```
1 1. Pandas⊢
     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, 사회
     과학 등 많은 분야의 data 처리에 적합하다.
 4
 5
 6 2. Pandas의 특징
 7
     1)Pandas의 주요 기능을 설명
 8
       -쉬운 결손값(missing data)처리
 9
       -Label 위치를 자동적/명시적으로 정리한 data 작성
10
       -Data 집약
11
       -고도의 label base의 slicing, 추출, 큰 dataset의 subset화
12
       -직감적인 dataset 결합
       -Dataset의 유연한 변환 및 변형
13
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: <a href="http://pandas.pydata.org">http://pandas.pydata.org</a>
       -Repository: <a href="https://github.com/pandas-dev/pandas">https://github.com/pandas-dev/pandas</a>
21
       -PyPI: https://pypi.python.org/pypi/pandas
22
23
24
     3)설치 여부 확인
25
       -Jupyter 환경
26
         -!conda list | grep pandas
27
         -Windows에서는 grep명령어를 사용할 수 없음.
28
         -따라서 Windows에서는
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
     1)Series는 index라고 불리는 label을 가진 동일한 data형을 가지는 1차원 data이다.
43
44
       -Series는 DataFrame과 함께 pandas에서 제공하는 데이터 구조 중 하나이다.
45
       -DataFrame이 2차원 자료구조라면 Series는 1차원 자료구조이다.
       -DataFrame과 비슷하지만 1차원 자료구조이기 때문에 columns 속성이 없고 index 속성만 있다.
46
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 = [ ] )
       -제1인수에는 다음과 같은 1차원의 data를 넘겨준다.
61
62
         --List
63
         --Tuple
64
         --Dirctionary
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
         2
         dtype: float64
76
77
78
         ser.index = ['a', 'b', 'c']
79
         ser
80
81
         a 10.0
82
         b 20.0
         c 0.3
83
84
         dtype: float64
85
86
         ser1 = pd.Series([1,2,3], index=['a', 'b', 'c'])
87
88
89
         a 1
90
         b 2
91
            3
         С
92
         dtype: int64
93
         ser2 = pd.Series( [1, 2, 3, 4], index = ['USA', 'Germany', 'France', 'Japan'] )
94
95
         ser2
96
97
         USA
98
         Germany 2
99
         France
                  3
```

```
100
           Japan
101
           dtype: int64
102
           ser3 = pd.Series([1, 2, 5, 4], index = ['USA', 'Germany', 'Italy', 'Japan'])
103
104
           ser3
105
106
           USA
                    1
107
           Germany
                     2
108
           Italy
                   5
           Japan
109
                    4
110
           dtype: int64
111
112
           fruits = Series([2500,3800,1200,6000], index=['apple','banana','peer','cherry'])
113
           fruits
114
115
           apple
                        2500
116
           banana
                        3800
117
                           1200
           peer
118
           cherry
                      6000
119
           dtype: int64
120
121
          fruits.values
122
123
           array([2500, 3800, 1200, 6000], dtype=int64)
124
125
           fruits.index
126
           Index(['apple', 'banana', 'peer', 'cherry'], dtype='object')
127
128
129
        -dict로 Series 생성하기
130
           --Series형태는 dict와 매우 유사 하여 key가 index로 바뀌었다고 착각할 수 있다.
131
           --그러나 사실상 데이터 구조 자체가 매우 다르기 때문에 dict와는 사용하는 방식이 다르므로 주의해야 한다.
132
133
           ser4 = pd.Series( {'a':100, 'b':200, 'c':300} )
134
           ser4
135
136
              100
           a
137
              200
           b
              300
138
           С
           dtype: int64
139
140
141
           fruits_dic = {'apple': 2500,'banana':3800,'peer':1200,'cherry':6000}
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
158
          2
159
160
        -loc를 사용하지 않는 서식
161
          --loc를 사용하지 않는 다음과 같은 서식도 있다.
162
163
            ser['b']
164
165
            2
166
167
        -Label의 범위 지정
168
          --Label의 범위를 지정해서 slice를 할 수 있다.
169
           ser.loc['b': 'c']
170
171
172
           b 2
           c 3
173
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
            ser.loc[['a', 'c']]
182
183
184
            a 1
185
            c 3
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
195
196
        -iloc의 slice는 Python 표준 list나 tuple에 대한 slice와 같이 동작한다.
197
        -위치를 slice로 지정
198
          ser.iloc[1:3]
199
200
201
          b 2
```

```
202
          c 3
203
          dtype: int64
204
205
206
      7)논리값을 사용해서 data 선택하기
        -loc와 iloc에는 논리값의 list를 넘길 수 있다.
207
208
209
          ser.loc[[True, False, True]]
210
211
          a 1
212
          c 3
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
          b False
221
222
          c True
223
          dtype: bool
224
225
        -이것을 이용해서 data를 추출할 수 있다.
226
227
          ser.loc[ser != 2]
228
229
          a 1
230
          c 3
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, 6000],
          index=['apple','banana','peer','cherry'])
238
239
          fruits
240
241
                       2500
          apple
242
          banana
                       3800
243
          peer
                         1200
244
          cherry
                     6000
245
          dtype: int64
246
247
          new_fruits = fruits.drop('banana')
248
          new_fruits
249
250
          apple
                       2500
251
          peer
                         1200
252
          cherry
                     6000
```

```
253
          dtype: int64
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
                         5
264
          apple
          banana
265
                         9
266
          cherry
                     10
                           3
267
          peer
268
          dtype: int64
269
270
          fruits2
271
272
                         3
          apple
                         2
273
          orange
274
          banana
                         9
275
          cherry
276
          mango
                         10
277
          dtype: int64
278
279
        -Series의 연산은 index에 의해 연산된다.
        -아래의 결과를 보면 'mango' index와 'orange' index는 fruits2 Series에만 있으므로 그 결과가 NaN이
280
        되고
281
        -'peer' index는 fruits1 Series만 있으므로 그 결과가 NaN이 된다.
282
283
          fruits1 + fruits2
284
285
          apple
                         8.0
286
          banana
                       18.0
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
          cherry
                     6000
                       3800
301
          banana
302
                       2500
          apple
```

```
303
                        1200
         peer
304
         dtype: int64
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
                  0
                  5
314
         apple
315
         banana
                  9
         cherry 10
316
317
         peer
                  3
318
319
       -열 단위의 DataFrame을 행단위로 바꾸려면 T 속성 또는 transpose() 함수를 이용.
320
       -T 속성과 transpose() 함수는 행렬의 전치행렬을 반환한다.
321
322
         fruits1.to_frame().T
323
324
             apple banana cherrypeer
             5 9
325
                          10
326
327
         fruits1.to_frame().transpose()
328
329
             apple banana cherrypeer
330
         0 5
                  9
                          10
                                 3
331
332
333
334 4. DataFrame
335
      1)DataFrame은 행과 열에 label을 가진 2차원 data이다.
336
      2)Data형은 열마다 다른 형을 가질 수 있다.
337
      3)1차원 data인 Series의 집합으로 인식하는 것도 가능하다.
      4)Series의 특징을 포함해서 DataFrame에는 다음과 같은 특징이 있다.
338
339
       -행과 열에 label을 가진 2차원 data
340
       -열마다 다른 형태를 가질 수 있음.
       -Table형 data에 대해 불러오기 / data 쓰기가 가능
341
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
356
           d = \{ 'col1': [1, 2], 'col2': [3, 4] \}
357
          df = pd.DataFrame(data=d)
           df
358
359
360
              col1col2
361
          0
              1
                     3
              2
                     4
362
           1
363
364
         -dict는 다음처럼 list에 저장되어 있어도 쉽게 DataFrame으로 만들 수 있다.
365
366
           d = [\{'col1': 1, 'col2': 3\}, \{'col1': 2, 'col2': 4\}]
           df = pd.DataFrame(data=d)
367
368
           df
369
370
              col1col2
371
          0
             1
                     3
372
           1
              2
                     4
373
374
         -만일 list내의 dict의 요소의 수가 다를 경우에는 NaN으로 채워진다.
375
376
           d = [\{'col1': 1, 'col2': 3\}, \{'col1': 2, 'col2': 4\}, \{'col1': 2\}]
           df = pd.DataFrame(data=d)
377
378
           df
379
380
              col1col2
381
                     3.0
          0
              1
382
              2
                     4.0
           1
383
           2
              2
                    NaN
384
385
           emp_list = [
               {'name<sup>'</sup>:'John', 'age' : 25, 'job' : 'Manager'},
386
               {'name':'Smith', 'age': 30, 'job': 'Salesman'}
387
388
           ]
389
390
           df = pd.DataFrame(emp_list)
391
392
393
                    job
             age
                                  name
394
           0 25
                    Manager
                                  John
                    Salesman
395
           1 30
                                  Smith
396
           #list의 순서와 맞지 않음. key의 alphabet 순서와 동일
397
398
           df = df[['name','age','job']]
399
           df.head()
400
                      age job
401
              name
                      25
402
              John
                              Manager
           0
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
           Γ
410
              ('name', ['John', 'Smith']),
411
              ('age', [25, 30]),
              ('job', ['Manager', 'Salesman']),
412
413
           1
414
        )
415
416
        df = pd.DataFrame.from_dict(emp_ordered_list)
417
        df.head()
418
419
          name age job
420
                  25
        0 John
                          Manager
421
         1 Smith
                  30
                        Salesman
422
423
424
       6)list를 이용해 DataFrame 만들기
425
426
        df = pd.DataFrame(
427
              [[1, 10, 100], [2, 20, 200], [3, 30, 300]], index=['r1', 'r2', 'r3'],
428
              columns=['c1','c2','c3'])
429
        df
430
431
              c1
                    c2 c3
432
        r1 1
                  10
                         100
433
        r2 2
                  20
                         200
434
        r3 3
                  30
                        300
435
436
        emp_list = [
             _
['John', 25, 'Manager'],
437
438
             ['Smith', 30, 'Salesman']
439
           ]
440
441
        column_names = ['name', 'age', 'job']
        df = pd.DataFrame.from_records(emp_list, columns =column_names)
442
443
        df.head()
444
445
                    age
                          iob
             name
446
           0 John
                      25
                            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
```

```
df = pd.DataFrame.from items(emp list)
455
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.
        """Entry point for launching an IPython kernel.
458
459
460
        df.head()
461
462
                 age job
          name
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
471
        ser1 = pd.core.series.Series(list)
472
        ser2 = pd.core.series.Series(['one', 'two', 'three'])
        pd.DataFrame(data=dict(num=ser1, word=ser2))
473
474
475
          num word
476
        0 1
               one
477
        1 2
               two
478
        2 3
               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
          member df = pd.read csv("member data.csv", sep=",")
486
487
          member df
488
489
            Name Age Email
                                            Address
                        kildong@hong.com 서울시 강동구
490
          0 홍길동
                   20
                   25
                        kilseo@hong.com
491
          1 홍길서
                                          서울시 강서구
492
          2 홍길남
                   26
                        south@hong.com
                                           서울시 강남구
                        book@hong.com
493
          3 홍길북
                   27
                                           서울시 강북구
494
495
496
        -csv파일에 헤더 정보가 없을 경우 header=None 인수를 포함하면 열 이름이 0, 1, 2, ...순으로 자동 지정
        된다.
497
498
499
      9)sklearn.datasets module data를 DataFrame으로 변환하기
        -Scikit-learn package에는 학습을 위한 많은 dataset이 제공된다.
500
        -Scikit-learn에서 제공하는 dataset은 dict 형식으로 되어 있다.
501
502
```

```
503
         import numpy as np
504
         import pandas as pd
505
         from sklearn import datasets
         iris = datasets.load iris()
506
507
         iris
508
509
         {'data': array([[5.1, 3.5, 1.4, 0.2],
510
         [4.9, 3., 1.4, 0.2],
511
         [4.7, 3.2, 1.3, 0.2],
512
         [4.6, 3.1, 1.5, 0.2],
513
         [5., 3.6, 1.4, 0.2],
         ... 생략 ...
514
515
         [6.7, 3., 5.2, 2.3],
         [6.3, 2.5,5., 1.9],
516
         [6.5, 3., 5.2, 2.],
517
518
         [6.2, 3.4, 5.4, 2.3],
         [5.9, 3., 5.1, 1.8]]),
519
520
521
         522
523
         524
         525
         526
         527
         2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2]),
528
         'target names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
529
         'DESCR': 'Iris Plants Database...
530
531
         'feature names': ['sepal length(cm)',
532
         'sepal width (cm)',
533
         'petal length(cm)',
534
         'petal width (cm)'],
535
         'filename':
         'c:\\pythonhome\\projectenv\\lib\\site-packages\\sklearn\\datasets\\data\\iris.csv'}
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
         iris df.head()
541
542
543
            sepal length (cm)
                             sepal width (cm)
                                              petal length (cm)
                                                               petal width (cm)
            species
544
         0
            5.1
                             3.5
                                                1.4
                                                                0.2
            setosa
545
         1
            4.9
                             3.0
                                                1.4
                                                                0.2
            setosa
                             3.2
546
            4.7
                                                1.3
                                                                0.2
         2
            setosa
547
                             3.1
                                                1.5
                                                                0.2
         3
            4.6
            setosa
```

```
548
              5.0
                                   3.6
                                                                             0.2
          4
                                                         1.4
              setosa
549
550
551
552 6. 이름 지정하기
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
                  이름 나이 이메일
561
                                                  주소
562
           0
                  홍길동 20 <u>kildong@hong.com</u> 서울시 강동구
                  홍길서 25 <u>kilseo@hong.com</u> 서울시 강서구
홍길남 26 <u>south@hong.com</u> 서울시 강남구
563
           1
                  홍길남 26 <u>south@hong.com</u>
564
           2
565
           3
                  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
        member df.index = ["동", "서", "남", "북"]
574
575
        member df
576
577
            이름 나이 이메일
                                             주소
578
        동 홍길동 20 <u>kildong@hong.com</u> 서울시 강동구

      서
      홍길서
      25
      kilseo@hong.com
      서울시 강서구

      남
      홍길남
      26
      south@hong.com
      서울시 강남구

      북
      홍길북
      27
      book@hong.com
      서울시 강북구

579
580
581
582
583
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 구
         분할 수 있다.
587
        -level의 이름은 names 속성을 이용할 수 있다.
588
        -열의 이름을 지정할 때에는 columns 속성을 이용하고, 열의 level을 지정하려면 columns.names 속성을
589
        -행(인덱스)의 이름을 지정할 때에는 index 속성을 이용하고, 행의 레벨을 지정하려면 index.names 속성을
        이용한다.
590
           member df = pd.read csv("member data.csv", comment='#')
591
          member_df.columns = [["기본정보", "기본정보", "추가정보", "추가정보"],
592
                                       ["이름", "나이","이메일", "주소"]]
593
594
           member df.columns.names = ["정보구분", "상세정보"]
```

```
member_df.index = [["좌우","좌우","상하","상하"],
["동", "서", "남", "북"]]
595
596
         member_df.index.names = ["위치구분", "상세위치"]
597
         member df
598
599
                정보구분 기본정보 추가정보
600
               상세정보 이름 나이 이메일
601
                                                  주소
602
         위치구분 상세위치
603
         좌우
               동 홍길동 20 kildong@hong.com 서울시 강동구
                   홍길서 25 kilseo@hong.com 서울시 강서구
홍길남 26 south@hong.com 서울시 강남구
               서
604
               남 홍길남 26 <u>south@hong.com</u> 서울시 강남
북 홍길북 27 <u>book@hong.com</u> 서울시 강북구
605
         상하
606
607
608
609
610 7. 부분 데이터 조회
     member df = pd.read csv("member data.csv")
611
612
     member_df
613
614
     1)단일 열 조회
615
       -참조 형식(.) 또는 배열 형식([])을 이용하면 열 하나를 조회할 수 있다.
616
       -DataFrame의 열 이름을 참조 형식(. 연산자)을 이용해서 정보를 조회할 수 있다.
617
618
         member_df.Name
619
620
         0
               홍길동
621
         1
               홍길서
         2
622
               홍길남
623
         3
               홍길북
624
         Name: Name, dtype: object
625
       -열 정보를 조회할 때에는 배열 형식(["열이름"])으로도 가능하다.
626
627
       member_df["Name"]
628
       -----
629
             홍길동
630
       1
              홍길서
       2
631
             홍길남
632
              홍길북
       3
633
       Name: Name, dtype: object
634
635
636
     2)loc를 이용한 이름으로 조회
       -loc[]를 이용하면 행 또는 열 이름으로 부분 데이터셋을 조회할 수 있다.
637
638
       -콜론(:)은 사이의 모든 행을 선택한다.
639
640
         member_df.loc[0:2]
641
            Name Age Email
642
                                          Address
         0 홍길동
                   20 <u>kildong@hong.com</u> 서울시 강동구
643
                        kilseo@hong.com 서울시 강서구
            홍길서 25
644
         1
645
         2
            홍길남
                   26
                        south@hong.com
                                          서울시 강남구
```

```
646
647
       -loc에서의 숫자는 인덱스가 아니다.
648
       -그러므로 0:2는 인덱싱 할 때의 from:to 개념을 적용한 0행부터 2행까지를 의미하는 것이 아니다.
       -loc() 함수에서 0:2는 0, 1, 2 행을 의미한다.
649
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
656
         member_df.loc[0:2, "Name":"Email"]
657
658
           Name Age Email
659
         0 홍길동
                  20
                       kildong@hong.com
                       kilseo@hong.com
                  25
660
         1 홍길서
661
         2 홍길남
                  26
                       south@hong.com
662
663
       -행 또는 열의 이름으로 데이터를 조회려면 리스트 형식으로 지정한다.
664
         member_df.loc[[0,2], ["Name","Email"]]
665
666
667
            Name Email
                   kildong@hong.com
668
         0
            홍길동
669
         2
             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
         0
            홍길동
                   20
                         kildong@hong.com 서울시 강동구
                   25
                        kilseo@hong.com 울시 강서구
677
         1
            홍길서
678
         2
            홍길남
                   26
                        south@hong.com
                                            서울시 강남구
679
680
       -모든 행을 지정하는 경우도 같다.
681
682
         member_df.loc[:, 'Email']
683
684
           kildong@hong.com
         0
             kilseo@hong.com
685
         1
             south@hong.com
686
         2
687
         3
             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@hong.com
         1
699
            26
                  south@hong.com
700
701
         member_df.iloc[0:3, 0:3]
702
            -----
703
            Name Age Email
                        kildong@hong.com
704
            홍길동
                   20
         0
705
         1
            홍길서
                   25
                         kilseo@hong.com
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
            Name Age Email
                                            Address
                        book@hong.com 서울시 강북구
715
       3
            홍길북
                   27
                        south@hong.com
716
       2
            홍길남
                   26
                                          서울시 강남구
                                           서울시 강서구
717
       1
                   25
                         kilseo@hong.com
            홍길서
718
            홍길동
                   20
                        kildong@hong.com 서울시 강동구
719
720
       member_df.iloc[0::2,[1,3]]
721
722
           Age Address
723
       0 20
                서울시 강동구
724
       2
          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
         iris = sm.datasets.get_rdataset("iris", package="datasets")
733
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.
         --data: 데이터를 담고 있는 DataFrame.
741
         -- doc : 데이터에 대한 설명 문자열.
742
         --이 설명은 R package의 내용을 그대로 가져온 것이므로 예제 코드가 R로 되어 있어 Python에서 바로 사
743
         용할 수 없다.
744
       -statsmodels 패키지의 qet rdataset() 함수로 불러온 데이터의 data 속성은 데이터를 담고 있는
```

```
DataFrame이다.
745
746
           iris df = iris.data
           iris df.head()
747
748
                                              Petal.Length Petal.Width
749
               Sepal.Length
                               Sepal.Width
                                                                           Species
750
               5.1
                               3.5
                                                                 0.2
           0
                                                1.4
                                                                               setosa
751
               4.9
                                3.0
                                                1.4
                                                                 0.2
           1
                                                                               setosa
752
           2
               4.7
                               3.2
                                                1.3
                                                                 0.2
                                                                               setosa
753
           3
               4.6
                                3.1
                                                1.5
                                                                 0.2
                                                                               setosa
754
           4
               5.0
                                3.6
                                                1.4
                                                                 0.2
                                                                               setosa
755
756
         -다음 코드는 versicolor 종의 데이터만 조회한다.
757
758
           iris df.loc[iris df['Species']=='versicolor'].head()
759
760
               Sepal.Length
                               Sepal.Width
                                              Petal.Length
                                                            Petal.Width
                                                                           Species
761
           50 7.0
                                3.2
                                                4.7
                                                                 1.4
                                                                               versicolor
762
           51 6.4
                               3.2
                                                4.5
                                                                 1.5
                                                                               versicolor
763
           52 6.9
                               3.1
                                                4.9
                                                                 1.5
                                                                               versicolor
           53 5.5
764
                                2.3
                                                4.0
                                                                 1.3
                                                                               versicolor
765
           54 6.5
                                2.8
                                                4.6
                                                                 1.5
                                                                               versicolor
766
767
         -다음 코드는 versicolor 종의 Sepal.Length열과 Species 열 정보만 조회한다.
768
769
           iris_df.loc[iris_df['Species']=='versicolor', ['Sepal.Length', 'Species']].head()
770
771
                   Sepal.Length Species
772
           50
                   7.0
                                 versicolor
773
           51
                   6.4
                                 versicolor
774
           52
                   6.9
                                 versicolor
775
           53
                   5.5
                                 versicolor
776
           54
                   6.5
                                 versicolor
777
778
         -다음 코드는 versicolor 종들 중에서 Sepal.Length가 6.5보다 큰 데이터만 조회한다.
779
           iris df.loc[(iris df['Species']=='versicolor') & (iris df['Sepal.Length'].astype(float) >
780
           6.5)].head()
781
782
                                 Sepal.Width
                                                              Petal.Width
               Sepal.Length
                                                Petal.Length
                                                                             Species
           50 7.0
783
                                  3.2
                                                  4.7
                                                                   1.4
                                                                                 versicolor
784
           52 6.9
                                 3.1
                                                  4.9
                                                                   1.5
                                                                                 versicolor
785
           58 6.6
                                 2.9
                                                  4.6
                                                                   1.3
                                                                                 versicolor
786
           65 6.7
                                  3.1
                                                  4.4
                                                                   1.4
                                                                                 versicolor
787
           75 6.6
                                  3.0
                                                  4.4
                                                                   1.4
                                                                                 versicolor
788
789
         -좀 더 쉬운 다른 예제를 사용해 보자.
790
791
           user list = [
792
              {'Name':'John', 'Age':25, 'Gender': 'male', 'Address':'Chicago'},
793
             {'Name':'Smith', 'Age':35, 'Gender': 'male', 'Address':'Boston'},
```

```
794
            {'Name':'Jenny', 'Age':45, 'Gender': 'female', 'Address':'Dallas'}
795
          1
796
797
          df = pd.DataFrame(user list)
798
          df = df[['Name', 'Age', 'Gender','Address']]
          df.head()
799
800
801
            Name
                   Age Gender
                                   Address
          0 John
802
                    25
                           male
                                       Chicago
          1 Smith
                    35
                         male
                                     'Boston'
803
804
          2 Jenny
                    45
                         female
                                     Dallas
805
806
        -age가 30보다 많은 사람의 정보
807
808
          df[df.Age > 30]
809
810
                         Gender
                                   Address
            Name
                   Age
811
          1 Smith
                    35
                         male
                                     Boston
812
          2 Jenny
                    45
                         female
                                     Dallas
813
814
          df.query('Age > 30')
815
816
            Name
                   Age
                         Gender
                                   Address
                         male
817
          1 Smith
                    35
                                     Boston
818
          2 Jenny
                   45
                         female
                                     Dallas
819
        -age가 30보다 많고 이름이 'Smith'인 사람의 정보
820
821
822
          df[(df.Age > 30) \& (df.Name == 'Smith')]
823
          # or df.loc[(df.Age > 30) & (df.Name == 'Smith')]
824
825
                   Age Gender
                                   Address
            Name
826
          1 Smith
                   35
                         male
                                     Boston
827
828
829
        -Column name이 Name과 Gender인 column만 가져오기
830
831
          df.filter(items=['Name', 'Gender'])
832
833
              Name
                     Gender
834
          0
              John
                        male
835
          1
              Smith
                     male
836
          2
                     female
              Jenny
837
        -Column name이 'A'라는 글자가 있는 Column만 가져오기
838
839
840
          df.filter(like = 'A')
841
842
                Age
                    Address
                25
843
                      Chicago
          0
844
          1
                35
                      Boston
```

```
2
                45
                     Dallas
845
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
853
          1
                Smith
                       35
854
          2
                       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
862
          1 Boston 35
863
          2 Dallas 45
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.
876
        -inplace: bool 타입.
877
          --False(기본값)이면 삭제된 결과 DataFrame을 리턴하며, True 이면 현재 DataFrame에서 데이터를
          삭제하고 None을 반환.
878
      3)row index를 사용하여 삭제하기
879
880
881
        user_list = [
           {'Age':25, 'Gender': 'male', 'Address': 'Chicago'},
882
883
           {'Age':35, 'Gender': 'male', 'Address': 'Boston'},
           {'Age':45, 'Gender' : 'female', 'Address':'Dallas'}
884
885
        1
886
887
        df = pd.DataFrame(user_list,
                     index = ['John', 'Smith', 'Jenny'],
888
889
                     columns = ['Age', 'Gender', 'Address'])
890
        df.head()
891
892
893
                   Gender
                             Address
              Age
894
               25
        John
                     male
                                 Chicago
```

```
895
         Smith 35
                                Boston
                    male
896
        Jenny 45
                    female
                                Dallas
897
898
        df.drop(['John', 'Jenny'])
899
900
                      Gender
                                Address
                Age
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
                Age Gender
                                Address
909
               35
                      male
         Smith
                                  Boston
910
911
912
      4)inplace keyword 인수 이용하기
913
914
        df = pd.DataFrame(user_list,
                      index = ['John', 'Smith', 'Jenny'],
915
916
                      columns = ['Age', 'Gender', 'Address'])
917
        df.drop(['John', 'Jenny'], inplace=True)
918
919
        df
920
921
                Age Gender
                                Address
922
         Smith 35
                      male
                                  Boston
923
924
925
       5)row가 숫자로 indexing되어 있을 때 삭제하기
926
927
        user_list = [
928
           {'Name':'John', 'Age':25, 'Gender': 'male', 'Address':'Chicago'},
           {'Name':'Smith', 'Age':35, 'Gender': 'male', 'Address':'Boston'},
929
           {'Name':'Jenny', 'Age':45, 'Gender': 'female', 'Address':'Dallas'}
930
931
        1
932
933
        df = pd.DataFrame(user_list)
934
935
        df = df[['Name', 'Age', 'Gender','Address']]
936
937
        df
938
                                    Address
939
            Name
                    Age Gender
940
        0
            John
                      25
                            male
                                        Chicago
941
         1
            Smith
                    35
                          male
                                      Boston
         2
                          female
                                      Dallas
942
            Jennv
                    45
943
944
        df = df.drop(df.index[[0, 2]])
945
        df
```

```
946
947
         Name Age Gender Address
       1 Smith 35
948
                    male Boston
949
950
     6)Condition으로 삭제하기
951
952
953
       df.head()
954
955
            Name Age Gender
                                Address
956
       0
            John 25 male
                                   Chicago
       1
            Smith 35 male
957
                                  Boston
958
       2
            Jenny 45
                        female
                                  Dallas
959
       -Age가 30 이상인 사람의 정보만 저장함으로 나머지 정보는 삭제하는 방법
960
961
         df = df[df.Age > 30]
962
963
         df
964
965
            Name Age Gender
                                Address
            Smith 35 male
                                  Boston
966
         1
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"]
973
       member_df.index = ["동", "서", "남", "북"]
974
975
       member df
976
          이름 나이 이메일
977
                                        주소
978
       동 홍길동 20 <u>kildong@hong.com</u> 서울시 강동구
                                      서울시 강서구
서울시 강남구
       서 홍길서 25 <u>kilseo@hong.com</u>
979
       남 홍길남 26 south@hong.com
북 홍길북 27 book@hong.com
980
                    south@hong.com서울시 강남book@hong.com서울시 강북구
981
982
983
     8)단일 행 삭제하기
984
985
       member df = member df.drop('북') #axis=0(기본값)이면행에서 찾아 삭제
986
       member df
987
          이름 나이 이메일
988
                                       주소
989
       동 홍길동 20 kildong@hong.com 서울시 강동구
       서 홍길서 25 kilseo@hong.com 서울시 강서구
990
991
       남 홍길남 26 south@hong.com
                                      서울시 강남구
992
       -행의 이름을 지정하지 않았다면 기본값은 아마도 숫자일 것이다.
993
       -이 경우 행 번호가 행의 이름이 된다.
994
995
996
         member_df2 = pd.read_csv("member_data.csv")
```

```
997
           member_df2.drop(2,axis=0)
998
999
       9) 단일 열 삭제하기
1000
1001
         -axis인자의 값이 1이면 열을 삭제한다.
1002
1003
           member_df = member_df.drop('주소', axis=1)
1004
           member df
1005
              이름 나이 이메일
1006
1007
           동 홍길동 20 kildong@hong.com
           서 홍길서 25 kilseo@hong.com
1008
          남 홍길남 26 south@hong.com
1009
1010
1011
1012
           user list = [
             {'Name':'John', 'Age':25, 'Gender': 'male', 'Address':'Chicago'},
1013
1014
             {'Name': 'Smith', 'Age': 35, 'Gender': 'male', 'Address': 'Boston'},
             {'Name':'Jenny', 'Age':45, 'Gender': 'female', 'Address':'Dallas'}
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
                John
                       25
1023
           0
                               male
                                          Chicago
1024
                       35
           1
                Smith
                             male
                                        Boston
1025
           2
                Jenny 45
                             female
                                        Dallas
1026
1027
           df.drop('Age', axis= 1)
1028
1029
              Name Gender Address
1030
                                  Chicago
           0
              John
                     male
              Smith male
1031
           1
                                 Boston
1032
              Jenny female
                                Dallas
1033
           df.drop('Age', axis= 1, inplace=True)
1034
1035
1036
1037
              Name Gender
                               Address
1038
          0
              John
                     male
                                  Chicago
1039
              Smith male
                                 Boston
           1
1040
              Jenny female
                                Dallas
1041
1042
1043
       10) 복수 열 삭제하기
         -여러 개 열을 삭제하려면 labels 인자를 이용한다.
1044
         -axis 인자가 1일 경우 열을 의미한다.
1045
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
           동 홍길동 20
1056
           서 홍길서 25
           남 홍길남 26
1057
1058
           북 홍길북 27
1059
1060
1061
1062 9. DataFrame 요소 추가
1063
       1)열 추가
1064
         -DataFrame에 없는 열을 지정해서 값을 할당하면 새로운 열이 만들어 진다.
1065
          -이 때 열은 가장 오른쪽에 만들어진다.
1066
1067
            member_df = pd.read_csv("member_data.csv")
            member_df["BirthYear"] = 2000
1068
1069
            member_df
            -----
1070
1071
               Name Age Email
                                                  Address
                                                               BirthYear
               홍길동 20 <u>kildong@hong.com</u> 서울시 강동구
1072
           0
                                                               2000

      1
      홍길서
      25
      kilseo@hong.com
      서울시 강서구

      2
      홍길남
      26
      south@hong.com
      서울시 강남구

      3
      홍길북
      27
      book@hong.com
      서울시 강북구

1073
                                                                 2000
1074
                                                                 2000
1075
                                                               2000
1076
1077
         -열을 추가할 때 행별로 다른 값을 지정할 수 있다.
1078
1079
            member_df = pd.read_csv("member_data.csv")
            member_df["BirthYear"] = [2001, 2002, 2003, 2004]
1080
           member_df
1081
1082
               Name Age Email
1083
                                                   Address
                                                               BirthYear
           0 홍길동 20 <u>kildong@hong.com</u> 서울시 강동구
1084
                                                               2001

      1
      홍길서
      25
      kilseo@hong.com
      서울시 강서구

      2
      홍길남
      26
      south@hong.com
      서울시 강남구

1085
                                                                 2002
1086
                                                                 2003
                             book@hong.com 서울시 강북구
            3
               홍길북 27
                                                               2004
1087
1088
1089
         -만일 누락되어야 하는 값이 있다면 None으로 지정한다.
1090
1091
            member_df = pd.read_csv("member_data.csv")
            member_df["BirthYear"] = [2001, 2002, 2003, None]
1092
1093
            member df
1094
1095
               Name Age Email
                                                   Address
                                                               BirthYear
               홍길동
                       20
                             kildong@hong.com 서울시 강동구
1096
           0
                                                               2001.0
               홍길서 25
                             kilseo@hong.com 서울시 강서구
1097
                                                                 2002.0
           1
1098
            2 홍길남
                       26
                             south@hong.com
                                                  서울시 강남구
                                                                 2003.0
```

```
book@hona.com
1099
              홍길북
                     27
                                             서울시 강북구
                                                          NaN
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
         member_df = pd.read_csv("member_data.csv")
         member_df["BirthYear"] = pd.Series([2001,2002,2004], index=[0,1,3])
1107
1108
         member df
1109
1110
            Name Age Email
                                             Address
                                                        BirthYear
                           kildong@hong.com 서울시 강동구
1111
                     20
                                                          2001.0
           0
              홍길동
                           kilseo@hong.com 서울시 강서구
                     25
1112
          1
              홍길서
                                                            2002.0
1113
           2
              홍길남
                     26
                           south@hong.com
                                              서울시 강남구
                                                            NaN
           3
                           book@hong.com
1114
              홍길북
                     27
                                             서울시 강북구
                                                          2004.0
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")
           new_member = {"Name": "한지민", "Age": 23, "Email": "jimin@naver.com",
1121
           "Address": "서울시 송파구"}
1122
         -추가할 데이터가 Series가 아니라면 ignore index=True를 설정해야 한다.
1123
1124
1125
           new df = member df.append(new member, ignore index=True)
1126
           new_df
1127
1128
              Name Age Email
                                               Address
1129
          0
              홍길동
                     20
                           kildong@hong.com 서울시 강동구
                           kilseo@hong.com 서울시 강서구
south@hong.com 서울시 강남구
book@hong.com 서울시 강북구
1130
           1
              홍길서
                     25
1131
           2
              홍길남
                     26
                      27
1132
           3
              홍길북
           4
                      23
1133
              하지민
                           jimin@naver.com 서울시 송파구
1134
1135
       4)다른 예제로 연습해 보자.
1136
1137
         user list = [
1138
             {'Name':'John', 'Age':15, 'Gender': 'male', 'Address':'Chicago', 'Job': 'Student'},
             {'Name':'Smith', 'Age':25, 'Gender': 'male', 'Address':'Boston', 'Job': 'Teachter'},
1139
             {'Name':'Jenny', 'Age':17, 'Gender': 'female', 'Address':'Dallas', 'Job': 'Student'}
1140
1141
         1
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
```

```
Chicago
                                                Student
1148
         0
             John
                     15
                          male
1149
         1
             Smith
                     25
                          male
                                    Boston
                                                Teachter
1150
         2
             Jenny
                     17
                          female
                                    Dallas
                                              Student
1151
1152
         -column 새로 추가하기
           df['Salary'] = 0
1153
1154
1155
           df.head()
1156
               Name Age Gender
1157
                                      Address Job
                                                           Salary
1158
           0
               John
                       15
                              male
                                          Chicago
                                                   Student
                                                              0
                       25
1159
           1
               Smith
                            male
                                        Boston
                                                   Teacher
                                                               0
1160
           2
               Jenny
                       17
                            female
                                        Dallas
                                                  Student
                                                             0
1161
         -기존 column 값을 이용
1162
1163
         -'Job'에 따라 'Salary' 여부 Column으로 수정
1164
1165
           df['Salary'] = np.where(df['Job'] != 'Student', 'yes', 'no')
1166
           df.head()
1167
1168
                                      Address Job
               Name Age Gender
                                                       Salary
1169
           0
               John
                       15
                              male
                                          Chicago Student no
                       25
1170
           1
               Smith
                            male
                                        Boston
                                                  Teacher
                                                           yes
1171
           2
               Jenny
                       17
                            female
                                        Dallas
                                               Student
1172
         -'Total' column 추가
1173
1174
           student list = [
1175
              {'Name':'John', 'Midterm':95, 'Final': 85},
1176
1177
              {'Name':'Smith', 'Midterm':85, 'Final': 80},
1178
              {'Name':'Jenny', 'Midterm':30, 'Final': 10},
1179
           1
1180
           df = pd.DataFrame(student list, columns= ['Name', 'Midterm', 'Final'])
1181
1182
1183
           df.head()
1184
1185
             Name
                     Midterm Final
                     95
1186
           0 John
                              85
           1 Smith
                     85
                            80
1187
1188
           2 Jenny
                     30
                            10
1189
1190
           df['Total'] = df['Midterm'] + df['Final']
1191
           df
1192
1193
             Name
                     Midterm Final
                                    Total
1194
                     95
           0 John
                              85
                                    180
                     85
1195
           1 Smith
                            80
                                  165
1196
                                  40
           2 Jenny
                     30
                            10
1197
         -'Average' column 추가하기
1198
```

```
1199
1200
           df['Average'] = df['Total'] / 2
1201
           df.head()
1202
1203
             Name
                     Midterm Final
                                       Total
                                              Average
1204
           0 John
                       95
                                 85
                                       180
                                               90.0
1205
           1 Smith
                     85
                               80
                                     165
                                             82.5
1206
           2 Jenny
                     30
                               10
                                     40
                                             20.0
1207
1208
         -'Grade' column 추가하기
1209
1210
           grade list = []
           for row in df['Average']:
1211
1212
              if row \leq 100 and row \geq 90:
1213
                 grade list.append('A')
1214
              elif row < 90 and row >= 80:
1215
                 grade list.append('B')
1216
              elif row < 80 and row >= 70:
1217
                grade_list.append('C')
1218
              elif row < 70 and row >= 60:
1219
                 grade_list.append('D')
1220
              else : grade_list.append('F')
1221
1222
           df['Grade'] = grade_list
1223
           df
1224
1225
             Name
                     Midterm Final
                                       Total
                                              Average Grade
1226
                       95
                                              90.0
           0 John
                               85
                                       180
                                                        Α
1227
           1 Smith
                     85
                             80
                                     165
                                             82.5
                                                      В
1228
           2 Jenny
                     30
                             10
                                     40
                                             20.0
                                                      F
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
       4)Syntax
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)
1242
         DataFrame.sort_values(by, axis=0, ascending=True, inplace=False,
           kind='quicksort', na_position='last')
1243
1244
1245
         -label: 정렬할 level을 지정.
1246
         -axis: int 타입 또는 축의 이름. (0 또는 'index') 와 (1 또는 'columns') 중 하나
         -ascending: True(기본값) 이면 오름차순, False 이면 내림차순으로 정렬.
1247
1248
         -inplace: bool 타입이며, False(기본값)이면 삭제된 결과 DataFrame을 리턴하며, True 이면 현재
         DataFrame에서 데이터를 삭제하고 None을 반환.
```

```
1249
         -kind : 정렬 알고리즘을 지정.
1250
           --정렬 방법은 "quicksort", "mergesort", "heapsort" 중 하나.
           --기본값은 "quicksort"이며 이 옵션은 단일 열 또는 단일 index를 정렬할 때만 적용.
1251
         -na position: NaN 값을 놓을 위치를 "first" 또는 "last"로 지정.
1252
1253
           --기본값은 NaN을 마지막에 두는 "last".
1254
         -sort remaining: 만일 True(기본값)이고 레벨(level)로 정렬하며 index가 멀티레벨일 경우 지정한 레벨
         로 정렬 할 후 다른 레벨을 이용해도 정렬.
1255
         -by: sort_values() 함수에서 by 인자는 정렬의 기준이 되는 열 또는 행의 이름을 지정.
1256
1257
           member_df = pd.read_csv("member_data.csv")
           member df.index = ["동", "서", "남", "북"]
1258
1259
           member df
1260
1261
               Name Age Email
                                                 Address
           동 홍길동 20 kildong@hong.com 서울시 강동구
1262

      서
      홍길서
      25
      kilseo@hong.com
      서울시 강서구

      남
      홍길남
      26
      south@hong.com
      서울시 강남구

      북
      홍길북
      27
      book@hong.com
      서울시 강북구

1263
1264
1265
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
1274
               Name Age Email
                                                 Address
                                               서울시 강남구
1275
                            south@hong.com
           남 홍길남 26
                            kildong@hong.com 서울시 강동구
1276
           동 홍길동
                      20
           북 홍길북 27
1277
                            book@hong.com 서울시 강북구
1278
           서 홍길서
                      25
                            kilseo@hong.com 서울시 강서구
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
           동 서울시 강동구 20 <u>kildong@hong.com</u> 홍길동
서 서울시 강서구 25 <u>kilseo@hong.com</u> 홍길
1287
1288
                                                         홍길서
1289
          남 서울시 강남구 26 <u>south@hong.com</u>
                                                         홍길남
               서울시 강북구 27 book@hong.com
1290
           북
                                                      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
                       book@honq.com 서울시 강북구
         북 홍길북
                   27
                       kildong@hong.com 서울시 강동구
1300
         동 홍길동
                   20
1301
                   25
                       kilseo@hong.com 서울시 강서구
         서 홍길서
         남 홍길남
                       south@hona.com
1302
                   26
                                       서울시 강남구
1303
       -여러 열을 이용해서 정렬 하고 싶을 때는 by 인자의 값을 여러 column 이름을 갖는 list 형식으로 지정하면 된
1304
       다.
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", "이
         메일", "주소"]]
1312
         member_df.columns.names = ["정보구분", "상세정보"]
         member_df.index = [["좌우","좌우","상하","상하"], ["동", "서", "남", "북"]]
1313
         member_df.index.names = ["위치구분", "상세위치"]
1314
1315
         member df
1316
            정보구분 기본정보
1317
                              추가정보
1318
            상세정보 이름 나이 이메일
                                                      주소
1319
            위치구분 상세위치
                                 kildong@hong.com 서울시 강동구
1320
         좌우
               동 홍길동 20
                                 kilseo@hong.com서울시 강서구south@hong.com서울시 강남구book@hong.com서울시 강북구
                      홍길서 25
1321
                 서
1322
                      홍길남 26
         상하
                 남
1323
                 북
                      홍길북 27
1324
1325
         member_df.sort_index(level=["위치구분"])
1326
1327
               정보구분
                     기본정보 추가정보
1328
                        이름 나이 이메일
                                               주소
         상세정보
1329
         위치구분 상세위치
1330
         상하 남 홍길남 26 south@hong.com 서울시 강남구
                      홍길북 27 book@hong.com 서울시 강북구
1331
1332
                      홍길동 20 kildong@hong.com서울시 강동구
         좌우
               동
1333
                      홍길서 25 kilseo@hong.com 서울시 강서구
1334
         member_df.sort_index(level=["상세위치"])
1335
1336
              정보구분 기본정보 추가정보
1337
1338
              상세정보 이름 나이 이메일
                                               주소
1339
         위치구분 상세위치
1340
         상하 남
                    홍길남 26 <u>south@hong.com</u>
                                            서울시 강남구
         좌우 동
                   홍길동 20 <u>kildong@hong.com</u> 서울시 강동구
1341
1342
                   홍길북 27 <u>book@hong.com</u> 서울시 강북구
         상하 북
         좌우 서 홍길서 25 kilseo@hong.com
1343
                                            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 구
         분하여 그룹화 한 후 기초 통계 함수 등을 적용 할 수 있도록 한다.
1350
         -Svntax
1351
1352
           DataFrame.groupby(by=None, axis=0, level=None, as_index=True,
1353
                                 sort=True,group_keys=True, squeeze=False,
1354
                                 observed=False, **kwargs)
1355
1356
           import statsmodels.api as sm
           iris = sm.datasets.get rdataset("iris", package="datasets")
1357
1358
           iris df = iris.data
1359
           iris df.head()
1360
1361
              Sepal.Length
                             Sepal.Width Petal.Length Petal.Width
                                                                   Species
1362
             5.1
                             3.5
                                            1.4
                                                          0.2
           0
                                                                       setosa
1363
           1 4.9
                             3.0
                                            1.4
                                                          0.2
                                                                       setosa
1364
           2 4.7
                             3.2
                                            1.3
                                                          0.2
                                                                       setosa
1365
           3 4.6
                             3.1
                                           1.5
                                                          0.2
                                                                       setosa
1366
          4 5.0
                             3.6
                                            1.4
                                                          0.2
                                                                       setosa
1367
1368
1369
       2)단일 열로 그룹화
         -groupby() 함수를 이용하여 그룹화 할 열을 지정.
1370
1371
         -그룹이 지정되면 그 그룹에 기초통계 분석 함수를 사용하면 된다.
1372
1373
           iris grouped = iris df.groupby(iris df.Species)
1374
           iris grouped
1375
1376
           <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001F477FF7808>
1377
1378
           iris_grouped.mean()
                                  #평균
1379
1380
                     Sepal.Length Sepal.Width Petal.Length Petal.Width
1381
           Species
1382
                      5.006
           setosa
                                                   1.462
                                                                  0.246
                                    3.428
1383
           versicolor 5.936
                                   2.770
                                                   4.260
                                                                  1.326
1384
           virginica 6.588
                                  2.974
                                                 5.552
                                                                2.026
1385
1386
1387
       3)다중 열로 그룹화
1388
         -그룹을 두 가지 이상으로 지정하고 싶을 때는 list를 이용해 그룹을 지정하면 된다.
         -열 이름에 점(.)이 포함되었다면 df["열이름"] 형식으로 열을 지정해야 한다.
1389
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"]])
          iris_grouped2
1394
1395
1396
           <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001F479369C08>
1397
```

```
1398
           iris grouped2.mean().head()
1399
1400
                                   Sepal.Width Petal.Length Petal.Width
                    Sepal.Length
1401
           Species
1402
           setosa
                    4.3
                                   3.000000
                                                  1.100000
                                                                 0.100
                    4.4
                                                                 0.200
1403
                                   3.033333
                                                  1.333333
1404
                    4.5
                                   2.300000
                                                  1.300000
                                                                 0.300
1405
                    4.6
                                   3.325000
                                                  1.325000
                                                                 0.225
1406
                    4.7
                                   3.200000
                                                  1.450000
                                                                 0.200
1407
1408
         -그룹된 객체를 이용해 요약정보를 볼 수도 있다.
1409
1410
           iris_grouped.describe()
1411
         -요약정보에서 생략된 부분(...)의 내용을 보려면 디스플레이 옵션을 설정한다.
1412
1413
1414
           pd.options.display.max columns = 999
1415
1416
       4)그룹간 반복 처리
1417
         -그룹화 된 데이터에서 그룹의 타입과 그룹 객체를 반복문을 이용해 처리할 수 있다.
1418
1419
           for type, group in iris_grouped:
1420
             print(type, '\n', group.head())
1421
1422
1423
       5) DataFrame group indexing
1424
         -DataFrame group에서 indexing을 위해서는 take() 함수를 이용.
1425
         -Syntax
1426
           DataFrameGroupBy.take(indices, axis, is copy)
1427
1428
         -indices : 가져올 index를 list 형식으로 지정.
1429
         -axis : 기본값 0이면 행 index를 이용해 가져오고, 1이면 열 index를 이용해 가져온다.
1430
         -is_copy: 기본값 True이면 객체의 복사본이 반환.
1431
         -다음 코드는 iris 데이터를 불러와 종(Species) 별로 grouping한 결과에서 부분집합을 가져온다.
1432
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.Width Petal.Length Petal.Width
                             Sepal.Length
1448
           Species
```

```
5.1
1449
                              0
                                                        3.5
                                                                                        0.2
              setosa
                                                                        1.4
1450
                              1
                                     4.9
                                                        3.0
                                                                        1.4
                                                                                        0.2
1451
                              2
                                                        3.2
                                                                        1.3
                                                                                        0.2
                                     4.7
1452
                              50
                                                        3.2
                                                                        4.7
                                                                                        1.4
              versicolor
                                     7.0
1453
                              51
                                     6.4
                                                        3.2
                                                                        4.5
                                                                                        1.5
1454
                              52
                                     6.9
                                                                        4.9
                                                                                        1.5
                                                        3.1
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"},
1462
                   {'Name': 'Nate', 'Major': "Computer Science", 'Gender': "male"},
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': 'Edward', 'Major': "Computer Science", 'Gender': "male"},
1468
1469
                   {'Name': 'Zara', 'Major': "Psychology", 'Gender': "female"},
1470
                   {'Name': 'Wendy', 'Major': "Économics", 'Gender': "female"},
1471
                   {'Name': 'Sera', 'Major': "Psychology", 'Gender': "female"}
1472
1473
             1
1474
1475
           df = pd.DataFrame(student list, columns = ['Name', 'Major', 'Gender'])
1476
           df
1477
1478
                Name
                         Major
                                              Gender
1479
           0
                John
                           Computer Science
                                                   male
1480
           1
                Nate
                            Computer Science
                                                   male
1481
           2
                Abraham Physics
                                                male
1482
           3
                         Psychology
                Brian
                                                male
1483
           4
                Janny
                         Economics
                                              female
           5
1484
                Yuna
                         Economics
                                              female
1485
           6
                Jeniffer Computer Science
                                                female
                Edward Computer Science
1486
           7
                                                 male
           8
                            Psychology
                                                   female
1487
                Zara
           9
                Wendy
                            Economics
                                                female
1488
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
              'Economics': Int64Index([4, 5, 9], dtype='int64'),
1497
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
           Computer Science: 4
1507
1508
              Name Major
                                         Gender
1509
           0
                John
                        Computer Science
                                           male
1510
           1
                Nate
                        Computer Science male
1511
              Jeniffer
                        Computer Science
                                           female
           6
1512
           7
              Edward
                        Computer Science
                                           male
1513
1514
           Economics: 3
1515
              Name
                        Major
                                   Gender
                        Economics female
1516
           4
              Janny
1517
           5
              Yuna
                      Economics female
1518
              Wendy Economics female
1519
1520
           Physics: 1
1521
               Name
                        Major
                                   Gender
1522
              Abraham Physics
                                   male
1523
1524
           Psychology: 3
1525
              Name
                                     Gender
                        Major
1526
           3
              Brian
                        Psychology
                                     male
                      Psychology female
1527
           8
              Zara
1528
           10
                        Psychology female
                Sera
1529
1530
         -전공별 명수
1531
1532
           df_major_cnt = pd.DataFrame({'Count':groupby_major.size()})
1533
           df major cnt
1534
1535
                            Count
1536
           Major
1537
           Computer Science 4
1538
           Economics
                            3
1539
           Physics
                             3
1540
           Psychology
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
1548
              Computer Science
           0
                                 4
1549
              Economics
           1
                               3
1550
           2
              Physics
                               1
```

```
1551
              Psychology
                                 3
1552
1553
         -성별로 group by
1554
1555
           groupby_gender = df.groupby('Gender')
1556
1557
           for name, group in groupby_gender:
1558
             print(name + ": " + str(len(group)))
1559
            print(group)
1560
            print()
1561
1562
           female: 6
1563
              Name
                      Major
                                        Gender
1564
                                          female
               Janny
                         Economics
1565
           5
               Yuna
                                          female
                         Economics
1566
           6
              Jeniffer
                        Computer Science
                                            female
1567
                                          female
           8
               Zara
                         Psychology
1568
           9
               Wendy
                        Economics
                                          female
1569
           10
                Sera
                         Psychology
                                          female
1570
1571
           male: 5
1572
              Name
                        Major
                                          Gender
1573
           0
                        Computer Science
                                          male
                John
1574
           1
                Nate
                        Computer Science
                                          male
1575
           2
              Abraham Physics
                                        male
                        Psychology
1576
           3
              Brian
                                          male
                        Computer Science
1577
           7
              Edward
                                          male
1578
1579
1580
1581 12. DataFrame에 함수 적용하기
1582
       -DataFrame의 데이터에서 합계나 평균 등 일반적인 통계는 DataFrame의 함수들을 사용하면 되지만, 판다스에
       서 제공하지 않는 기능을 커스텀 함수(custom function)로 구현해서 DataFrame에 적용하려면 apply(),
       applymap(), map() 등의 함수를 사용한다.
       -적용할 함수의 이름은 다음과 같다.
1583
1584
         method
                  통용대상
                                      반환값
1585
         map
                    Series(값별)
                                        Series
1586
                  DataFrame(열 또는 행별) Series
         apply
1587
                    DataFrame(값별)
         applymap
                                        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
         -raw
1598
           --False(기본값)일 경우 각 행이나 열을 Series로 함수에 전달.
1599
           --True이면 전달 된 함수는 대신 ndarray 객체를 받는다.
```

1649 1650

```
-result type: 'expand', 'reduce', 'broadcast', None 중 하나를 사용.
1600
1601
          --기본값은 None.
1602
          --이것은 axis=1(columns)인 경우에만 작동.
          --'expand' : 목록과 같은 결과가 열로 바뀐다.
1603
1604
          --'reduce': 목록과 같은 결과를 확장하지 않고 가능한 경우 Series를 반환한다.
            ---이것은 '확장'의 반대.
1605
1606
          --'broadcast' : 결과가 DataFrame의 원래 모양으로 브로드 캐스팅되고 원본 인덱스와 열은 유지된다.
          --None: 적용 함수의 반환 값에 따라 다르다.
1607
1608
            ---목록 같은 결과는 일련의 결과로 반환된다.
1609
            --- 그러나 apply 함수가 시리즈를 리턴하면 이들은 열로 확장된다.
1610
         -arqs: 배열/시리즈 외에도 func에 전달할 위치 인수를 tuple형식으로 지정.
         -**kwds: func에 전달할 추가 키워드 인수를 지정.
1611
1612
1613
          import statsmodels.api as sm
          iris = sm.datasets.get rdataset("iris", package="datasets")
1614
1615
          iris df = iris.data
          iris df.head()
1616
1617
1618
              Sepal.Length
                            Sepal.Width Petal.Length Petal.Width
                                                                  Species
1619
          0 5.1
                            3.5
                                          1.4
                                                         0.2
                                                                      setosa
1620
          1 4.9
                            3.0
                                           1.4
                                                         0.2
                                                                      setosa
          2 4.7
                                           1.3
1621
                            3.2
                                                         0.2
                                                                     setosa
          3 4.6
1622
                            3.1
                                           1.5
                                                         0.2
                                                                     setosa
1623
          4
              5.0
                            3.6
                                           1.4
                                                         0.2
                                                                     setosa
1624
1625
        -iris 데이터에서 종(Species) 정보를 제외한 나머지 열 정보를 조회한다.
         -이것은 apply() 함수를 사용하기 전/후를 비교하기 위해서이다.
1626
1627
1628
          iris_df.iloc[:,:-1].head()
1629
1630
              Sepal.Length
                            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
          2 4.7
1633
                            3.2
                                         1.3
                                                       0.2
1634
          3 4.6
                            3.1
                                         1.5
                                                       0.2
1635
          4
              5.0
                            3.6
                                                       0.2
                                         1.4
1636
1637
        -다음 코드는 iris 데이터에 np.round 함수를 적용해서 데이터를 반올림 합니다.
1638
1639
          import numpy as np
          iris_df.iloc[:,:-1].apply(np.round).head()
1640
1641
1642
              Sepal.Length
                            Sepal.Width Petal.Length Petal.Width
1643
          0 5.0
                            4.0
                                        1.0
                                                     0.0
          1 5.0
                            3.0
                                         1.0
                                                     0.0
1644
          2 5.0
1645
                            3.0
                                        1.0
                                                     0.0
1646
          3
             5.0
                            3.0
                                         2.0
                                                     0.0
          4
1647
              5.0
                            4.0
                                                     0.0
                                         1.0
1648
```

-apply() 함수에 사용하는 함수가 요소별로 동작하는 함수라면 위의 예에서처럼 각 요소에 함수를 적용한다. -그러나 만일 함수가 요소별로 동작하지 않는 함수라면 axis 매개변수의 값에 따라 적용된 결과는 달라질 수 있

```
다.
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
1665
           Sepal.Length
                              876.5
1666
           Sepal.Width
                            458.6
1667
           Petal.Length
                            563.7
1668
           Petal.Width
                              179.9
1669
           dtype: float64
1670
1671
         -다음 코드는 행 별 합계를 출력한다.
1672
1673
           iris_df.iloc[:,:-1].apply(np.sum, axis=1) # 행 별 합계 출력
1674
1675
           0
                      10.2
                      9.5
1676
           1
           2
                      9.4
1677
1678
           3
                      9.4
1679
           4
                      10.2
1680
           ... 생략 ...
1681
           145
                      17.2
1682
           146
                      15.7
1683
           147
                      16.7
1684
           148
                      17.3
                      15.8
1685
           149
           Length: 150, dtype: float64
1686
1687
1688
         -다음 코드는 iris 데이터의 열 별 평균을 계산하고 각 데이터와 평균과의 차이를 apply() 함수를 이용해 계산한
         다.
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
           Sepal.Width
1695
                            3.057333
           Petal.Length
                            3.758000
1696
1697
           Petal.Width
                              1.199333
           dtype: float64
1698
1699
```

```
1700
           iris_df2.apply(lambda x : x-iris_avg,axis=1).head()
1701
           -----
1702
           0
               -0.743333 0.442667
                                      -2.358
                                                -0.999333
1703
           1
               -0.943333
                          -0.057333-2.358
                                              -0.999333
1704
           2
                          0.142667
                                                -0.999333
               -1.143333
                                      -2.458
           3
1705
               -1.243333
                          0.042667
                                      -2.258
                                                -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},
1713
            {'Name':'Smith', 'Midterm':85, 'Final': 80},
1714
            {'Name':'Jenny', 'Midterm':30, 'Final': 10},
1715
         1
1716
1717
         df = pd.DataFrame(student_list, columns= ['Name', 'Midterm', 'Final'])
1718
1719
         df.head()
1720
1721
           Name
                   Midterm Final
1722
         0 John
                   95
                            85
1723
         1 Smith
                   85
                           80
1724
         2 Jenny
                   30
                           10
1725
         df['Total'] = df['Midterm'] + df['Final']
1726
1727
         df
1728
1729
           Name
                   Midterm Final
                                  Total
1730
         0 John
                     95
                           85
                                  180
1731
         1 Smith
                   85
                           80
                                165
1732
         2 Jenny
                   30
                           10
                                40
1733
1734
         -'Average' column 추가하기
1735
1736
           df['Average'] = df['Total'] / 2
1737
           df.head()
1738
1739
             Name
                     Midterm Final
                                      Total
                                              Average
1740
           0 John
                       95
                                85
                                      180
                                              90.0
1741
           1 Smith
                     85
                              80
                                    165
                                            82.5
1742
           2 Jenny
                     30
                              10
                                    40
                                            20.0
1743
         -'Grade' column 추가하기
1744
1745
1746
           grade_list = []
           for row in df['Average']:
1747
              if row \leq 100 and row \geq 90:
1748
1749
                grade_list.append('A')
1750
              elif row < 90 and row >= 80:
```

```
1751
                 grade list.append('B')
1752
              elif row < 80 and row >= 70:
1753
                 grade list.append('C')
1754
              elif row < 70 and row >= 60:
1755
                 grade list.append('D')
              else : grade_list.append('F')
1756
1757
1758
            df['Grade'] = grade_list
1759
            df
1760
1761
              Name Midterm Final
                                        Total
                                               Average Grade
                     95
1762
            0 John
                           85
                                        180
                                                90.0
                                                          Α
1763
            1 Smith
                      85
                              80
                                      165
                                              82.5
                                                        В
                                              20.0
1764
            2 Jenny
                      30
                              10
                                      40
                                                        F
1765
1766
          -'Result' column 추가하기
1767
1768
            def pass_or_fail(row):
1769
              if row != 'F':
1770
                 return 'Pass'
1771
              else:
1772
                 return 'Fail'
1773
1774
            df['Result'] = df.Grade.apply(pass_or_fail)
1775
            df
1776
1777
              Name
                     Midterm Final
                                       Total
                                                Average Grade
                                                                Result
                       95
                               85
                                       180
                                                90.0
1778
            0 John
                                                                  Pass
                                                          Α
1779
            1 Smith
                      85
                              80
                                      165
                                              82.5
                                                                Pass
                                                        В
1780
            2 Jenny
                     30
                              10
                                     40
                                              20.0
                                                        F
                                                                Fail
1781
1782
          -Column 추가하면서 각각의 값 조작하기
1783
1784
            date list = [
1785
              { 'yyyy-mm-dd' : '2019-01-05'},
1786
              { 'yyyy-mm-dd' : '2019-01-10'}
            1
1787
1788
1789
            df = pd.DataFrame(date_list, columns = ['yyyy-mm-dd'])
1790
            df
1791
1792
              yyyy-mm-dd
1793
            0 2019-01-05
1794
            1 2019-01-10
1795
           def extract_year(row):
1796
1797
              return row.split('-')[0]
1798
1799
            df['Year'] = df['yyyy-mm-dd'].apply(extract_year)
           df
1800
1801
```

```
1802
               yyyy-mm-dd
                                Year
1803
           0
               2019-01-05
                              2019
1804
               2019-01-10
           1
                              2019
1805
1806
1807
         -passing keyword parameter to apply function
1808
1809
           date_list = [{'Jumin': '2000-06-27'},
1810
                {'Jumin': '2002-09-24'},
1811
                {'Jumin': '2005-12-20'}]
           df = pd.DataFrame(date list, columns = ['Jumin'])
1812
1813
                    _____
1814
1815
             Jumin
           0 2000-06-27
1816
1817
           1 2002-09-24
           2 2005-12-20
1818
1819
1820
           def extract_year(row):
1821
             return row.split('-')[0]
1822
1823
           df['Born_Year'] = df['Jumin'].apply(extract_year)
1824
1825
1826
                        Born_Year
             Jumin
           0 2000-06-27
                         2000
1827
           1 2002-09-24
1828
                          2002
           2 2005-12-20
1829
                          2005
1830
1831
           def calc_age(year, current_year):
1832
             return current_year - int(year)
1833
1834
           df['Age'] = df['Born_Year'].apply(calc_age, current_year=2019)
1835
1836
                     Born_Year Age
1837
             Jumin
1838
           0 2000-06-27 2000
                                      19
           1 2002-09-24
1839
                          2002
                                      17
           2 2005-12-20
                          2005
                                      14
1840
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
           df
1847
1848
                     Born_Year Age Introduce
             Jumin
           0 2000-06-27 2000
                                           I am 19 years old.
1849
                                      19
                                           I am 17 years old.
1850
           1 2002-09-24
                                      17
                          2002
           2 2005-12-20
                                      14
                                           I am 14 years old.
1851
                          2005
1852
```

1901

import numpy as np

```
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
1857
        wise) 방식으로 적용하는 DataFrame의 각 요소마다 사용자 정의 함수를 수행한다.
        -이때 사용자 정의 함수는 반드시 단일 값(Single value)을 반환해야 한다.
1858
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
          iris df.iloc[:,:-1].applymap(np.sum).head()
1865
1866
1867
              Sepal.Length Sepal.Width Petal.Length Petal.Width
1868
              5.1
                          3.5
                                      1.4
          0
                                                   0.2
1869
          1
            4.9
                          3.0
                                      1.4
                                                   0.2
1870
          2
             4.7
                          3.2
                                       1.3
                                                   0.2
1871
          3
                                      1.5
                                                   0.2
             4.6
                          3.1
1872
          4
              5.0
                          3.6
                                      1.4
                                                   0.2
1873
1874
        -이 결과는 각 요소에 대한 합계이므로 원래의 데이터와 같은 값이다.
        -다음 코드는 lambda식을 이용해서 각 요소의 값을 제곱한다.
1875
1876
1877
          iris df.iloc[:,:-1].applymap(lambda x : x**2).head()
          -----
1878
1879
              Sepal.Length Sepal.Width Petal.Length Petal.Width
1880
              26.01
                          12.25
                                      1.96
                                                   0.04
          0
1881
          1
              24.01
                          9.00
                                       1.96
                                                   0.04
          2
              22.09
                          10.24
                                                   0.04
1882
                                      1.69
1883
          3 21.16
                          9.61
                                      2.25
                                                   0.04
1884
              25.00
                          12.96
                                      1.96
                                                   0.04
1885
1886
        -한번에 DataFrame에 있는 모든 요소들을 수정하고자 할 때 사용하자.
1887
1888
          data_list = [
            {'x': 5.5, 'y': -5.6},
1889
            {'x': -5.2, 'y': 5.5},
1890
            {'x': -1.6, 'y': -4.5}
1891
1892
          1
1893
          df = pd.DataFrame(data_list)
1894
          df
1895
1896
              Χ
                   У
          0 5.5 -5.6
1897
          1 -5.2 5.5
1898
          2
              -1.6 -4.5
1899
1900
```

```
1902
1903
          df = df.applymap(np.around) #반올림함수 사용
1904
          df
1905
1906
              Χ
1907
          0 6.0 -6.0
1908
          1 -5.0 6.0
          2 -2.0 -4.0
1909
1910
1911
1912
       4)map()
1913
        -map() 함수는 Series 데이터의 요소 각각에 대해 함수 또는 dict 또는 다른 Series를 적용한다.
1914
         -map() 함수는 DataFrame에는 사용할 수 없다.
1915
        -다음 구문에서 보는 것처럼 Series 타입에서만 사용할 수 있다.
1916
        -Svntax
1917
          Series.map(arg, na_action=None)
        -arg: 시리즈의 값 하나 하나에 적용할 함수, dict 또는 Series.
1918
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.
1922
          --None(기본값)이면 NA는 그대로 함수 또는 딕셔너리에 NA로 전달된다.
1923
         -map() 함수는 Series의 값 하나하나에 접근하면서 해당 함수를 수행한다.
1924
1925
          import pandas as pd
1926
          x = pd.Series(['Hello', 'Python', 'World'],index=[1,2,3])
1927
1928
1929
                   Hello
          1
1930
          2
                 Pvthon
1931
          3
                   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
          --Series에 함수를 적용하면 함수 인자에 시리즈의 요소 하나가 전달된다.
1940
1941
1942
            x.map(my_func)
1943
            1
                     (Hello, 5)
1944
1945
            2
                   (Python, 6)
                     (World, 5)
1946
            3
1947
            dtype: object
1948
1949
        -dict 사용
1950
          --Series에 dict를 적용하면 dict의 key별로 Series의 값에 적용된다.
1951
1952
            z = {"Hello": 'A', "Python": 'B', "World": 'C'}
```

```
1953
             x.map(z)
1954
             1
                     Α
1955
             2
                     В
1956
             3
                     C
1957
             dtype: object
1958
1959
         -Series 사용
1960
           -Series에 Series를 적용하면 원본 Series의 값에 적용할 Series의 index별로 적용된다.
1961
1962
             y = pd.Series(['foo', 'bar', 'baz'],index=['Hello','Python', 'World'])
1963
             x.map(y)
1964
1965
             2
1966
                     bar
             3
1967
                     baz
1968
             dtype: object
1969
1970
1971
       5)다른 예제로 map()을 연습해 보자.
1972
1973
         date_list = [{'Date': '2000-06-27'},
1974
                {'Date': '2002-09-24'},
                {'Date': '2005-12-20'}]
1975
         df = pd.DataFrame(date_list, columns = ['Date'])
1976
1977
         df
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
         df
1993
           Date
1994
                         Year
         0 2000-06-27
1995
                         2000
1996
         1 2002-09-24 2002
         2 2005-12-20
1997
                         2005
1998
1999
         -map() 응용하기
2000
2001
           data list = [
             {'Name':'John', 'Age':15, 'Gender':'male', 'Job':'Student'},
2002
2003
             {'Name':'Smith', 'Age':25, 'Gender':'male', 'Job':'Teacher'},
```

```
2004
            {'Name':'Jenny', 'Age':27, 'Gender':'female', 'Job':'Developer'},
2005
           1
2006
           df = pd.DataFrame(data list, columns = ['Name', 'Age', 'Gender', 'Job'])
2007
2008
2009
2010
            Name Age Gender Job
                    15
2011
           0 John
                           male
                                    Student
2012
           1 Smith
                    25
                         male
                                  Teacher
2013
           2 Jenny 27
                         female
                                  Developer
2014
          df.Job = df.Job.map({'Student':1, 'Teacher':2, 'Developer':3})
2015
2016
2017
2018
            Name Age Gender Job
2019
           0 John
                    15
                           male 1
2020
                   25
           1 Smith
                         male
                                  2
2021
           2 Jenny 27
                         female
                                  3
2022
2023
2024
       6) na_action
2025
         -Series가 NaN 값을 포함할 경우 na_action 인자의 값에 따라 결과는 달라진다.
2026
2027
          s = pd.Series([1, 2,3,None])
2028
          S
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
          s.map(lambda x:(x, x**2))
2038
2039
2040
                  (1.0, 1.0)
          0
2041
                  (2.0, 4.0)
           1
           2
                  (3.0, 9.0)
2042
2043
           3
                  (nan, nan)
2044
           dtype: object
2045
2046
         -na_action='ignore'일 경우 적용한 결과가 NA가 된다.
2047
          s.map(lambda x:(x, x^{**}2), na_action='ignore')
2048
2049
2050
                  (1.0, 1.0)
          0
2051
                  (2.0, 4.0)
          1
                  (3.0, 9.0)
2052
           2
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,**kwargs)
2065
         -value : scalar, dict, Series, 또는 DataFrame을 지정.
2066
           --결측치를 채우는데 사용할 값.
2067
           --이 값은 list로 지정할 수 없다.
         -method : {'backfill', 'bfill', 'pad', 'ffill', None}
2068
2069
           --기본값 None
2070
           --재 색인화된 채우기에 사용될 방법을 지정한다.
2071
           --'ffill': 이전의 유효한 관측값을 이용해 채운다.
2072
           --bfill : 이후의 유효한 관측값을 사용해서 채운다.
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
2076
           --method가 지정되고 있는 경우, 이것은 순방향/역방향으로 채울 수 있는 최대 연속 NaN 개수이다.
2077
           --method가 지정되지 않은 경우, 이것은 NaN이 채워지는 축 전체의 최대 항목 수이다.
2078
           --없으면 0보다 커야한다.
2079
         -downcast : dict, 기본값 None
           --항목들을 downcast='infer'일 경우 적당한 동등한 타입으로 형변환(다운캐스트) 된다.
2080
2081
           --예를 들면 변수가 float64일 경우 자동으로 int64로 변경된다.
2082
           --원래의 값들이 정수이더라도 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],
2086
                             [np.nan, np.nan, np.nan, 5],
2087
                             [np.nan, 3,np.nan, 4]],
2088
                              columns=list('ABCD'))
           df
2089
2090
2091
                  Α
                         В
                                C
                                      D
2092
           0
                  NaN 2.0
                            NaN 0
2093
           1
                  3.0
                         4.0
                              NaN 1
2094
           2
                  NaN
                       NaNNaN 5
           3
2095
                  NaN
                       3.0
                            NaN 4
2096
2097
         -다음구문은 모든 NaN을 0으로 채운다.
2098
2099
           df.fillna(0)
2100
2101
                           C
                Α
                     В
                                D
                    2.0
2102
           0
                0.0
                           0.0
                                0
2103
                3.0
                     4.0
                           0.0
                                1
           1
           2
                                5
2104
                0.0
                     0.0
                           0.0
2105
           3
                0.0
                     3.0
                           0.0
                                4
```

```
2106
2107
         -널(null)이 아닌 이전 또는 이후의 값을 이용해 채울 수 있다.
2108
         -ffill 은 이전의 널이 아닌 값을 이용해서 채운다.
2109
2110
           df.fillna(method='ffill')
2111
2112
                           C
                Α
                     В
                                D
2113
           0
                NaN2.0
                         NaN0
2114
           1
                3.0 4.0
                           NaN1
2115
           2
                3.0
                    4.0
                           NaN5
2116
           3
                3.0
                     3.0
                           NaN4
2117
2118
         -모든 열 'A', 'B', 'C', 'D'의 NaN 값을 각각 0, 1, 2, 3 값으로 채운다.
2119
2120
           values= {'A': 0,'B': 1,'C': 2,'D': 3}
2121
          df.fillna(value=values)
2122
2123
                           C
                Α
                     В
                                D
2124
                0.0 2.0
                           2.0
                                0
           0
2125
           1
                3.0 4.0
                           2.0
                                1
           2
2126
                    1.0
                           2.0
                                5
                0.0
           3
2127
                0.0
                     3.0
                           2.0
2128
2129
         -limit는 채울 NaN의 개수를 지정한다.
2130
         -limit=1로 하면 다음은 처음의 NaN 값 하나만 채운다.
2131
2132
           df.fillna(value=values, limit=1)
2133
2134
                           С
                Α
                     В
                                D
2135
           0
                0.0
                    2.0
                           2.0
                                0
2136
           1
                3.0 4.0
                           NaN1
           2
                NaN1.0
                         NaN5
2137
2138
           3
                NaN3.0
                         NaN4
2139
2140
2141
       2)replace()
2142
         -replace() 함수는 to replace 속성에 주어진 값을 value 값으로 바꾼다.
         -DataFrame의 값은 다른 값으로 동적으로 대체e된다.
2143
         -이것은 .loc 또는 .iloc를 사용하여 업데이트하는 것과는 다르다.
2144
2145
         -일부 값으로 업데이트 할 위치를 지정해야 한다.
2146
         -Syntax
2147
           DataFrame.replace(to_replace=None, value=None,
2148
                               inplace=False, limit=None,
2149
                                regex=False, method='pad')
         -to_replace: str, regex, list, dict, Series, int, float, 또는 None, 대체 될 값을 찾는 방법을 숫자
2150
         (numeric), 문자(str) 또는 정규식(regex)으로 지정한다.
2151
           --숫자, 문자 또는 정규식의 리스트로 지정하면 to_replace와 value가 모두 목록인 경우
            a. 길이가 같아야 한다.
2152
2153
            b. regex=True이면 두 list에 있는 모든 문자열은 정규 표현식으로 해석된다.
2154
           --dict를 사용하여 다른 기존 값에 대해 다른 대체 값을 지정할 수 있다.
2155
           --예를 들어 {'a':'b', 'y':'z'}는 'a'값을 'b'로, 'y'를 'z'로 바꾼다.
```

```
2156
          --이 방법으로 dict를 사용하려면 value 매개변수는 None 이어야 한다.
2157
          --DataFrame의 경우 dict는 다른 값이 다른 열에서 대체되어야 한다고 지정할 수 있다.
2158
          --예를 들어 {'a':1, 'b':'z'}는 'a'열의 값 1과 'b'열의 'z'값을 찾고 value에 지정된 값으로 대체한다.
2159
          --이 경우 value 매개변수는 None이 아니어야 한다.
2160
          --{'a':{'b':np.nan}}과 같은 중첩 dict의 경우 'a'열에서 'b'값을 찾아 NaN으로 바꾼다.
          --이 방법으로 중첩된 dict를 사용하려면 value 매개변수가 None 이어야 한다.
2161
2162
          --정규식도 중첩 할 수 있다.
2163
          --그러나 열 이름(중첩된 dict의 최상위 dict key)은 정규식이 될 수 없다.
2164
          --즉, regex 인수는 문자열, 컴파일된 일반 표현식 또는 list, dict, ndarray 또는 Series와 같은 요소여야
          하다.
2165
          --value도 None이면 중첩된 dict이나 Series여야 한다.
         -inplace: boolean, 기본값 False, 만일 True 이면 현재 객체를 수정.
2166
2167
         -limit: int, 기본값 None, method가 지정되고 있는 경우, 이것은 순방향/역방향으로 채울 수 있는 최대 연
         속 NaN 개수다.
2168
        -regex : bool 또는 to replace와 같은 타입.
2169
          --기본값 False: to replace 또는 value를 정규식으로 해석할지 여부
          --이 값이 True이면 to replace는 문자열이어야 한다.
2170
2171
          --또는 정규 표현식 또는 list, dict 또는 정규 표현식의 배열이 될 수 있다.
2172
          --이 경우 to replace는 None이어야 한다.
2173
         -method: {'pad', 'ffill', 'bfill', None}
          --to_replace가 scala, list 또는 tuple이고 값이 None 인 경우 대체 할 때 사용할 방법이다.
2174
2175
          s = pd.Series([0, 1,2,3,4])
2176
2177
          s.replace(0, 5)
2178
2179
                  5
          0
2180
          1
                 1
          2
                  2
2181
2182
          3
                  3
2183
          4
                 4
2184
          dtype: int64
2185
2186
         -다음 구문은 DataFrame 객체에서 O값을 5로 바꾼다.
2187
          df = pd.DataFrame(\{'A': [0, 1,2,3,4],
2188
2189
                              'B': [5, 6,7,8,9],
                                'C': ['a', 'b', 'c', 'd', 'e']})
2190
2191
          df.replace(0, 5)
2192
2193
              A B C
             5
                 5 a
2194
          0
2195
          1
             1
                 6 b
             2
2196
          2
                 7
                     С
             3
2197
          3
                 8
                     d
              4
          4
2198
                 9
2199
2200
         -다음 구문은 to replace를 list로 지정한 예이다.
2201
         -DataFrame의 모든 0, 1, 2, 3을 4로 바꾼다.
2202
          df.replace([0, 1, 2,3], 4)
2203
2204
```

```
2205
                    C
                В
             Α
2206
             4
                5 a
          0
2207
             4
                6 b
          1
             4
                7 c
          2
2208
2209
          3
             4
                8 d
          4
             4
                 9
2210
                    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
             Α
                в с
             4
                5 a
2218
          0
            3
2219
                6 b
          1
2220
          2
            2
                7 c
          3
            1
                8 d
2221
2222
             4 9
                    е
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
          2
                 3
2230
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
               Α
                    В
                    5
2241
          0
               10
                          а
2242
               100 6
          1
                          b
2243
          2
                    7
               2
                          С
2244
          3
               3
                    8
                          d
                    9
2245
2246
2247
        -다음 구문은 A열의 O값과 B열의 5값을 100으로 바꾼다.
2248
2249
          df.replace({'A': 0,'B': 5}, 100)
2250
2251
                          C
             Α
                    В
2252
          0 100
                    100
                         а
2253
             1
                    6
                          b
2254
            2
                    7
                          С
2255
             3
                    8
                          d
```

```
2256
              4
           4
                      9
                           е
2257
2258
         -다음 구문은 A열의 0은 100으로 바꾸고 4는 400으로 바꾼다.
2259
2260
           df.replace({'A': {0: 100, 4: 400}})
2261
2262
              Α
                    В
                      С
                    5
2263
             100
           0
                       а
2264
           1
              1
                    6
                       b
2265
           2
              2
                    7
                       С
             3
2266
           3
                    8
                       d
           4
             400 9
2267
2268
2269
         -다음 구문은 to_replace에 정규표현식을 사용한 예이다.
2270
2271
           df = pd.DataFrame({'A': ['bat', 'foo', 'bait'],
                             'B': ['abc', 'bar', 'xyz']})
2272
2273
2274
         -다음 구문은 ba로 시작하고 마지막 문자가 임의의 문자인 문자열을 'new'로 바꾼다.
2275
2276
           df.replace(to_replace=r'^ba.$', value='new', regex=True)
2277
2278
                Α
                       В
2279
           0
                new abc
2280
           1
                foo
                       new
2281
           2
                bait
                       XYZ
2282
         -다음 구문은 A열에서 ba로 시작하고 마지막 문자가 임의의 문자인 문자열을 'new'로 바꾼다.
2283
2284
2285
           df.replace({'A': r'^ba.$'}, {'A': 'new'}, regex=True)
2286
2287
                Α
                       В
2288
           0
                new abc
2289
           1
                foo
                       bar
2290
           2
                bait
                       xyz
2291
2292
         -다음 구문은 regex 속성에 정규표현식을 지정한 예이다.
2293
2294
           df.replace(regex=r'^ba.$', value='new')
2295
2296
                Α
                        В
2297
           0
                new abc
2298
           1
                foo
                       new
2299
           2
                bait
                       XYZ
2300
2301
         -regex 속성에 dict 형식으로 지정하면 정규표현식과 바꿀 값을 같이 지정할 수 있다.
2302
2303
           df.replace(regex={r'^ba.$': 'new', 'foo': 'xyz'})
2304
2305
                       В
2306
           0
                new abc
```

```
2307
          1
                XYZ
                       new
2308
          2
                bait
                       xyz
2309
2310
         -regex 속성이 list일 경우 value 속성이 지정되어야 한다.
2311
          df.replace(regex=[r'^ba.$', 'foo'], value='new')
2312
2313
2314
                Α
                       В
2315
          0
                new
                     abc
2316
          1
                new
                     new
2317
          2
                bait
                       XYZ
2318
2319
         -여러 개의 논리(bool)값 또는 날짜시간(datetime64) 객체를 바꿀 때 to_replace 매개변수의 데이터 유형
         이 바꿀 값의 데이터 유형과 일치해야 한다.
2320
         -다음 구문을 오류를 발생한다.
2321
          df = pd.DataFrame({'A': [True, False, True],
2322
2323
                             'B': [False, True, False]})
          df.replace({'astring': 'new value', True: False})
2324
                                                        # raises
2325
2326
          TypeError
                                         Traceback (most recentcall last)
2327
          <ipython-input-3-2313e57ab11c> in < module>
          ----> 1 df.replace({'a string': 'new value', True: False})
2328
          ... 생략 ...
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 동작
         을 비교/이해해야 한다.
2333
         -만일 다음과 같은 데이터가 있을 경우
2334
2335
          s = pd.Series([10, 'a', 'a', 'b', 'a'])
2336
2337
         와 같다.
2338
         -즉 모든 a의 값을 None으로 바꾼다.
2339
2340
          s.replace({'a': None})
2341
2342
                     10
          0
2343
          1
                     None
2344
          2
                     None
2345
          3
2346
          4
                     None
2347
          dtype: object
2348
2349
         -value=None 이고 to_replace가 scala, list, tuple일 경우 replace() 함수는 method 매개변수에 기
         본값('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
          0
                  10
2355
          1
                  10
2356
          2
                  10
          3
2357
                   b
2358
          4
                   b
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
         -Svntax
          DataFrame.where(cond, other=nan, inplace=False,
2366
2367
                            axis=None, level=None, errors='raise',
2368
                              try cast=False, raise on error=None)
2369
         -cond: boolean NDFrame, array-like, 또는 호출가능객체(callable).
          --만일 cond가 True이면 원래 값을 유지.
2370
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.
          --other가 callable 이면 NDFrame에서 계산되고 스칼라 또는 NDFrame을 반환해야 한다.
2376
2377
          --callable 객체는 입력 된 NDFrame을 변경하면 안된다.
2378
         -inplace: boolean, 기본값 False.
2379
          --만일 True 이면 현재 객체를 수정.
2380
         -axis : {0 또는 'index', 1 또는 'columns'}, 축을 지정한다.
         -level: int, 기본값 None, 정렬 수준을 지정한다.
2381
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.
2387
         -raise on error: boolean, 기본값 True.
          --유효한 데이터타입이 아니면 예외를 발생시킨다.
2388
         -where 메소드는 if-then 관용구의 응용 프로그램이다.
2389
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
          0
                  NaN
2402
          1
                  1.0
          2
                  2.0
2403
2404
          3
                  3.0
```

```
2405
                 4.0
          4
2406
          dtype: float64
2407
        -반면, mask() 함수는 해당 조건을 만족하는 데이터에 대해 NaN을 반환한다.
2408
2409
2410
          s.mask(s > 0)
2411
2412
          0
                 0.0
2413
          1
                 NaN
2414
          2
                 NaN
2415
          3
                 NaN
2416
          4
                 NaN
          dtype: float64
2417
2418
2419
        -다음 코드는 s 객체에서 1보다 큰 값을 반환하고 그렇지 않으면 10을 반환한다.
2420
2421
          s.where(s > 1,10)
2422
2423
          0
                 10
2424
          1
                 10
2425
          2
                 2
          3
2426
                 3
          4
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
            Α
               В
2437
        0 0
               -1
2438
        1 -2 3
2439
         2 -4 -5
         3 6 -7
2440
2441
        4
            -8 9
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
            Α
2448
           True
                     True
        0
2449
        1
            True
                     True
2450
         2
            True
                     True
2451
         3
            True
                     True
2452
        4
            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
            Α
                     В
2460
           True
                     True
        0
2461
            True
                     True
        1
        2
2462
            True
                     True
2463
        3
            True
                    True
2464
            True
                     True
2465
2466
2467
       4)dropna
         -dropna() 함수는 결측치(누락된 값)를 포함한 행 또는 열을 제거해 준다.
2468
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.
2474
          --0, 또는 'index': 누락된 값이 있는 행을 삭제한다.
2475
          --1, 또는 'columns' : 누락된 값이 포함 된 열을 삭제한다.
2476
         -how: {'any', 'all'}, 기본값 'any'.
          --'any': NA 값이 있는 경우 해당 행 또는 열을 삭제한다.
2477
2478
          --'all': 모든 값이 NA이면 행 또는 열을 삭제한다.
        -thresh: int, 선택사항.
2479
          --NaN이 아닌 항목의 최소 개수를 지정한다.
2480
          --예를 들어 thresh=2 이면 NaN을 포함하더라도 NaN이 최소 2개 이상이면 삭제하지 않는다.
2481
        -subset: array-like, 선택사항.
2482
2483
          --부분집합을 뽑을 다른 축의 이름이다.
2484
          --예: 행을 삭제하면 포함 할 열의 목록이 된다.
2485
         -inplace: boolean, 기본값 False.
          --만일 True 이면 현재 객체를 수정한다.
2486
2487
          df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'], "toy": [np.nan,
2488
          'Batmobile', 'Bullwhip'],
                            "born": [pd.NaT, pd.Timestamp("1940-04-25"), pd.NaT]})
2489
2490
          df
2491
2492
                 name
                                   toy
                                                   born
2493
          0
                 Alfred
                                   NaN
                                                  NaT
2494
          1
                 Batman
                                   Batmobile
                                                  1940-04-25
2495
          2
                 Catwoman
                                   Bullwhip
                                                  NaT
2496
2497
        -다음 구문은 최소 하나 이상의 요소가 누락된 값이 있는 행을 제거한다.
2498
2499
          df.dropna()
2500
2501
              name
                        toy
                                       born
2502
              Batman Batmobile
                                     1940-04-25
2503
2504
        -다음 구문은 누락된 값을 포함한 열을 제거해 준다.
```

```
2505
          df.dropna(axis='columns')
2506
2507
          -----
2508
                name
2509
          0
                Alfred
2510
          1
                 Batman
2511
          2
                 Catwoman
2512
2513
        -다음 구문은 행의 모든 요소가 누락된 값일 경우 행을 제거해 준다.
2514
2515
          df.dropna(how='all')
2516
                        toy
2517
               name
                                     born
2518
          0
               Alfred
                          NaN
                                   NaT
2519
          1
               Batman
                          Batmobile 1940-04-25
2520
          2
               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
               name
2527
                         Batmobile
          1
               Batman
                                     1940-04-25
2528
          2
               Catwoman
                           Bullwhip
                                     NaT
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
2536
               Batman
                         Batmobile 1940-04-25
          1
2537
2538
        -누락된 값을 제거하고 현재 DataFrame이 변경된다.
2539
2540
          df.dropna(inplace=True)
          df
2541
2542
2543
             name
                       toy
                                   born
2544
                       Batmobile 1940-04-25
          1
             Batman
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 특정
          유형으로 형변환하는 numpy.dtype 또는 Python의 datatype 이다.
```

```
2554
         -copy: bool, 기본값 True.
2555
           --copy=True 일 때 복사본을 반환한다.
         -errors : str, {\raise', \ignore'}, 기본값 raise.
2556
2557
           --errors='raise'이면 예외를 발생시킨다.
2558
           --errors='ignore' 이면 예외를 억제한다.
2559
           --오류 시 원본 개체를 반환한다.
2560
         -kwargs : 생성자에 전달할 keyword 인수이다.
2561
2562
           ser = pd.Series([1, 2], dtype='int32')
2563
           ser
2564
           0 1
2565
2566
           1
                  2
2567
           dtype: int32
2568
2569
           ser.astype('int64')
2570
2571
           0
                 1
2572
           1
                  2
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
2580
           s1 # note thats1[0] has changedtoo
2581
2582
           0
                  10
2583
                    2
           1
2584
           dtype: int64
2585
2586
2587
       6) 중복된 값 제거하기
2588
2589
         data list = [
           {'Name':'John', 'Gender':'male', 'Job':'Student'},
2590
           {'Name':'Smith','Gender':'male', 'Job':'Teacher'},
2591
           {'Name':'Jenny','Gender':'female', 'Job':'Developer'},
2592
2593
           {'Name':'Smith','Gender':'male', 'Job':'Teacher'}
2594
2595
         df = pd.DataFrame(data list, columns = ['Name', 'Gender', 'Job'])
2596
2597
         df.head()
2598
2599
             Name Gender Job
2600
         0 John male
                               Student
         1 Smith male
                             Teacher
2601
         2 Jenny female
2602
                             Developer
            Smith male
                             Teacher
2603
                                           #중복된 값
2604
```

```
2605
           -중복된 값 확인하기
2606
2607
             df.duplicated()
2608
2609
             0
                 False
2610
             1
                 False
2611
             2
                 False
2612
             3
                   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
             0 John
                       male
                                     Student
2621
             1 Smith male
                                   Teacher
2622
             2 Jenny female
                                   Developer
2623
2624
2625
        7)기타 Null 처리하기
2626
2627
           student list = [
             {'Name': 'John', 'Major': 'Computer Science', 'Gender': 'male', 'Age': 40},
2628
2629
               {'Name': 'Nate', 'Major': None, 'Gender': "male", 'Age':35},
                    {'Name': 'Abraham', 'Major': 'Physics', 'Gender': 'male', 'Age':37},
2630
                   {'Name': 'Brian', 'Major': 'Psychology', 'Gender': 'male', 'Age':None},
2631
                   {'Name': 'Janny', 'Major': None, 'Gender': 'female', 'Age':10},
2632
                   {'Name': 'Yuna', 'Major': None, 'Gender': 'female', 'Age':12},
2633
                   {'Name': 'Jeniffer', 'Major': 'Computer Science', 'Gender': 'female', 'Age':45}, {'Name': 'Edward', 'Major': 'Computer Science', 'Gender': 'male', 'Age':None}, {'Name': 'Zara', 'Major': 'Psychology', 'Gender': 'female', 'Age':25},
2634
2635
2636
                   {'Name': 'Wendy', 'Major': 'Economics', 'Gender': 'female', 'Age':37},
2637
                   {'Name': 'Sera', 'Major': None, 'Gender': 'female', 'Age': None}
2638
2639
               1
           df = pd.DataFrame(student_list, columns = ['Name', 'Major', 'Gender', 'Age'])
2640
2641
2642
2643
                        Major
                                           Gender
               Name
2644
               John
                          Computer Science male
                                                             40.0
           0
2645
           1
               Nate
                          None
                                              male
                                                           35.0
2646
           2
               Abraham Physics
                                              male
                                                           37.0
           3
2647
               Brian
                        Psychology
                                              male
                                                           NaN
2648
           4
               Janny
                        None
                                           female
                                                         10.0
           5
2649
               Yuna
                        None
                                           female
                                                         12.0
2650
               Jeniffer Computer Science
                                              female
                                                           45.0
2651
           7
               Edward Computer Science
                                              male
                                                           NaN
2652
           8
               Zara
                          Psychology
                                                female
                                                             25.0
                          Economics
2653
           9
               Wendy
                                              female
                                                           37.0
2654
           10 Sera
                          None
                                              female
                                                           NaN
2655
```

```
2656
          df.shape
2657
2658
          (11,4)
2659
2660
          df.info()
2661
2662
          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 11 entries, 0 to 10
2663
2664
          Data columns (total 4 columns):
2665
          Name
                   11 non-null object
2666
          Major
                   7 non-null object
          Gender 11 non-null object
2667
2668
          Age
                  8 non-null float64
2669
          dtypes: float64(1), object(3)
2670
          memory usage: 432.0+ bytes
2671
          -Null 확인하기
2672
2673
2674
            df.isna()
2675
2676
                Name Major Gender Age
2677
            0
                False False False
                                    False
                False True
2678
            1
                              False
                                       False
2679
            2
                False False False
                                    False
2680
                False False False
            3
                                     True
            4
2681
                False True
                              False
                                      False
            5
                False True
                                      False
2682
                              False
            6
                False False False
2683
                                    False
2684
            7
                False False False
                                    True
2685
            8
                False False False
                                    False
2686
            9
                False False False
                                    False
            10 False True
                              False
                                      True
2687
2688
2689
            df.isnull()
2690
2691
                Name Major Gender Age
2692
                False False False
            0
                                    False
2693
                False True
                                      False
            1
                              False
            2
                False False False
2694
                                    False
2695
            3
                False False False
                                     True
            4
2696
                False True
                              False
                                      False
2697
            5
                False True
                              False
                                      False
2698
            6
                False False False
                                    False
2699
            7
                False False False
                                    True
                False False False
2700
            8
                                     False
2701
            9
                False False False
                                     False
2702
            10 False True
                              False
                                      True
2703
          -None값을 다른 값으로 변경하기
2704
2705
2706
            df.Age = df.Age.fillna(0)
                                       #숫자 NaN을 0으로
```

```
df.Major = df.Major.fillna('Unknown') #글자 None을 Unknown으로
2707
2708
2709
2710
               Name Major
                                         Gender
                                                  Age
2711
               John
                         Computer Science
                                                        40.0
           0
                                            male
                                                        35.0
2712
           1
               Nate
                         Unknown
                                            male
2713
           2
               Abraham Physics
                                          male
                                                      37.0
           3
2714
               Brian
                       Psychology
                                         male
                                                      0.0
2715
               Janny Unknown
                                          female
                                                      10.0
           5
2716
               Yuna
                       Unknown
                                          female
                                                      12.0
2717
           6
               Jeniffer Computer Science female
                                                      45.0
           7
               Edward Computer Science male
2718
                                                      0.0
2719
           8
               Zara
                         Psychology
                                            female
                                                        25.0
           9
                         Economics
                                                      37.0
2720
               Wendy
                                          female
2721
           10 Sera
                         Unknown
                                            female
                                                        0.0
2722
       8)Unique와 갯수 알아보기
2723
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
                              3
2737
           Computer Science
2738
           Psychology
                               2
           Physics
                               1
2739
2740
           Economics
           Name: Major, dtype: int64
2741
2742
2743
2744
2745 14. DataFrame 합치기
2746
2747
         list1 = [
           {'Name': 'John', 'Job': 'Teacher'},
2748
2749
           {'Name': 'Nate', 'Job': 'Student'},
           {'Name': 'Fred', 'Job': 'Developer'}
2750
2751
         1
2752
2753
         list2 = \Gamma
           {'Name': 'Ed', 'Job': 'Dentist'},
2754
2755
           {'Name': 'Jack', 'Job': 'Farmer'},
2756
           {'Name': 'Ted', 'Job': 'Designer'}
```

```
2757
         ]
2758
2759
         df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
         df2 = pd.DataFrame(list2, columns = ['Name', 'Job'])
2760
2761
         -concat()로 합치기
2762
2763
2764
           result = pd.concat([df1, df2])
2765
           result
2766
2767
               Name Job
2768
           0
               John
                       Teacher
2769
           1
               Nate
                       Student
           2
2770
               Fred
                       Developer
                                         #0, 1, 2가 반복
2771
           0
              Ed
                     Dentist
2772
           1
               Jack
                     Farmer
           2
2773
               Ted
                     Designer
2774
2775
           result = pd.concat([df1, df2], ignore_index = True)
2776
           result
2777
2778
               Name Job
2779
           0
               John
                       Teacher
2780
           1
               Nate
                       Student
           2
2781
               Fred
                       Developer
2782
           3
               Ed
                     Dentist
2783
           4
               Jack
                       Farmer
           5
2784
               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
           2
                       Developer
               Fred
2795
           0
               Ed
                     Dentist
                                          #0, 1, 2가 반복
2796
           1
                       Farmer
               Jack
           2
2797
               Ted
                     Designer
2798
2799
           result = pd.append([df1, df2], ignore_index = True)
2800
           result
2801
2802
               Name Job
                       Teacher
2803
           0
               John
2804
           1
               Nate
                       Student
2805
           2
                       Developer
               Fred
           3
                     Dentist
2806
               Ed
2807
           4
               Jack
                       Farmer
```

```
2808
               Ted
                    Designer
2809
2810
         -Column으로 합치기
           --두개의 DataFrame의 column이 서로 일치하지 않음.
2811
2812
2813
             list1 = [
               {'Name': 'John', 'Job': 'Teacher'}, {'Name': 'Nate', 'Job': 'Student'},
2814
2815
2816
               {'Name': 'Jack', 'Job': 'Developer'}
2817
2818
2819
             list2 = [
2820
               {'Age': 25, 'Country': 'U.S'},
               {'Age': 30, 'Country': 'U.K'},
2821
               {'Age': 45, 'Country': 'Korea'}
2822
2823
             1
2824
2825
             df1 = pd.DataFrame(list1, columns = ['Name', 'Job'])
2826
             df2 = pd.DataFrame(list2, columns = ['Age', 'Country'])
2827
2828
             result = pd.concat([df1, df2], axis=1, ignore_index=True)
2829
             result
2830
2831
                 0
                      1
                                2 3
                        Teacher 25 U.S
2832
             0
                John
                        Student 30 U.K
2833
             1
                Nate
             2
2834
                 Jack
                        Developer 45 Korea
2835
2836
2837
2838 15. 기초 통계 분석
2839
       1)Pandas는 데이터를 보다 좀 더 편하게 다룰 수 있게 하는 데이터 구조 측면에서의 장점을 가진 패키지이다.
2840
       2)Pandas에서 제공하는 통계분석은 기본적인 기술통계 및 데이터 요약이다.
2841
       3)고급 통계 기법을 사용하고 싶다면 Scikit-learn 이나 다른 통계 패키지를 이용하여 수행 할 수 있다.
2842
       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
2850
         -quantile : 분위수
2851
         -corr : 상관관계
2852
         -var : 표본분산
2853
         -std : 표본 정규분산
2854
2855
       -다음 코드는 기술 통계량을 확인해 보기 위해 데이터를 불러오자.
2856
       -Statsmodels 패키지를 이용해 iris 데이터를 불러온다.
2857
2858
         import statsmodels.api as sm
```

```
iris = sm.datasets.get rdataset("iris", package="datasets")
2859
2860
         iris df = iris.data
2861
         iris df.head()
         ......
2862
2863
             Sepal.Length Sepal.Width Petal.Length Petal.Width Species
                          3.5
                                      1.4
2864
            5.1
                                                   0.2
         0
                                                               setosa
2865
         1
            4.9
                          3.0
                                       1.4
                                                   0.2
                                                               setosa
         2
            4.7
                                                   0.2
2866
                         3.2
                                      1.3
                                                               setosa
         3 4.6
2867
                         3.1
                                      1.5
                                                   0.2
                                                               setosa
2868
           5.0
                          3.6
                                       1.4
                                                   0.2
                                                               setosa
2869
2870
2871
       5)최솟값, 최댓값, 평균, 중위수
2872
         -Syntax
           DataFrame.min(axis=None, skipna=None, level=None, numeric only=None,
2873
           **kwarqs)
         -axis: 0이면 index, 1이면 columns를 의미다.
2874
2875
         -skipna: True(기본값)이면 NA 또는 null 값을 계산에서 제외한다.
         -level: 다중 index일 경우 level을 지정한다.
2876
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
         Sepal.Length
2883
                                 4.3
                               2
2884
         Sepal.Width
                               1
2885
         Petal.Length
2886
         Petal.Width
                                 0.1
2887
         Species
                                 setosa
2888
         dtype: object
2889
2890
         iris df.max()
2891
         Sepal.Length
2892
                                 7.9
         Sepal.Width
2893
                               4.4
                               6.9
         Petal.Length
2894
                                 2.5
2895
         Petal.Width
                                 virginica
2896
         Species
2897
         dtype: object
2898
         iris_df.mean()
2899
2900
2901
         Sepal.Length
                           5.843333
         Sepal.Width
2902
                         3.057333
2903
         Petal.Length
                         3.758000
2904
         Petal.Width
                           1.199333
         dtype: float64
2905
2906
2907
         iris_df.median()
2908
```

2000	
2909	Sepal.Length 5.80
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	-Syntax
2925	DataFrame.describe(percentiles=None, include=None, exclude=None)
2926	- $percentiles$: 출력에 포함될 백분위 수를 $0{\sim}1$ 사이의 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.
2936	-기본적으로 하위 백분위 수는 25이고 상위 백분위 수는 75이다.
2937	-50 백분위 수는 중앙값과 같다.
2938	-객체 데이터(예: 문자열 또는 timestamp)의 경우 결과 색인에 count, unique, top 그리고 freq가 포함된
	다.
2939	-top가 가장 일반적인 값이다.
2940	-freq는 가장 일반적인 값의 빈도수이다.
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 임의
	로 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 분
	석을 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
                  Sepal.Length Sepal.Width
                                           Petal.Length Petal.Width
2956
          count
                 150.000000
                              150.000000
                                             150.000000
                                                           150.000000
2957
          mean
                 5.843333
                              3.057333
                                             3.758000
                                                           1.199333
2958
                 0.828066
                              0.435866
                                             1.765298
                                                           0.762238
          std
2959
                 4.300000
                              2.000000
                                             1.000000
                                                           0.100000
          min
2960
          25%
                 5.100000
                              2.800000
                                             1.600000
                                                           0.300000
2961
          50%
                 5.800000
                              3.000000
                                             4.350000
                                                           1.300000
2962
          75%
                 6.400000
                              3.300000
                                             5.100000
                                                           1.800000
2963
                 7.900000
                              4.400000
                                             6.900000
                                                           2.500000
          max
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 출력한
         다.
2967
2968
          iris df.Species.describe()
2969
2970
                         150
          count
2971
                         3
          unique
2972
          top
                         virginica
2973
                       50
          freq
2974
          Name: Species, dtype: object
2975
2976
2977
       9)include와 rexclude
         -다음 데이터의 요약 통계량을 출력하면 a열과 b열의 요약 통계량만 출력된다.
2978
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
                              С
2985
          count 6.000000
                              6.000000
2986
          mean 1.500000
                              3.000000
2987
          std
                 0.547723
                              1.095445
2988
                 1.000000
                              2.000000
          min
          25%
                              2.000000
2989
               1.000000
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.
         -int64 유형을 include 시키거나 나머지 유형들을 exclude 시키면 된다.
2997
2998
        -앞의 a,b,c열을 갖는 데이터에서 다음 두 구문은 같은 결과를 출력할 것이다.
2999
          df.describe(include=["int64"])
3000
          df.describe(exclude=["bool", "float64"])
3001
3002
3003
                   а
```

```
3004
           count
                    6.000000
3005
                    1.500000
           mean
3006
                    0.547723
           std
3007
           min
                    1.000000
           25%
3008
                    1.000000
3009
           50%
                    1.500000
3010
           75%
                    2.000000
3011
                    2.000000
           max
3012
3013
         -모든 요소에 대해 요약통계량을 출력하려면 include='all'을 이용한다.
3014
3015
           df.describe(include='all')
3016
3017
                                 b
                                         С
                    а
                    6.000000
                                 6
3018
           count
                                         6.000000
3019
                    NaN
                               2
           unique
                                       NaN
3020
                    NaN
                               True
                                       NaN
           top
3021
                             3
                                     NaN
           frea
                  NaN
3022
                    1.500000
                                 NaN
                                      3.000000
           mean
3023
                    0.547723
                                 NaN
                                       1.095445
           std
3024
                    1.000000
                                 NaN
                                       2.000000
           min
3025
           25%
                    1.000000
                                 NaN
                                       2.000000
3026
           50%
                    1.500000
                                 NaN
                                       3.000000
3027
           75%
                    2.000000
                                 NaN 4.000000
3028
                    2.000000
                                 NaN 4.000000
           max
3029
3030
         -다음처럼 include와 exclude에 같은 유형을 사용하면 오류가 발생한다.
3031
3032
           df.describe(include=["int64"], exclude=["int64", "float64"])
3033
3034
                                     Traceback (most recentcall last)
3035
           <ipython-input-64-52780e7f55bd> in <module>()
3036
3037
3038
       10)분산,표준편차
3039
         -var()는 분산(variance)을, std()는 표준편차(standard deviation)를 계산한다.
3040
         -Svntax
3041
           DataFrame.var(axis=None, skipna=None, level=None, ddof=1,
                            numeric_only=None, **kwargs)
3042
           DataFrame.std(axis=None, skipna=None, level=None, ddof=1,
3043
                            numeric only=None,**kwargs)
3044
3045
         -ddof: 델타 자유도(Delta Degree of Freedom)를 지정한다.
3046
           --기본값은 1이다.
3047
           --계산에 사용되는 제수는 N-ddof이다.
           --여기서 N은 요소의 수를 나타낸다.
3048
3049
3050
           iris df.var()
3051
3052
                             0.685694
           Sepal.Length
           Sepal.Width
3053
                            0.189979
           Petal.Length
                            3.116278
3054
```

```
3055
          Petal.Width
                            0.581006
3056
          dtype: float64
3057
3058
          iris df.std()
3059
3060
          Sepal.Length
                            0.828066
3061
          Sepal.Width
                           0.435866
3062
          Petal.Length
                           1.765298
          Petal.Width
3063
                            0.762238
3064
          dtype: float64
3065
3066
       11) 공분산(covariance), 상관계수(correlation)
3067
         -cov()는 각 열들의 공분산 쌍을 계산한다.
         -corr()는 각 열들의 상관계수 쌍을 계산한다.
3068
3069
         -Svntax
3070
           DataFrame.cov(min_periods=None)
           DataFrame.corr(method='pearson', min_periods=1)
3071
3072
         -min_periods : 유효한 결과를 얻기 위해 열 쌍당 필요한 최소 관측 수를 지정한다.
3073
         -method : 상관계수를 계산할 방법을 지정한다.
3074
          --"pearson", "kendall", "spearman" 중 하나를 지정할 수 있다.
3075
3076
          iris_df.cov()
3077
3078
                           Sepal.Length
                                         Sepal.Width Petal.Length Petal.Width
3079
          Sepal.Length
                         0.685694
                                         -0.042434 1.274315
                                                                0.516271
          Sepal.Width
                      -0.042434
                                     0.189979
3080
                                                  -0.329656 -0.121639
          Petal.Length
                                     -0.329656 3.116278
3081
                         1.274315
                                                               1.295609
          Petal.Width
                          0.516271
3082
                                         -0.121639 1.295609
                                                                0.581006
3083
3084
        iris_df.corr()
3085
                       Sepal.Length Sepal.Width Petal.Length Petal.Width
3086
3087
         Sepal.Length 1.000000
                                   -0.117570 0.871754
                                                           0.817941
         Sepal.Width -0.117570 1.000000
                                            -0.428440 -0.366126
3088
3089
         Petal.Length 0.871754 -0.428440 1.000000
                                                         0.962865
        Petal.Width
3090
                       0.817941
                                  -0.366126 0.962865
                                                           1.000000
3091
3092
3093
       12)통계 산출하기 복습
3094
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
          df
3100
            학번
                   이름
                         국어 영어 수학 전산
3101
3102
          0 1101
                   한송이 78 87 83 78
          1 1102 정다워 88 83 57 98
3103
3104
3105
          . . .
```

```
3106
3107
          df.loc[:, '국어':'전산'].mean()
3108
3109
          국어
              84.916667
3110
          영어
              80.416667
          수학
3111
              75.333333
3112
          전산
              79.750000
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 백분
        위수가 중앙값이다)이다.
3122
3123
          df.describe().round(1) #round() 반올림함수
3124
3125
               국어 영어 수학 전산
3126
          count 12.0 12.0
                             12.0
                                     12.0
                              75.3
3127
          mean 84.9
                      80.4
                                     79.8
3128
          std
               10.7
                      15.3
                             15.2
                                     11.6
3129
                                     55.0
          min 68.0
                      56.0
                              53.0
                 77.5
                        67.0
                               61.5
3130
          25%
                                      76.8
                 87.5
                        85.0
                               80.5
                                      78.0
3131
          50%
3132
          75%
                 91.2
                        91.0
                               87.2
                                      88.0
3133
                 98.0
                        99.0
                               93.0
                                      98.0
          max
3134
3135
        -백분위수 값을 변경할 경우에는 keyword 인수 percentiles의 list 요소에 1이상의 소수 값을 지정한다.
3136
3137
          df.describe(percentiles = [0.1, 0.9]).round(1)
3138
3139
                국어
                     영어 수학 전산
3140
          count 12.0
                     12.0
                              12.0
                                     12.0
                      80.4
                             75.3
3141
          mean 84.9
                                     79.8
                      15.3
                              15.2
3142
                                     11.6
          std
               10.7
                                     55.0
3143
               68.0
                      56.0
                              53.0
          min
          10%
                 68.8
                        58.0
                               53.4
                                      66.7
3144
                 87.5
3145
          50%
                        85.0
                               80.5
                                      78.0
                                      88.9
3146
          90%
                 98.0
                        97.0
                               92.5
3147
                 98.0
                        99.0
                               93.0
                                      98.0
          max
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.
          --count : 결손값을 제외한 data 수
3152
3153
          --unique: unique한 data 수
3154
          --top: data의 수가 가장 많은 값
3155
          --freq: top의 data 수
```

```
3156
3157
         df[['학번', '이름']].describe()
3158
3159
               학번 이름
3160
               12
                    12
         count
3161
         unique 12 12
3162
               1102 한산섬
         top
3163
         freq
               1
                    1
3164
3165
       -논리값으로 data 추출하기
3166
         df.loc[df['국어'] > 90].head()
3167
3168
            학번 이름 국어 영어 수학 전산
3169
                  튼튼이 98 97 93 88
         5 1106
3170
         7 1108 더크게 98 67 93 78
3171
                 한산섬 98 89 73 78
3172
         10 1111
3173
3174
         df.query('학번 == 1104')
3175
3176
                  이름 국어 영어 수학 전산
3177
         3
            1104 고아라 83 57 88 73
3178
3179
       -where method로 data 추출하기
3180
3181
         df.where(df['영어'] < 70)
3182
3183
            학번
                 이름
                      국어
                          영어 수학
                                    전산
3184
            NaN NaN NaN NaN NaN
         0
3185
            NaN NaN NaN NaN NaN
         1
3186
            1103.0 그리운 76.0
                             56.0
                                    87.0
                                          78.0
            1104.0 고아라 83.0
         3
                              57.0
                                    0.88
                                          73.0
3187
3188
         4
            NaN NaN NaN NaN NaN
         5
            NaN NaN NaN NaN NaN
3189
3190
         6
            1107.0 한아름 68.0
                            67.0
                                    83.0
                                          89.0
         7
            1108.0 더크게 98.0 67.0
3191
                                   93.0
                                          78.0
3192
         8
            NaN NaN NaN NaN NaN
3193
         9
            NaN
                      NaN NaN NaN NaN
                 NaN
3194
         10 NaN
                 NaN
                      NaN
                          NaN NaN NaN
         11 NaN NaN
3195
                      NaN
                         NaN NaN NaN
3196
3197
       -값 변경하기
3198
3199
         df.head(3)
3200
3201
            학번
                 이름 국어 영어 수학 전산
         0 1101 한송이 78 87 83 78
3202
                  정다워 88 83 57 98
3203
         1 1102
                 그리운 76 56 87 78
3204
         2
            1103
3205
         df.loc[1, '전산'] = np.nan
3206
```

```
3207
3208
          df.loc[1, '전산']
3209
3210
          nan
3211
3212
          df.head(3)
3213
3214
             학번
                    이름 국어 영어 수학 전산
3215
          0 1101 한송이 78 87 83 78.0
3216
                    정다워 88 83 57 NaN
          1
             1102
3217
          2
             1103 그리운 76 56 87 78.0
3218
3219
        -복수의 값 변경
3220
          df.loc[df['학번'] > 1110, '수학'] = np.nan
3221
3222
3223
          df.tail(2)
3224
3225
             학번
                  이름 국어 영어 수학 전산
          10 1111 한산섬 98 89 NaN 78.0
3226
3227
          11 1112 하나로 89 97 NaN 88.0
3228
3229
        -결손값 제외하기
          df.loc[df['수학'].isnull()]
3230
3231
3232
             학번 이름 국어 영어 수학
          10 1111
                    한산섬 98 89 NaN 78.0
3233
3234
          11 1112 하나로 89 97 NaN 88.0
3235
3236
        -결손값이 포함되어 있는 data 제외
3237
3238
          df.dropna().loc[8:] #8번째 이후 data 중 NaN값이 있는 data제외
3239
           학번
3240
                  이름 국어 영어 수학 전산
          8 1109 더높이 88 99 53.0 88.0
3241
          9 1110 아리랑 68 79 63.0 66.0
3242
3243
3244
        -dropna()는 비파괴적 조작이다.
        -따라서 df에는 이전 data가 남아있다.
3245
3246
        -DataFrame의 내용을 파괴적으로 다시 쓰는 경우
3247
3248
          df.dropna(inplace = True)
3249
3250
        -Data 형
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
         --Series의 data 형을 확인할 때에는 dtype을 참조한다.
3254
3255
3256
           df['국어'].dtype
3257
```

```
3258
            dtype('int64')
3259
3260
        -DataFrame의 data 형을 확인하는 경우
3261
        -DataFrame의 data 형을 확인할 때는 dtypes를 참조한다.
3262
3263
          df.dtypes
3264
          _____
3265
          학번 int64
          이름 object
국어 int64
영어 int64
3266
3267
3268
          수학 float64
3269
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
              1101
          0
3279
          1 1102
          2
3280
              1103
3281
          3
              1104
3282
          ...
3283
          . . .
          10
3284
              1111
3285
          11 1112
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
          학번
                 int64
3296
          이름 object
          국어 int64
영어 int64
3297
3298
3299
          수학
              float64
                              #변경되지 않음. 비파괴적이어서...
3300
          전산
               float64
3301
          dtype: object
3302
3303
        -DataFrame을 다시 쓰는 경우
3304
          df['수학'] = df['수학'].astype(np.str)
3305
3306
          df.dtypes
3307
3308
          학번 int64
```

```
3309
           이름
                  object
3310
           국어
                  int64
3311
           영어
                  int64
3312
           수학
                  object
                              #변경됐음.
3313
           전산
                float64
           dtype: object
3314
3315
3316
         -Sort 하기
3317
3318
           del df
3319
           df = pd.read csv('sungjuk utf8.csv', header = None,
               names = ['학번', '이름', '국어', '영어', '수학', '전산'])
3320
3321
3322
           df.sort_values('국어', ascending=False)
3323
3324
              학번
                      이름
                           국어 영어 수학 전산
                      튼튼이 98 97 93 88
3325
              1106
3326
           7
              1108 더크게 98 67 93 78
3327
           10 1111
                      한산섬 98 89 73 78
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
3345
         -Clipboard
3346
         -Pickle
         -공공데이터포털: https://www.data.go.kr/
3347
3348
         -7|E+(http://pandas.pydata.org/pandas-docs/stable/io.html)
3349
3350
3351
3352 17. CSV file 불러오기
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
3362
3363
        0 John 20 student
3364
        1
            Jenny
                   30
                        developer
3365
        2
           Nate
                     30
                          teacher
3366
        3
            Julia
                     40
                          dentist
3367
            Brian
                   45
                        manager
3368
3369
3370
       4) Data Frame. head()는 앞에서 5행분의 Data Frame을 넘겨준다.
       5)인수에 정수값을 넘기는 방식으로 행수를 지정할 수 있다.
3371
3372
3373
        df = pd.read_csv('friend_list.csv')
3374
        df.head(2)
3375
3376
          name age job
3377
        0 John 20 student
3378
        1 Jenny 30 developer
3379
3380
3381
       6)뒤에서부터 읽을 행의 갯수를 지정할 수도 있다.
3382
        df = pd.read_csv('friend_list.csv')
3383
3384
        df.tail(2)
3385
3386
                   age job
            name
                   45
3387
        4
          Brian
                        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
       8)지정한 열을 DataFrame의 index로 하기
3398
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
                      developer
3408
          Jenny
                 30
                   30
                        teacher
3409
          Nate
```

```
3410
           Julia
                    40
                          dentist
3411
           Brian
                  45
                        manager
3412
           #index로 지정할 열을 열 이름으로 지정
3413
3414
           df = pd.read_csv('friend_list.csv', index_col = 'job')
3415
           df.head()
3416
3417
                    name age
3418
           job
3419
           student
                               20
                      John
3420
           developer
                      Jenny
                             30
3421
           teacher
                    Nate
                             30
3422
           dentist
                      Julia
                               40
3423
                             45
           manager
                      Brian
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
3429
           #형 지정
3430
           df = pd.read_csv('friend_list.csv', dtype={'age':float})
3431
           df.head()
3432
                name age job
3433
3434
           0
                John
                        20.0
                                 student
                        30.0
3435
           1
                Jenny
                               developer
           2
3436
                Nate
                          30.0
                                 teacher
3437
           3
                         40.0
                Julia
                                 dentist
3438
           4
                Brian
                        45.0
                               manager
3439
3440
3441
       10)형식이 유사한 txt file 읽어오기
        -CSV file처럼 txt file도 각 열의 구분을 ','로 할 경우
3442
3443
3444
           df = pd.read_csv('friend_list.txt')
           df.head()
3445
3446
3447
                name
                        age job
3448
           0
                John
                        20
                               student
3449
           1
                Jenny
                        30
                             developer
3450
           2
                Nate
                         30
                               teacher
3451
           3
                Julia
                          40
                               dentist
3452
           4
                Brian
                        45
                             manager
3453
3454
3455
       11)만일 file의 column들이 쉼표로 구분되어 있지 않은 경우
         -delimiter parameter에 구분자를 지정해서 column을 나눠야 한다.
3456
3457
3458
           df = pd.read_csv('friend_list_tab.txt')
3459
           df.head()
```

```
3460
3461
                name age job
          0 John\t20\tstudent
3462
          1 Jenny\t30\tdeveloper
3463
3464
          2 Nate\t30\tteacher
          3 Julia\t40\tdentist
3465
3466
          4 Brian\t45\tmanager #구분이 어려움.
3467
3468
          df = pd.read csv('friend list tab.txt', delimiter = '\t')
3469
          df.head()
3470
          -----
                name age job
3471
3472
          0
                John 20 student
                       30
               Jenny
3473
          1
                            developer
          2
3474
                Nate 30
                            teacher
3475
          3
                Julia
                       40
                              dentist
3476
          4
                Brian
                       45
                            manager
3477
3478
3479
       12)file에 data header가 없는 csv file을 사용할 때
         -만일 data header가 없으면, header = None으로 지정해야 첫번째 data가 data header로 들어가는
3480
         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
                Nate 30 teacher
          1
3487
3488
          2
                Julia
                       40 dentist
3489
          3
                Brian 45 manager
3490
          4
                Chris 25 intern
3491
          df = pd.read csv('friend list no head.csv', header = None)
3492
3493
          df
3494
3495
              0 1 2
3496
          0 John 20 student
          1 Jenny 30 developer
3497
          2 Nate
                     30 teacher
3498
          3
                     40 dentist
3499
              Julia
3500
          4
              Brian 45 manager
              Chris 25 intern
3501
          5
3502
3503
        -만일 header가 없는 data를 호출했을 경우, DataFrame 생성 후, column header를 지정할 수 있다.
3504
3505
          df.columns = ['Name', 'Age', 'Job']
          df.index = ['1101', '1102', '1103', '1104', '1105', '1106']
3506
3507
3508
3509
                  Name Age Job
```

```
3510
           1101
                   John
                             20
                                   student
3511
           1102
                   Jenny
                           30
                                 developer
3512
           1103
                   Nate
                             30
                                   teacher
3513
           1104
                   Julia
                             40
                                   dentist
3514
           1105
                   Brian
                           45
                                 manager
3515
           1106
                   Chris
                           25
                                 intern
3516
         -file을 열 때 동시에 header에 column을 지정해야 할 경우
3517
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
3525
        13)외부 CSV file 읽기
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.
3532
         -http://panda.pydata.org/pandas-docs/stable/io.html#io-read-csv-table
3533
3534
3535
        15)DataFrame csv file로 저장하기
3536
3537
         import pandas as pd
3538
3539
         user list = [
            {'Name':'John', 'Age':25, 'Gender': 'male', 'Address':'Chicago'},
3540
            {'Name':'Smith', 'Age':35, 'Gender': 'male', 'Address':None},
3541
            {'Name':'Jenny', 'Age':45, 'Gender': 'female', 'Address':'Dallas'}
3542
         1
3543
3544
3545
         df = pd.DataFrame(user list)
3546
3547
         df = df[['Name', 'Age', 'Gender','Address']]
3548
3549
         df.head()
3550
3551
         Name
                         Gender
                                   Address
                   Age
3552
         0 John
                    25
                           male
                                       Chicago
3553
                   35
         1 Smith
                         male
                                     None
3554
         2 Jenny
                   45
                         female
                                     Dallas
3555
3556
         df.to csv('user list.csv')
3557
3558
         -기본적으로 to_csv()는 index = True, header = True가 설정되어 있다.
3559
         -만일 index = False로 설정할 경우
```

```
3560
3561
           df.to_csv('user_list.csv', index=False)
3562
         -각 행의 index 즉 0, 1, 2가 사라진 csv file이 생성된다.
3563
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.
       2)사전준비를 위해 xlrd module을 설치여부를 확인한다.
3577
3578
3579
         !conda list | grep xlrd #Windows에서는 사용 불가
3580
3581
         $ conda install -y xrld==1.2.0 #설치안되어 있으면 설치
3582
3583
         # Excel file 불러오기
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. SOL을 사용해서 불러오기
       1)pandas.read_sql() 함수를 사용한다.
3597
3598
       2)첫번째 인수에 query를 실행하는 SQL문, 두번째 인수에
       SQLAlchemy(http://docs.sglalchemy.org/en/latest/dialects/index) 또는 DBAPI2(PEP 249,
       Python Database API Specification v2.0, https://www.python.org/dev/peps/pep-0249)의 접
       속 instance를 넘겨준다.
3599
3600
       3)MariaDB with Python
3601
         -설치여부 확인하기
3602
           !conda list | grep mysql-connector-python
3603
           #Windows에서는 grep 명령어 사용 불가
3604
3605
3606
         -mysql-connector-python 설치하기
3607
```

```
3608
            --In Anaconda Prompt,
3609
3610
              $ conda install -y mysgl-connector-python
3611
3612
              import mysgl.connector as mariadb
3613
3614
              mariadb_connection = mariadb.connect(user='root', password='javamariadb',
              host='localhost', database='world')
3615
              cursor = mariadb connection.cursor()
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'
3620
                % (ID, Name, 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
3625
              ID = 2334, Name = Taequ, CountryCode = KOR, District = Taequ, Popluation =
              2548568
              ID = 2335, Name = Taejon, CountryCode = KOR, District = Taejon, Popluation =
3626
              1425835
3627
3628
3629
              mylist = []
3630
              for ID, Name, Country Code, District, Population in cursor:
                list = []
3631
3632
                list.append(ID); list.append(Name); list.append(CountryCode)
                list.append(District); list.append(Population)
3633
3634
                mylist.append(list)
3635
              df = pd.DataFrame(data = mylist, columns = ['ID', 'Name', 'CountryCode',
3636
              'District','Population'])
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
          -Install on Windows
3643
3644
            $ python -m pip install cx_Oracle --upgrade
3645
          -In Anaconda Prompt
3646
3647
            $ conda install cx oracle
3648
3649
          -Connection
```

```
import cx Oracle
3650
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
             (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CO
3663
             NNECT DATA=(SID=XE)))
3664
             conn2 = cx_Oracle.connect('scott', 'tiger', dsn_tns)
3665
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
             -----
             '11.2.0.2.0'
3672
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
                   from employees e inner join departments d on e.department id =
3678
                   d.department id
                   inner join locations I on d.location id = I.location id"""
3679
3680
             cursor.execute(sql)
3681
3682
             for employee_id, first_name, salary, hire_date, department_name, city in cursor:
               print(employee_id, first_name, salary, hire_date, department_name, city)
3683
3684
             cursor.close()
3685
```