df.loc[(df['c1'] > 1) & (df['c3'] < 300)]
265       ------------------------------------------
266           c1     c2     c3
267       r2     2    20   200
268
269   11)Series를 이용한 DataFrame 생성하기
270
271     list = [1,2,3]
272
273     ser1 = pd.core.series.Series(list)
274
275     ser2 = pd.core.series.Series(['one', 'two', 'three'])
276
277     pd.DataFrame(data=dict(num=ser1, word=ser2))
278     ----------------------------------------------------
279       num   word
280     0 1    one
281     1 2    two
282     2 3    three
283
284   12)Create DataFrame from your python code
285     -Python Dictionary를 이용해서 DataFrame 생성하기
286
287       emp_list = [
288          {'name':'John', 'age' : 25, 'job' : 'Manager'},
289          {'name':'Smith', 'age': 30, 'job' : 'Salesman'}
290       ]
291
292       df = pd.DataFrame(emp_list)
293       df
294       ---------------------------------------
295         age   job          name
296       0 25    Manager       John
297       1 30    Salesman    Smith              #list의 순서와 맞지 않음. key의 alphabet 순서
          와 동일
298
299       df = df[['name','age','job']]
300
301       df.head()
302       ----------------------------------------
303           name    age    job
304       0    John       25     Manager
```

```
305         1    Smith    30     Salesman
306
307     -OrderDictionary 이용하기(key 순서 보장)
308
309         from collections import OrderedDict
310
311         emp_ordered_list = OrderedDict(
312             [
313                 ('name', ['John', 'Smith']),
314                 ('age', [25, 30]),
315                 ('job', ['Manager', 'Salesman']),
316             ]
317         )
318
319         df = pd.DataFrame.from_dict(emp_ordered_list)
320
321         df.head()
322         ------------------------------------------------------
323           name    age    job
324         0 John       25     Manager
325         1 Smith    30     Salesman
326
327     -Python List를 이용해서 DataFrame 생성하기
328
329         emp_list = [
330             ['John', 25, 'Manager'],
331             ['Smith', 30, 'Salesman']
332         ]
333
334         column_names = ['name', 'age', 'job']
335
336         df = pd.DataFrame.from_records(emp_list, columns =column_names)
337
338         df.head()
339         ------------------------------------------------
340           name    age    job
341         0 John       25     Manager
342         1 Smith    30     Salesman
343
344         emp_list = [
345             ['name', ['John', 'Smith']],
346             ['age', [25, 30]],
347             ['job', ['Manager', 'Salesman']],
348         ]
349
350         df = pd.DataFrame.from_items(emp_list)
351         ------------------------------------------------
352         C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
            FutureWarning: from_items is deprecated. Please use
            DataFrame.from_dict(dict(items), ...) instead.
            DataFrame.from_dict(OrderedDict(items)) may be used to preserve the key
            order.
```

```
    """Entry point for launching an IPython kernel.

    df.head()
    ---------------------------------------------
      name   age   job
    0 John     25    Manager
    1 Smith   30    Salesman

13)DataFrame에서 column condition으로 select 하기

  user_list = [
     {'Name':'John', 'Age':25, 'Gender' : 'male', 'Address':'Chicago'},
     {'Name':'Smith', 'Age':35, 'Gender' : 'male', 'Address':'Boston'},
     {'Name':'Jenny', 'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
  ]

  df = pd.DataFrame(user_list)

  df = df[['Name', 'Age', 'Gender','Address']]

  df.head()
  ---------------------------------------
    Name   Age   Gender    Address
  0 John     25    male         Chicago
  1 Smith   35    male        'Boston'
  2 Jenny   45    female      Dallas

  -age가 30보다 많은 사람의 정보

    df[df.Age > 30]
    ------------------------------------------------
      Name   Age   Gender    Address
    1 Smith   35    male         Boston
    2 Jenny   45    female      Dallas

    df.query('Age > 30')
    ------------------------------------------------
      Name   Age   Gender    Address
    1 Smith   35    male         Boston
    2 Jenny   45    female      Dallas

  -age가 30보다 많고 이름이 'Smith'인 사람의 정보

    df[(df.Age > 30) & (df.Name == 'Smith')]
    ---------------------------------------------------
      Name   Age   Gender    Address
    1 Smith   35    male         Boston

14)Column filter 하기

  -Column name이 Name과 Gender인 column만 가져오기
```

```
    df.filter(items=['Name', 'Gender'])
    -------------------------------------------
        Name   Gender
    0   John     male
    1   Smith  male
    2   Jenny  female

  -Column name이 'A'라는 글자가 있는 Column만 가져오기

    df.filter(like = 'A')
    -------------------------------
          Age  Address
    0     25   Chicago
    1     35   Boston
    2     45   Dallas

  -정규식을 이용하여 column name이 'e'로 끝나는 Column만 가져오기

    df.filter(regex = 'e$')
    -------------------------
          Name  Age
    0     John    25
    1     Smith  35
    2     Jenny  45

  -정규식을 이용하여 column name이 'A'로 시작하는 Column만 가져오기

    df.filter(regex = 'A^')

15)DataFrame의 row와 column 삭제하기
  -row index를 사용하여 삭제하기

    user_list = [
       {'Age':25, 'Gender' : 'male', 'Address':'Chicago'},
       {'Age':35, 'Gender' : 'male', 'Address':'Boston'},
       {'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
    ]

    df = pd.DataFrame(user_list,
                      index = ['John', 'Smith', 'Jenny'],
                      columns = ['Age', 'Gender', 'Address'])

    df.head()
    ----------------------------------------
          Age  Gender   Address
    John    25    male       Chicago
    Smith 35    male       Boston
    Jenny 45    female     Dallas

    df.drop(['John', 'Jenny'])
    ------------------------------------
            Age  Gender   Address
```

```
457         Smith   35    male         Boston
458
459         df.head()  #하지만 여전히 df는 예전값을 갖고 있다.
460
461         df = df.drop(['John', 'Jenny'])   #이렇게 하면 제거된 결과를 df가 갖게 된다.
462         df.head()
463         --------------------------------------
464                 Age   Gender    Address
465         Smith   35    male         Boston
466
467       -inplace keyword 인수 이용하기
468
469         df = pd.DataFrame(user_list,
470                           index = ['John', 'Smith', 'Jenny'],
471                           columns = ['Age', 'Gender', 'Address'])
472
473         df.drop(['John', 'Jenny'], inplace=True)
474         df
475         ---------------------------------------------
476                 Age   Gender    Address
477         Smith   35    male         Boston
478
479       -row가 숫자로 indexing되어 있을 때 삭제하기
480
481         user_list = [
482            {'Name':'John', 'Age':25, 'Gender' : 'male', 'Address':'Chicago'},
483            {'Name':'Smith', 'Age':35, 'Gender' : 'male', 'Address':'Boston'},
484            {'Name':'Jenny', 'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
485         ]
486
487         df = pd.DataFrame(user_list)
488
489         df = df[['Name', 'Age', 'Gender','Address']]
490
491         df
492         ----------------------------------
493             Name   Age   Gender    Address
494         0   John      25    male         Chicago
495         1   Smith   35    male         Boston
496         2   Jenny   45    female       Dallas
497
498         df = df.drop(df.index[[0, 2]])
499         df
500         -----------------------------------
501           Name   Age   Gender    Address
502         1 Smith   35    male         Boston
503
504       -Condition으로 삭제하기
505
506         df.head()
507         -------------------------------
508               Name   Age   Gender    Address
```

```
0    John     25    male     Chicago
1    Smith    35    male     Boston
2    Jenny    45    female   Dallas

-Age가 30 이상인 사람의 정보만 저장함으로 나머지 정보는 삭제하는 방법

  df = df[df.Age > 30]
  df
  ---------------------------
     Name   Age   Gender   Address
  1  Smith  35    male     Boston
  2  Jenny  45    female   Dallas

-특정 column 삭제하기

  df.head()
  -------------------------------
        Name   Age   Gender   Address
  0     John     25    male     Chicago
  1     Smith  35    male     Boston
  2     Jenny  45    female   Dallas

  df.drop('Age', axis= 1)   <--- axis의 값이 1이면 column을 지칭
  ---------------------------------
     Name   Gender   Address
  0  John     male     Chicago
  1  Smith  male     Boston
  2  Jenny  female   Dallas

  df.drop('Age', axis= 1,  inplace=True)
  df
  -------------------------------------
     Name   Gender   Address
  0  John     male     Chicago
  1  Smith  male     Boston
  2  Jenny  female   Dallas

16)DataFrame의 row, column 생성 및 수정하기
  -column 새로 추가하기

  df.head()
  ----------------------------------------
     Name   Age   Gender  Address Job
  0  John     15    male     Chicago Student
  1  Smith  25    male     Boston    Teacher
  2  Jenny  17    female   Dallas   Student

  df['Salary'] = 0

  df.head()
  ------------------------------------------------------
     Name   Age   Gender   Address   Job        Salary
```

```
561       0   John      15    male        Chicago   Student     0
562       1   Smith   25    male        Boston      Teacher     0
563       2   Jenny   17    female      Dallas    Student     0
564
565     -기존 column 값을 이용
566     -'Job'에 따라 'Salary' 여부 Column으로 수정
567
568       df['Salary'] = np.where(df['Job'] != 'Student', 'yes', 'no')
569       df.head()
570       ------------------------------------------------------------
571           Name   Age   Gender    Address Job       Salary
572       0   John      15    male        Chicago Student no
573       1   Smith   25    male        Boston    Teacher yes
574       2   Jenny   17    female      Dallas  Student no
575
576     -'Total' column 추가
577
578       student_list = [
579          {'Name':'John', 'Midterm':95, 'Final' : 85},
580          {'Name':'Smith', 'Midterm':85, 'Final' : 80},
581          {'Name':'Jenny', 'Midterm':30, 'Final' : 10},
582       ]
583
584       df = pd.DataFrame(student_list, columns= ['Name', 'Midterm', 'Final'])
585
586       df.head()
587       ------------------------------------------------
588         Name   Midterm Final
589       0 John      95      85
590       1 Smith   85      80
591       2 Jenny   30      10
592
593       df['Total'] = df['Midterm'] + df['Final']
594       df
595       ---------------------------------------
596         Name   Midterm Final    Total
597       0 John      95      85    180
598       1 Smith   85      80    165
599       2 Jenny   30      10    40
600
601     -'Average' column 추가하기
602
603       df['Average'] = df['Total'] / 2
604       df.head()
605       ----------------------------------------
606         Name   Midterm Final      Total    Average
607       0 John      95        85    180      90.0
608       1 Smith   85        80    165      82.5
609       2 Jenny   30        10    40       20.0
610
611     -'Grade' column 추가하기
612
```

```
    grade_list = []
    for row in df['Average']:
        if row <= 100 and row >= 90 :
            grade_list.append('A')
        elif row < 90 and row >= 80 :
            grade_list.append('B')
        elif row < 80 and row >= 70 :
            grade_list.append('C')
        elif row < 70 and row >= 60 :
            grade_list.append('D')
        else : grade_list.append('F')

    df['Grade'] = grade_list
    df
    ------------------------------------------
      Name   Midterm Final    Total   Average Grade
    0 John     95      85      180     90.0      A
    1 Smith  85      80      165     82.5      B
    2 Jenny  30      10      40      20.0      F
```

-apply function 사용하기
-'Result' column 추가하기

```
    def pass_or_fail(row):
        if row != 'F':
            return 'Pass'
        else :
            return 'Fail'

    df['Result'] = df.Grade.apply(pass_or_fail)
    df
    ------------------------------------------
      Name   Midterm Final    Total   Average Grade   Result
    0 John     95      85      180     90.0      A       Pass
    1 Smith  85      80      165     82.5      B       Pass
    2 Jenny  30      10      40      20.0      F       Fail
```

-Column 추가하면서 각각의 값 조작하기

```
    date_list = [
        { 'yyyy-mm-dd' : '2019-01-05'},
        { 'yyyy-mm-dd' : '2019-01-10'}
    ]

    df = pd.DataFrame(date_list, columns = ['yyyy-mm-dd'])
    df
    ------------------------------------
      yyyy-mm-dd
    0 2019-01-05
    1 2019-01-10

    def extract_year(row):
```

```
665         return row.split('-')[0]
666
667     df['Year'] = df['yyyy-mm-dd'].apply(extract_year)
668     df
669     ------------------------------------
670         yyyy-mm-dd      Year
671     0   2019-01-05      2019
672     1   2019-01-10      2019
673
674   -Row 추가하기
675
676     user_list = [
677        {'Name':'John', 'Age':15, 'Gender' : 'male', 'Address':'Chicago'},
678        {'Name':'Smith', 'Age':25, 'Gender' : 'male', 'Address':'Boston'},
679        {'Name':'Jenny', 'Age':17, 'Gender' : 'female', 'Address':'Dallas'}
680     ]
681
682     df = pd.DataFrame(user_list, columns = ['Name', 'Age', 'Gender', 'Address'])
683     df.head()
684     ------------------------------------------------------------------
685         Name    Age    Gender    Address
686     0   John      15     male        Chicago
687     1   Smith   25     male        Boston
688     2   Jenny   17     female      Dallas
689
690     df2 = pd.DataFrame(
691        [['Peter', 35, 'male', 'Seoul']], columns = ['Name', 'Age', 'Gender',
           'Address']
692     )
693
694     df2.head()
695     ----------------------------------------
696         Name    Age    Gender  Address
697     0   Peter   35     male      Seoul
698
699     df.append(df2, ignore_index = True)
700     ------------------------------------------------------
701         Name    Age    Gender    Address
702     0   John      15     male        Chicago
703     1   Smith   25     male        Boston
704     2   Jenny   17     female      Dallas
705     3   Peter   35     male        Seoul
706
707
708   -passing keyword parameter to apply function
709
710     date_list = [{'Jumin': '2000-06-27'},
711          {'Jumin': '2002-09-24'},
712          {'Jumin': '2005-12-20'}]
713     df = pd.DataFrame(date_list, columns = ['Jumin'])
714     df
715     ------------------------------------------------------
```

```
    Jumin
0 2000-06-27
1 2002-09-24
2 2005-12-20

def extract_year(row):
    return row.split('-')[0]

df['Born_Year'] = df['Jumin'].apply(extract_year)
df
-------------------------------------------
    Jumin       Born_Year
0 2000-06-27    2000
1 2002-09-24    2002
2 2005-12-20    2005

def calc_age(year, current_year):
    return current_year - int(year)

df['Age'] = df['Born_Year'].apply(calc_age, current_year=2019)
df
-----------------------------------------------------
    Jumin       Born_Year   Age
0 2000-06-27    2000        19
1 2002-09-24    2002        17
2 2005-12-20    2005        14

def get_introduce(age, prefix, suffix):
    return prefix + str(age) + suffix

df['introduce'] = df['age'].apply(get_introduce, prefix="I am ", suffix=" years
old.")
df
-------------------------------------------------------------
    Jumin       Born_Year   Age   Introduce
0 2000-06-27    2000        19    I am 19 years old.
1 2002-09-24    2002        17    I am 17 years old.
2 2005-12-20    2005        14    I am 14 years old.

17)How to use map function
  -apply()와 유사한 map()

date_list = [{'Date': '2000-06-27'},
      {'Date': '2002-09-24'},
      {'Date': '2005-12-20'}]
df = pd.DataFrame(date_list, columns = ['Date'])
df
-------------------------------------------
    Date
0 2000-06-27
1 2002-09-24
2 2005-12-20
```

```
767
768       def extract_year(date):
769         return date.split('-')[0]
770
771       df['Year'] = df['Date'].map(extract_year)
772       df
773       ---------------------------------------
774         Date          Year
775       0 2000-06-27    2000
776       1 2002-09-24    2002
777       2 2005-12-20    2005
778
779     -map() 응용하기
780
781       data_list = [
782         {'Name':'John', 'Age':15, 'Gender':'male', 'Job':'Student'},
783         {'Name':'Smith', 'Age':25, 'Gender':'male', 'Job':'Teacher'},
784         {'Name':'Jenny', 'Age':27, 'Gender':'female', 'Job':'Developer'},
785       ]
786
787       df = pd.DataFrame(data_list, columns = ['Name', 'Age', 'Gender', 'Job'])
788       df
789       ------------------------------------------------------------
790         Name   Age   Gender  Job
791       0 John      15     male       Student
792       1 Smith   25     male       Teacher
793       2 Jenny   27     female     Developer
794
795       df.Job = df.Job.map({'Student':1, 'Teacher':2, 'Developer':3})
796       df
797       -----------------------------------------
798         Name   Age   Gender  Job
799       0 John      15     male       1
800       1 Smith   25     male       2
801       2 Jenny   27     female     3
802
803   18)Applymap 사용하기
804     -한번에 DataFrame에 있는 모든 요소들을 수정하고자 할 때
805
806       data_list = [
807         {'x': 5.5, 'y': -5.6},
808         {'x': -5.2, 'y': 5.5},
809         {'x': -1.6, 'y': -4.5}
810       ]
811       df = pd.DataFrame(data_list)
812       df
813       --------------------------------
814           x      y
815       0   5.5   -5.6
816       1   -5.2  5.5
817       2   -1.6  -4.5
818
```

```
    import numpy as np

    df = df.applymap(np.around)  #반올림함수 사용
    df
    -------------------------------------
        x      y
    0   6.0    -6.0
    1   -5.0   6.0
    2   -2.0   -4.0

19)Group By

  student_list = [
      {'Name': 'John', 'Major': "Computer Science", 'Gender': "male"},
         {'Name': 'Nate', 'Major': "Computer Science", 'Gender': "male"},
         {'Name': 'Abraham', 'Major': "Physics", 'Gender': "male"},
         {'Name': 'Brian', 'Major': "Psychology", 'Gender': "male"},
         {'Name': 'Janny', 'Major': "Economics", 'Gender': "female"},
         {'Name': 'Yuna', 'Major': "Economics", 'Gender': "female"},
         {'Name': 'Jeniffer', 'Major': "Computer Science", 'Gender': "female"},
         {'Name': 'Edward', 'Major': "Computer Science", 'Gender': "male"},
         {'Name': 'Zara', 'Major': "Psychology", 'Gender': "female"},
         {'Name': 'Wendy', 'Major': "Economics", 'Gender': "female"},
         {'Name': 'Sera', 'Major': "Psychology", 'Gender': "female"}
   ]
  df = pd.DataFrame(student_list, columns = ['Name', 'Major', 'Gender'])
  df
  ------------------------------------------------------------
      Name    Major               Gender
  0   John      Computer Science    male
  1   Nate      Computer Science    male
  2   Abraham   Physics             male
  3   Brian     Psychology          male
  4   Janny     Economics           female
  5   Yuna      Economics           female
  6   Jeniffer  Computer Science    female
  7   Edward    Computer Science    male
  8   Zara        Psychology          female
  9   Wendy     Economics           female
  10  Sera      Psychology          female

  -전공별 Group By
    groupby_major = df.groupby('Major')

    groupby_major.groups
    ----------------------------------------------
    {'Computer Science': Int64Index([0, 1, 6, 7], dtype='int64'),
     'Economics': Int64Index([4, 5, 9], dtype='int64'),
     'Physics': Int64Index([2], dtype='int64'),
     'Psychology': Int64Index([3, 8, 10], dtype='int64')}

  -보기 좋게
```

```
  for name, group in groupby_major:
    print(name + ": " + str(len(group)))
    print(group)
    print()
  ----------------------------------------------------
  Computer Science: 4
       Name    Major                 Gender
  0      John     Computer Science    male
  1      Nate     Computer Science    male
  6   Jeniffer    Computer Science      female
  7   Edward    Computer Science    male

  Economics: 3
       Name      Major         Gender
  4   Janny      Economics   female
  5   Yuna     Economics   female
  9   Wendy   Economics   female

  Physics: 1
       Name      Major           Gender
  2   Abraham  Physics       male

  Psychology: 3
       Name      Major           Gender
  3   Brian      Psychology     male
  8   Zara     Psychology  female
  10    Sera     Psychology  female
```

-전공별 명수

```
  df_major_cnt = pd.DataFrame({'Count':groupby_major.size()})
  df_major_cnt
  ------------------------------------------------
                      Count
  Major
  Computer Science    4
  Economics             3
  Physics             1
  Psychology            3
```

-전공도 count와 같이

```
  df_major_cnt = pd.DataFrame({'Count':groupby_major.size()}).reset_index()
  df_major_cnt
  ------------------------------------------------
       Major               Count
  0   Computer Science    4
  1   Economics             3
  2   Physics             1
  3   Psychology            3
```

-성별로 group by

```
923
924         groupby_gender = df.groupby('Gender')
925
926         for name, group in groupby_gender:
927           print(name + ": " + str(len(group)))
928           print(group)
929           print()
930         ------------------------------------------------
931         female: 6
932              Name    Major                  Gender
933         4     Janny       Economics           female
934         5     Yuna        Economics           female
935         6    Jeniffer    Computer Science       female
936         8     Zara         Psychology         female
937         9     Wendy     Economics             female
938         10     Sera        Psychology         female
939
940         male: 5
941              Name       Major                  Gender
942         0      John     Computer Science    male
943         1      Nate     Computer Science    male
944         2    Abraham  Physics               male
945         3    Brian       Psychology           male
946         7    Edward    Computer Science    male
947
948     20)중복된 값 제거하기
949
950       data_list = [
951         {'Name':'John', 'Gender':'male', 'Job':'Student'},
952         {'Name':'Smith','Gender':'male', 'Job':'Teacher'},
953         {'Name':'Jenny','Gender':'female', 'Job':'Developer'},
954         {'Name':'Smith','Gender':'male', 'Job':'Teacher'}
955       ]
956       df = pd.DataFrame(data_list, columns = ['Name', 'Gender', 'Job'])
957
958       df.head()
959       ------------------------------------------------------
960            Name Gender     Job
961       0    John      male          Student
962       1    Smith male          Teacher
963       2    Jenny female        Developer
964       3    Smith male          Teacher           #중복된 값
965
966       -중복된 값 확인하기
967
968         df.duplicated()
969         ----------------------------------
970         0    False
971         1    False
972         2    False
973         3      True              #여기가 중복된 값이 있다는 뜻
974         dtype: bool
```

-중복된 값 제거

```
df.drop_duplicates()
-----------------------------------
   Name Gender     Job
0 John      male          Student
1 Smith male          Teacher
2 Jenny female        Developer
```

21)None Value(NaN)을 찾고 원하는 값으로 변경하기

```
student_list = [
  {'Name': 'John', 'Major': 'Computer Science', 'Gender': 'male', 'Age': 40},
    {'Name': 'Nate', 'Major': None, 'Gender': "male", 'Age':35},
        {'Name': 'Abraham', 'Major': 'Physics', 'Gender': 'male', 'Age':37},
        {'Name': 'Brian', 'Major': 'Psychology', 'Gender': 'male', 'Age':None},
        {'Name': 'Janny', 'Major': None, 'Gender': 'female', 'Age':10},
        {'Name': 'Yuna', 'Major': None, 'Gender': 'female', 'Age':12},
        {'Name': 'Jeniffer', 'Major': 'Computer Science', 'Gender': 'female',
        'Age':45},
        {'Name': 'Edward', 'Major': 'Computer Science', 'Gender': 'male',
        'Age':None},
        {'Name': 'Zara', 'Major': 'Psychology', 'Gender': 'female', 'Age':25},
        {'Name': 'Wendy', 'Major': 'Economics', 'Gender': 'female', 'Age':37},
        {'Name': 'Sera', 'Major': None, 'Gender': 'female', 'Age':None}
    ]
df = pd.DataFrame(student_list, columns = ['Name', 'Major', 'Gender', 'Age'])
df
-----------------------------------------------------------
    Name    Major               Gender    Age
0   John      Computer Science    male        40.0
1   Nate      None                male        35.0
2   Abraham   Physics             male        37.0
3   Brian     Psychology          male        NaN
4   Janny     None              female      10.0
5   Yuna      None              female      12.0
6   Jeniffer  Computer Science    female      45.0
7   Edward    Computer Science    male        NaN
8   Zara      Psychology          female      25.0
9   Wendy     Economics           female      37.0
10  Sera      None              female      NaN

df.shape
-----------------------------------
(11,4)

df.info()
------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 4 columns):
```

```
Name      11 non-null object
Major     7 non-null object
Gender    11 non-null object
Age       8 non-null float64
dtypes: float64(1), object(3)
memory usage: 432.0+ bytes

-Null 확인하기

  df.isna()
  ----------------------------------
      Name Major Gender  Age
  0   False  False  False    False
  1   False  True     False    False
  2   False  False  False    False
  3   False  False  False    True
  4   False  True     False    False
  5   False  True     False    False
  6   False  False  False    False
  7   False  False  False    True
  8   False  False  False    False
  9   False  False  False    False
  10  False  True     False    True

  df.isnull()
  ----------------------------------
      Name Major Gender  Age
  0   False  False  False    False
  1   False  True     False    False
  2   False  False  False    False
  3   False  False  False    True
  4   False  True     False    False
  5   False  True     False    False
  6   False  False  False    False
  7   False  False  False    True
  8   False  False  False    False
  9   False  False  False    False
  10  False  True     False    True

-None값을 다른 값으로 변경하기

  df.Age = df.Age.fillna(0)     #숫자 NaN을 0으로
  df.Major = df.Major.fillna('Unknown')   #글자 None을 Unknown으로
  df
  ---------------------------------------------
      Name    Major                Gender    Age
  0   John       Computer Science    male         40.0
  1   Nate       Unknown              male         35.0
  2   Abraham  Physics              male         37.0
  3   Brian    Psychology           male         0.0
  4   Janny    Unknown              female       10.0
  5   Yuna     Unknown              female       12.0
```

```
      6   Jeniffer  Computer Science   female      45.0
      7   Edward    Computer Science   male        0.0
      8   Zara      Psychology         female      25.0
      9   Wendy     Economics          female      37.0
      10  Sera      Unknown            female      0.0
```

22)Unique와 갯수 알아보기
-Unique 즉, 중복제거된 값 알아보기

```
df.Major.unique()
-------------------------------------
array(['Computer Science', 'Unknown', 'Physics', 'Psychology', 'Economics'],
dtype=object)

df.Gender.unique()
-------------------------------------
array(['male', 'female'], dtype=object)

df.Major.value_counts()
-------------------------------------
Unknown             4
Computer Science    3
Psychology          2
Physics             1
Economics           1
Name: Major, dtype: int64
```

23)DataFrame 합치기

```
list1 = [
  {'Name': 'John', 'Job': 'Teacher'},
  {'Name': 'Nate', 'Job': 'Student'},
  {'Name': 'Fred', 'Job': 'Developer'}
]

list2 = [
  {'Name': 'Ed', 'Job': 'Dentist'},
  {'Name': 'Jack', 'Job': 'Farmer'},
  {'Name': 'Ted', 'Job': 'Designer'}
]

df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
df2 = pd.DataFrame(list2, columns = ['Name', 'Job'])
```

-concat()로 합치기

```
result = pd.concat([df1, df2])
result
-----------------------------------
    Name Job
0   John    Teacher
1   Nate    Student
```

```
    2   Fred    Developer
    0   Ed    Dentist                #0, 1, 2가 반복
    1   Jack    Farmer
    2   Ted   Designer

    result = pd.concat([df1, df2], ignore_index = True)
    result
    -----------------------------------
        Name Job
    0   John    Teacher
    1   Nate    Student
    2   Fred    Developer
    3   Ed    Dentist
    4   Jack    Farmer
    5   Ted   Designer

  -append()로 합치기

    result = df1.append(df2)
    result
    --------------------------------------
        Name Job
    0   John    Teacher
    1   Nate    Student
    2   Fred    Developer
    0   Ed    Dentist                #0, 1, 2가 반복
    1   Jack    Farmer
    2   Ted   Designer

    result = pd.append([df1, df2], ignore_index = True)
    result
    -----------------------------------
        Name Job
    0   John    Teacher
    1   Nate    Student
    2   Fred    Developer
    3   Ed    Dentist
    4   Jack    Farmer
    5   Ted   Designer

  -Column으로 합치기
    --두개의 DataFrame의 column이 서로 일치하지 않음.

      list1 = [
        {'Name': 'John', 'Job': 'Teacher'},
        {'Name': 'Nate', 'Job': 'Student'},
        {'Name': 'Jack', 'Job': 'Developer'}
      ]

      list2 = [
        {'Age': 25, 'Country': 'U.S'},
        {'Age': 30, 'Country': 'U.K'},
```

```
        {'Age': 45, 'Country': 'Korea'}
    ]

    df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
    df2 = pd.DataFrame(list2, columns = ['Age', 'Country'])

    result = pd.concat([df1, df2], axis=1, ignore_index=True)
    result
    ----------------------------------------------------------------
        0       1           2    3
    0   John    Teacher     25   U.S
    1   Nate    Student     30   U.K
    2   Jack    Developer   45   Korea
```

4. 다양한 data 불러오기
  1)Pandas는 다음과 같이 다양한 형식의 data를 불러올 수 있다.
    -CSV
    -Excel
    -Database
    -JSON
    -MessagePack
    -HTML
    -Google BigQuery
    -Clipboard
    -Pickle
    -기타(http://pandas.pydata.org/pandas-docs/stable/io.html)

5. CSV file 불러오기
  1)pandas.read_csv() 함수를 사용한다.
  2)첫번째 인수에 file 경로를 넘겨주면 DataFrame 형 object를 넘겨준다.
  3)File 경로 또는 URL 형식으로 지정할 수 있다.

```
    import pandas as pd

    df = pd.read_csv('pandas_data/friend_list.csv')
    df.head()
    ------------------------------------------------------
        name    age    job
    0   John      20     student
    1   Jenny   30     developer
    2   Nate      30     teacher
    3   Julia     40     dentist
    4   Brian   45     manager
```

  4)DataFrame.head()는 앞에서 5행분의 DataFrame을 넘겨준다.
  5)인수에 정수값을 넘기는 방식으로 행수를 지정할 수 있다.

```
    df = pd.read_csv('pandas_data/friend_list.csv')
    df.head(2)
    ---------------------------------
```

```
   name age    job
0 John     20     student
1 Jenny 30     developer
```

6)뒤에서부터 읽을 행의 갯수를 지정할 수도 있다.

```
df = pd.read_csv('pandas_data/friend_list.csv')
df.tail(2)
---------------------------------------
    name    age    job
4    Brian    45     manager
5    Chris    25     intern
```

7)각 열은 Series이다.

```
type(df.job)
----------------------------
pandas.core.series.Series
```

8)지정한 열을 DataFrame의 index로 하기
-아래 예제처럼 keyword 인수 index_col에 수치 또는 열 이름을 지정하여 지정한 열을 DataFrame의 index로 한다.

```
# index로 지정할 열을 번호로 지정
df = pd.read_csv('pandas_data/friend_list.csv', index_col = 0)
df.head()
-----------------------------------------------------------------
         age    job
name
John       20     student
Jenny    30     developer
Nate       30     teacher
Julia       40     dentist
Brian    45     manager

#index로 지정할 열을 열 이름으로 지정
df = pd.read_csv('pandas_data/friend_list.csv', index_col = 'job')
df.head()
-------------------------------------------------------
           name    age
job
student      John       20
developer    Jenny    30
teacher    Nate       30
dentist       Julia       40
manager     Brian    45
```

9)지정한 열을 지정한 형으로 불러오기
-Keyword 인수 dtype에 열 이름(key)과 형(값)을 사전형으로 지정하여 지정한 열을 지정한 형으로 불러올 수 있다.

```
#형 지정
```

```
1282        df = pd.read_csv('pandas_data/friend_list.csv', dtype={'age':float})
1283        df.head()
1284        ----------------------------------------
1285              name    age    job
1286        0     John      20.0     student
1287        1     Jenny   30.0     developer
1288        2     Nate      30.0     teacher
1289        3     Julia     40.0     dentist
1290        4     Brian   45.0     manager
1291
1292    10)형식이 유사한 txt file 읽어오기
1293      -CSV file처럼 txt file도 각 열의 구분을 ','로 할 경우
1294
1295        df = pd.read_csv('pandas_data/friend_list.txt')
1296        df.head()
1297        ----------------------------------------
1298              name    age    job
1299        0     John      20     student
1300        1     Jenny   30     developer
1301        2     Nate      30     teacher
1302        3     Julia     40     dentist
1303        4     Brian   45     manager
1304
1305
1306    11)만일 file의 column들이 쉼표로 구분되어 있지 않은 경우
1307      -delimiter parameter에 구분자를 지정해서 column을 나눠야 한다.
1308
1309        df = pd.read_csv('pandas_data/friend_list_tab.txt')
1310        df.head()
1311        ----------------------------------------
1312              name age job
1313        0 John\t20\tstudent
1314        1 Jenny\t30\tdeveloper
1315        2 Nate\t30\tteacher
1316        3 Julia\t40\tdentist
1317        4 Brian\t45\tmanager     #구분이 어려움.
1318
1319        df = pd.read_csv('pandas_data/friend_list_tab.txt', delimiter = '\t')
1320        df.head()
1321        ----------------------------------------
1322              name    age    job
1323        0     John      20     student
1324        1     Jenny   30     developer
1325        2     Nate      30     teacher
1326        3     Julia     40     dentist
1327        4     Brian   45     manager
1328
1329    12)file에 data header가 없는 csv file을 사용할 때
1330      -만일 data header가 없으면, header = None으로 지정해야 첫번째 data가 data header로
          들어가는 것을 막을 수 있다.
1331
1332        df = pd.read_csv('pandas_data/friend_list_no_head.csv')
```

```
1333        df.head()
1334        ------------------------------------------------------------
1335              John     20   student          #첫 번째 행이 header가 돼버림.
1336        0     Jenny 30   developer
1337        1     Nate     30   teacher
1338        2     Julia    40   dentist
1339        3     Brian 45   manager
1340        4     Chris  25   intern
1341
1342        df = pd.read_csv('pandas_data/friend_list_no_head.csv', header = None)
1343        df
1344        ------------------------------------------------------
1345            0      1    2
1346        0   John     20   student
1347        1   Jenny 30   developer
1348        2   Nate     30   teacher
1349        3   Julia    40   dentist
1350        4   Brian 45   manager
1351        5   Chris  25   intern
1352
1353      -만일 header가 없는 data를 호출했을 경우, DataFrame 생성 후, column header를 지정할
          수 있다.
1354
1355        df.columns = ['Name', 'Age', 'Job']
1356        df.index = ['1101', '1102', '1103', '1104', '1105', '1106']
1357        df
1358        ----------------------------------------
1359                Name   Age   Job
1360        1101    John     20    student
1361        1102    Jenny   30    developer
1362        1103    Nate     30    teacher
1363        1104    Julia    40    dentist
1364        1105    Brian   45    manager
1365        1106    Chris    25    intern
1366
1367      -file을 열 때 동시에 header에 column을 지정해야 할 경우
1368
1369        df = pd.read_csv('pandas_data/friend_list_no_head.csv', header = None,
          names=['Name', 'Age', 'Job'])
1370        df.head()
1371        ------------------------------------
1372        위의 결과와 동일
1373
1374    13)외부 CSV file 읽기
1375      -https://github.com/vincentarelbundock/Rdatasets/tree/master/csv/datasets
1376
1377
1378    14)기타 option
1379      -read_csv()에는 다수의 option이 준비되어 있다.
1380      -상세한 문서의 내용은 문서를 참조한다.
1381      -http://panda.pydata.org/pandas-docs/stable/io.html#io-read-csv-table
1382
```

15)DataFrame csv file로 저장하기

```
import pandas as pd

user_list = [
   {'Name':'John', 'Age':25, 'Gender' : 'male', 'Address':'Chicago'},
   {'Name':'Smith', 'Age':35, 'Gender' : 'male', 'Address':None},
   {'Name':'Jenny', 'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
]

df = pd.DataFrame(user_list)

df = df[['Name', 'Age', 'Gender','Address']]

df.head()
-------------------------------------
Name      Age   Gender    Address
0 John      25    male        Chicago
1 Smith   35    male        None
2 Jenny   45    female      Dallas

df.to_csv('user_list.csv')
```

-기본적으로 to_csv()는 index = True, header = True가 설정되어 있다.
--만일 index = False로 설정할 경우

```
   df.to_csv('user_list.csv', index=False)
```

--각 행의 index 즉 0, 1, 2가 사라진 csv file이 생성된다.
--또한 header = False로 설정하면 각 열의 header가 없어진다.

-Smith의 거주지가 None이기 때문에 빈칸으로 값이 들어간다.
-만일 csv로 file 저장시 빈칸대신 '-'를 넣어서 뭔가 있다는 것을 설정하려면,

```
df.to_csv('user_list.csv', na_rep = '-')
```

-이렇게 설정하면 해당 칸에는 '-'가 들어가게 된다.

6. Excel file 불러오기

1)pandas.read_excel() 함수를 사용한다.
2)사전준비를 위해 xlrd module을 설치여부를 확인한다.

```
!conda list | grep xlrd    #Windows에서는 사용 불가

$ conda install -y xrld==1.2.0   #설치안되어 있으면 설치

# Excel file 불러오기
df = pd.read_excel('pandas_data/재무실적.xlsx')
df.head()
```

3)불러오는 sheet 지정하기

-기본 설정으로는 첫 번ㅉ sheet를 불러온다.
-Sheet 이름을 지정해서 불러올 때는 keyword 인수 sheetname에 sheet이름을 지정한다.

```
df = pd.read_excel('pandas_data/재무실적.xlsx', sheetname = 'data2')
df.head()
```

7. SQL을 사용해서 불러오기
1)pandas.read_sql() 함수를 사용한다.
2)첫번째 인수에 query를 실행하는 SQL문, 두번째 인수에 SQLAlchemy(http://docs.sqlalchemy.org/en/latest/dialects/index) 또는 DBAPI2(PEP 249, Python Database API Specification v2.0, https://www.python.org/dev/peps/pep-0249)의 접속 instance를 넘겨준다.

3)MariaDB with Python
-설치여부 확인하기

```
!conda list | grep mysql-connector-python        #Windows에서는 grep 명령어 사용 불가
```

-mysql-connector-python 설치하기

--In Anaconda Prompt,

```
$ conda install -y mysql-connector-python

import mysql.connector as mariadb

mariadb_connection = mariadb.connect(user='root', password='javamariadb', host='localhost', database='world')
cursor = mariadb_connection.cursor()

cursor.execute("SELECT ID, Name, CountryCode, District, Population FROM city WHERE CountryCode='KOR'")

for ID,Name,CountryCode,District,Population in cursor:
  print('ID = %d, Name = %s, CountryCode = %s, District = %s, Popluation = %d' % (ID, Name, CountryCode,District,Population))
----------------------------------------------------
ID = 2331, Name = Seoul, CountryCode = KOR, District = Seoul, Popluation = 9981619
ID = 2332, Name = Pusan, CountryCode = KOR, District = Pusan, Popluation = 3804522
ID = 2333, Name = Inchon, CountryCode = KOR, District = Inchon, Popluation = 2559424
ID = 2334, Name = Taegu, CountryCode = KOR, District = Taegu, Popluation = 2548568
ID = 2335, Name = Taejon, CountryCode = KOR, District = Taejon, Popluation = 1425835
...
...
```

4)Oracle with Python

```
-Oracle cx_Oracle 7
-https://www.oracle.com/database/technologies/appdev/python.html
-Install on Windows
  $ python -m pip install cx_Oracle --upgrade

-In Anaconda Prompt
  $ conda install cx_oracle

-Connection
  import cx_Oracle

  --conn = cx_Oracle.connect('scott', 'tiger', 'localhost:1521/XE')

  --conn1 = cx_Oracle.connect('scott/tiger@localhost:1521/XE')

    print(conn1)
    ------------------------------------------------------------
    <cx_Oracle.Connection to scott@localhost:1521/XE>

  --dsn_tns = cx_Oracle.makedsn('localhost', 1521, 'XE')
    print(dsn_tns)
    ------------------------------------------------------------
    (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=152
    1))(CONNECT_DATA=(SID=XE)))

    conn2 = cx_Oracle.connect('scott', 'tiger', dsn_tns)
    print(conn2)
    ------------------------------------------------
    <cx_Oracle.Connection to
    scott@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(POR
    T=1521))(CONNECT_DATA=(SID=XE)))>

    conn.version
    --------------------------
    '11.2.0.2.0'

  -Cursor Objects
    cursor = conn.cursor()
    cursor.execute('SELECT empno, ename, job, hiredate, mgr, sal, comm,
    deptno FROM emp')
    ------------------------------------------------------------
    <cx_Oracle.Cursor on <cx_Oracle.Connection to
    scott@localhost:1521/XE>>

    for empno, ename, job, hiredate, mgr, sal, comm, deptno in cursor:
      print(empno, ename, job, hiredate, mgr, sal, comm,deptno)
    ------------------------------------------------------------------------
    7369 SMITH CLERK 1980-12-17 00:00:00 7902 800.0 None 20
    7499 ALLEN SALESMAN 1981-02-20 00:00:00 7698 1600.0 300.0 30
    7521 WARD SALESMAN 1981-02-22 00:00:00 7698 1250.0 500.0 30
    7566 JONES MANAGER 1981-04-02 00:00:00 7839 2975.0 None 20
    7654 MARTIN SALESMAN 1981-09-28 00:00:00 7698 1250.0 1400.0 30
```

```
        7698 BLAKE MANAGER 1981-05-01 00:00:00 7839 2850.0 None 30
        7782 CLARK MANAGER 1981-06-09 00:00:00 7839 2450.0 None 10
        7788 SCOTT ANALYST 1987-07-13 00:00:00 7566 3000.0 None 20
        7839 KING PRESIDENT 1981-11-17 00:00:00 None 5000.0 None 10
        7844 TURNER SALESMAN 1981-09-08 00:00:00 7698 1500.0 0.0 30
        7876 ADAMS CLERK 1987-07-13 00:00:00 7788 1100.0 None 20
        7900 JAMES CLERK 1981-12-03 00:00:00 7698 950.0 None 30
        7902 FORD ANALYST 1981-12-03 00:00:00 7566 3000.0 None 20
        7934 MILLER CLERK 1982-01-23 00:00:00 7782 1300.0 None 10

        cursor.close()


8. HTML file 불러오기
  1)pandas.read_html() 함수 사용하기
    -다음의 third-party package들의 설치여부를 확인한다.
      --html5lib, lxml, BeautifulSoup4
    -HTML file의 table 요소를 DataFrame으로 불러온다.
    -DataFrame이 들어 있던 list가 return된다.
    -Table 요소가 여러 개 있는 경우, 여러 개의 DataFrame이 저장된다.

      import pandas as pd

      url = 'http://www.fdic.gov/bank/individual/failed/banklist.html'

      dfs = pd.read_html(url)

      dfs[0].info()
      ------------------------------------------------------------------------------
      --
      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 555 entries, 0 to 554
      Data columns (total 7 columns):
      Bank Name              555 non-null object
      City                   555 non-null object
      ST                     555 non-null object
      CERT                   555 non-null int64
      Acquiring Institution  555 non-null object
      Closing Date           555 non-null object
      Updated Date           555 non-null object
      dtypes: int64(1), object(6)
      memory usage: 30.4+ KB

  2)request library 이용하기
    -Python HTTP for Humans'
    -http://docs.python-requests.org/en/master/
    -Install
      pip install requests

    import requests
    url = 'https://www.naver.com'
    naver = requests.get(url)
```

```
1573      print(naver.text)
1574
1575    3)pprint module 이용하기
1576      -대량의 data를 보기 쉽게 표시해주는 표준 module
1577      -pprint(prettyprint)
1578
1579        import requests
1580        import pprint
1581        url = 'https://www.naver.com'
1582        naver = requests.get(url)
1583        pprint.pprint(naver.text)
1584
1585    4)requests를 사용하여 API에 접근하기
1586      -기상청 RSS(http://www.weather.go.kr/weather/lifenindustry/sevice_rss.jsp)를 이
          용하자.
1587      -2013년 동네 예보 RSS
1588        http://www.weather.go.kr/images/weather/lifenindustry/dongnaeforecast_rss.
          pdf
1589      -주소선택후 rss button click하면 zone을 알 수 있다.
1590        --예:서울특별시 강남구 청담동
1591          http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500
1592
1593        import requests
1594        import pprint
1595        api_uri = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1168056500'
1596        weather_data = requests.get(api_uri).text
1597        pprint.pprint(weather_data)
1598        ------------------------------------------------------------------------------
1599        ('<?xml version="1.0" encoding="UTF-8" ?>\n'
1600         '<rss version="2.0">\n'
1601         '<channel>\n'
1602         '<title>기상청 동네예보 웹서비스 - 서울특별시 강남구 청담동 도표예보</title>\n'
1603         '<link>http://www.kma.go.kr/weather/main.jsp</link>\n'
1604         '<description>동네예보 웹서비스</description>\n'
1605         '<language>ko</language>\n'
1606         '<generator>동네예보</generator>\n'
1607         '<pubDate>2019년 01월 15일 (화)요일 14:00</pubDate>\n'
1608         ' <item>\n'
1609         '<author>기상청</author>\n'
1610         '<category>서울특별시 강남구 청담동</category>\n'
1611         '<title>동네예보(도표) : 서울특별시 강남구 청담동 '
1612         '[X=61,Y=126]</title><link>http://www.kma.go.kr/weather/forecast/timese
          ries.jsp?searchType=INTEREST&amp;dongCode=1168056500</link>\n'
1613         '<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=I
          NTEREST&amp;dongCode=1168056500</guid>\n'
1614         '<description>\n'
1615         ...
1616         ...
1617
1618      -get method의 params option을 활용하기
1619
1620        api_url = 'http://www.kma.go.kr/wid/queryDFSRSS.jsp'
```

```
1621
1622        payload = {'zone':'1168056500'}
1623
1624        weather_data = requests.get(api_url, payload).text
1625
1626        weather_data
1627        -------------------------------------------
1628        '<?xml version="1.0" encoding="UTF-8" ?>\n<rss
            version="2.0">\n<channel>\n<title>기상청 동네예보 웹서비스 - 서울특별시 강남구 청
            담동 도표예보
            </title>\n<link>http://www.kma.go.kr/weather/main.jsp</link>\n<descriptio
            n>동네예보 웹서비스</description>\n<language>ko</language>\n<generator>
            동네예보</generator>\n<pubDate>2019년 01월 15일 (화)요일
            14:00</pubDate>\n <item>\n<author>기상청</author>\n<category>서울특별
            시 강남구 청담동</category>\n<title>동네예보(도표) : 서울특별시 강남구 청담동
            [X=61,Y=126]</title><link>http://www.kma.go.kr/weather/forecast/timeseri
            es.jsp?searchType=INTEREST&amp;dongCode=1168056500</link>\n<guid>
            http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTERES
            T&amp;dongCode=1168056500</guid>\n<description>\n <header>\n
            <tm>201901151400</tm>\n  <ts>4</ts>\n
1629        ...
1630        ...
1631
1632    5)xml.etree.ElementTree module 사용하기
1633      -Python에서 xml data를 다루기 위한 module
1634
1635        import xml.etree.ElementTree as ET
1636
1637        xml_data = ET.fromstring(weather_data)
1638
1639        for tag in xml_data.iter('data'):
1640          print(tag.find('hour').text + "/" + tag.find('temp').text)
1641        ----------------------------------------------------------------------
1642        18/-2.0
1643        21/-4.0
1644        24/-5.0
1645        ...
1646        ...
1647
1648        list = []
1649        for tag in xml_data.iter('data'):
1650          dic = {'hour': tag.find('hour').text,
1651          'day' : tag.find('day').text,
1652          'temp' : tag.find('temp').text,
1653          'tmx' : tag.find('tmx').text,
1654          'tmn' : tag.find('tmn').text,
1655          'sky' : tag.find('sky').text,
1656          'pty' : tag.find('pty').text,
1657          'wfKor' : tag.find('wfKor').text,
1658          'wfEn' : tag.find('wfEn').text}
1659        list.append(dic)
1660
```

```
    df = pd.DataFrame(list, columns=['hour', 'day', 'temp','tmx','tmn','sky', 'pty',
    'wfKor', 'wfEn'])

    df
    ------------------------------------------------------------
      hour  day temp    tmx    tmn    sky pty  wfKor    wfEn
    0 18  0    -2.0    -999.0  -999.0  2    0    구름 조금 Partly Cloudy
    1 21  0    -4.0    -999.0  -999.0  2    0    구름 조금 Partly Cloudy
    2 24  0    -5.0    -999.0  -999.0  1    0    맑음      Clear
    3 3   1    -6.0    -1.0   -8.0   1    0    맑음      Clear
    4 6   1    -7.0    -1.0   -8.0   1    0    맑음      Clear

6)전자신문 RSS
  import pandas as pd
  import xml.etree.ElementTree as ET
  import requests

  api_url = 'http://rss.etnews.com/Section901.xml'

  etnews_data = requests.get(api_url).text
  etnews_data
  ------------------------------------------------
  '<?xml version="1.0" encoding="utf-8" ?>\n<rss version="2.0">\r\n
  <channel>\r\n   ...
  ...

  xml_data = ET.fromstring(etnews_data)
  for tag in xml_data.iter('item'):
    print(tag.find('title').text + "," + tag.find('pubDate').text)
  ------------------------------------------------------------------
  애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍,Tue, 15 Jan 2019 17:00:00
  +0900
  지난해 드론 자격증 취득자 전년비 4배 증가…실효성 지적도,Tue, 15 Jan 2019 17:00:00
  +0900
  화웨이, 韓 스마트워치 시장 진출,Tue, 15 Jan 2019 17:00:00 +0900
  ...
  ...

  list = []

  for tag in xml_data.iter('item'):
     dic = {'Title':tag.find('title').text,
          'Link':tag.find('link').text,
          'Author':tag.find('author').text,
          'PubDate':tag.find('pubDate').text,
          'Guid':tag.find('guid').text}
     list.append(dic)

  list
  ---------------------------------------
  [{'Title': "애플 납품 업체들, 줄줄이 실적 하향... '아이폰 쇼크' 후폭풍",
     'Link': 'http://www.etnews.com/20190115000273',
```

```
        'Author': '윤건일',
        'PubDate': 'Tue, 15 Jan 2019 17:00:00 +0900',
        'Guid': '20190115000273'},
        ...
        ...

    df = pd.DataFrame(list, columns=['Title', 'Link', 'Author', 'PubDate', 'Guid'])
    df.head()
    ----------------------------------------------
    ...
    ...

    df.info()
    ---------------------------------------
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 30 entries, 0 to 29
    Data columns (total 5 columns):
    Title      30 non-null object
    Link       30 non-null object
    Author     30 non-null object
    PubDate    30 non-null object
    Guid       30 non-null object
    dtypes: object(5)
    memory usage: 1.2+ KB


9. Crawling & Scraping
  1)Crawling
    -Web Site의 data를 그대로 취득하는 것
  2)Scraping
    -Crawling하여 모든 data에서 필요한 것만 추출하거나 변환하는 처리를 포함.
  3)BeautifulSoup4
    -Scraping을 위한 module
    -일반적으로 HTML tag들이 start tag와 end tag가 서로 pair 되지 않을 경우가 많다.
    -pair 되지 않아도 아름답게 처리해 주는 module
    -https://www.crummy.com/software/BeautifulSoup/bs4/doc/

    from bs4 import BeautifulSoup

    html_data = requests.get('https://www.naver.com')

    soup = BeautifulSoup(html_data.text, 'html.parser')

    soup.title
    ---------------------------------
    <title>NAVER</title>

  4)Naver 영화 평점 Scraping 하기

    from bs4 import BeautifulSoup

    html_data =
```

```
    requests.get('https://movie.naver.com/movie/point/af/list.nhn?page=1')

soup = BeautifulSoup(html_data.text, 'html.parser')

titles = soup.find_all(class_='movie')

title_list = []
for title in titles:
  print(title.text)
----------------------------------------
대학살의 신
스타워즈: 라스트 제다이
내안의 그놈
말모이
말모이
내안의 그놈
언니
내안의 그놈
존 윅
마이 리틀 자이언트

for title in titles:
  title_list.append(title.text)

point_list = []
points = soup.find_all(class_='point')
for point in points:
  point_list.append(point.text)

review_list = []
reviews = soup.find_all(class_='title')
for review in reviews:
   rev = review.text
   rev = rev.strip()
   rev = rev.replace('\t', '')
   rev = rev.replace('\n', '')
   rev = rev.replace('신고', '')
   review_list.append(rev)

df = pd.DataFrame(title_list, columns=['Title'])
df['Point'] = point_list
df['Review'] = review_list
df
------------------------------------------------
  Title                    Point    Review
0 대학살의 신              7        대학살의 신자식싸움에 부모등 터진다
1 스타워즈: 라스트 제다이  1        스타워즈: 라스트 제다이이거 보느니 로그원 열번 보는게
낫다
2 내안의 그놈              10       내안의 그놈재밌어요 유치해도 빵빵터짐 진영 연기 잘하네
요 라미란과 잘...
3 말모이                 10       말모이후반부에 보고 울었습니다 감동적임
4 말모이                 10       말모이재미를 떠나서 역사는 무조건 10점이다
```

```
5 내안의 그놈 10 내안의 그놈뻔한스토리이지만 재밌게 잘 보고왔어요최고의 성형은 다이어트
6 언니 5 언니론다 로우지가 언니 역활했으면 그나마 공감이 됐을듯... 개연성도 떨어지고 액션도...
7 내안의 그놈 9 내안의 그놈ㅋㅋ재밌었음 생각보다 안정적인 연기력 소소하게 웃기 좋은 영화
8 존 윅 10 존 윅액션의 선두주자 키아누리브스!!
9 마이 리틀 자이언트 7 마이 리틀 자이언트동화보다 더 환상적인 거인
```

5)Naver 평점 1page부터 100page까지 scraping 하기

```
from bs4 import BeautifulSoup

url = 'https://movie.naver.com/movie/point/af/list.nhn?page='

title_list = []
point_list = []
review_list = []

for pge in range(1, 101):
    url = url + str(pge)
    print(url)
    html_data = requests.get(url)
    soup = BeautifulSoup(html_data.text, 'html.parser')
    titles = soup.find_all(class_='movie')
    points = soup.find_all(class_='point')
    reviews = soup.find_all(class_='title')
    for title in titles:
        title_list.append(title.text)
    for point in points:
        point_list.append(point.text)
    for review in reviews:
        rev = review.text
        rev = rev.strip()
        rev = rev.replace('\t', '')
        rev = rev.replace('\n', '')
        rev = rev.replace('신고', '')
        review_list.append(rev)
    url = url.split('=')[0] + '='

df = pd.DataFrame(title_list, columns=['Title'])
df['Point'] = point_list
df['Review'] = review_list

df.info()
-------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 3 columns):
Title     1020 non-null object
Point     1020 non-null object
Review    1020 non-null object
```

```
dtypes: object(3)
memory usage: 24.0+ KB
```