```
2
      1)Pandas는 label이 부여된 data를 쉽고 직관적으로 취급할 수 있도록 설계된 Python third-party package이다.
 3
      2)Pandas의 2가지 주요 data 구조인 Series(1차원 data)와 DataFrame(2차원 data)은 금융, 통계, 사회과학 등 많은 분야의 data 처리에 적합하다.
 4
 5
 6
    2. Pandas의 특징
      1)Pandas의 주요 기능을 설명
 7
 8
         -쉬운 결손값(missing data)처리
 9
         -Label 위치를 자동적/명시적으로 정리한 data 작성
10
         -고도의 label base의 slicing, 추출, 큰 dataset의 subset화
11
12
         -직감적인 dataset 결합
13
         -Dataset의 유연한 변환 및 변형
14
         -축의 계층적 label 붙임
15
         -여러가지 data 형식에 대응한 강력한 I/O
16
         -시계열 data 고유의 처리
17
18
      2)Pandas의 package 정보
19
         -Version: 0.25.1
20
         -공식 site : http://pandas.pydata.org
21
         -Repository: https://github.com/pandas-dev/pandas
22
         -PyPI: <a href="https://pypi.python.org/pypi/pandas">https://pypi.python.org/pypi/pandas</a>
23
24
      3)설치 여부 확인
25
         -Jupyter 환경
26
            -!conda list | grep pandas
27
            -Windows에서는 grep명령어를 사용할 수 없음.
            -따라서 Windows에서는
28
29
              !conda list
                           #목록 중에서 찾아야 함.
30
31
         -PIP 환경
32
           -pip -V #version 확인
33
            -pip install pandas
                               #관리자 권한으로 설치할 것
34
            -numpy도 같이 설치됨.
35
           -설치 후 pip list로 확인
36
37
      4)import pandas
38
         import pandas as pd
39
40
41
42
    3. Series
43
      1)Series는 index라고 불리는 label을 가진 동일한 data형을 가지는 1차원 data이다.
44
         -Series는 DataFrame과 함께 pandas에서 제공하는 데이터 구조 중 하나이다.
45
         -DataFrame이 2차원 자료구조라면 Series는 1차원 자료구조이다.
46
         -DataFrame과 비슷하지만 1차원 자료구조이기 때문에 columns 속성이 없고 index 속성만 있다.
47
48
      2)다음의 특징이 있다.
49
         -Index(label)를 가지는 1차원 data
50
         -Index는 중복 가능
51
         -Label 또는 data의 위치를 지정한 추출가능.
52
         -Index에 대한 slice가 가능
53
         -산술 연산이 가능.
54
         -통계량을 산출하는 merit를 가지고 있음.
55
56
      3)Python 표준 list나 tuple 등에서 사용되는 index라는 언어와의 혼동을 피하기 위해 series의 index를 label이라고 한다.
57
58
      4)Series 작성하기
59
         -Series의 작성에는 pandas.series class를 사용한다.
60
         -pd.Series( data = [ ] , index = [ ] )
61
         -제1인수에는 다음과 같은 1차원의 data를 넘겨준다.
62
           --List
63
            --Tuple
           --Dirctionary
64
65
            --numpy.ndarray
66
         -아래와 같이 keyword 인수 index에 label이 되는 값을 넘기는 것으로 data를 표시한다.
67
68
         -list로 Series 생성하기
69
70
           ser = pd.Series([10, 20, 30])
71
           ser #index를 생략한 경우 : 0부터 차례대로 정수가 할당된다.
72
73
           0 10.0
74
            1 20.0
75
                0.3
76
           dtype: float64
77
78
           ser.index = ['a', 'b', 'c']
79
           ser
80
81
           a 10.0
82
           b
               20.0
83
               0.3
```

84

dtype: float64

Pandas⊢

```
86
             ser1 = pd.Series([1,2,3], index=['a', 'b', 'c'])
 87
             ser1
 89
             a 1
 90
             b
             c 3
 91
 92
             dtype: int64
 93
             ser2 = pd.Series([1, 2, 3, 4], index = ['USA', 'Germany', 'France', 'Japan'])
 94
 95
 96
 97
             USA
 98
             Germany 2
             France
 99
100
             Japan
101
             dtype: int64
102
             ser3 = pd.Series( [1, 2, 5, 4], index = ['USA', 'Germany', 'Italy', 'Japan'] )
103
104
105
             USA
106
             Germany 2
107
108
             Italy
109
             Japan
             dtype: int64
110
111
112
             fruits = Series([2500,3800,1200,6000], index=['apple','banana','peer','cherry'])
113
114
115
                             2500
             apple
116
                            3800
             banana
117
                               1200
             peer
                            6000
118
             cherry
119
             dtype: int64
120
121
             fruits.values
122
123
             array([2500, 3800, 1200, 6000], dtype=int64)
124
125
             fruits.index
126
127
             Index(['apple', 'banana', 'peer', 'cherry'], dtype='object')
128
129
           -dict로 Series 생성하기
130
             --Series형태는 dict와 매우 유사 하여 key가 index로 바뀌었다고 착각할 수 있다.
131
             	ext{--}그러나 사실상 데이터 구조 자체가 매우 다르기 때문에 \operatorname{dict}와는 사용하는 방식이 다르므로 주의해야 한다.
132
133
             ser4 = pd.Series( {'a':100, 'b':200, 'c':300} )
134
             ser4
135
136
             a 100
137
             b 200
138
                 300
139
             dtype: int64
140
             fruits_dic = {'apple': 2500,'banana':3800,'peer':1200,'cherry':6000}
141
142
             fruits = Series(fruits_dic)
143
             type(fruits dic)
144
145
             dict
146
147
             type(fruits)
148
149
             pandas.core.series.Series
150
151
152
        5)Label을 사용해서 data를 선택하기
153
           -Series.loc를 사용해서 label에서 data를 선택한다.
154
155
             ser = pd.Series([1,2,3], index=['a', 'b', 'c'])
156
             ser.loc['b']
157
             2
158
159
160
           -loc를 사용하지 않는 서식
161
             --loc를 사용하지 않는 다음과 같은 서식도 있다.
162
163
                ser['b']
164
                2
165
166
167
           -Label의 범위 지정
168
             --Label의 범위를 지정해서 slice를 할 수 있다.
```

```
170
               ser.loc['b' : 'c']
171
172
173
               c 3
174
               dtype: int64
175
176
            --Label에 따른 slice는 label의 시작 위치와 종료 위치를 포함한다.
            --Python의 list나 tuple에 대한 slice와의 동작이 다름에 주의한다.
177
178
179
          -복수의 요소 지정
180
            --복수의 요소를 list로 지정할 수 있다.
181
182
               ser.loc[['a', 'c']]
183
               a 1
c 3
184
185
186
               dtype: int64
187
188
189
       6)위치를 지정해서 data 선택하기
190
          -Series.iloc를 사용해서 data의 위치를 정수값으로 지정하고 data를 선택할 수 있다.
191
192
            ser.iloc[1]
193
194
            2
195
196
          -iloc의 slice는 Python 표준 list나 tuple에 대한 slice와 같이 동작한다.
197
          -위치를 slice로 지정
198
199
            ser.iloc[1:3]
200
            b 2
c 3
201
202
203
            dtype: int64
204
205
206
       7)논리값을 사용해서 data 선택하기
207
          -loc와 iloc에는 논리값의 list를 넘길 수 있다.
208
209
            ser.loc[[True, False, True]]
210
            a 1
c 3
211
212
213
            dtype: int64
214
215
          -인수에 부여된 논리값 list는 Series의 index 위치에 대하여 True에 지정된 위치만 되돌아간다.
          -Series에 대한 비교 연산을 통해 논리값을 되돌려준다.
216
217
218
            ser != 2
219
220
            a True
221
            b False
222
            c True
223
            dtype: bool
224
225
         -이것을 이용해서 data를 추출할 수 있다.
226
227
            ser.loc[ser != 2]
228
            a 1
c 3
229
230
231
            dtype: int64
232
233
234
       8)Series data 삭제하기
235
          -drop()은 데이터 구조의 row(행) 또는 column(열) 요소를 삭제.
236
237
            fruits = Series([2500, 3800, 1200,
            index=['apple','banana','peer','cherry'])
238
239
            fruits
240
241
            apple
                           3800
242
            banana
243
            peer
                              1200
244
                           6000
            cherry
            dtype: int64
245
246
247
            new_fruits = fruits.drop('banana')
248
            new_fruits
249
            -----
250
                   2500
            apple
251
                             1200
            peer
                           6000
252
            cherry
```

```
254
255
256
       9)Series data의 기본 연산
257
          -다음 코드는 시리즈 데이터의 + 연산자로 더하는 예이다.
258
          -연산하는 Series들의 index 중 하나라도 NaN(결측치)가 존재하면 연산 결과도 무조건 NaN으로 나온다.
259
260
            fruits1 = Series([5,9,10,3], index=['apple','banana','cherry','peer'])
261
            fruits2 = Series([3,2,9,5,10], index=['apple', 'orange', 'banana', 'cherry', 'mango'])
262
            fruits1
263
264
                           5
            apple
265
                             9
            banana
266
                          10
            cherry
267
            peer
            dtype: int64
268
269
270
            fruits2
271
            -----
272
            apple
273
            orange
274
                             9
            banana
275
                             5
            cherry
276
                             10
            mango
277
            dtype: int64
278
279
          -Series의 연산은 index에 의해 연산된다.
280
          -아래의 결과를 보면 'mango' index와 'orange' index는 fruits2 Series에만 있으므로 그 결과가 NaN이 되고
281
          -'peer' index는 fruits1 Series만 있으므로 그 결과가 NaN이 된다.
282
283
            fruits1 + fruits2
284
285
                             8.0
            annle
                           18.0
286
            banana
287
                          15.0
            cherry
288
            mango
                                NaN
289
            orange
                             NaN
290
            peer
                                NaN
291
            dtype: float64
292
293
294
       10)Series data 정렬
295
          -정렬은 sort_values()를 이용.
296
            fruits = Series([2500,3800,1200,6000], index=['apple','banana','peer','cherry'])
297
298
            fruits.sort_values(ascending=False)
299
300
                           6000
            cherry
301
            banana
                           3800
                           2500
302
            apple
303
                             1200
            peer
            dtype: int64
304
305
306
307
       11)Series를 DataFrame임으로
308
          -to_frame() 함수를 이용.
309
310
            fruits1 = Series([5,9,10,3], index=['apple','banana','cherry','peer'])
311
            fruits1.to frame()
312
            -----
313
314
            apple
315
                      9
            banana
            cherry
                      10
316
317
            peer
                      3
318
          -열 단위의 DataFrame을 행단위로 바꾸려면 T 속성 또는 transpose() 함수를 이용.
319
320
          -T 속성과 transpose() 함수는 행렬의 전치행렬을 반환한다.
321
322
            fruits1.to_frame().T
323
            -----
324
                 apple banana cherry peer
325
326
327
            fruits1.to_frame().transpose()
328
            -----
329
                 apple banana cherry peer
                           10
330
                 5 9
331
332
333
334
     4. DataFrame
335
       1)DataFrame은 행과 열에 label을 가진 2차원 data이다.
```

336

2)Data형은 열마다 다른 형을 가질 수 있다.

dtype: int64

```
337
        3)1차원 data인 Series의 집합으로 인식하는 것도 가능하다.
338
        4)Series의 특징을 포함해서 DataFrame에는 다음과 같은 특징이 있다.
339
           -행과 열에 label을 가진 2차원 data
340
           -열마다 다른 형태를 가질 수 있음.
341
           -Table형 data에 대해 불러오기 / data 쓰기가 가능
342
           -DataFrame끼리 여러가지 조건을 사용한 결합 처리가 가능
343
           -Cross 집계가 가능
344
345
        5)Python 표준 list나 tuple 등에서 사용되는 index라는 언어와의 혼동을 피하기 위해서 DataFrame의 index를 label이라고 한다.
346
347
348
349
     5. DataFrame 생성하기
350
        1)DataFrame의 생성에는 pandas.DataFrame class를 사용한다.
351
        2)제1인수에는 1차원 또는 2차원 data를 넘긴다.
352
        3)Keyword 인수 index(행) 및 columns(열)에 label이 되는 값을 넘기는 것으로 data를 표시한다.
353
        4)dict를 이용한 DataFrame 생성하기
354
           -dict를 를 이용하면 dict의 key가 열 이름이 된다.
355
             d = {'col1': [1, 2], 'col2': [3,
356
357
             df = pd.DataFrame(data=d)
358
             df
359
360
                  col1 col2
361
             0
                          3
                          4
362
363
364
           -dict는 다음처럼 list에 저장되어 있어도 쉽게 DataFrame으로 만들 수 있다.
365
366
             d = [\{'col1': 1, 'col2': 3\}, \{'col1': 2, 'col2': 4\}]
367
             df = pd.DataFrame(data=d)
368
             df
369
                  col1 col2
370
371
                  1
372
373
374
           -만일 list내의 dict의 요소의 수가 다를 경우에는 NaN으로 채워진다.
375
376
             d = [\{'col1': 1, 'col2': 3\}, \{'col1': 2, 'col2': 4\}, \{'col1': 2\}]
377
             df = pd.DataFrame(data=d)
378
             df
379
380
                  col1 col2
381
             0
                  1
                          3.0
382
             1
                          4.0
383
                          NaN
384
385
             emp list = [
                  {'name':'John', 'age' : 25, 'job' : 'Manager'},
{'name':'Smith', 'age': 30, 'job' : 'Salesman'}
386
387
388
389
390
             df = pd.DataFrame(emp_list)
391
             df
392
393
                age
                         job
                                            name
394
             0 25
                        Manager
                                        John
395
             1 30
                       Salesman
                                        Smith
396
             #list의 순서와 맞지 않음. key의 alphabet 순서와 동일
397
             df = df[['name','age','job']]
398
399
             df.head()
400
401
                                  job
                   name
                             25
                                  Manager
402
                  John
403
             1
                  Smith
                             30
                                  Salesman
404
405
        5)OrderDictionary 이용하기(key 순서 보장)
406
407
          from collections import OrderedDict
408
          emp_ordered_list = OrderedDict(
409
                 ('name', ['John', 'Smith']),
('age', [25, 30]),
410
411
412
                 ('job', ['Manager', 'Salesman']),
413
414
415
          df = pd.DataFrame.from_dict(emp_ordered_list)
416
          df.head()
417
418
419
                             job
             name age
                       25 Manager
420
          0 John
```

```
423
424
       6)list를 이용해 DataFrame 만들기
425
426
          df = pd.DataFrame(
                 [[1, 10, 100], [2, 20, 200], [3, 30, 300]], index=['r1', 'r2', 'r3'],
427
428
                 columns=['c1','c2','c3'])
429
          df
430
                      _____
431
                           c3
                 c1
                      c2
432
                       10
                           100
          r1
               1
433
          r2
               2
                       20
                            200
434
          r3
               3
                       30
                            300
435
436
          emp list = [
437
                ['John', 25, 'Manager'],
438
                ['Smith', 30, 'Salesman']
439
            ]
440
441
          column names = ['name', 'age', 'job']
442
          df = pd.DataFrame.from_records(emp_list, columns =column_names)
443
          df.head()
444
                              job
445
               name age
            0 John
                         25
446
                              Manager
447
            1 Smith
                         30
                              Salesman
448
449
          emp_list = [
             ['name', ['John', 'Smith']],
450
451
             ['age', [25, 30]],
             ['job', ['Manager', 'Salesman']],
452
453
          1
454
455
          df = pd.DataFrame.from_items(emp_list)
456
457
          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.
458
          """Entry point for launching an IPython kernel.
459
460
          df.head()
461
462
            name
                    age
                           job
463
          0 John
                      25
                           Manager
464
          1 Smith
                       30
                            Salesman
465
466
467
       7)Series를 이용한 DataFrame 생성하기
468
469
          list = [1,2,3]
470
          ser1 = pd.core.series.Series(list)
471
          ser2 = pd.core.series.Series(['one', 'two', 'three'])
472
473
          pd.DataFrame(data=dict(num=ser1, word=ser2))
474
475
            num word
476
          0 1
                    one
477
          1 2
                    two
          2 3
478
                    three
479
480
481
       8)read_csv()함수를 이용해서 DataFrame 만들기
482
          -read_csv() 함수는 CSV 파일을 읽어 DataFrame으로 만든다.
483
          -CSV 파일을 읽을 때 SEP 매개변수로 구분자를 지정할 수 있다.
484
485
            import pandas as pd
486
            member_df = pd.read_csv("member_data.csv", sep=",")
487
            member_df
488
             -----
489
                         Age
               Name
                                Email
                                                          Address
490
            0 홍길동
                       20
                           kildong@hong.com
                                                서울시 강동구
                       25
                            kilseo@hong.com
491
            1 홍길서
                                                  서울시 강서구
492
                       26
                            south@hong.com
            2 홍길남
                                                  서울시 강남구
493
            3 홍길북
                       27
                            book@hong.com
                                                서울시 강북구
494
495
496
          -CSV파일에 헤더 정보가 없을 경우 header=None 인수를 포함하면 열 이름이 0, 1, 2, ...순으로 자동 지정된다.
497
498
499
       9)sklearn.datasets module data를 DataFrame으로 변환하기
500
          -Scikit-learn package에는 학습을 위한 많은 dataset이 제공된다.
501
          -Scikit-learn에서 제공하는 dataset은 dict 형식으로 되어 있다.
```

422

502

1 Smith

30

Salesman

```
504
          import pandas as pd
505
          from sklearn import datasets
506
          iris = datasets.load_iris()
507
508
          {'data': array([[5.1, 3.5, 1.4, 0.2],
509
          [4.9, 3., 1.4, 0.2],
[4.7, 3.2, 1.3, 0.2],
[4.6, 3.1, 1.5, 0.2],
510
511
512
          [5. , 3.6, 1.4, 0.2],
513
514
          ... 생략 ...
          [6.7, 3., 5.2, 2.3],
[6.3, 2.5, 5., 1.9],
515
516
          [6.5, 3., 5.2, 2.],
[6.2, 3.4, 5.4, 2.3],
[5.9, 3., 5.1, 1.8]]),
517
518
519
520
          521
          522
          523
524
          525
526
          527
528
          'DESCR': 'Iris Plants Database...
529
530
531
          'feature_names':['sepal length (cm)',
           'sepal width (cm)',
532
          'petallength (cm)'
533
534
          'petal width (cm)'],
535
          536
537
538
          x = pd.DataFrame(iris.data, columns=iris.feature_names)
539
          y = pd.DataFrame(iris['target_names'][iris['target']], columns=["species"])
          iris_df = pd.concat([x,y], axis=1)
540
541
          iris_df.head()
542
               sepal length (cm) sepal width (cm) petal length (cm)
543
                                                                  petal width (cm) species
544
          setosa
545
             4.9
                                    3.0
                                                            1.4
                                                                                 0.2
          setosa
546
             4.7
                                    3.2
                                                            1.3
                                                                                 0.2
          setosa
547
                                    3.1
                                                                                 0.2
             4.6
                                                            1.5
          setosa
548
                                                                                 0.2
          4 5.0
                                    3.6
                                                           1.4
          setosa
549
550
551
552
553
      1)DataFrame이 열 또는 행의 이름을 지정하면 이름으로 데이터의 부분집합을 얻거나 정렬할 수 있다.
554
      2)행의 이름은 index, 열의 이름은 columns 속성을 이용한다.
555
      3)열 이름 지정하기
        -DataFrame의 columns 속성을 이용하면 열의 이름들을 지정할 수 있다.
556
557
558
          member_df.columns = ["이름", "나이", "이메일", "주소"]
559
          member_df
560
561
                   이름 나이 이메일
                   홍길동 20 kildong@hong.com
          n
562
                                          서울시 강동구
                   홍길서 25 kilseo@hong.com 서울시 강서구
홍길남 26 south@hong.com 서울시 강남구
563
          1
564
565
                   홍길북 27 book@hong.com
                                          서울시 강북구
566
567
          member_df.columns
568
          Index(['이름', '나이', '이메일', '주소'], dtype='object')
569
570
571
      4)행 이름 지정하기
572
        -행은 index 속성을 이용해 이름을 지정할 수 있다.
573
574
        member_df.index = ["동", "서", "남", "북"]
575
        member_df
576
577
            이름 나이 이메일
            홍길동 20 kildong@hong.com
578
                                      서울시 강동구
            홍길서 25 kilseo@hong.com 서울시 강서구
홍길남 26 south@hong.com 서울시 강남구
579
        서
580
        남
                                        서울시 강남구
            홍길북 27 book@hong.com
581
                                    서울시 강북구
```

import numpy as np

```
584
      5)level 이름 지정하기
585
         -DataFrame의 열 이름과 행 이름은 group을 지어 지정할 수 있다.
586
         -이렇게 하면 한 개의 열은 두 개 이상의 열 이름 또는 행 이름을 가질 수 있는데 이때 level을 이용해 이름을 구분할 수 있다.
587
         -level의 이름은 names 속성을 이용할 수 있다.
588
         -열의 이름을 지정할 때에는 columns 속성을 이용하고, 열의 level을 지정하려면 columns.names 속성을 이용한다.
589
         -행(인덱스)의 이름을 지정할 때에는 index 속성을 이용하고, 행의 레벨을 지정하려면 index.names 속성을 이용한다.
590
           591
592
593
           member_df.columns.names = ["정보구분"
594
                                           "상세정보"]
          595
596
597
598
           member_df
599
600
                   정보구분 기본정보 추가정보
601
                   상세정보 이름 나이 이메일
          위치구분
602
                   상세위치
                          홍길동 20 kildong@hong.com
                                                  서울시 강동구
603
                   동
                          흥길서 25 kilseo@hong.com 서울시 강서구
흥길남 26 south@hong.com 서울시 강남구
604
                   서
605
                          홍길남 26 south@hong.com
          상하
                   남
                         홍길북 27 book@hong.com 서울시 강북구
606
607
608
609
610
    7. 부분 데이터 조회
      member_df = pd.read_csv("member_data.csv")
612
      member df
613
614
      1)단일 열 조회
        -참조 형식(.) 또는 배열 형식([\ ])을 이용하면 열 하나를 조회할 수 있다.
615
616
         -DataFrame의 열 이름을 참조 형식(. 연산자)을 이용해서 정보를 조회할 수 있다.
617
618
          member_df.Name
619
          0
620
                  홍길동
621
          1
                   홍길서
          2홍길남3홍길북
622
623
          Name: Name, dtype: object
624
625
626
        -열 정보를 조회할 때에는 배열 형식(["열이름"])으로도 가능하다.
627
       member_df["Name"]
628
        0
629
630
                홍길서
           > e시
홍길남
-
631
        2
632
633
        Name: Name, dtype: object
634
635
636
      2)loc를 이용한 이름으로 조회
637
        -loc[]를 이용하면 행 또는 열 이름으로 부분 데이터셋을 조회할 수 있다.
638
        -콜론(:)은 사이의 모든 행을 선택한다.
639
640
           member_df.loc[0:2]
641
               Name Age Email
홍길동 20 kildong@hong.com 서울시 강동구
642
                                                     Address
643
              홍길서 25 kilseo@hong.com
                                           서울시 강서구
644
                    26 south@hong.com
645
              홍길남
                                             서울시 강남구
646
647
        -loc에서의 숫자는 인덱스가 아니다.
648
         -그러므로 0:2는 인덱싱 할 때의 from:to 개념을 적용한 0행부터 2행까지를 의미하는 것이 아니다.
649
         -loc() 함수에서 0:2는 0, 1, 2 행을 의미한다.
650
         -loc[]는 기본적으로 행의 이름을 이용해서 부분 데이터셋을 조회한다.
651
652
           member_df.loc["Name":"Email"] # nothing
653
654
         -행과 열의 이름을 모두 이용해서 부분 데이터셋을 조회할 수 있다.
655
           member_df.loc[0:2, "Name":"Email"]
656
657
             Name Age Email
658
          0 홍길동 20 kildong@hong.com
1 홍길서 25 kilseo@hong.com
659
          1 홍길서 25 kilseo@hong.com
2 홍길남 26 south@hong.com
660
661
662
663
        -행 또는 열의 이름으로 데이터를 조회려면 리스트 형식으로 지정한다.
664
665
           member_df.loc[[0,2], ["Name","Email"]]
```

```
666
            _____
667
                Name
                          Email
            n
                        kildong@hong.com
668
                 홍길동
669
                 홍길남
                        south@hong.com
670
671
         -모든 열을 지정하는 경우 요소에 [:]를 넘긴다.
672
673
            member_df.loc[0:2, :]
674
675
                Name
                          Age
                                 Email
                                                          Address
676
                        20
                             kildong@hong.com
                 홍길동
                                                서울시 강동구
                        25
                             kilseo@hong.com
677
            1
                 홍길서
                                                  울시 강서구
678
            2
                        26
                             south@hong.com
                 홍길남
                                                  서울시 강남구
679
680
         -모든 행을 지정하는 경우도 같다.
681
682
           member_df.loc[:, 'Email']
683
            0
684
               kildong@hong.com
685
            1
                kilseo@hong.com
686
                south@hong.com
687
                 book@hong.com
688
            Name: Email, dtype: object
689
690
       3)iloc를 이용한 index로 조회
691
         -iloc[from_index : to_index]는 index를 이용해서 부분 데이터셋을 조회한다.
692
         -to_index는 포함되지 않는다.
693
         -indexing 방법은 Python list의 indexing 방법을 사용할 수 있다.
694
695
            member_df.iloc[1:3, 1:3]
696
697
                Age
                        Email
698
                 25
                     kilseo@hona.com
            1
699
            2
                 26
                     south@hong.com
700
701
            member_df.iloc[0:3, 0:3]
702
703
                Name
                          Age
                                 Fmail
704
            0
                        20
                             kildong@hong.com
                 홍길동
705
            1
                        25
                             kilseo@hong.com
                 홍길서
706
            2
                 홍길남
                        26
                             south@hong.com
707
708
709
       4)iloc[from_index: to_index: by] 형식을 사용할 수 있다.
710
          -이것은 from_index 부터 to_index까지 매 by마다 데이터를 조회한다.
711
712
         member df.iloc[::-1]
713
714
                               Email
                Name
                          Age
                                                          Address
                        27
715
         3
                             book@hong.com
                 홍길북
                                                서울시 강북구
         2
716
                        26
                             south@hong.com
                 홍길남
                                                  서울시 강남구
717
         1
                        25
                             kilseo@hong.com
                 홍길서
                                                  서울시 강서구
718
         0
                        20
                             kildong@hong.com
                 홍길동
                                                서울시 강동구
719
720
         member_df.iloc[0::2,[1,3]]
721
722
              Age
                     Address
723
         0
              20
                   서울시 강동구
724
              26
                  서울시 강남구
725
726
       5)조건으로 조회하기
727
         -조건으로 데이터 조회를 설명하기 전에 package를 import하고 예제로 사용할 데이터를 불러와 DataFrame으로 만든다.
728
         -iris data는 sklearn package의 dataset에서 불러올 수도 있지만 DataFrame으로 사용하려면 statsmodels package의 dataset에서 불러오는 것이
         더 쉽다.
729
730
            import numpy as np
731
            import pandas as pd
732
            import statsmodels.api as sm #pip install statsmodels
733
           iris = sm.datasets.get_rdataset("iris", package="datasets")
734
           iris
735
736
            <class 'statsmodels.datasets.utils.Dataset'>
737
738
         -get_rdataset() 함수로 불러온 데이터는 다음 속성을 가지고 있다.
739
            --package : 데이터를 제공하는 R package 이름.
740
            --title : 데이터의 이름.
741
            --data : 데이터를 담고 있는 DataFrame.
742
            --__doc__ : 데이터에 대한 설명 문자열.
743
            --이 설명은 R package의 내용을 그대로 가져온 것이므로 예제 코드가 R로 되어 있어 Python에서 바로 사용할 수 없다.
744
          -statsmodels 패키지의 get_rdataset() 함수로 불러온 데이터의 data 속성은 데이터를 담고 있는 DataFrame이다.
745
746
            iris_df = iris.data
747
           iris df.head()
748
                          -----
```

```
Sepal.Width
        Sepal.Length
                                            Petal.Length
                                                             Petal.Width
                                                                               Species
  0
        5.1
                                3.5
                                                                               0.2
                                                                                                    setosa
                                                        1.4
                                                                               0.2
   1
        4.9
                                3.0
                                                        1.4
                                                                                                    setosa
        4.7
                                3.2
                                                        1.3
                                                                               0.2
                                                                                                    setosa
        4.6
   3
                                3.1
                                                        1.5
                                                                               0.2
                                                                                                    setosa
   4
        5.0
                                3.6
                                                        1.4
                                                                               0.2
                                                                                                    setosa
-다음 코드는 versicolor 종의 데이터만 조회한다.
  iris_df.loc[iris_df['Species']=='versicolor'].head()
                                                             Petal.Width
                          Sepal.Width
                                            Petal.Length
        Sepal.Length
                                                                               Species
   507.0
                             3.2
                                                     4.7
                                                                             1.4
                                                                                                 versicolor
   516.4
                             3.2
                                                     4.5
                                                                            1.5
                                                                                                 versicolor
  526.9
                             3.1
                                                     4.9
                                                                            1.5
                                                                                                 versicolor
  535.5
                             2.3
                                                     4.0
                                                                            1.3
                                                                                                 versicolor
  546.5
                             2.8
                                                                            1.5
                                                                                                 versicolor
                                                     4.6
-다음 코드는 versicolor 종의 Sepal.Length열과 Species 열 정보만 조회한다.
  iris_df.loc[iris_df['Species']=='versicolor', ['Sepal.Length', 'Species']].head()
              Sepal.Length Species
   50
           7.0
                                versicolor
   51
           6.4
   52
           6.9
                                versicolor
  53
           5.5
                                versicolor
  54
           6.5
                                versicolor
-다음 코드는 versicolor 종들 중에서 Sepal.Length가 6.5보다 큰 데이터만 조회한다.
  iris_df.loc[(iris_df['Species']=='versicolor') & (iris_df['Sepal.Length'].astype(float) > 6.5)].head()
                             Sepal.Width
                                               Petal.Length
        Sepal.Length
                                                                Petal.Width
                                                                                  Species
   507.0
                                3.2
                                                        4.7
                                                                               1.4
                                                                                                    versicolor
   526.9
                                3.1
                                                        4.9
                                                                               1.5
                                                                                                    versicolor
  586.6
                                2.9
                                                        4.6
                                                                               1.3
                                                                                                    versicolor
  656.7
                                                        4.4
                                3.1
                                                                               1.4
                                                                                                    versicolor
   756.6
                                3.0
                                                        4.4
                                                                               1.4
                                                                                                    versicolor
-좀 더 쉬운 다른 예제를 사용해 보자.
  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'}
  1
  df = pd.DataFrame(user list)
  df = df[['Name', 'Age', 'Gender','Address']]
   df.head()
     Name
                 Age
                         Gender
                                      Address
   0
     John
                 25
                       male
                                      Chicago
     Smith
                 35
                                      'Boston'
   1
                       male
                 45
   2 Jenny
                       female
-age가 30보다 많은 사람의 정보
   df[df.Age > 30]
                 Age
     Name
                       Gender
                                      Address
   1 Smith
                 35
                                      Boston
                       male
                 45
                       female
                                      Dallas
   2 Jenny
   df.query('Age > 30')
     Name
                 Age
                          Gender
                                      Address
   1 Smith
                 35
                       male
                                      Boston
   2 Jenny
                 45
                       female
-age가 30보다 많고 이름이 'Smith'인 사람의 정보
   df[(df.Age > 30) & (df.Name == 'Smith')]
   # or df.loc[(df.Age > 30) & (df.Name == 'Smith')]
     Name
                 Age
                          Gender
                                      Address
   1 Smith
                 35
                       male
                                      Boston
-Column name이 Name과 Gender인 column만 가져오기
   df.filter(items=['Name', 'Gender'])
```

750

751

752

753

754

755 756

757 758

759 760

761

762

763

764

765

766

767 768 769

770

772

773

774

775

776

777 778

779 780

781

782

783

784

785

786

787

788 789

790 791

796 797

798

799

800 801

802

803

804

805 806

807 808

809

810

811

812

813 814

815 816

817

818

819

820 821

822 823

824 825

826

827 828 829

830 831

```
0
834
                  John
                             male
835
             1
                  Smith
                             male
836
                  Jenny
                             female
837
838
           -Column name이 'A'라는 글자가 있는 Column만 가져오기
839
840
             df.filter(like = 'A')
841
842
                     Age
                             Address
843
             0
                          Chicago
844
                          Boston
             1
                     35
845
             2
                     45
                          Dallas
846
847
          -정규식을 이용하여 column name이 'e'로 끝나는 Column만 가져오기
848
849
             df.filter(regex = 'e$')
850
851
                     Name
                                Age
852
             0
                     John
                                25
                                35
853
                     Smith
             1
854
                                45
                     Jenny
855
856
          -정규식을 이용하여 column name이 'A'로 시작하는 Column만 가져오기
857
858
             df.filter(regex = '^A')
859
860
                Address Age
             0 Chicago 25
861
             1 Boston 35
862
             2 Dallas 45
863
864
865
866
867
     8. 데이터 삭제하기
868
        1)drop() 함수는 데이터 구조의 row(행) 또는 column(열) 요소를 삭제한다.
869
        2)Syntax
870
          DataFrame.drop(labels=None, axis=0, inplace=False)
871
872
           -labels : 삭제할 index 또는 컬럼의 이름.
873
           -axis : int 타입 또는 축의 이름.
874
             --(0 또는 'index') 와 (1 또는 'columns') 중 하나
875
             --1이면 열을 삭제.
           -inplace : bool 타입.
876
877
             --False(기본값)이면 삭제된 결과 DataFrame을 리턴하며, True 이면 현재 DataFrame에서 데이터를 삭제하고 None을 반환.
878
879
        3)row index를 사용하여 삭제하기
880
881
          user list = [
              {'Age':25, 'Gender' : 'male', 'Address':'Chicago'},
{'Age':35, 'Gender' : 'male', 'Address':'Boston'},
{'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
882
883
884
885
          ]
886
887
          df = pd.DataFrame(user_list,
888
                             index = ['John', 'Smith', 'Jenny'],
889
                             columns = ['Age', 'Gender', 'Address'])
890
891
          df.head()
892
          -----
893
                  Age
                       Gender
                                     Address
894
          John
                  25
                       male
                                     Chicago
895
          Smith 35
                        male
896
          Jenny 45
                       female
                                     Dallas
897
898
          df.drop(['John', 'Jenny'])
899
900
                     Age
                           Gender
                                      Address
901
          Smith
                     35
                          male
                                        Boston
902
903
          df.head() #하지만 여전히 df는 예전값을 갖고 있다.
904
905
          df = df.drop(['John', 'Jenny']) #이렇게 하면 제거된 결과를 df가 갖게 된다.
906
          df.head()
907
908
                           Gender Address
                     Age
909
          Smith
                     35
                          male
                                        Boston
910
911
912
        4)inplace keyword 인수 이용하기
913
914
          df = pd.DataFrame(user_list,
915
                             index = ['John', 'Smith', 'Jenny'],
916
                             columns = ['Age', 'Gender', 'Address'])
```

Name

Gender

```
918
          df.drop(['John', 'Jenny'], inplace=True)
919
920
                    Age Gender Address
921
922
          Smith
                    35 male
                                      Boston
923
924
925
       5)row가 숫자로 indexing되어 있을 때 삭제하기
926
927
          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'}
928
929
930
931
932
933
          df = pd.DataFrame(user_list)
934
935
          df = df[['Name', 'Age', 'Gender','Address']]
936
937
          df
938
939
               Name
                         Age Gender
                                           Address
940
          0
               John
                         25 male
                                           Chicago
941
               Smith
                         35
                              male
                                           Boston
                         45
                                           Dallas
942
          2
                             female
               Jenny
943
944
          df = df.drop(df.index[[0, 2]])
945
946
            Name Age Gender
947
                                        Address
948
                      35 male
949
950
951
       6)Condition으로 삭제하기
952
953
          df.head()
954
955
                 Name
                            Age Gender
                                             Address
956
          0
                 John
                            25
                                male
                                              Chicago
957
          1
                  Smith
                            35
                                 male
                                              Boston
958
                 Jenny
                            45
                                female
                                              Dallas
959
960
          -Age가 30 이상인 사람의 정보만 저장함으로 나머지 정보는 삭제하는 방법
961
962
            df = df[df.Age > 30]
963
964
965
                            Age
                                 Gender
                 Name
                                             Address
966
                            35
                                             Boston
             1
                 Smith
                                male
967
             2
                  Jenny
                            45
                                female
                                             Dallas
968
969
970
       7)앞에서 사용했던 member_data.csv 파일 데이터를 이용해 보자.
971
972
          member_df = pd.read_csv("member_data.csv")
          member_df.columns = ["이름", "나이", "이메일", "주소"]
member_df.index = ["동", "서", "남", "북"]
973
974
975
          member df
976
          _____
977
               이름 나이 이메일
978
               홍길동 20 kildong@hong.com
                                                서울시 강동구
                                               서울시 강서구
979
               홍길서 25
                         kilseo@hong.com
          서
                         south@hong.com
980
               홍길남 26
          남
                                                  서울시 강남구
981
               홍길북 27
                         book@hong.com
                                                서울시 강북구
982
       8)단일 행 삭제하기
983
984
985
          member_df = member_df.drop('북') #axis=0(기본값)이면 행에서 찾아 삭제
986
          member_df
987
               이름 나이 이메일
988
               홍길동 20 kildong@hong.com 서울시 강동구
989
               홍길서 25
990
                         kilseo@hong.com
                                              서울시 강서구
991
               홍길남 26
                         south@hong.com
          남
                                                서울시 강남구
992
993
          -행의 이름을 지정하지 않았다면 기본값은 아마도 숫자일 것이다.
994
          -이 경우 행 번호가 행의 이름이 된다.
995
996
            member_df2 = pd.read_csv("member_data.csv")
997
            member_df2.drop(2, axis=0)
998
999
```

```
1002
1003
             member_df = member_df.drop('주소',axis=1)
1004
             member_df
1005
             -----
1006
                  이름 나이 이메일
1007
                  홍길동 20 kildong@hong.com
1008
                  홍길서 25 kilseo@hong.com
1009
             남
                  홍길남 26 south@hong.com
1010
1011
1012
             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'}
1013
1014
1015
1016
             1
1017
1018
             df = pd.DataFrame(user list)
             df = df[['Name', 'Age', 'Gender','Address']]
1019
1020
             df.head()
1021
1022
                    Name Age Gender
                                               Address
             0
                    John
                              25
                                               Chicago
1023
                                   male
1024
                    Smith
                              35
                                               Boston
                                   male
             1
1025
             2
                    Jenny
                              45
                                   female
                                               Dallas
1026
1027
             df.drop('Age', axis= 1)
1028
1029
                  Name
                            Gender
                                   Address
1030
             0
                  John
                                        Chicago
                            male
1031
                                        Boston
                  Smith
                            male
             1
1032
                  Jenny
                            female
                                        Dallas
1033
             df.drop('Age', axis= 1, inplace=True)
1034
1035
1036
             _____
                  Name Gender Address
1037
1038
             0
                  lohn
                           male
                                       Chicago
1039
             1
                  Smith
                            male
                                        Boston
1040
             2
                  Jenny
                            female
                                        Dallas
1041
1042
1043
        10) 복수 열 삭제하기
1044
           -여러 개 열을 삭제하려면 labels 인자를 이용한다.
1045
           -axis 인자가 1일 경우 열을 의미한다.
1046
1047
             member_df.drop(labels=["이메일", "주소"], axis=1)
1048
1049
           -axis=1과 axls="columns"는 같은 의미이다.
1050
           위의 코드와 다음 코드 실행 결과는 같다.
1051
1052
             member_df.drop(labels=["이메일", "주소"], axis="columns")
1053
1054
                  이름 나이
1055
                  홍길동 20
1056
                  홍길서 25
             서
1057
                  홍길남 26
             남
1058
             북
                  홍길북 27
1059
1060
1061
      9. DataFrame 요소 추가
1062
1063
        1)열 추가
1064
           -DataFrame에 없는 열을 지정해서 값을 할당하면 새로운 열이 만들어 진다.
1065
           -이 때 열은 가장 오른쪽에 만들어진다.
1066
             member_df = pd.read_csv("member_data.csv")
1067
1068
             member_df["BirthYear"] = 2000
1069
             member_df
1070
             _____
                           Age Email
1071
                                                           Address
                                                                        BirthYear
                  Name
             0
                         20 kildong@hong.com
1072
                  홍길동
                                                 서울시 강동구 2000
                                                서울시 강서구
1073
                         25
                              kilseo@hong.com
             1
                                                                2000
                  홍길서
                              south@hong.com
                                                                2000
1074
             2
                         26
                  홍길남
                                                   서울시 강남구
1075
                         27
                              book@hong.com
                                                 서울시 강북구
                                                              2000
                  홍길북
1076
1077
           -열을 추가할 때 행별로 다른 값을 지정할 수 있다.
1078
             member_df = pd.read_csv("member_data.csv")
1079
             member_df["BirthYear"] = [2001, 2002, 2003, 2004]
1080
1081
             member_df
1082
                          Age Email
1083
                                                           Address
                                                                        BirthYear
                  Name
             0
                         20 kildong@hong.com 서울시 강동구
1084
                                                              2001
```

-axis인자의 값이 1이면 열을 삭제한다.

```
1085
             1
                          25
                               kilseo@hong.com
                                                                   2002
                   홍길서
                                                      서울시 강서구
1086
             2
                          26
                               south@hong.com
                                                                  2003
                   홍길남
                                                      서울시 강남구
1087
                                                                2004
                          27
                               book@hong.com
                                                   서울시 강북구
                   홍길북
1088
1089
           -만일 누락되어야 하는 값이 있다면 None으로 지정한다.
1090
             member_df = pd.read_csv("member_data.csv")
1091
             member_df["BirthYear"] = [2001, 2002, 2003, None]
1092
1093
             member df
1094
1095
                           Age Email
                                                                          BirthYear
                  Name
                                                             Address
                          20 kildong@hong.com
1096
             0
                                                   서울시 강동구 2001.0
                  홍길동
1097
             1
                          25
                               kilseo@hong.com 서울시 강서구
                                                                  2002.0
                   홍길서
                               south@hong.com
1098
                          26
             2
                                                                   2003.0
                   홍길남
                                                     서울시 강남구
1099
              3
                          27
                               book@hong.com
                                                   서울시 강북구
1100
1101
           -None 값을 포함하면 NaN은 실수 유형으로 간주되기 때문에 정수 자료형 값들은 모두 실수 자료형으로 바뀌어 저장된다.
1102
1103
         2)Series를 이용한 열 추가
1104
           -Series를 이용하면 index의 이름(행번호)을 지정해서 값을 할당할 수 있다.
1105
1106
                       = pd.read_csv("member_data.csv")
           member_df
1107
           member df["BirthYear"] = pd.Series([2001,2002,
                                                          2004], index=[0,1,3])
           member_df
1108
1109
                                                          Address
1110
                                Email
                Name
                          Age
                                                                       BirthYear
1111
             0
                  홍길동
                          20 kildong@hong.com
                                                   서울시 강동구 2001.0
1112
             1
                          25
                               kilseo@hong.com
                                                                2002.0
                                                   서울시 강서구
                   홍길서
1113
             2
                   홍길남
                          26
                               south@hong.com
                                                     서울시 강남구
                                                                  NaN
             3
                          27
                               book@hong.com
                                                                2004.0
1114
                   홍길북
                                                   서울시 강북구
1115
1116
         3)dict로 행 추가
1117
           -DataFrame의 행으로 새로운 데이터를 추가할 수 있다.
1118
           -추가할 데이터가 다음처럼 dict로 되어 있다면 쉽게 DataFrame에 행으로 추가할 수 있다.
1119
1120
              member_df = pd.read_csv("member_data.csv")
1121
             new_member = {"Name": "한지민", "Age": 23, "Email": "jimin@naver.com", "Address": "서울시 송파구"}
1122
1123
           -추가할 데이터가 Series가 아니라면 ignore_index=True를 설정해야 한다.
1124
1125
             new_df = member_df.append(new_member, ignore_index=True)
1126
             new df
1127
                   Name Age Email
1128
                                                             Address
                          20 kildong@hong.com
1129
             0
                  홍길동
                                                   서울시 강동구
                                                   서울시 강서구
1130
                   홍길서
                          25
                               kilseo@hong.com
1131
             2
                          26
                               south@hong.com
                                                     서울시 강남구
                   홍길남
1132
             3
                          27
                               book@hong.com
                   홍길북
                                                   서울시 강북구
1133
                               jimin@naver.com
                   한지민
                                                      서울시 송파구
1134
1135
         4)다른 예제로 연습해 보자.
1136
1137
           user_list = [
                 {'Name':'John', 'Age':15, 'Gender' : 'male', 'Address':'Chicago', 'Job' : 'Student'}, {'Name':'Smith', 'Age':25, 'Gender' : 'male', 'Address':'Boston', 'Job' : 'Teachter'}, {'Name':'Jenny', 'Age':17, 'Gender' : 'female', 'Address':'Dallas', 'Job' : 'Student'}
1138
1139
1140
1141
1142
1143
           df = pd.DataFrame(user list)
           df = df[['Name', 'Age', 'Gender', 'Address', 'Job']]
1144
1145
           df.head()
1146
1147
                Name Age Gender Address
                                                        Job
                       15 male Chicago
1148
           0
                lohn
                                                   Student
                          25 male
1149
           1
                Smith
                                         Boston
                                                      Teachter
1150
                Jenny
                          17
                              female
                                         Dallas
                                                      Student
1151
1152
           -column 새로 추가하기
             df['Salary'] = 0
1153
1154
1155
             df.head()
1156
1157
                   Name
                            Age
                                  Gender Address
                                                                       Salary
                                                                     Λ
1158
             0
                  lohn
                            15
                                 male
                                              Chicago
                                                        Student
1159
                   Smith
                             25
                                  male
                                              Boston
                                                           Teacher
                                                                        0
1160
                                                           Student
             2
                             17
                                              Dallas
                  Jenny
                                  female
                                                                        0
1161
1162
           -기존 column 값을 이용
           -'Job'에 따라 'Salary' 여부 Column으로 수정
1163
1164
1165
              df['Salary'] = np.where(df['Job'] != 'Student', 'yes', 'no')
1166
              df.head()
1167
1168
                  Name
                            Age Gender Address Job
                                                                   Salary
```

```
1169
               0
                     John
                                15
                                      male
                                                    Chicago Student
1170
                     Smith
                                25
                                      male
                                                    Boston
                                                                Teacher
               1
                                                                           ves
1171
                                17
                     Jenny
                                      female
                                                    Dallas
                                                                Student
                                                                           no
1172
1173
             -'Total' column 추가
1174
1175
               student list = [
                   {'Name':'John', 'Midterm':95, 'Final': 85},
1176
                   {'Name':'Smith', 'Midterm':85, 'Final' : 80}, 
{'Name':'Jenny', 'Midterm':30, 'Final' : 10},
1177
1178
1179
1180
1181
               df = pd.DataFrame(student_list, columns= ['Name', 'Midterm', 'Final'])
1182
1183
               df.head()
1184
1185
                             Midterm Final
                  Name
                              95
1186
               0 John
                              85
1187
               1 Smith
                                      80
1188
                 Jenny
                              30
                                      10
1189
               df['Total'] = df['Midterm'] + df['Final']
1190
1191
1192
1193
                              Midterm Final Total
                              95
                                      85
1194
               0 John
                                           180
1195
               1
                  Smith
                              85
                                      80
                                           165
1196
               2 Jenny
                              30
                                      10
                                           40
1197
1198
            -'Average' column 추가하기
1199
1200
               df['Average'] = df['Total'] / 2
1201
               df.head()
1202
1203
                  Name
                              Midterm Final
                                                Total
                                                          Average
                              95
1204
               0 John
                                         85
                                              180
                                                          90.0
1205
                  Smith
                              85
                                         80
                                              165
                                                          82.5
1206
                             30
                                         10
                                              40
                                                       20.0
               2 Jenny
1207
1208
             -'Grade' column 추가하기
1209
1210
               grade list = []
               for row in df['Average']:
1211
1212
                   if row <= 100 and row >= 90:
1213
                      grade_list.append('A')
1214
                   elif row < 90 and row >= 80:
1215
                      grade_list.append('B')
                   elif row < 80 and row > = 70 :
1216
1217
                      grade list.append('C')
                   elif row < 70 and row >= 60:
1218
1219
                      grade_list.append('D')
1220
                   else: grade_list.append('F')
1221
1222
               df['Grade'] = grade_list
1223
               df
1224
1225
                             Midterm Final
                  Name
                                                 Total
                                                          Average Grade
1226
               0 John
                             95
                                      85
                                               180
                                                          90.0
                                                                     Α
1227
               1 Smith
                              85
                                      80
                                               165
                                                          82.5
                                                                     В
                                      10
                                                                   F
                              30
                                                       20.0
1228
               2 Jenny
                                               40
1229
1230
1231
1232
       10. 정렬
1233
          1)DataFrame을 정렬하려면 sort_index() 또는 sort_value() 함수를 이용한다.
1234
          2)정렬의 경우 에도 inplace 인자를 사용할 수 있다.
1235
          3)inplace=True인 경우 원본 데이터프레임이 변경된다.
1236
1237
1238
             DataFrame.sort_index(axis=0, level=None, ascending=True,
1239
                                                  inplace=False, kind='quicksort',
1240
                                                  na_position='last', sort_remaining=True,
1241
                                                  by=None)
            DataFrame.sort_values(by, axis=0, ascending=True, inplace=False,
1242
1243
               kind='quicksort', na_position='last')
1244
1245
             -label : 정렬할 level을 지정.
1246
             -axis : int 타입 또는 축의 이름. (0 또는 'index') 와 (1 또는 'columns') 중 하나
1247
             -ascending : True(기본값) 이면 오름차순, False 이면 내림차순으로 정렬.
1248
             -inplace: bool 타입이며, False(기본값)이면 삭제된 결과 DataFrame을 리턴하며, True 이면 현재 DataFrame에서 데이터를 삭제하고 None을 반환.
             -kind : 정렬 알고리즘을 지정.
1249
            -- 저희 발고리듬을 시청.
-- 저럴 방법은 "quicksort", "mergesort", "heapsort" 중 하나.
--기본값은 "quicksort"이며 이 옵션은 단일 열 또는 단일 index를 정렬할 때만 적용.
-na_position: NaN 값을 놓을 위치를 "first" 또는 "last"로 지정.
1250
1251
1252
```

```
1253
            --기본값은 NaN을 마지막에 두는 "last".
          -sort_remaining : 만일 True(기본값)이고 레벨(level)로 정렬하며 index가 멀티레벨일 경우 지정한 레벨로 정렬 할 후 다른 레벨을 이용해도 정렬.
1254
1255
          -by : sort_values() 함수에서 by 인자는 정렬의 기준이 되는 열 또는 행의 이름을 지정.
1256
            member_df = pd.read_csv("member_data.csv")
1257
1258
            member_df.index = ["동", "서", "남", "북"]
1259
            member df
1260
1261
                 Name
                         Age
                               Email
                                                       Address
                       20 kildong@hong.com
1262
                 홍길동
                                              서울시 강동구
1263
                       25
                            kilseo@hong.com
            서
                 홍길서
                                                서울시 강서구
1264
                       26
                            south@hong.com
            남
                 홍길남
                                                서울시 강남구
1265
                       27
                            book@hong.com
                 홍길북
                                              서울시 강북구
1266
1267
1268
        5)행 이름으로 정렬
1269
          -DataFrame을 index 기준으로 정렬하려면 sort_index()를 이용.
1270
          -sort index() 함수는 DataFrame의 행 이름을 이용해서 정렬.
1271
1272
            member_df.sort_index()
1273
                                                       Address
1274
                 Name
                                Email
1275
                 홍길남
                       26
                            south@hong.com
                                                서울시 강남구
            남
1276
                       20
                            kildong@hong.com
            동
                 홍길동
                                              서울시 강동구
1277
                       27
                            book@hong.com
            보
                 홍길북
                                              서울시 강북구
                       25
                            kilseo@hong.com
1278
            서
                                                서울시 강서구
                 홍길서
1279
1280
1281
       6)열 이름으로 열 순서 바꾸기
1282
          -DataFrame을 열 이름을 기준으로 정렬하려면 sort_index(axis=1)를 이용.
1283
1284
            member_df.sort_index(axis=1)
1285
1286
                   Address
                              Age
                                     Email
                                                            Name
1287
                              20
                                   kildong@hong.com
            동
                   서울시 강동구
                                                     홍길동
1288
            서
                   서울시 강서구
                              25
                                   kilseo@hong.com
                                                       홍길서
1289
            남
                   서울시 강남구
                              26
                                   south@hong.com
                                                       홍길남
1290
                                   book@hong.com
            북
                   서울시 강북구
                              27
                                                     홍길북
1291
1292
1293
       7)값으로 정렬
1294
          -DataFrame의 값을 기준으로 정렬하려면 sort values()를 이용.
1295
1296
            member_df.sort_values(by=["Email"])
1297
1298
                 Name
                          Age
                               Email
                                                       Address
1299
                       27
                            book@hong.com
                                              서울시 강북구
                 홍깈북
            보
1300
                       20
            동
                 홍길동
                            kildong@hong.com
                                              서울시 강동구
1301
                       25
            서
                            kilseo@hong.com
                                                서울시 강서구
                 홍길서
                            south@hong.com
1302
                 홍길남
                       26
                                                서울시 강남구
1303
          -여러 열을 이용해서 정렬 하고 싶을 때는 by 인자의 값을 여러 column 이름을 갖는 list 형식으로 지정하면 된다.
1304
1305
1306
1307
        8)level로 정렬
1308
          -DataFrame의 열 이름 또는 행 이름에 level이 지정되어 있을 경우 level로 정렬할 수 있다.
1309
1310
            member df
                       = pd.read_csv("member_data.csv")
1311
            member_df.columns = [["기본정보", "기본정보", "추가정보", "추가정보"], ["이름", "나이", "이메일", "주소"]]
            1312
1313
1314
1315
            member_df
1316
1317
                 정보구분 기본정보
                                       추가정보
1318
                 상세정보
                              이름
                                       나이
                                              이메일
                                                                       주소
1319
                 위치구분
                       상세위치
1320
                                     20
                                         kildong@hong.com
            좌우
                       동
                              홍길동
                                                              서울시 강동구
                                         kilseo@hong.com
1321
                                     25
                       서
                              홍길서
                                                                서울시 강서구
1322
                              홍길남
                                     26
                                          south@hong.com
                                                                서울시 강남구
                       남
1323
                                     27
                                         book@hong.com
                       북
                              홍길북
                                                              서울시 강북구
1324
1325
            member_df.sort_index(level=["위치구분"])
1326
1327
                     정보구분
                              기본정보 추가정보
1328
            상세정보
                              이름
                                    나이 이메일
                                                              주소
1329
            위치구분
                     상세위치
1330
                              홍길남 26 south@hong.com
                                                       서울시 강남구
            상하
                     남
                              홍길북 27 book@hong.com 서울시 강북구
1331
                     북
                              홍길동 20 kildong@hong.com 서울시 강동구
1332
            좌우
                     동
1333
                              홍길서 25 kilseo@hong.com
                                                       서울시 강서구
1334
1335
            member_df.sort_index(level=["상세위치"])
1336
```

```
1337
                    정보구분 기본정보 추가정보
1338
                    상세정보
                              이름 나이 이메일
                                                                  주소
1339
             위치구분 상세위치
1340
             상하
                              홍길남 26 south@hong.com
                                                           서울시 강남구
                    남
1341
            좌우
                             홍길동 20 kildong@hong.com 서울시 강동구
1342
                              홍길북 27 book@hong.com
                                                       서울시 강북구
             상하
                    북
1343
                              홍길서 25 kilseo@hong.com
             좌우
                                                          서울시 강서구
1344
1345
1346
1347
      11. 데이터 그룹화 및 집계
1348
        1)Group by
1349
           -groupby() 함수는 데이터를 구분 할 수 있는 열(column)의 값들을 이용하여 데이터를 여러 기준에 의해 구분하여 그룹화 한 후 기초 통계 함수 등을 적용 할 수
           있도록 한다.
1350
           -Syntax
1351
1352
             DataFrame.groupby(by=None, axis=0, level=None, as_index=True,
1353
                                          sort=True, group_keys=True, squeeze=False,
                                          observed=False, **kwargs)
1354
1355
1356
             import statsmodels.api as sm
             iris = sm.datasets.get_rdataset("iris", package="datasets")
1357
             iris_df = iris.data
1358
1359
             iris_df.head()
1360
                                                             Petal.Width
1361
                  Sepal.Length Sepal.Width Petal.Length
                                                                            Species
1362
                  5.1
                                     3.5
                                                        1.4
                                                                            0.2
                                                                                             setosa
1363
                  4.9
                                     3.0
                                                                            0.2
             1
                                                        1.4
                                                                                             setosa
1364
                 4.7
                                     3.2
                                                         1.3
                                                                            0.2
                                                                                             setosa
1365
                  4.6
                                     3.1
                                                         1.5
                                                                            0.2
                                                                                             setosa
1366
                  5.0
                                                                            0.2
                                     3.6
                                                         1.4
                                                                                             setosa
1367
1368
1369
        2)단일 열로 그룹화
1370
          -groupby() 함수를 이용하여 그룹화 할 열을 지정.
1371
           -그룹이 지정되면 그 그룹에 기초통계 분석 함수를 사용하면 된다.
1372
1373
             iris_grouped = iris_df.groupby(iris_df.Species)
1374
1375
1376
             <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001F477FF7808>
1377
1378
                                    #평균
             iris_grouped.mean()
1379
                           Sepal.Length Sepal.Width
1380
                                                        Petal.Length
                                                                       Petal.Width
1381
             Species
1382
                         5.006
                                            3.428
                                                                1.462
                                                                                   0.246
             setosa
             versicolor 5.936
                                             2.770
1383
                                                                                   1.326
                                                                4.260
1384
                                          2.974
                                                              5.552
             virginica 6.588
                                                                                 2.026
1385
1386
1387
        3)다중 열로 그룹화
1388
           -그룹을 두 가지 이상으로 지정하고 싶을 때는 list를 이용해 그룹을 지정하면 된다.
1389
           -열 이름에 점(.)이 포함되었다면 df["열이름"] 형식으로 열을 지정해야 한다.
1390
           -다음 코드는 iris 데이터를 종(Species)별, 꽃받침 조각의 길이(Sepal.Length)별로 그룹화 하는 예이다.
1391
1392
             iris_grouped2 = iris_df.groupby([iris_df.Species,
1393
             iris_df["Sepal.Length"]])
1394
             iris_grouped2
1395
1396
             <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001F479369C08>
1397
1398
             iris_grouped2.mean().head()
1399
1400
                                            Sepal.Width
                                                           Petal.Length
                                                                         Petal.Width
1401
             Species
                       Sepal.Length
                                          3.000000
                                                           1.100000
1402
             setosa
                       4.3
                                                                            0.100
1403
                         4.4
                                            3.033333
                                                             1.333333
                                                                              0.200
1404
                                                             1.300000
                         4.5
                                            2.300000
                                                                              0.300
1405
                         4.6
                                            3.325000
                                                            1.325000
                                                                              0.225
                                                                              0.200
1406
                         4.7
                                             3.200000
                                                            1.450000
1407
1408
           -그룹된 객체를 이용해 요약정보를 볼 수도 있다.
1409
1410
             iris_grouped.describe()
1411
```

-요약정보에서 생략된 부분(...)의 내용을 보려면 디스플레이 옵션을 설정한다.

pd.options.display.max_columns = 999

4)그룹간 반복 처리

1412

1413 1414

1415 1416

1417

1418 1419 -그룹화 된 데이터에서 그룹의 타입과 그룹 객체를 반복문을 이용해 처리할 수 있다.

for type, group in iris_grouped:

```
1420
                     print(type, '\n', group.head())
1421
1422
1423
            5)DataFrame group indexing
               -DataFrame group에서 indexing을 위해서는 take() 함수를 이용.
1424
1425
               -Svntax
1426
                  DataFrameGroupBy.take(indices, axis, is copy)
1427
               -indices : 가져올 index를 list 형식으로 지정.
1428
1429
               -axis : 기본값 0이면 행 index를 이용해 가져오고, 1이면 열 index를 이용해 가져온다.
1430
               -is_copy: 기본값 True이면 객체의 복사본이 반환.
1431
1432
               -다음 코드는 iris 데이터를 불러와 종(Species) 별로 grouping한 결과에서 부분집합을 가져온다.
1433
1434
                  import statsmodels.api as sm
                  iris = sm.datasets.get_rdataset("iris", package="datasets")
1435
                  iris df = iris.data
1436
1437
                  iris df grouped = iris df.groupby(iris df.Species)
1438
1439
               -다음 코드는 각 종별로 0, 1, 2행을 가져온다.
1440
1441
                  iris_df_grouped.take([0,1,2])
1442
1443
               -연속적인 index는 range() 함수를 이용할 수 있다.
1444
1445
                  iris_df_grouped.take(range(0,3))
1446
1447
                                                   Sepal.Length
                                                                       Sepal.Width Petal.Length Petal.Width
1448
                  Species
1449
                                                                                                                         0.2
                  setosa
                                      0
                                                                                                                            0.2
1450
                                                                             3.0
                                                                                                     1 4
                                         1
1451
                                         2
                                                                                                                            0.2
                                                                             3.2
1452
                                      50
                                            7.0
                                                                       3.2
                                                                                              4.7
                                                                                                                     1.4
                  versicolor
                                         51
                                                                          3.2
                                                                                                 4.5
1453
                                               6.4
                                                                                                                        1.5
                                             6.9
1454
                                         52
                                                                          3.1
                                                                                                 4.9
                                                                                                                        1.5
1455
                  virginica
                                  100
                                            6.3
                                                                       3.3
                                                                                              6.0
                                                                                                                     2.5
1456
                                         101
                                                   5.8
                                                                             2.7
                                                                                                     5.1
                                                                                                                            1.9
1457
                                         102
                                                   7.1
                                                                             3.0
                                                                                                     5.9
                                                                                                                            2.1
1458
1459
           6)좀 더 쉬운 예제로 Group By를 다뤄보자.
1460
1461
               student list = [
                     {'Name': 'John', 'Major': "Computer Science", 'Gender': "male"},
{'Name': 'Nate', 'Major': "Computer Science", 'Gender': "male"},
1462
1463
                     {'Name': 'Abraham', 'Major': "Physics", 'Gender': "male"}, {'Name': 'Brian', 'Major': "Psychology", 'Gender': "male"}, {'Name': 'Janny', 'Major': "Economics", 'Gender': "female"}, {'Name': 'Yuna', 'Major': "Economics", 'Gender': "female"},
1464
1465
1466
1467
                     {'Name': 'Jeniffer', 'Major': "Computer Science", '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"},
1468
1469
1470
1471
                     {'Name': 'Sera', 'Major': "Psychology", 'Gender': "female"}
1472
1473
               ]
1474
1475
               df = pd.DataFrame(student_list, columns = ['Name', 'Major', 'Gender'])
1476
               df
1477
1478
                     Name
                                  Maior
                                                             Gender
               0
                                  Computer Science
1479
                     John
                                                             male
                                   Computer Science
1480
               1
                     Nate
                                                             male
1481
               2
                     Abraham Physics
                                                             male
1482
               3
                              Psychology
                                                          male
                     Brian
1483
               4
                     Janny
                                  Economics
                                                            female
               5
1484
                     Yuna
                               Economics
                                                          female
                     Jeniffer Computer Science
1485
               6
                                                          female
               7
                     Edward Computer Science
1486
                                                          male
1487
               8
                                  Psychology
                     Zara
                                                             female
1488
              9
                     Wendy
                                                             female
                                  Economics
1489
               10 Sera
                               Psychology
                                                          female
1490
1491
               -전공별 Group By
1492
                  groupby major = df.groupby('Major')
1493
1494
                  groupby_major.groups
1495
                  {'Computer Science': Int64Index([0, 1, 6, 7], dtype='int64'),
1496
1497
                   'Economics': Int64Index([4, 5, 9], dtype='int64'),
1498
                   'Physics': Int64Index([2], dtype='int64'),
1499
                   'Psychology': Int64Index([3, 8, 10], dtype='int64')}
1500
1501
               -보기 좋게
1502
                  for name, group in groupby_major:
    print(name + ": " + str(len(group)))
1503
```

```
1504
                print(group)
1505
                print()
1506
                              -----
1507
              Computer Science: 4
1508
                   Name
                          Major
                                                  Gender
1509
              0
                   John
                             Computer Science
                                                  male
1510
                             Computer Science
              1
                   Nate
                                                  male
                                               female
1511
              6
                Jeniffer
                           Computer Science
                   Edward
1512
                             Computer Science
                                                  male
1513
1514
              Economics: 3
1515
                                          Gender
                   Name
                             Major
1516
                           Economics
                                       female
                Janny
1517
                             Economics female
                   Yuna
              9 Wendy Economics
1518
                                     female
1519
1520
              Physics: 1
1521
                   Name
                             Major
                                             Gender
              2 Abraham Physics
                                     male
1522
1523
1524
              Psychology: 3
1525
                   Name
                             Major
                                             Gender
                             Psychology
1526
                   Brian
                                          male
              8
                             Psychology
1527
                   Zara
                                          female
1528
              10
                   Sera
                             Psychology
                                          female
1529
1530
           -전공별 명수
1531
1532
              df_major_cnt = pd.DataFrame({'Count':groupby_major.size()})
1533
              df_major_cnt
1534
1535
                                     Count
1536
              Maior
              Computer Science
                                  4
1537
1538
                                   3
              Economics
1539
              Physics
                                  1
1540
              Psychology
                                   3
1541
1542
           -전공도 count와 같이
1543
1544
              df_major_cnt = pd.DataFrame({'Count':groupby_major.size()}).reset_index()
1545
              df major cnt
1546
1547
                                        Count
                   Computer Science
1548
              0
                                        4
1549
              1
                   Economics
                                        3
1550
              2
                   Physics
                                        1
1551
              3
                                        3
                   Psychology
1552
1553
           -성별로 group by
1554
1555
              groupby_gender = df.groupby('Gender')
1556
              for name, group in groupby_gender:
    print(name + ": " + str(len(group)))
1557
1558
1559
                print(group)
1560
                print()
1561
1562
              female: 6
                                                     Gender
1563
                   Name
                             Major
1564
                   Janny
                             Economics
                                               female
1565
              5
                   Yuna
                            Economics
                                               female
1566
              6
                   Jeniffer
                             Computer Science
                                                  female
                             Psychology
1567
              8
                                                  female
                   7ara
              9
1568
                   Wendy
                            Economics
                                               female
1569
              10
                   Sera
                             Psychology
                                                  female
1570
1571
              male: 5
1572
                   Name
                             Major
                                                    Gender
1573
              0
                   John
                             Computer Science
                                                  male
1574
                   Nate
                             Computer Science
                                                  male
              1
1575
              2
                Abraham Physics
                                             male
1576
                             Psychology
                   Brian
                                                  male
1577
              7
                             Computer Science
                   Edward
                                                  male
1578
1579
1580
      12. DataFrame에 함수 적용하기
1581
1582
         -DataFrame의 데이터에서 합계나 평균 등 일반적인 통계는 DataFrame의 함수들을 사용하면 되지만, 판다스에서 제공하지 않는 기능을 커스텀 함수(custom
         function)로 구현해서 DataFrame에 적용하려면 apply(), applymap(), map() 등의 함수를 사용한다.
1583
         -적용할 함수의 이름은 다음과 같다.
1584
           method
                      통용대상
                                               반환값
1585
                        Series(값별)
           map
                                               Series
1586
           apply
                      DataFrame(열 또는 행별) Series
```

```
1587
           applymap
                        DataFrame(값별)
                                               DataFrame
1588
1589
         1)apply()
1590
           -DataFrame에 사용자 정의 함수를 적용하기 위해서 apply() 함수를 사용한다.
1591
           -Syntax
1592
              DataFrame.apply(func, axis=0, raw=False,
1593
                                       result_type=None, args=(), **kwds)
1594
           -func : 각 열 또는 행에 적용할 함수.
1595
           -axis : 함수가 적용될 축.
1596
             --기본값(0 또는 'index')이면 각 열 별로 함수가 적용되며, 1 또는 'columns'이면 각 행 별로 함수가 적용.
1597
1598
             --False(기본값)일 경우 각 행이나 열을 Series로 함수에 전달.
           --True이면 전달 된 함수는 대신 ndarray 객체를 받는다.
-result_type : 'expand', 'reduce', 'broadcast', None 중 하나를 사용.
1599
1600
1601
             --기본값은 None.
1602
              --이것은 axis=1(columns)인 경우에만 작동.
1603
             --'expand' : 목록과 같은 결과가 열로 바뀐다.
1604
             --'reduce' : 목록과 같은 결과를 확장하지 않고 가능한 경우 Series를 반환한다.
1605
                ---이것은 '확장'의 반대.
1606
              --'broadcast' : 결과가 DataFrame의 원래 모양으로 브로드 캐스팅되고 원본 인덱스와 열은 유지된다.
1607
             --None : 적용 함수의 반환 값에 따라 다르다.
1608
                ---목록 같은 결과는 일련의 결과로 반환된다.
1609
                ---그러나 apply 함수가 시리즈를 리턴하면 이들은 열로 확장된다.
1610
           -args : 배열/시리즈 외에도 func에 전달할 위치 인수를 tuple형식으로 지정.
1611
           -**kwds: func에 전달할 추가 키워드 인수를 지정.
1612
1613
              import statsmodels.api as sm
1614
             iris = sm.datasets.get_rdataset("iris", package="datasets")
             iris_df = iris.data
1615
1616
             iris_df.head()
1617
1618
                   Sepal.Length
                                  Sepal.Width Petal.Length
                                                                Petal.Width
                                                                                Species
1619
             0
                   5.1
                                       3.5
                                                                                0.2
                                                           1.4
                                                                                                  setosa
                                                                                0.2
1620
             1
                   4.9
                                       3.0
                                                           1.4
                                                                                                  setosa
                                                                                0.2
1621
             2
                   4.7
                                       3.2
                                                           1.3
                                                                                                  setosa
1622
              3
                   4.6
                                       3.1
                                                           1.5
                                                                                0.2
                                                                                                  setosa
1623
              4
                   5.0
                                       3.6
                                                                                0.2
                                                                                                  setosa
1624
1625
           -iris 데이터에서 종(Species) 정보를 제외한 나머지 열 정보를 조회한다.
1626
           -이것은 apply() 함수를 사용하기 전/후를 비교하기 위해서이다.
1627
1628
             iris df.iloc[:,:-1].head()
1629
                   Sepal.Length
1630
                                  Sepal.Width Petal.Length
                                                              Petal.Width
1631
              0
                   5.1
                                       3.5
                                                         1.4
                                                                             0.2
1632
              1
                   4.9
                                       3.0
                                                         1.4
                                                                             0.2
1633
              2
                   4.7
                                       3.2
                                                         1.3
                                                                             0.2
1634
              3
                   4.6
                                                                             0.2
                                       3.1
                                                         1.5
1635
              4
                   5.0
                                       3.6
                                                                             0.2
1636
1637
           -다음 코드는 iris 데이터에 np.round 함수를 적용해서 데이터를 반올림 합니다.
1638
1639
             import numpy as np
1640
             iris_df.iloc[:,:-1].apply(np.round).head()
1641
1642
                                  Sepal.Width Petal.Length Petal.Width
1643
             0
                   5.0
                                                                           0.0
                                       4.0
                                                        1.0
1644
              1
                   5.0
                                       3.0
                                                         1.0
                                                                           0.0
1645
             2
                   5.0
                                       3.0
                                                                           0.0
                                                         1.0
              3
1646
                   5.0
                                       3.0
                                                         2.0
                                                                           0.0
1647
              4
                   5.0
                                       4.0
                                                         1.0
1648
1649
           -apply() 함수에 사용하는 함수가 요소별로 동작하는 함수라면 위의 예에서처럼 각 요소에 함수를 적용한다.
1650
           -그러나 만일 함수가 요소별로 동작하지 않는 함수라면 axis 매개변수의 값에 따라 적용된 결과는 달라질 수 있다.
1651
           -다음 코드는 열 별 합계를 출력한다.
1652
1653
             iris_df.iloc[:,:-1].apply(np.sum) # 열 별 합계 출력
1654
1655
             Sepal.Length
                                  876.5
1656
              Sepal.Width
                                  458.6
1657
              Petal.Length
                                  563.7
1658
              Petal.Width
                                    179.9
1659
              dtype: float64
1660
1661
           -위의 코드는 다음 lambda을 사용한 코드와 실행결과가 같다.
1662
1663
              iris_df.iloc[:,:-1].apply(lambda x : np.sum(x))
1664
              Sepal.Length
                                  876.5
1665
                                  458.6
              Sepal.Width
1666
              Petal.Length
1667
                                  563.7
1668
              Petal.Width
                                    179.9
1669
              dtype: float64
```

```
1672
1673
              iris_df.iloc[:,:-1].apply(np.sum, axis=1) # 행 별 합계 출력
1674
1675
                             10.2
1676
                             9.5
1677
                             9.4
              2
1678
              3
                             9.4
1679
              4
                             10.2
1680
              ... 생략 ...
1681
              145
                             17.2
              146
1682
                             15.7
1683
              147
                             16.7
1684
              148
                             17.3
1685
              149
                             15.8
              Length: 150, dtype: float64
1686
1687
1688
           -다음 코드는 iris 데이터의 열 별 평균을 계산하고 각 데이터와 평균과의 차이를 apply() 함수를 이용해 계산한다.
1689
1690
              iris_df2 = iris_df.iloc[:,:-1]
1691
              iris_avg = iris_df2.apply(np.average)
1692
              iris_avg
1693
1694
              Sepal.Length 5.843333
1695
              Sepal.Width
                                  3.057333
1696
              Petal.Length
                                  3.758000
1697
              Petal.Width
                                     1.199333
1698
              dtype: float64
1699
1700
              iris_df2.apply(lambda x : x-iris_avg, axis=1).head()
1701
1702
                   -0.743333
                              0.442667 -2.358 -0.999333
1703
                   -0.943333
                                -0.057333 -2.358 -0.999333
              1
1704
                                                    -0.999333
                                           -2.458
                   -1.143333
                                0.142667
1705
                                            -2.258
              3
                   -1.243333
                               0.042667
                                                       -0.999333
1706
                   -0.843333
                               0.542667
                                            -2.358
                                                       -0.999333
1707
1708
1709
         2)다른 예제로 apply()를 복습해보자.
1710
1711
           student_list = [
1712
               {'Name':'John', 'Midterm':95, 'Final': 85},
               {'Name':'Smith', 'Midterm':85, 'Final' : 80},
{'Name':'Jenny', 'Midterm':30, 'Final' : 10},
1713
1714
1715
1716
           df = pd.DataFrame(student_list, columns= ['Name', 'Midterm', 'Final'])
1717
1718
1719
           df.head()
1720
           -----
1721
              Name
                        Midterm Final
1722
           0 John
                        95
                              85
                        85
1723
           1 Smith
                                80
1724
           2 Jenny
                        30
                                10
1725
1726
           df['Total'] = df['Midterm'] + df['Final']
1727
1728
1729
              Name
                        Midterm Final Total
           0 John
1730
                        95
                              85 180
                        85
                                80
                                    165
1731
           1 Smith
1732
           2 Jenny
                        30
                                     40
                                10
1733
1734
           -'Average' column 추가하기
1735
1736
              df['Average'] = df['Total'] / 2
1737
              df.head()
1738
1739
                          Midterm Final
                                           Total
                                                    Average
                Name
1740
              0 John
                          95 85
                                         180
                                                    90.0
                                         165
40
              1 Smith
1741
                          85
                                    80
                                                    82.5
1742
              2 Jenny
                          30
                                    10
                                                  20.0
1743
           -'Grade' column 추가하기
1744
1745
1746
              grade_list = []
1747
              for row in df['Average']:
1748
                 if row \leq 100 and row \geq 90 :
1749
                    grade_list.append('A')
1750
                 elif row < 90 and row >= 80:
                    grade_list.append('B')
1751
1752
                 elif row < 80 and row >= 70:
1753
                    grade_list.append('C')
1754
                 elif row < 70 and row >= 60:
```

-다음 코드는 행 별 합계를 출력한다.

```
1756
                  else : grade list.append('F')
1757
              df['Grade'] = grade_list
1759
              df
1760
                            Midterm Final
1761
                                             Total
                                                       Average Grade
                 Name
                                            180
1762
              0
                 John
                            95
                                    85
                                                       90.0
                            85
                                                       82.5
                                                                  В
1763
               1 Smith
                                    80
                                            165
1764
                                                    20.0
               2 Jenny
                            30
                                    10
                                            40
1765
            -'Result' column 추가하기
1766
1767
1768
               def pass_or_fail(row):
1769
                  if row != 'F':
                     return 'Pass'
1770
1771
                  else :
1772
                     return 'Fail'
1773
1774
               df['Result'] = df.Grade.apply(pass_or_fail)
1775
1776
                 Name
1777
                            Midterm Final
                                              Total
                                                       Average Grade
                                                                          Result
1778
              0 John
                                            180
                            95
                                   85
                                                       90.0
                                                                 Α
                                                                            Pass
1779
               1 Smith
                            85
                                            165
                                                       82.5
                                                                            Pass
1780
               2 Jenny
                            30
                                    10
                                            40
                                                    20.0
                                                                          Fail
1781
1782
            -Column 추가하면서 각각의 값 조작하기
1783
1784
               date_list = [
                  { 'yyyy-mm-dd' : '2019-01-05'},
{ 'yyyy-mm-dd' : '2019-01-10'}
1785
1786
1787
              ]
1788
1789
              df = pd.DataFrame(date_list, columns = ['yyyy-mm-dd'])
1790
1791
1792
                 yyyy-mm-dd
1793
               0 2019-01-05
1794
               1 2019-01-10
1795
1796
              def extract year(row):
1797
                 return row.split('-')[0]
1798
1799
              df['Year'] = df['yyyy-mm-dd'].apply(extract_year)
1800
1801
1802
                    yyyy-mm-dd
                                       Year
1803
               0
                    2019-01-05
                                       2019
                    2019-01-10
1804
                                       2019
               1
1805
1806
1807
            -passing keyword parameter to apply function
1808
               date_list = [{'Jumin': '2000-06-27'},
1809
                     {'Jumin': '2002-09-24'},
{'Jumin': '2005-12-20'}]
1810
1811
1812
              df = pd.DataFrame(date_list, columns = ['Jumin'])
1813
              df
1814
               -----
1815
                 Jumin
              0 2000-06-27
1816
1817
               1 2002-09-24
              2 2005-12-20
1818
1819
1820
              def extract_year(row):
                 return row.split('-')[0]
1821
1822
              df['Born_Year'] = df['Jumin'].apply(extract_year)
1823
1824
              df
1825
1826
                 Jumin
                                 Born_Year
              0 2000-06-27
1827
                                 2000
1828
               1 2002-09-24
                                 2002
1829
               2 2005-12-20
                                 2005
1830
1831
               def calc_age(year, current_year):
1832
                 return current_year - int(year)
1833
              df['Age'] = df['Born_Year'].apply(calc_age, current_year=2019)
              df
1835
1836
1837
                                 Born Year
                                             Aae
                 Jumin
1838
              0 2000-06-27
                                 2000
                                               19
```

grade_list.append('D')

```
1840
              2 2005-12-20
                               2005
1841
1842
              def get_introduce(age, prefix, suffix):
1843
                return prefix + str(age) + suffix
1844
1845
              df['introduce'] = df['age'].apply(get_introduce, prefix="I am ", suffix=" years old.")
1846
1847
1848
                Jumin
                               Born_Year Age
1849
              0 2000-06-27
                                            19 I am 19 years old.
              1 2002-09-24
1850
                               2002
                                            17
                                                I am 17 years old.
1851
              2 2005-12-20
                               2005
                                                I am 14 years old.
1852
1853
1854
         3)applymap()
1855
           -applymap() 함수는 DataFrame의 함수이지만 apply() 함수처럼 각 행(row, axis=1) 또는 각 열(column, axis=0)별로 작동하는 함수가 아니다.
1856
           -applymap() 함수는 각 요소(element)별로 작동한다.
           -Vector에 scala를 연산하면, 벡터의 요소 하나하나에 해당 연산을 해주는 것처럼 엘리먼트 와이즈(Element wise) 방식으로 적용하는 DataFrame의 각
1857
           요소마다 사용자 정의 함수를 수행한다.
1858
           -이때 사용자 정의 함수는 반드시 단일 값(Single value)을 반환해야 한다.
1859
           -Syntax
1860
              DataFrame.applymap(func)
1861
1862
           -앞에서 apply() 함수의 예제에서 사용했던 np.sum() 함수를 applymap() 함수에 적용해 보자.
1863
           -그러면 applymap() 함수가 요소별로 동작하는 것을 확인할 수 있다.
1864
1865
             iris_df.iloc[:,:-1].applymap(np.sum).head()
1866
                   Sepal.Length Sepal.Width Petal.Length Petal.Width
1867
              n
1868
                   5.1
                                                                        0.2
                                    3.5
                                                      1.4
1869
                   4.9
                                     3.0
                                                      1.4
                                                                        0.2
1870
              2
                  4.7
                                                                        0.2
                                    3.2
                                                      1.3
              3
1871
                   4.6
                                     3.1
                                                      1.5
                                                                        0.2
1872
                   5.0
                                     3.6
                                                      1.4
                                                                        0.2
1873
1874
           -이 결과는 각 요소에 대한 합계이므로 원래의 데이터와 같은 값이다.
1875
           -다음 코드는 lambda식을 이용해서 각 요소의 값을 제곱한다.
1876
1877
             iris_df.iloc[:,:-1].applymap(lambda x : x**2).head()
1878
1879
                   Sepal.Length Sepal.Width Petal.Length Petal.Width
              0
                                                                     0.04
1880
                   26.01
                                    12.25
                                                      1.96
1881
                   24.01
                                     9.00
                                                    1.96
                                                                   0.04
              1
1882
              2
                   22.09
                                    10.24
                                                      1.69
                                                                     0.04
1883
              3
                   21.16
                                     9.61
                                                    2.25
                                                                   0.04
1884
              4
                   25.00
                                    12.96
                                                      1.96
                                                                     0.04
1885
1886
           -한번에 DataFrame에 있는 모든 요소들을 수정하고자 할 때 사용하자.
1887
              data_list = [
1888
                {'x': 5.5, 'y': -5.6},
1889
                {'x': -5.2, 'y': 5.5},
1890
1891
                {'x': -1.6, 'y': -4.5}
1892
1893
              df = pd.DataFrame(data_list)
1894
              df
1895
1896
                   5.5
                          -5.6
              0
1897
1898
                   -5.2
              1
                          5.5
1899
              2
                   -1.6
                          -4.5
1900
1901
             import numpy as np
1902
1903
              df = df.applymap(np.around) #반올림함수 사용
              df
1904
1905
1906
1907
              0
                   6.0
                          -6.0
1908
                  -5.0
                          6.0
              1
1909
              2
                   -2.0
                          -4.0
1910
1911
         4)map()
1912
1913
           -map() 함수는 Series 데이터의 요소 각각에 대해 함수 또는 dict 또는 다른 Series를 적용한다.
1914
           -map() 함수는 DataFrame에는 사용할 수 없다.
           -다음 구문에서 보는 것처럼 Series 타입에서만 사용할 수 있다.
1915
1916
           -Syntax
1917
              Series.map(arg, na_action=None)
1918
           -arg: 시리즈의 값 하나 하나에 적용할 함수, dict 또는 Series.
1919
           -na_action: NA 값이 매핑 함수의 영향을 받는지 여부를 제어.
1920
           -사용 가능한 값은 None 또는 'ignore'이다.
```

1921

--ignore'이면 NA일 경우 함수에 적용하지 않고 NA를 반환한다.

1 2002-09-24

2002

```
1923
            -map() 함수는 Series의 값 하나하나에 접근하면서 해당 함수를 수행한다.
1924
1925
               import pandas as pd
               x = pd.Series(['Hello', 'Python', 'World'], index=[1,2,3])
1926
1927
1928
1929
1930
                          Python
1931
                             World
1932
               dtype: object
1933
1934
            -사용자 정의 함수 사용
1935
               --사용자 정의 함수를 이 함수는 입력값과 입력값의 길이를 반환한다.
1936
1937
                  def my_func(data):
1938
                     return (data, len(str(data)))
1939
1940
               --Series에 함수를 적용하면 함수 인자에 시리즈의 요소 하나가 전달된다.
1941
1942
                  x.map(my_func)
1943
1944
                                (Hello, 5)
1945
                             (Python,6)
1946
                                (World, 5)
1947
                  dtype: object
1948
1949
            -dict 사용
1950
               --Series에 dict를 적용하면 dict의 key별로 Series의 값에 적용된다.
1951
1952
                  z = {\text{"Hello": 'A', "Python": 'B', "World": 'C'}}
1953
                  x.map(z)
1954
                  1
1955
                  2
                             В
1956
                             C
1957
                  dtype: object
1958
1959
            -Series 사용
1960
               -Series에 Series를 적용하면 원본 Series의 값에 적용할 Series의 index별로 적용된다.
1961
                  y = pd.Series(['foo', 'bar', 'baz'], index=['Hello', 'Python', 'World'])
1962
1963
                  x.map(y)
1964
1965
                  1
1966
                  2
                             bar
1967
1968
                  dtype: object
1969
1970
1971
          5)다른 예제로 map()을 연습해 보자.
1972
            date_list = [{'Date': '2000-06-27'},
1973
1974
                      {'Date': '2002-09-24'},
1975
                      {'Date': '2005-12-20'}]
1976
            df = pd.DataFrame(date_list, columns = ['Date'])
1977
1978
1979
               Date
1980
            0 2000-06-27
1981
            1 2002-09-24
1982
            2 2005-12-20
1983
1984
            def extract_year(date):
1985
               return date.split('-')[0]
1986
1987
            type(df['Date'])
1988
1989
            pandas.core.series.Series
1990
1991
            df['Year'] = df['Date'].map(extract_year)
1992
1993
1994
               Date
                                Year
1995
            0 2000-06-27
                                2000
1996
             1 2002-09-24
                                2002
1997
            2 2005-12-20
                                2005
1998
1999
            -map() 응용하기
2000
2001
               data_list = [
                  td_ist = [
{'Name':'John', 'Age':15, 'Gender':'male', 'Job':'Student'},
{'Name':'Smith', 'Age':25, 'Gender':'male', 'Job':'Teacher'},
{'Name':'Jenny', 'Age':27, 'Gender':'female', 'Job':'Developer'},
2002
2003
2004
               1
2005
```

--None(기본값)이면 NA는 그대로 함수 또는 딕셔너리에 NA로 전달된다.

```
2006
2007
              df = pd.DataFrame(data list, columns = ['Name', 'Age', 'Gender', 'Job'])
2008
2009
2010
                Name
                       Age Gender Job
2011
              0 John
                          15 male
                                         Student
2012
                          25
             1 Smith
                              male
                                         Teacher
2013
              2 Jenny
                          27
                             female
                                         Developer
2014
             df.Job = df.Job.map({'Student':1, 'Teacher':2, 'Developer':3})
2015
2016
2017
2018
                       Age Gender Job
               Name
2019
             0 John
                          15 male 1
                          25
2020
              1 Smith
                               male
                          27
2021
              2 Jenny
                               female
2022
2023
        6) na_action
2024
2025
           -Series가 NaN 값을 포함할 경우 na_action 인자의 값에 따라 결과는 달라진다.
2026
2027
              s = pd.Series([1, 2, 3, None])
2028
2029
2030
             0
                     1.0
2031
                        2.0
             1
2032
             2
                        3.0
2033
             3
                       NaN
2034
              dtype: float64
2035
2036
           -기본값(na_action=None)일 경우 NA는 그대로 함수의 인자로 전달된다.
2037
2038
              s.map(lambda x: (x, x^{**2}))
2039
2040
                        (1.0, 1.0)
2041
              1
                        (2.0, 4.0)
2042
                        (3.0, 9.0)
2043
                        (nan, nan)
2044
              dtype: object
2045
2046
           -na_action='ignore'일 경우 적용한 결과가 NA가 된다.
2047
2048
             s.map(lambda x: (x, x**2), na_action='ignore')
2049
2050
                        (1.0, 1.0)
2051
                        (2.0, 4.0)
2052
                        (3.0, 9.0)
2053
              3
                        NaN
2054
             dtype: obj
2055
2056
2057
2058 13. 데이터 전처리
2059
         1)fillna()
2060
           -fillna() 함수는 주어진 방법으로 NA 또는 NaN값을 채운다.
2061
           -Syntax
2062
              DataFrame.fillna(value=None, method=None, axis=None,
2063
                                    inplace=False, limit=None,
2064
                                       downcast=None, **kwarqs)
2065
           -value : scalar, dict, Series, 또는 DataFrame을 지정.
2066
              --결측치를 채우는데 사용할 값.
2067
              --이 값은 list로 지정할 수 없다.
2068
           -method : {'backfill', 'bfill', 'pad', 'ffill', None}
2069
              --기본값 None
2070
              --재 색인화된 채우기에 사용될 방법을 지정한다.
2071
              --`ffill': 이전의 유효한 관측값을 이용해 채운다.
             --bfill : 이후의 유효한 관측값을 사용해서 채운다.
2072
2073
           -axis : {0 또는 'index', 1 또는 'columns'}, 축을 지정한다.
2074
           -inplace : boolean, 기본값 False, 만일 True 이면 현재 객체를 수정한다.
2075
           -limit: int, 기본값 None
              --method가 지정되고 있는 경우, 이것은 순방향/역방향으로 채울 수 있는 최대 연속 NaN 개수이다.
2076
2077
              --method가 지정되지 않은 경우, 이것은 NaN이 채워지는 축 전체의 최대 항목 수이다.
2078
              --없으면 0보다 커야한다.
2079
           -downcast : dict, 기본값 None
2080
             --항목들을 downcast='infer'일 경우 적당한 동등한 타입으로 형변환(다운캐스트) 된다.
2081
              --예를 들면 변수가 float64일 경우 자동으로 int64로 변경된다.
2082
              	ext{--}원래의 값들이 정수이더라도 <math>	extbf{NaN} 값은 실수로 간주되기 때문에 모든 데이터들의 타입이 실수형으로 바뀐다.
2083
             --그럴 경우 유용하게 사용될 수 있다.
2084
2085
              df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                                       [np.nan, np.nan, np.nan, 5], [np.nan, 3, np.nan, 4]],
2086
2087
2088
                                         columns=list('ABCD'))
              df
2089
```

-limit: int, 기본값 None, method가 지정되고 있는 경우, 이것은 순방향/역방향으로 채울 수 있는 최대 연속 NaN 개수다.

 -regex : bool 또는 to_replace와 같은 타입.

--이 경우 to_replace는 None이어야 한다.

--기본값 False: to_replace 또는 value를 정규식으로 해석할지 여부 --이 값이 True이면 to_replace는 문자열이어야 한다. --또는 정규 표현식 또는 list, dict 또는 정규 표현식의 배열이 될 수 있다.

```
2173
           -method: {'pad', 'ffill', 'bfill', None}
2174
              --to_replace가 scala, list 또는 tuple이고 값이 None 인 경우 대체 할 때 사용할 방법이다.
2175
              s = pd.Series([0, 1, 2, 3, 4])
2176
2177
              s.replace(0, 5)
2178
2179
2180
                         1
2181
2182
              3
                         3
2183
                         4
              dtype: int64
2184
2185
2186
            -다음 구문은 DataFrame 객체에서 0값을 5로 바꾼다.
2187
               \begin{split} df = & \text{ pd.DataFrame}(\{ \text{'A':} [0, & 1, 2, 3, 4], \\ & \text{'B':} & [5, & 6, 7, 8, 9], \\ & \text{'C':} & [\text{'a', 'b', 'c', 'd', 'e'}] \}) \end{split} 
2188
2189
2190
2191
              df.replace(0, 5)
2192
2193
                            С
2194
              0
                    5
                         5
                            a
2195
                    1
                         6
                              b
2196
                    2
                         7
                              С
2197
              3
                    3
                         8
2198
              4
                    4
                         9
                              е
2199
2200
            -다음 구문은 to_replace를 list로 지정한 예이다.
2201
            -DataFrame의 모든 0, 1, 2, 3을 4로 바꾼다.
2202
2203
              df.replace([0, 1, 2, 3], 4)
2204
2205
                   Α
                         В
                             С
2206
                    4
                         5
                              а
2207
              1
                         6
                             b
2208
              2
                    4
                         7
                              С
2209
              3
                    4
                         8
                              d
2210
                         9
                              е
2211
2212
            -다음 구문은 to_replace와 value를 list로 지정한 예이다.
2213
            -0, 1, 2, 3을 각각 4, 3, 2,1로 바꾼다.
2214
2215
              df.replace([0, 1, 2, 3], [4, 3, 2, 1])
2216
2217
                        В
                            С
2218
                         5
2219
              1
                    3
                         6
                              b
2220
              2
                    2
                         7
                              С
2221
                         8
                    1
2222
2223
2224
            -다음 구문은 1과 2 값을 이후의 1 또는 2가 아닌 값으로 채운다.
2225
2226
              df.replace([1, 2], method='bfill')
2227
2228
                         0
2229
                         3
              1
2230
                         3
2231
              3
                         3
2232
                         4
2233
              dtype: int64
2234
2235
            -to_replace를 딕셔너리 형식으로 지정한 예이다.
2236
            -0 값은 10으로, 1값은 100으로 바꾼다.
2237
2238
              df.replace({0: 10, 1: 100})
2239
              -----
2240
                                      C
2241
              0
                      10
                           5
2242
              1
                      100
                              6
                                      b
2243
              2
                      2
                              7
                                      C
2244
              3
                      3
                              8
                                      d
2245
2246
2247
            -다음 구문은 A열의 0값과 B열의 5값을 100으로 바꾼다.
2248
2249
              df.replace({'A': 0, 'B': 5}, 100)
2250
                                      С
2251
                              В
2252
                    100
                              100
                                      а
2253
                                      b
              1
                              6
                    1
2254
                    2
                              7
                                      С
```

d

```
2257
2258
           -다음 구문은 A열의 0은 100으로 바꾸고 4는 400으로 바꾼다.
2259
2260
              df.replace({'A': {0: 100, 4: 400}})
2261
2262
2263
                   100
                           5
                                а
2264
                           6
                                b
2265
                                C
2266
              3
                   3
                           8
                                d
2267
                   400
2268
2269
           -다음 구문은 to_replace에 정규표현식을 사용한 예이다.
2270
                                           'foo', 'bait'],
2271
              df = pd.DataFrame({'A':['bat',
2272
                                        'B':
                                            ['abc', 'bar', 'xyz']})
2273
2274
           -다음 구문은 ba로 시작하고 마지막 문자가 임의의 문자인 문자열을 'new'로 바꾼다.
2275
2276
              df.replace(to_replace=r'^ba.$', value='new', regex=True)
2277
2278
                                В
2279
              0
                     new
                             abc
2280
              1
                     foo
                                new
2281
                     bait
                                XVZ
2282
2283
           -다음 구문은 A열에서 ba로 시작하고 마지막 문자가 임의의 문자인 문자열을 'new'로 바꾼다.
2284
2285
              df.replace({'A': r'^ba.$'}, {'A': 'new'}, regex=True)
2286
2287
2288
              0
2289
              1
                     foo
                                har
2290
                     bait
2291
2292
           -다음 구문은 regex 속성에 정규표현식을 지정한 예이다.
2293
2294
              df.replace(regex=r'^ba.$', value='new')
2295
2296
                                В
                     Α
2297
                     new
                             abc
2298
                     foo
                               new
2299
                     bait
                                xyz
2300
2301
           -regex 속성에 \operatorname{dict} 형식으로 지정하면 정규표현식과 바꿀 값을 같이 지정할 수 있다.
2302
              df.replace(regex={r'^ba.$':'new', 'foo': 'xyz'})
2303
2304
2305
              0
2306
                     new
                             abc
2307
                     xyz
                                new
2308
                     bait
                                xyz
2309
2310
           -regex 속성이 list일 경우 value 속성이 지정되어야 한다.
2311
2312
              df.replace(regex=[r'^ba.$', 'foo'], value='new')
2313
2314
2315
              0
                     new
                             abc
2316
              1
                     new
                             new
2317
                      bait
2318
2319
           -여러 개의 논리(bool)값 또는 날짜시간(datetime64) 객체를 바꿀 때 to_replace 매개변수의 데이터 유형이 바꿀 값의 데이터 유형과 일치해야 한다.
2320
           -다음 구문을 오류를 발생한다.
2321
2322
              df = pd.DataFrame({'A':[True, False, True],
                                        'B': [False, True, False]})
2323
2324
              df.replace({'astring': 'new value', True: False}) # raises
2325
2326
              TypeError
                                                       Traceback (most recent call last)
2327
              <ipython-input-3-2313e57ab11c> in <module>
2328
              ----> 1 df.replace({'a string': 'new value', True: False})
2329
2330
              TypeError: Cannot compare types'ndarray(dtype=bool)' and 'str'
2331
2332
           -to_replace()의 매개변수 특성을 이해하려면 s.replace({'a': None})와 s.replace('a', None)의 동작을 비교/이해해야 한다.
2333
           -만일 다음과 같은 데이터가 있을 경우
2334
              s = pd.Series([10, 'a', 'a', 'b', 'a'])
2335
2336
           -s.replace({'a':None})는 s.replace(to_replace={'a': None}, value=None, method=None)와 같다.
2337
2338
           -즉 모든 a의 값을 None으로 바꾼다.
2339
2340
              s.replace({'a': None})
```

```
2342
             0
                            10
2343
             1
                            None
2344
                            None
2345
             3
                            h
2346
             4
                            None
2347
             dtype: object
2348
2349
           -value=None 이고 to_replace가 scala, list, tuple일 경우 replace() 함수는 method 매개변수에 기본값('pad')을 이용한다.
           -그래서 s.replace('a',None)은 s.replace(to_replace='a', value=None, method='pad')와 같다.
2350
2351
2352
             s.replace('a', None)
2353
2354
                       10
2355
             1
                       10
2356
                       10
2357
             3
                         h
2358
2359
             dtype: object
2360
2361
2362
        3)where()
2363
           -where() 함수는 하나 이상의 조건에 대한 DataFrame을 확인하고 그에 따라 결과를 반환하는 데 사용된다.
2364
           -기본적으로 조건을 만족하지 않는 행은 NaN 값으로 채워진다.
2365
2366
             DataFrame.where(cond, other=nan,inplace=False,
2367
                                      axis=None, level=None, errors='raise',
2368
                                        try_cast=False, raise_on_error=None)
2369
           -cond : boolean NDFrame, array-like, 또는 호출가능객체(callable).
2370
             --만일 cond가 True이면 원래 값을 유지.
2371
             --만일 cond가 False일 경우 other에서 해당 값으로 대체.
2372
             --cond가 callable이면 NDFrame에서 계산되고 논리 NDFrame 또는 배열을 반환해야 한다.
2373
              --callable 객체는 입력된 NDFrame을 변경하면 안된다.
2374
           -other : scalar, NDFrame, 또는 callable.
2375
             --cond가 False인 항목은 other의 해당 값으로 대체된다.
2376
             --other가 callable 이면 NDFrame에서 계산되고 스칼라 또는 NDFrame을 반환해야 한다.
             --callable 객체는 입력 된 NDFrame을 변경하면 안된다.
2377
2378
           -inplace : boolean, 기본값 False.
2379
             --만일 True 이면 현재 객체를 수정.
2380
           -axis : \{0 또는 'index', 1 또는 'columns'\}, 축을 지정한다.
2381
           -level : int, 기본값 None, 정렬 수준을 지정한다.
2382
           -errors : str, {'raise', 'ignore'}, 기본값 raise.
2383
             --raise : 예외를 발생시킨다.
2384
             --ignore : 예외를 억제한다. 오류 시 원본 개체를 반환한다.
2385
           -try_cast: boolean, 기본값 False.
2386
             --가능한 경우 결과를 입력 유형으로 다시 형 변환 한다.
           -raise_on_error : boolean, 기본값 True.
2387
2388
             --유효한 데이터타입이 아니면 예외를 발생시킨다.
2389
           -where 메소드는 if-then 관용구의 응용 프로그램이다.
2390
           -호출하는 DataFrame의 각 요소에 대해 cond가 True이면 요소가 사용된다.
2391
           -그렇지 않으면 DataFrame other의 해당 요소가 사용된다.
2392
           -DataFrame.where()는 numpy.where()와 다르다.
2393
           -df1.where(m, df2)는 np.where(m, df1,df2)와 동일하다.
2394
2395
             s = pd.Series(range(5))
2396
2397
           -다음 구문은 S 객체에서 0보다 큰 값을 반환하고 그렇지 않은 경우 NaN을 반환한다.
2398
2399
             s.where(s > 0)
2400
2401
                       NaN
2402
             1
                       1.0
2403
                       2.0
2404
             3
                       3.0
2405
             4
                       4.0
2406
             dtype: float64
2407
2408
           -반면, mask() 함수는 해당 조건을 만족하는 데이터에 대해 NaN을 반환한다.
2409
2410
             s.mask(s > 0)
2411
             n
2412
                       0.0
2413
                       NaN
             1
2414
             2
                       NaN
2415
             3
                       NaN
2416
                       NaN
2417
             dtype: float64
2418
2419
           -다음 코드는 S 객체에서 1보다 큰 값을 반환하고 그렇지 않으면 10을 반환한다.
2420
             s.where(s > 1, 10)
2421
2422
2423
             0
                       10
```

```
2425
                        2
2426
              3
                        3
2427
                        4
2428
              dtype: int64
2429
2430
           -다음 코드는 0부터 10까지(10 포함 안됨) 데이터를 이용해 2열 짜리 DataFrame을 만들고 DataFrame의 값이 3으로 나눈 나머지가 0인 경우 그 값을
           반환하며 그렇지 않으면 해당 값을 음수로 반환한다.
2431
2432
           df = pd.DataFrame(np.arange(10).reshape(-1, 2), columns=['A', 'B'])
2433
           m = df \% 3 == 0
2434
           df.where(m, -df)
2435
2436
                     В
2437
                0
                     -1
2438
                -2 3
           1
2439
           2
                -4 -5
2440
           3
                6
2441
                 -89
2442
2443
         -다음 코드는 DataFrame.where()와 numpy.where()를 비교한 것이다.
2444
2445
           df.where(m, -df) == np.where(m, df,-df)
2446
2447
                            B
2448
                          True
                True
2449
           1
                True
                           True
2450
           2
                True
                           True
2451
           3
                True
                           True
2452
                True
                          True
2453
2454
         -다음 코드는 where()와 mask() 함수를 비교한 것이다.
2455
         -mask() 함수의 조건에 not(~) 연산자가 붙어 있는 것을 확인할 것.
2456
2457
           df.where(m, -df) == df.mask(\sim m, -df)
2458
2459
                             В
2460
           0
                True
                           True
2461
                True
           1
                           True
2462
           2
                True
                          True
2463
           3
                True
                           True
2464
           4
                True
                           True
2465
2466
2467
         4)dropna
2468
           -dropna() 함수는 결측치(누락된 값)를 포함한 행 또는 열을 제거해 준다.
2469
2470
              DataFrame.dropna(axis=0, how='any', thresh=None,
2471
                                       subset=None, inplace=False)
2472
           -axis: {0 or 'index', 1 or 'columns'}, 기본값 0.
2473
              --결측치를 포함하는 행을 제거할 것인지 아니면 열을 제거할 것인지를 지정한다.
--0, 또는 'index': 누락된 값이 있는 행을 삭제한다.
2474
              --1, 또는 `columns' : 누락된 값이 포함 된 열을 삭제한다.
2475
2476
           -how: {`any', `all'}, 기본값 `any'.
              --'any': NA 값이 있는 경우 해당 행 또는 열을 삭제한다.
--'all': 모든 값이 NA이면 행 또는 열을 삭제한다.
2477
2478
2479
           -thresh : int, 선택사항.
2480
              --NaN이 아닌 항목의 최소 개수를 지정한다.
2481
              --예를 들어 thresh=2 이면 NaN을 포함하더라도 NaN이 최소 2개 이상이면 삭제하지 않는다.
2482
           -subset : array-like, 선택사항.
2483
              --부분집합을 뽑을 다른 축의 이름이다.
2484
              --예: 행을 삭제하면 포함 할 열의 목록이 된다.
2485
           -inplace : boolean, 기본값 False.
2486
              --만일 True 이면 현재 객체를 수정한다.
2487
              df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'], "toy":[np.nan, 'Batmobile', 'Bullwhip'],
2488
                                       "born": [pd.NaT, pd.Timestamp("1940-04-25"), pd.NaT]})
2489
2490
              df
2491
2492
                        name
                                                                      born
2493
              0
                                                  NaN
                                                                      NaT
2494
                        Batman
                                               Batmobile
                                                                 1940-04-25
              1
2495
                        Catwoman
                                            Bullwhip
                                                              NaT
2496
2497
           -다음 구문은 최소 하나 이상의 요소가 누락된 값이 있는 행을 제거한다.
2498
2499
              df.dropna()
2500
2501
                   name
                               tov
                                                    born
                                               1940-04-25
2502
                   Batman
                               Batmobile
2503
2504
           -다음 구문은 누락된 값을 포함한 열을 제거해 준다.
2505
2506
              df.dropna(axis='columns')
2507
```

```
2509
             0
                      Alfred
2510
             1
                       Batman
2511
                      Catwoman
2512
2513
          -다음 구문은 행의 모든 요소가 누락된 값일 경우 행을 제거해 준다.
2514
2515
             df.dropna(how='all')
2516
2517
                   name
                                                 born
2518
                                Batmobile 1940-04-25
2519
                    Batman
             1
2520
                    Catwoman Bullwhip NaT
2521
2522
          -누락된 값이 아닌 요소가 최소 2개 이상인 행은 삭제하지 않는다.
2523
2524
             df.dropna(thresh=2)
2525
2526
                             toy born
Batmobile 1940-04-25
                   name
2527
                    Batman
                    Catwoman Bullwhip NaT
2528
2529
2530
          -누락된 값을 찾을 열을 지정한다.
2531
          -다음 구문은 DataFrame에서 toy열은 누락된 값의 유/무를 확인하지 않는다.
2532
2533
             df.dropna(subset=['name', 'born'])
2534
2535
                                toy
                                                 born
                    name
                                Batmobile 1940-04-25
2536
                    Batman
2537
2538
          -누락된 값을 제거하고 현재 DataFrame이 변경된다.
2539
2540
             df.dropna(inplace=True)
2541
             df
2542
                 name toy
Batman Batmobile
2543
                                              born
2544
                                        1940-04-25
2545
2546
2547
        5)astype
2548
          -Pandas의 객체를 주어진 dtype 속성으로 형변환 한다.
2549
           -Syntax
2550
             DataFrame.astype(dtype, copy=True, errors='raise', **kwargs)
2551
           -dtype: datatype 또는 dict.
2552
             --numpy.dtype 또는 Python의 datatype을 사용하여 전체 pandas 객체를 같은 유형으로 형변환 할 수 있다.
2553
             --{col: dtype, ...}처럼 dict 형식을 사용했을 경우 col은 열의 이름이고 dtype은 하나 이상의 열을 특정유형으로 형변환하는 numpy.dtype 또는
             Python의 datatype 이다.
2554
           -copy: bool, 기본값 True.
2555
            --copy=True 일 때 복사본을 반환한다.
2556
          -errors : str, {`raise', `ignore'}, 기본값 raise.
2557
             --errors='raise'이면 예외를 발생시킨다.
             --errors='ignore' 이면 예외를 억제한다.
2558
2559
             --오류 시 원본 개체를 반환한다.
2560
          -kwargs : 생성자에 전달할 keyword 인수이다.
2561
2562
             ser = pd.Series([1, 2], dtype='int32')
2563
             ser
2564
                   1
2565
             1
2566
2567
             dtype: int32
2568
2569
             ser.astype('int64')
2570
2571
                       1
2572
             1
2573
             dtype: int64
2574
2575
          -copy=False이면 반환받은 객체를 변경했을 경우 원본 데이터도 같이 변하므로 주의해야 한다.
2576
2577
             s1 = pd.Series([1,2])
2578
             s2 = s1.astype('int64', copy=False)
2579
             s2[0] = 10
             s1 # note that s1[0] has changedtoo
2580
2581
2582
                      10
2583
2584
             dtype: int64
2585
2586
2587
        6)중복된 값 제거하기
2588
2589
             {'Name':'John', 'Gender':'male', 'Job':'Student'},
2590
```

name

```
{'Name':'Jenny','Gender':'female', 'Job':'Developer'},
{'Name':'Smith','Gender':'male', 'Job':'Teacher'}
2592
2593
2594
2595
               df = pd.DataFrame(data_list, columns = ['Name', 'Gender', 'Job'])
2596
               df.head()
2597
2598
2599
                      Name Gender
                                              Job
               0
2600
                                                  Student
                      John
                                male
2601
               1
                      Smith male
                                                  Teacher
2602
                                female
                                                  Developer
               2
                      Jenny
2603
               3
                      Smith
                                male
                                                  Teacher
                                                                       #중복된 값
2604
2605
               -중복된 값 확인하기
2606
2607
                   df.duplicated()
2608
                   0
2609
                          False
2610
                   1
                          False
2611
                          False
2612
                         True
                                               #여기가 중복된 값이 있다는 뜻
2613
                  dtype: bool
2614
2615
                -중복된 값 제거
2616
2617
                   df.drop_duplicates()
2618
2619
                      Name Gender
                                              Job
2620
                     John
                                male
                                                  Student
2621
                                                  Teacher
                                male
                   1
                      Smith
2622
                   2 Jenny
                                female
                                                  Developer
2623
2624
2625
            7)기타 Null 처리하기
2626
2627
               student_list = [
2628
                   {'Name': 'John', 'Major': 'Computer Science', 'Gender': 'male', 'Age': 40},
                      {'Name': 'Nate', 'Major': None, 'Gender': "male", 'Age':35},
2629
                             ! 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':None},
{'Name': 'Zeward', 'Major': 'Computer Science', 'Gender': 'male', 'Age':None},
2630
2631
2632
2633
2634
2635
                             {'Name': 'Zara', 'Major': 'Psychology', 'Gender': 'female', 'Age':25}, {'Name': 'Wendy', 'Major': 'Economics', 'Gender': 'female', 'Age':37}, {'Name': 'Sera', 'Major': None, 'Gender': 'female', 'Age':None}
2636
2637
2638
2639
2640
               df = pd.DataFrame(student_list, columns = ['Name', 'Major', 'Gender', 'Age'])
2641
2642
                                    Major
2643
                      Name
                                                               Gender
2644
               0
                      John
                                    Computer Science
                                                                male
                                                                                 40.0
2645
                                    None
                                                                                 35.0
               1
                      Nate
                                                               male
2646
                      Abraham Physics
                                                                                 37.0
                                                                male
               3
2647
                                Psychology
                                                            male
                                                                              NaN
                      Brian
2648
               4
                      Janny
                                    None
                                                                female
                                                                                 10.0
2649
               5
                      Yuna
                                None
                                                            female
                                                                              12.0
               6
                      Jeniffer Computer Science
                                                            female
2650
                                                                              45.0
2651
               7
                      Edward Computer Science
                                                            male
                                                                              NaN
2652
               8
                                    Psychology
                                                                female
                                                                                 25.0
                      7ara
2653
               9
                      Wendy
                                    Economics
                                                                female
                                                                                 37.0
               10 Sera
2654
                                None
                                                            female
                                                                              NaN
2655
2656
               df.shape
                         _____
2657
2658
               (11,4)
2659
2660
               df.info()
2661
                <class 'pandas.core.frame.DataFrame'>
2662
                RangeIndex: 11 entries, 0 to 10
2663
                Data columns (total 4 columns):
2664
2665
                Name
                           11 non-null object
2666
                          7 non-null object
               Major
2667
               Gender 11 non-null object
2668
                         8 non-null float64
               dtypes: float64(1), object(3)
2669
2670
               memory usage: 432.0+ bytes
2671
2672
                -Null 확인하기
2673
2674
                   df.isna()
```

{'Name':'Smith','Gender':'male', 'Job':'Teacher'},

```
2676
                   Name MajorGender Age
2677
              0
                   False False False
2678
              1
                   False True
                               False False
2679
                   False False False
2680
                   False False False
                                     True
                   False True False False
2681
              5
2682
                   False True
                                False
              6
2683
                   False False False
              7
2684
                   False False False
                                      True
2685
              8
                   False False False
2686
              9
                   False False False
                                      False
2687
              10 False True
                             False
                                      True
2688
2689
              df.isnull()
2690
                   Name MajorGender Age
2691
2692
                   False False False
2693
                   False True False False
              1
2694
                    False False False
2695
              3
                   False False False
                                     True
2696
                   False True
                               False
                                        False
              5
                                 False
2697
                   False True
                                         False
                   False False False
2698
              6
2699
                    False False False
2700
              8
                   False False False
                                      False
2701
              9
                   False False False
                                      False
2702
              10 False True
                                      True
                            False
2703
2704
            -None값을 다른 값으로 변경하기
2705
2706
              df.Age = df.Age.fillna(0) #숫자 NaN을 0으로
2707
              df.Major = df.Major.fillna('Unknown') #글자 None을 Unknown으로
2708
              df
2709
                   _____
2710
                   Name
                             Major
                                                    Gender
2711
                   John
                              Computer Science
                                                   male
                                                                 40.0
2712
                              Unknown
                                                                   35.0
              1
                   Nate
                                                    male
2713
                    Abraham Physics
                                                   male
                                                                 37.0
2714
              3
                   Brian Psychology
                                                 male
                                                              0.0
2715
                                                                    10.0
                   Janny
                              Unknown
                                                     female
2716
                           Unknown
                                                                 12.0
                   Yuna
                                                   female
                                                              45.0
2717
              6
                   Jeniffer Computer Science
                                                female
2718
                    Edward Computer Science
                                                 male
2719
              8
                   Zara
                              Psychology
                                                   female
                                                                 25.0
2720
              9
                   Wendy
                              Economics
                                                   female
                                                                 37.0
2721
              10 Sera
                            Unknown
                                                    female
                                                                 0.0
2722
2723
         8)Unique와 갯수 알아보기
2724
            -Unique 즉, 중복제거된 값 알아보기
2725
2726
              df.Major.unique()
2727
2728
              array(['Computer Science', 'Unknown', 'Physics', 'Psychology', 'Economics'], dtype=object)
2729
2730
              df.Gender.unique()
2731
              array(['male', 'female'], dtype=object)
2732
2733
2734
              df.Major.value_counts()
2735
2736
              Unknown
2737
              Computer Science
2738
              Psychology
                                   2
2739
              Physics
                                   1
2740
              Economics
2741
              Name: Major, dtype: int64
2742
2743
2744
2745
      14. DataFrame 합치기
2746
2747
              {'Name': 'John', 'Job': 'Teacher'},
{'Name': 'Nate', 'Job': 'Student'},
{'Name': 'Fred', 'Job': 'Developer'}
2748
2749
2750
2751
2752
2753
           list2 = [
2754
              {'Name': 'Ed', 'Job': 'Dentist'},
              {'Name': 'Jack', 'Job': 'Farmer'},
{'Name': 'Ted', 'Job': 'Designer'}
2755
2756
2757
           1
2758
```

```
2759
            df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
2760
            df2 = pd.DataFrame(list2, columns = ['Name', 'Job'])
2761
2762
             -concat()로 합치기
2763
2764
               result = pd.concat([df1, df2])
2765
               result
2766
2767
                     Name Job
2768
               0
                     John
                             Teacher
2769
                     Nate
                             Student
2770
                             Developer
               2
                     Fred
2771
               0
                     Ed Dentist
                                                #0, 1, 2가 반복
2772
                     Jack
                             Farmer
               1
2773
               2
                     Ted
                             Designer
2774
2775
               result = pd.concat([df1, df2], ignore_index = True)
2776
               result
2777
2778
                     Name Job
2779
                     John
                             Teacher
2780
               1
                     Nate
                             Student
2781
               2
                     Fred
                             Developer
2782
               3
                         Dentist
                     Ed
2783
               4
                     Jack
                             Farmer
2784
               5
                     Ted
                             Designer
2785
2786
             -append()로 합치기
2787
2788
               result = df1.append(df2)
2789
               result
2790
2791
                     Name Job
2792
               0
                     John
                             Teacher
2793
               1
                     Nate
                             Student
2794
                             Developer
               2
                     Fred
2795
               0
                     Ed
                         Dentist
                                                 #0, 1, 2가 반복
2796
               1
                     Jack
                             Farmer
2797
                     Ted
                             Designer
2798
2799
               result = pd.append([df1, df2], ignore_index = True)
2800
2801
2802
                     Name Job
2803
               0
                     John
                             Teacher
2804
               1
                     Nate
                             Student
2805
               2
                     Fred
                             Developer
2806
               3
                     Ed Dentist
2807
                     Jack
                             Farmer
               5
2808
                     Ted
                             Designer
2809
2810
             -Column으로 합치기
2811
               --두개의 DataFrame의 column이 서로 일치하지 않음.
2812
2813
                     {'Name': 'John', 'Job': 'Teacher'},
{'Name': 'Nate', 'Job': 'Student'},
{'Name': 'Jack', 'Job': 'Developer'}
2814
2815
2816
2817
                  1
2818
2819
                  list2 = [
                     {'Age': 25, 'Country': 'U.S'},
{'Age': 30, 'Country': 'U.K'},
2820
2821
                     {'Age': 45, 'Country': 'Korea'}
2822
2823
2824
                  df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
2825
2826
                  df2 = pd.DataFrame(list2, columns = ['Age', 'Country'])
2827
2828
                  result = pd.concat([df1, df2], axis=1, ignore_index=True)
2829
                  result
2830
2831
                                Teacher
                                           25 U.S
                  0
                        John
2832
2833
                  1
                        Nate
                                Student
                                           30 U.K
2834
                  2
                                Developer 45 Korea
                        Jack
2835
2836
2837
2838
      15. 기초 통계 분석
2839
```

2841

¹⁾Pandas는 데이터를 보다 좀 더 편하게 다룰 수 있게 하는 데이터 구조 측면에서의 장점을 가진 패키지이다.

²⁾Pandas에서 제공하는 통계분석은 기본적인 기술통계 및 데이터 요약이다.

³⁾고급 통계 기법을 사용하고 싶다면 Scikit-learn 이나 다른 통계 패키지를 이용하여 수행 할 수 있다.

⁴⁾기술통계함수 목록

```
2843
           -count :NA를 제외한 개수
2844
           -min : 최솟값
2845
           -max : 최댓값
2846
           -sum : 합
2847
           -cumprod : 누적합
2848
           -mean: 평균
2849
           -median : 중앙값
           -quantile : 분위수
2850
2851
           -corr : 상관관계
2852
           -var : 표본분산
2853
           -std : 표본 정규분산
2854
2855
         -다음 코드는 기술 통계량을 확인해 보기 위해 데이터를 불러오자.
2856
         -Statsmodels \mbox{ 패키지를 이용해 iris 데이터를 불러온다.}
2857
2858
           import statsmodels.api as sm
2859
           iris = sm.datasets.get_rdataset("iris", package="datasets")
           iris df = iris.data
2860
           iris_df.head()
2861
2862
2863
                 Sepal.Length Sepal.Width Petal.Length Petal.Width
                                                                    Species
2864
           0
                5.1
                                  3.5
                                                   1.4
                                                                       0.2
                                                                                      setosa
2865
           1
                4.9
                                   3.0
                                                     1.4
                                                                       0.2
                                                                                      setosa
2866
           2
                4.7
                                   3.2
                                                     13
                                                                       0.2
                                                                                      setosa
2867
           3
                                                                       0.2
                 4.6
                                                                                       setosa
           4
2868
                5.0
                                                                       0.2
                                   3.6
                                                     1.4
                                                                                       setosa
2869
2870
2871
         5) 최솟값, 최댓값, 평균, 중위수
2872
           -Syntax
2873
              DataFrame.min(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
2874
            -axis : 0이면 index, 1이면 columns를 의미다.
2875
           -skipna : True(기본값)이면 NA 또는 null 값을 계산에서 제외한다.
2876
           -level : 다중 index일 경우 level을 지정한다.
2877
           -numeric_only: True일 경우 float, int, boolean 유형의 열들만 포함시킨다.
2878
              --기본값 None은 모든 열에 대해 연산을 시도한다.
2879
              --이 인자는 시리즈(Series)는 지원하지 않는다.
2880
2881
           iris_df.min()
2882
2883
           Sepal.Length
                                        4.3
2884
           Sepal.Width
                                        2
2885
           Petal.Length
                                        1
2886
           Petal.Width
                                          0.1
2887
           Species
                                          setosa
2888
           dtype: object
2889
2890
           iris_df.max()
2891
                                        7.9
2892
           Sepal.Length
2893
           Sepal.Width
                                        4.4
2894
           Petal.Length
                                        6.9
2895
           Petal.Width
                                          2.5
2896
           Species
                                          virginica
2897
           dtype: object
2898
2899
           iris df.mean()
2900
            _____
2901
           Sepal.Length
                               5.843333
           Sepal.Width
                                3.057333
2902
2903
                                3.758000
           Petal.Length
2904
           Petal.Width
                                  1.199333
2905
           dtype: float64
2906
2907
           iris_df.median()
2908
                                5.80
2909
           Sepal.Length
2910
           Sepal.Width
                                3.00
2911
           Petal.Length
                                4.35
2912
           Petal.Width
                                  1.30
2913
           dtype: float64
2914
2915
2916
         6)자세한 내용는 pandas documentation API Reference를 참조한다.
2917
            -Series Computations / Descriptive Stats :
           http://pandas.pydata.org/pandas-docs/stable/api.html \# computations-descriptive-stats
2918
           -DataFrame Computations / Descriptive Stats : http://pandas.pydata.org/pandas-docs/stable/api.html#api-dataframe-stats
2919
2920
2921
         7)요약 통계량
2922
           -describe() 함수는 요약 통계량을 출력한다.
2923
            -요약 통계량은 데이터의 개수, 평균, 표준편차, 최솟값, 25%, 50%, 75%, 그리고 최댓값 정보이다.
2924
            -Svntax
2925
              DataFrame.describe(percentiles=None, include=None, exclude=None)
```

```
2926
           -percentiles : 출력에 포함될 백분위 수를 0 \sim 1사이의 값으로 지정한다.
2927
             --기본 값은 [.25, .5, .75].
2928
             --이것은 25%, 50%, 75% 위치 데이터를 출력한다.
2929
           -include : 출력에 포함될 데이터의 유형을 지정한다.
2930
             --None(기본값) 이면 모든 숫자타입 열들을 출력에 포함시킨다.
2931
             --"all"이면 모든 열을 포함한다.
2932
             --정수형이면 "int64", 논리형이면 "bool", 실수형이면 "float64" 등으로 지정한다.
2933
           -exclude : 출력에서 제외할 데이터의 유형을 지정한다.
2934
             --None(기본값) 이면 아무것도 제외시키지 않는다.
2935
          -숫자 데이터의 경우 결과의 인덱스에는 count, mean, std, min, max 및 하위 백분위 수, 상위 백분위 수 및 상위 백분율이 포함된다.
          -기본적으로 하위 백분위 수는 25이고 상위 백분위 수는 75이다.
2936
2937
          -50 백분위 수는 중앙값과 같다.
2938
           -객체 데이터(예: 문자열 또는 timestamp)의 경우 결과 색인에 count, unique, top 그리고 freq가 포함된다.
2939
           -top가 가장 일반적인 값이다.
          -freq는 가장 일반적인 값의 빈도수이다.
2940
2941
           -timestamp는 첫 번째 요소와 마지막 요소도 포함한다.
2942
           -가장 높은 count를 갖는 값이 여러 개 일 경우, count와 top 결과는 가장 높은 count를 갖는 값 중에서 임의로 선택된다.
2943
           -DataFrame을 통해 제공되는 혼합 데이터 유형의 경우 기본값은 숫자 열의 분석만 반환한다.
2944
           -DataFrame이 숫자 열이 없는 개체 및 범주 데이터로만 구성된 경우 기본값은 개체열과 범주 형 열 모두의 분석을 반환하는 것이다.
2945
           -include='all' 매개변수가 제공되면 결과에는 각 유형의 속성이 결합된다.
2946
2947
2948
        8)기본 요약 통계량
2949
          -다음 코드는 iris 데이터의 요약 통계량을 출력한다.
2950
          -iris 데이터의 요약 통계량에는 종(Species) 정보는 출력되지 않는다.
2951
           -기본적으로 숫자 데이터의 요약 통계량이 출력된다.
2952
2953
             iris df.describe()
2954
2955
                                                 Petal.Length
                                                               Petal.Width
                      Sepal.Length Sepal.Width
                                                                    150.000000
2956
                   150.000000
                                  150.000000
                                                   150.000000
             count
2957
             mean 5.843333
                                  3.057333
                                                   3.758000
                                                                    1.199333
2958
             std
                      0.828066
                                    0.435866
                                                     1.765298
                                                                      0.762238
2959
             min
                      4.300000
                                     2.000000
                                                     1.000000
                                                                      0.100000
2960
                    5.100000
                                  2.800000
             25%
                                                   1.600000
                                                                    0.300000
                    5.800000
                                  3.000000
             50%
                                                   4.350000
2961
                                                                    1.300000
2962
             75%
                    6.400000
                                  3.300000
                                                   5.100000
                                                                    1.800000
2963
                    7.900000
                                  4.400000
                                                                    2.500000
             max
                                                   6.900000
2964
2965
          -다음은 종(Species) 정보의 요약 통계량을 출력한다.
2966
           -count(전체 데이터의 수), unique(데의 종류), top(가장 많은 요소), freq(가장 많은 요소의 수)를 출력한다.
2967
2968
             iris df.Species.describe()
2969
2970
             count
                             150
2971
             unique
2972
             top
                               virginica
                             50
2973
             frea
2974
             Name: Species, dtype: object
2975
2976
2977
        9)include와 SOL exclude
2978
           -다음 데이터의 요약 통계량을 출력하면 a열과 b열의 요약 통계량만 출력된다.
2979
           -b열은 논리값을 가지므로 기본요약 통계량 출력에서 제외된다.
2980
2981
             df = pd.DataFrame(\{'a': [1, 2] * 3, 'b': [True, False] * 3, 'c': [2.0, 4.0] * 3\})
2982
             df.describe()
2983
2984
                      а
             count 6.000000
                                  6.000000
2985
2986
             mean 1.500000
                                  3.000000
                                    1.095445
2987
                      0.547723
             std
2988
                      1.000000
                                     2.000000
             min
2989
             25%
                    1.000000
                                  2.000000
2990
             50%
                    1.500000
                                  3.000000
             75%
2991
                    2.000000
                                  4.000000
2992
                    2.000000
                                  4.000000
             max
2993
2994
          -include 및 exclude 매개변수를 사용하여 DataFrame에서 출력용으로 분석되는 열을 제한할 수 있다.
2995
           -Series를 분석 할 때 매개변수는 무시된다.
2996
           -다음은 정수유형 열에 대해서만 요약통계량을 출력한다.
2997
          -int64 유형을 include 시키거나 나머지 유형들을 exclude 시키면 된다.
2998
           -앞의 a,b,C열을 갖는 데이터에서 다음 두 구문은 같은 결과를 출력할 것이다.
2999
3000
             df.describe(include=["int64"])
3001
             df.describe(exclude=["bool", "float64"])
3002
3003
                      6.000000
3004
             count
3005
                      1.500000
             mean
3006
                         0.547723
             std
3007
                         1.000000
             min
3008
             25%
                      1.000000
             50%
                      1.500000
3009
```

```
3011
             max
                       2.000000
3012
3013
           -모든 요소에 대해 요약통계량을 출력하려면 include='all'을 이용한다.
3014
3015
             df.describe(include='all')
3016
3017
                       6.000000
                                                6.000000
3018
             count
                                      6
3019
             unique
                       NaN
                                      2
                                                NaN
3020
                                        True
             top
                         NaN
                                                NaN
                                      3
3021
             freq
                       NaN
                                                NaN
3022
             mean
                       1.500000
                                      NaN
                                             3.000000
                                      NaN
3023
             std
                         0.547723
                                               1.095445
                          1.000000
                                                2.000000
3024
             min
                                        NaN
3025
             25%
                       1.000000
                                             2.000000
                                      NaN
             50%
                       1.500000
                                             3.000000
3026
                                      NaN
3027
             75%
                       2.000000
                                             4.000000
                                      NaN
                       2.000000
                                      NaN
                                             4.000000
3028
             max
3029
3030
           -다음처럼 include와 exclude에 같은 유형을 사용하면 오류가 발생한다.
3031
3032
             df.describe(include=["int64"], exclude=["int64", "float64"])
3033
3034
                                             Traceback (most recent call last)
3035
             <ipython-input-64-52780e7f55bd> in <module>()
3036
3037
3038
        10)분산,표준편차
           -var()는 분산(variance)을, std()는 표준편차(standard deviation)를 계산한다.
3039
3040
           -Syntax
3041
             DataFrame.var(axis=None, skipna=None, level=None, ddof=1,
3042
                                   numeric only=None, **kwargs)
3043
             DataFrame.std(axis=None, skipna=None, level=None, ddof=1,
3044
                                   numeric_only=None, **kwargs)
3045
           -ddof : 델타 자유도(Delta Degree of Freedom)를 지정한다.
3046
             --기본값은 1이다.
3047
             --계산에 사용되는 제수는 N-ddof이다.
3048
             --여기서 N은 요소의 수를 나타낸다.
3049
3050
             iris_df.var()
3051
3052
             Sepal.Length 0.685694
3053
             Sepal.Width
                                0.189979
3054
             Petal.Length
                                3.116278
3055
             Petal.Width
                                   0.581006
3056
             dtype: float64
3057
3058
             iris df.std()
3059
             Sepal.Length 0.828066
3060
3061
             Sepal.Width
                                 0.435866
3062
             Petal.Length
                                 1.765298
3063
             Petal.Width
                                   0.762238
             dtype: float64
3064
3065
3066
        11)공분산(covariance), 상관계수(correlation)
3067
           -COV()는 각 열들의 공분산 쌍을 계산한다.
3068
           -corr()는 각 열들의 상관계수 쌍을 계산한다.
3069
           -Syntax
3070
             DataFrame.cov(min_periods=None)
3071
             DataFrame.corr(method='pearson', min_periods=1)
           -min_periods : 유효한 결과를 얻기 위해 열 쌍당 필요한 최소 관측 수를 지정한다.
3072
           -method : 상관계수를 계산할 방법을 지정한다.
--"pearson", "kendall", "spearman" 중 하나를 지정할 수 있다.
3073
3074
3075
3076
             iris_df.cov()
3077
3078
                                                  Sepal.Width Petal.Length Petal.Width
                                   Sepal.Length
                              0.685694
3079
             Sepal.Length
                                               -0.042434 1.274315
                                                                           0.516271
3080
             Sepal.Width
                              -0.042434
                                             0.189979
                                                            -0.329656
                                                                        -0.121639
                                               -0.329656
3081
             Petal.Length
                              1.274315
                                                            3.116278
                                                                           1.295609
             Petal.Width
                                0.516271
                                                  -0.121639 1.295609
3082
                                                                             0.581006
3083
3084
           iris_df.corr()
3085
           -----
3086
                             Sepal.Length Sepal.Width Petal.Length Petal.Width
3087
           Sepal.Length 1.000000
                                     -0.117570 0.871754
                                                                   0.817941
                                      1.000000
                                                     -0.428440
3088
           Sepal.Width
                         -0.117570
                                                                 -0.366126
                                                    1.000000
3089
                         0.871754
                                        -0.428440
           Petal.Length
                                                                   0.962865
3090
           Petal.Width
                            0.817941
                                           -0.366126 0.962865
                                                                      1.000000
3091
3092
```

75%

2.000000

```
3095
          -Series나 DataFrame에는 일반적인 수학적, 통계학적 계산을 실행하는 method를 사용할 수 있다.
3096
3097
             df = pd.read_csv('pandas_data/sungjuk_utf8.csv', header = None,
3098
                  names = ['학번', '이름', '국어', '영어', '수학', '전산'])
3099
3100
                      이름 국어 영어 수학 전산
3101
              학번
                     한송이 78878378
3102
             0 1101
3103
             1 1102
                      정다워 88835798
3104
             . . .
3105
3106
3107
             df.loc[:, '국어':'전산'].mean()
3108
3109
                 84.916667
             국어
3110
                 80.416667
             영어
             수학 75.333333
3111
             전산 79.750000
3112
3113
             dtype: float64
3114
3115
          -Series에서도 같은 method를 사용할 수 있다.
3116
3117
             df['국어'].sum()
3118
             1019
3119
3120
3121
          -percent 표시는 백위수 값(전체를 100으로 작은 쪽부터세어서 몇 번째가 되는지 나타내는 수치이다. 50 백분위수가 중앙값이다)이다.
3122
3123
             df.describe().round(1) #round() 반올림함수
3124
3125
                         영어
                                        전산
                                12.0
3126
             count12.0
                        12.0
                                       12.0
3127
             mean84.9
                        80.4
                                75.3
                                       79.8
3128
                   10.7
                          15.3
                                 15.2
             std
                                         11.6
3129
             min
                    68.0
                           56.0
                                  53.0
                                         55.0
3130
             25%
                    77.5
                           67.0
                                  61.5
                                         76.8
             50%
3131
                    87.5
                           85.0
                                  80.5
                                         78.0
3132
             75%
                    91.2
                           91.0
                                  87.2
                                         88.0
3133
             max
                    98.0
                           99.0
                                  93.0
                                         98.0
3134
3135
          -백분위수 값을 변경할 경우에는 keyword 인수 percentiles의 list 요소에 <math>1이상의 소수 값을 지정한다.
3136
3137
             df.describe(percentiles = [0.1, 0.9]).round(1)
3138
3139
                  국어 영어
                                        전산
3140
             count12.0
                        12.0
                                12.0
                                       12.0
                                       79.8
3141
             mean84.9
                        80.4
                                75.3
3142
                   10.7
                          15.3
                                 15.2
             std
                                         11.6
                    68.0
                                  53.0
3143
                                         55.0
             min
                           56.0
3144
             10%
                    68.8
                           58.0
                                  53.4
3145
             50%
                    87.5
                           85.0
                                  80.5
                                         78.0
             90%
3146
                    98.0
                           97.0
                                  92.5
                                         88.9
3147
             max
                    98.0
                           99.0
                                  93.0
                                         98.0
3148
3149
          -위의 예에서는 2개의 값을 지정하고 있지만 3개 이상 지정하는 것도 가능하다.
3150
          -DataFrame에 대해서 통계적인 연산을 하는 method를 실행한 경우에는 수치형의 열이 대상이 된다.
3151
           -비수치열에 대해 describe()를 사용하는 경우에는 다음과 같은 기본 통계량이 산출된다.
3152
             --count : 결손값을 제외한 data 수
3153
             --unique : unique한 data 수
3154
             --top: data의 수가 가장 많은 값
3155
             --freq : top의 data \div
3156
             df[['학번', '이름']].describe()
3157
3158
3159
                      학번
             count 12
3160
                        12
3161
                      12 12
             unique
                      1102 한산섬
3162
             top
3163
             freq
3164
3165
          -논리값으로 data 추출하기
3166
3167
             df.loc[df['국어'] > 90].head()
3168
3169
                                  국어 영어 수학 전산
                          이름
                       튼튼이 98979388
3170
                 1106
3171
                 1108 더크게 98679378
3172
             101111 한산섬 98897378
3173
             df.query('학번 == 1104')
3174
3175
3176
                                 국어 영어 수학 전산
                          이름
                         고아라 83 57 88 73
3177
                 1104
```

```
3179
          -where method로 data 추출하기
3180
3181
            df.where(df['go|'] < 70)
3182
3183
                                           수학
                        이름
                               국어
                                      영어
                 학번
                                                  전산
                                            NaN
                                                   NaN
                        NaN
3184
                 NaN
                               NaN
                                      NaN
            1 NaN NaN
3185
                             NaN NaN NaN
                                                  NaN
                                    56.0 87.0
57.0 88.0
            2 1103.0 그리운 76.0
3 1104.0 고아라 83.0
                                                  78.0
3186
3187
                                                  73.0
            4 NaN
                             NaN NaN NaN NaN
3188
                        NaN
3189
            5 NaN
                             NaN NaN NaN
                        NaN
                                                   NaN
                 1107.0 한아름 68.0 67.0 83.0
1108.0 더크게 98.0 67.0 93.0
3190
            6
                                                  89.0
                1108.0 더크게 98.0
3191
            7
                                                  78.0
            8 NaN NaN NaN NaN NaN
9 NaN NaN NaN NaN NaN
                                                  NaN
NaN
3192
3193
            10 NaN NaN NaN NaN
3194
                                                  NaN
3195
                                    NaN
                                           NaN
            11 NaN
                      NaN
                           NaN
                                                  NaN
3196
3197
          -값 변경하기
3198
            df.head(3)
3199
3200
                       이름 국어 영어 수학 전산
3201
                 학번
                       한송이 78878378
3202
               1101
                       정다워 88835798
그리운 76568778
3203
                 1102
             1
3204
                 1103
3205
            df.loc[1, '전산'] = np.nan
3206
3207
3208
            df.loc[1, '전산']
3209
3210
            nan
3211
3212
            df.head(3)
3213
                 학번 이름 국어 영어 수학 전산
3214
                 1101 한송이 78878378.0
1102 정다워 888357NaN
3215
            0
3216
3217
                 1103 그리운 76 56 87 78.0
3218
3219
          -복수의 값 변경
3220
3221
             df.loc[df['학번'] > 1110, '수학'] = np.nan
3222
             df.tail(2)
3223
3224
            학번 이름 국어 영어 수학
101111 한산섬 9889NaN 78.0
                                                  전산
3225
3226
            11 1112 하나로 89 97 NaN
                                      88.0
3227
3228
3229
          -결손값 제외하기
3230
            df.loc[df['수학'].isnull()]
3231
            학번 이름 국어 영어 수학
101111 한산섬 9889NaN 78.0
3232
                                                  전산
3233
            11 1112 하나로 89 97 NaN
3234
                                      88.0
3235
3236
          -결손값이 포함되어 있는 data 제외
3237
3238
             df.dropna().loc[8:] #8번째 이후 data 중 NaN값이 있는 data제외
3239
                      이름 국어 영어 수학 전산
3240
                     더높이 88 99 53.0 88.0
            8 1109
3241
3242
            9 1110
                     아리랑 68 79 63.0 66.0
3243
3244
          -dropna()는 비파괴적 조작이다.
3245
          -따라서 df에는 이전 data가 남아있다.
3246
          -DataFrame의 내용을 파괴적으로 다시 쓰는 경우
3247
3248
            df.dropna(inplace = True)
3249
3250
3251
            --Series나 DataFrame은 작성된 시점에 data형이 자동으로 설정된다.
3252
             --수치 data는 NumPy의 data형이 저장되고, 문자열 등의 data는 object 형으로 취급된다.
3253
            --Series의 data 형을 확인하는 경우
3254
            --Series의 data 형을 확인할 때에는 dtype을 참조한다.
3255
3256
               df['국어'].dtype
3257
3258
               dtype('int64')
3259
3260
          -DataFrame의 data 형을 확인하는 경우
3261
          -DataFrame의 data 형을 확인할 때는 dtypes를 참조한다.
```

```
3263
             df.dtypes
3264
3265
             학번
                    int64
3266
             이름
                    object
3267
             국어
                    int64
3268
                    int64
             영어
3269
             수학
                    float64
3270
             전산
                    float64
3271
             dtype: object
3272
3273
          -형을 변환하는 경우에는 astype() 을 사용한다.
3274
           -인수에는 type형 또는 NumPy의 data 형을 지정한다.
3275
3276
             df['학번'].astype(np.str)
3277
3278
             0
                1101
3279
                1102
3280
                 1103
3281
             3
                 1104
3282
             ...
3283
3284
             10
                 1111
             11 1112
3285
3286
             Name: 학번, dtype: object
3287
3288
          -복수열의 형을 변경하는 경우
3289
          -인수에 사전을 지정한다.
3290
3291
             df.astype({'영어':np.float64, '수학':np.str})
3292
3293
             df.dtypes
3294
3295
                    int64
             학번
3296
             이름
                    object
3297
             국어
                    int64
3298
             영어
                    int64
3299
                    float64
             수학
                                   #변경되지 않음. 비파괴적이어서...
3300
             전산
                    float64
3301
             dtype: object
3302
3303
          -DataFrame을 다시 쓰는 경우
3304
             df[' \rightarrow \phi'] = df[' \rightarrow \phi'].astype(np.str)
3305
3306
             df.dtypes
3307
             -----
3308
             학번
                    int64
3309
                    object
             이름
3310
                    int64
             국어
3311
                    int64
             영어
3312
             수학
                    object
                                #변경됐음.
3313
                    float64
             전산
             dtype: object
3314
3315
3316
           -Sort 하기
3317
3318
             del df
             df = pd.read_csv('sungjuk_utf8.csv', header = None,
3319
3320
                  names = ['학번', '이름', '국어', '영어', '수학', '전산'])
3321
3322
             df.sort_values('국어', ascending=False)
3323
             -----
3324
                                국어 영어 수학 전산
                         이름
                1106 튼튼이 98 97 93 88
1108 더크게 98 67 93 78
3325
             5
3326
3327
             101111 한산섬 98897378
3328
3329
3330
3331
           -sort_values() 역시 비파괴적 조작이다.
3332
           -덮어쓰려면 inplace에 True를 할당한다.
3333
3334
3335
3336
      16. 다양한 data 불러오기
3337
        1)Pandas는 다음과 같이 다양한 형식의 data를 불러올 수 있다.
3338
           -CSV
3339
           -Excel
3340
           -Database
3341
           -JSON
3342
           -MessagePack
3343
           -HTML
3344
           -Google BigQuery
           -Clipboard
3345
```

```
3348
           -기타(http://pandas.pydata.org/pandas-docs/stable/io.html)
3349
3350
3351
      17. CSV file 불러오기
3352
3353
        1)pandas.read_csv() 함수를 사용한다.
3354
        2)첫번째 인수에 file 경로를 넘겨주면 DataFrame 형 object를 넘겨준다.
3355
        3)File 경로 또는 URL 형식으로 지정할 수 있다.
3356
3357
          import pandas as pd
3358
3359
           df = pd.read_csv('friend_list.csv')
3360
           df.head()
3361
               name age job
John 20 student
3362
3363
               Jenny 30 developer
Nate 30 teacher
Julia 40 dentist
3364
          1
3365
           2
3366
          3
               Brian 45 manager
3367
3368
3369
3370
        4)DataFrame.head()는 앞에서 5행분의 DataFrame을 넘겨준다.
3371
        5)인수에 정수값을 넘기는 방식으로 행수를 지정할 수 있다.
3372
3373
           df = pd.read_csv('friend_list.csv')
3374
          df.head(2)
3375
3376
             nameage job
           0 John 20 student
3377
3378
          1 Jenny 30 developer
3379
3380
3381
        6)뒤에서부터 읽을 행의 갯수를 지정할 수도 있다.
3382
3383
          df = pd.read_csv('friend_list.csv')
3384
           df.tail(2)
3385
3386
               name age job
3387
               Brian 45 manager
3388
          5
               Chris 25 intern
3389
3390
3391
        7)각 열은 Series이다.
3392
3393
          type(df.job)
3394
3395
           pandas.core.series.Series
3396
3397
3398
        8)지정한 열을 DataFrame의 index로 하기
3399
           -아래 예제처럼 keyword 인수 index_col에 수치 또는 열 이름을 지정하여 지정한 열을 DataFrame의 index로 한다.
3400
3401
             # index로 지정할 열을 번호로 지정
3402
             df = pd.read_csv('friend_list.csv', index_col = 0)
3403
             df.head()
3404
3405
                       age job
3406
             name
                       20 student
3407
             John
3408
             Jenny
                       30 developer
                      30 teacher
40 dentist
3409
             Nate
3410
             Julia
                   45 manager
3411
             Brian
3412
3413
             #index로 지정할 열을 열 이름으로 지정
3414
             df = pd.read_csv('friend_list.csv', index_col = 'job')
3415
3416
             _____
3417
                         name age
3418
             job
3419
             student
                         lohn
                                   20
3420
             developer
                         Jenny
3421
                                30
             teacher Nate
3422
             dentist
                        Julia
                                 40
                         Brian 45
3423
             manager
3424
3425
3426
        9)지정한 열을 지정한 형으로 불러오기
3427
           -Keyword 인수 dtype에 열 이름(key)과 형(값)을 사전형으로 지정하여 지정한 열을 지정한 형으로 불러올 수 있다.
3428
```

#형 지정

3429

3346

3347

-Pickle

-공공데이터포털: https://www.data.go.kr/

```
3430
             df = pd.read_csv('friend_list.csv', dtype={'age':float})
3431
             df.head()
3432
                    name age job
John 20.0 student
3433
             n
3434
3435
             1
                    Jenny
                              30.0
                                     developer
                            30.0
3436
             2
                    Nate
                                     teacher
3437
             3
                    Julia
                             40.0
                                     dentist
                           45.0 manager
3438
             4
                    Brian
3439
3440
3441
        10)형식이 유사한 txt file 읽어오기
3442
           -CSV file처럼 txt file도 각 열의 구분을 ','로 할 경우
3443
3444
             df = pd.read_csv('friend_list.txt')
3445
             df.head()
3446
3447
                    name age job
                            20 student
30 developer
3448
             0
                    John
3449
             1
                    Jenny
                           30 teacher
40 dentist
3450
             2
                    Nate
3451
             3
                    Julia
3452
             4
                    Brian 45 manager
3453
3454
3455
        11)만일 file의 column들이 쉼표로 구분되어 있지 않은 경우
3456
           -delimiter parameter에 구분자를 지정해서 column을 나눠야 한다.
3457
3458
             df = pd.read_csv('friend_list_tab.txt')
3459
             df.head()
3460
3461
                    nameage job
3462
             0 John\t20\tstudent
3463
             1 Jenny\t30\tdeveloper
3464
             2 Nate\t30\tteacher
3465
             3 Julia\t40\tdentist
3466
             4 Brian\t45\tmanager
                                   #구분이 어려움.
3467
3468
             df = pd.read_csv('friend_list_tab.txt', delimiter = '\t')
3469
             df.head()
3470
             _____
3471
                    name age job
                            20 student
30 developer
             0
3472
                    John
3473
             1
                    Jenny
                              30 teacher
3474
             2
                    Nate
                             40 dentist
3475
             3
                    Julia
                           45 manager
3476
             4
                    Brian
3477
3478
        12)file에 data header가 없는 csv file을 사용할 때
3479
3480
           -만일 data header가 없으면, header = None으로 지정해야 첫번째 data가 data header로 들어가는 것을 막을 수 있다.
3481
3482
             df = pd.read_csv('friend_list_no_head.csv')
3483
             df.head()
3484
             _____
3485
                    John 20 student
                                          #첫 번째 행이 header가 돼버림.
                    Jenny 30 developer
3486
             0
3487
             1
                    Nate
                            30 teacher
3488
             2
                    Julia
                            40 dentist
                    Brian 45 manager
3489
             3
3490
             4
                    Chris 25 intern
3491
3492
             df = pd.read_csv('friend_list_no_head.csv', header = None)
3493
             df
3494
             ______
3495
3496
             0
                  John 20 student
                  Jenny 30 developer
Nate 30 teacher
3497
             1
3498
                         30 teacher
             2
3499
             3
                  Julia
                         40 dentist
                  Brian 45 manager
3500
             4
             5
                  Chris 25 intern
3501
3502
3503
           -만일 header가 없는 data를 호출했을 경우, DataFrame 생성 후, column header를 지정할 수 있다.
3504
3505
             df.columns = ['Name', 'Age', 'Job']
3506
             df.index = ['1101', '1102', '1103', '1104', '1105', '1106']
3507
3508
3509
                      Name Age Job
                              20 student
3510
             1101
                    John
3511
             1102
                    Jenny
                              30
                                   developer
3512
             1103
                              30 teacher
                    Nate
3513
             1104
                    Julia
                              40 dentist
```

```
3515
              1106
                      Chris
                             25
                                  intern
3516
3517
           -file을 열 때 동시에 header에 column을 지정해야 할 경우
3518
3519
              df = pd.read_csv('friend_list_no_head.csv', header = None, names=['Name', 'Age', 'Job'])
3520
              df.head()
3521
3522
              위의 결과와 동일
3523
3524
         13)외부 CSV file 읽기
3525
3526
            -https://github.com/vincentarelbundock/Rdatasets/tree/master/csv/datasets
3527
3528
3529
         14)기타 option
3530
           -read_csv()에는 다수의 option이 준비되어 있다.
3531
            -상세한 문서의 내용은 문서를 참조한다.
           -http://panda.pydata.org/pandas-docs/stable/io.html#io-read-csv-table
3532
3533
3534
         15)DataFrame csv file로 저장하기
3535
3536
3537
           import pandas as pd
3538
3539
           user_list = [
               {'Name':'John', 'Age':25, 'Gender': 'male', 'Address':'Chicago'},
3540
3541
               {'Name':'Smith', 'Age':35, 'Gender' : 'male', 'Address':None},
               {'Name':'Jenny', 'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
3542
3543
3544
3545
           df = pd.DataFrame(user_list)
3546
3547
           df = df[['Name', 'Age', 'Gender', 'Address']]
3548
3549
           df.head()
3550
3551
                        Age
                              Gender
           Name
                                          Address
3552
           0 John
                        25
                             male
                                          Chicago
3553
           1 Smith
                        35
                             male
                                          None
3554
                        45
           2 Jenny
                             female
                                          Dallas
3555
3556
           df.to csv('user list.csv')
3557
           -기본적으로 to_csv()는 index = True, header = True가 설정되어 있다.
3558
3559
           -만일 index = False로 설정할 경우
3560
3561
              df.to_csv('user_list.csv', index=False)
3562
3563
           -각 행의 index 즉 0, 1, 2가 사라진 csv file이 생성된다.
3564
           -또한 header = False로 설정하면 각 열의 header가 없어진다.
3565
3566
           -Smith의 거주지가 None이기 때문에 빈칸으로 값이 들어간다.
3567
           -만일 CSV로 file 저장시 빈칸대신 '-'를 넣어서 뭔가 있다는 것을 설정하려면,
3568
3569
              df.to_csv('user_list.csv', na_rep = '-')
3570
3571
           -이렇게 설정하면 해당 칸에는 '-'가 들어가게 된다.
3572
3573
3574
3575
      18. Excel file 불러오기
3576
         1)pandas.read_excel() 함수를 사용한다.
3577
         2)사전준비를 위해 xlrd module을 설치여부를 확인한다.
3578
3579
           !conda list | grep xlrd #Windows에서는 사용 불가
3580
3581
           $ conda install -y xrld==1.2.0 #설치안되어 있으면 설치
3582
           # Excel file 불러오기
3583
3584
           df = pd.read_excel('재무실적.xlsx')
3585
           df.head()
3586
3587
         3)불러오는 sheet 지정하기
3588
            -기본 설정으로는 첫 번째 Sheet를 불러온다.
3589
           -Sheet 이름을 지정해서 불러올 때는 keyword 인수 sheet_name에 sheet이름을 지정한다.
3590
3591
              df = pd.read_excel('재무실적.xlsx', sheet_name = 'data2')
3592
              df.head()
3593
3594
3595
3596
      19. SQL을 사용해서 불러오기
3597
         1)pandas.read_sql() 함수를 사용한다.
```

1105

Brian

45

manager

```
DBAPI2(PEP 249, Python Database API Specification v2.0, https://www.python.org/dev/peps/pep-0249)의 접속 instance를 넘겨준다.
3599
3600
         3)MariaDB with Python
3601
            -설치여부 확인하기
3602
3603
              !conda list | grep mysql-connector-python
3604
               #Windows에서는 grep 명령어 사용 불가
3605
3606
            -mysql-connector-python 설치하기
3607
3608
              -- In Anaconda Prompt,
3609
3610
                 $ conda install -y mysql-connector-python
3611
3612
                 import mysql.connector as mariadb
3613
                 mariadb connection = mariadb.connect(user='root', password='javamariadb', host='localhost', database='world')
3614
                 cursor = mariadb_connection.cursor()
3615
3616
3617
                 cursor.execute("SELECT ID, Name, CountryCode, District, Population FROM city WHERE CountryCode='KOR'")
3618
3619
                 for ID, Name, Country Code, District, Population in cursor:
                    print('ID = %d, Name = %s, CountryCode = %s, District = %s, Popluation = %d' % (ID, Name,
3620
                    CountryCode,District,Population))
3621
3622
                 ID = 2331, Name = Seoul, CountryCode = KOR, District = Seoul, Popluation = 9981619
3623
                 ID = 2332, Name = Pusan, CountryCode = KOR, District = Pusan, Popluation = 3804522
3624
                 ID = 2333, Name = Inchon, CountryCode = KOR, District = Inchon, Popluation = 2559424
                 ID = 2334, Name = Taegu, CountryCode = KOR, District = Taegu, Popluation = 2548568
3625
3626
                 ID = 2335, Name = Taejon, CountryCode = KOR, District = Taejon, Popluation = 1425835
3627
                 . . .
3628
3629
                 mylist = []
3630
                 for ID, Name, Country Code, District, Population in cursor:
3631
                    list = []
3632
                    list.append(ID); list.append(Name); list.append(CountryCode)
3633
                    list.append(District); list.append(Population)
3634
                    mylist.append(list)
3635
                 df = pd.DataFrame(data = mylist, columns = ['ID', 'Name', 'CountryCode', 'District','Population'])
3636
3637
                 print(df)
3638
3639
3640
         4)Oracle with Python
3641
            Oracle cx_oracle 7
3642
            -https://www.oracle.com/database/technologies/appdev/python.html
3643
            -Install on Windows
3644
              $ python -m pip install cx Oracle --upgrade
3645
3646
            -In Anaconda Prompt
3647
              $ conda install cx_oracle
3648
3649
            -Connection
3650
              import cx_Oracle
3651
3652
              --conn = cx Oracle.connect('hr', 'hr', 'localhost:1521/XE')
3653
3654
              --conn1 = cx Oracle.connect('scott/tiger@localhost:1521/XE')
3655
3656
                 print(conn1)
3657
3658
                 <cx_Oracle.Connection to hr@localhost:1521/XE>
3659
3660
              --dsn_tns = cx_Oracle.makedsn('localhost', 1521, 'XE')
3661
                 print(dsn tns)
3662
3663
                 (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SID=XE)))
3664
3665
                 conn2 = cx_Oracle.connect('scott', 'tiger', dsn_tns)
3666
                 print(conn2)
3667
3668
                 <cx Oracle.Connection to
                 hr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SID=XE)))>
3669
3670
                 conn.version
3671
3672
                 '11.2.0.2.0'
3673
3674
              -Cursor Objects
3675
                 conn = cx_Oracle.connect('hr', 'hr', 'localhost:1521/XE')
3676
                 cursor = conn.cursor()
3677
                 sql = """SELECT employee_id, first_name, salary, to_char(hire_date, 'yyyy-mm-dd'), department_name, city
3678
                         from employees e inner join departments d on e.department_id = d.department_id
```

2)첫번째 인수에 query를 실행하는 SQL문, 두번째 인수에 SQLAlchemy(http://docs.sqlalchemy.org/en/latest/dialects/index) 또는

inner join locations I on d.location_id = I.location_id"""	
3680 cursor.execute(sql)	
3681	
for employee_id, first_name, salary, hire_date, department_name, city in curs	or:
print(employee_id, first_name, salary, hire_date, department_name, city)	
3684	
3685 cursor.close()	