CH. 9. 판다스

Section 01 판다스 개요

❑ 판다스의 데이터 구조

- **파이썬에도 데이터를 저장하기 위한 많은 데이터 구조가 있지만 판다스는 데이터 저장을 위하여 다음과 같은 2가지의 데이터 구조를 파이썬에 추가한다.**

  ➢ **시리즈**(Series) : **시리즈는 동일한 유형의 데이터를 저장하는 1차원 배열이다. 예를 들면, 다음과 같은 데이터가 시리즈이다.**

11	73	53	27	52	65	74	98	13	72

  ➢ **데이터 프레임** (DataFrame) : **데이터 프레임은 동일한 유형의 데이터가 2차원적으로 모인것이다. 예를 들면 다음과 같이, 행과 열로 되어 있는 2차원 테이블이 데이터 프레임이다. 각 열은 시리즈로 되어 있다.**

	이름	나이	성별	평점
0	김철수	19	Male	3.45
1	김영희	22	Female	4.1
2	김영수	20	Male	3.9
3	최지영	26	Female	4.5

행(row)

열(column)

Section 01 판다스 개요

❑ Index와 columns 객체

- **데이터 프레임에서는 행이나 열에 붙인 레이블을 중요시한다.**
- **데이터 프레임에서는 레이블을 나타내는** i ndex**와** columns **객체를 정의하여 사용한다.**
- index **객체는 행들의 레이블**(label)**이고** columns **객체는 열들의 레이블이 저장된 객체이다.**

	이름	나이	성별	평점
0	김철수	19	Male	3.45
1	김영희	22	Female	4.1
2	김영수	20	Male	3.9
3	최지영	26	Female	4.5

columns

index

- **위의 테이블에서 각 행에 붙은** 0, 1, 2, 3, … **이** index **객체이다.**
- **또 각 열에 붙은 "이름", "나이", … 레이블이** columns **객체이다.**

Section 02 판다스 시작하기

❑ 타이타닉 데이터셋

- 우리가 판다스로 무엇을 할 수 있는지를 조금 구체적으로 이번 절에서 살펴보지 실제 데이터를 적재해서 간단하게 분석해보자.
- 우리는 판다스 튜토리얼 웹페이지에서 타이타닉 탑승자에 대한 데이터셋(titanic.csv)를 다운로드 받을 수 있다.
  - https://pandas.pydata.org/pandasdocs/sta ble/getting _started/
- titanic.csv의 파일은 실제 타이티닉 승객 891명에 대한 데이터를 포함하고 있다.
- 테이블의 각 행은 탑승자 1인의 정보를 나타낸다.
- 열에는 탑승자의 생존 여부를 포힘하여 탑승자에 대한 다양한 정보기 들어있다.

Section 02 판다스 시작하기

❑ 타이타닉 데이터셋

- **타이타닉 데이터셋의 구조**

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
1	0	3	Braund, Mr. Owen Harris	Male	22	1	0
2	1	1	Cumings, Mrs. John Bradley	Female	38	1	0
3	1	3	Heikkinen, Miss. Laina	Female	26	0	0
4	1	1	Futrelle, Mrs. Jacques Heath	Female	35	1	0
5	0	3	Allen, Mr. William Henry	Male	35	0	0

Section 02 판다스 시작하기

❑ 타이타닉 데이터셋

- **타이타닉 데이터셋의 구조**
  - Passengerld: **승객의** ID**이다**
  - Survived: **생존 여부**
  - Pclass: **탑승 등급을 나타낸다**. **클래스** 1, **클래스** 2, **클래스** 3**의** 3**가지 클래스가 있다**.
  - Name: **승객의 이름**.
  - Sex: **승객의 성별**.
  - Age: **승객의 나이**.
  - SibSp: **승객에게 형제 자매와 배우자가 있음을 나타낸다**.
  - Parch: **승객이 혼자인지 또는 가족이 있는지 여부**.
  - Ticket: **승객의 티켓 변호**.
  - Fare: **운임**.
  - Cabin: **승객의 선실**.
  - Embeded: **탑승한 지역**

Section 02 판다스 시작하기

❑ 판다스를 사용하기 위해 제일 먼저 할 일은?

- 판다스를 사용하기 위해 먼저 다음과 같이 판다스 패키지를 읽어 들어야 한다.

```
import pandas as pd
```

- 판다스로 작업을 시작하려면 판다스 패키지를 가져와야 한다.
- 판다스 패키지에 pd라는 별명을 붙이는 것은 업계의 표준 관행이다.

Section 02 판다스 시작하기

❑ 타이타닉 CSV 파일을 읽으려면?

- 판다스는 read_csv()로 csv 파일로 저장된 데이터를 읽는 기능을 제공한다.
- 사실 판다스는 많은 다른 파일 형식을 읽을 수 있다.
- CSV, 엑셀, SQL, JSON 파일을 읽을 수 있다.
- 이들은 모두 read_xxx()와 같은 형태의 함수로 되어 있다.
- 다음 예제는 타이타닉 csv 파일을 읽어서 출력하는 프로그램이다.
- 타이타닉 csv 파일은 이미지를 다운받은 사이트에서 다운로드할 수 있다.
- 또한, 해당 파일은 파이썬 소스 파일이 있는 폴더내의 data 폴더에 위치해야 한다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv('./data/titanic.csv')
4  print(titanic)
```

Section 02 판다스 시작하기

❑ 타이타닉 승객들의 나이를 추출하려면?

- 데이터 프레임에서 열을 선택하려면 대괄호 사이에 열 레이블을 넣으면 된다.
- 이것은 딕셔너리에서 키를 가지고 값을 선택하는 것과 아주 유사하다.
- 데이터 프레임의 특정 열은 시리즈라고 할 수 있다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv('./data/titanic.csv')
4  print(titanic["Age"])
```

```
>>> %Run ex9_2.py
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
5       NaN
6      54.0
7       2.0
8      27.0
9      14.0
10      4.0
11     58.0
12     20.0
```

Section 02 판다스 시작하기

❑ 타이타닉 탑승객 중에서 최고령자를 알고 싶다면?

- **데이터 프레임에서** Age **열을 선택하고** max() **함수를 적용하여서 작업을 수행할 수 있다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv('./data/titanic.csv')
4  print(titanic["Age"].max())
```

```
>>> %Run ex9_3.py
  80.0
>>>
```

Section 02 판다스 시작하기

❑ 타이타닉 승객 데이터에 대한 기본 통계를 알고 싶다면?

- describe() **메소드는 숫자 데이터에 대한 간략한 개요를 제공한다.**
- **문자열 데이터는 처리하지 않는다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv('./data/titanic.csv')
4  print(titanic.describe())
```

```
>>> %Run ex9_4.py
       PassengerId    Survived      ...         Parch        Fare
count   891.000000  891.000000      ...    891.000000  891.000000
mean    446.000000    0.383838      ...      0.381594   32.204208
std     257.353842    0.486592      ...      0.806057   49.693429
min       1.000000    0.000000      ...      0.000000    0.000000
25%     223.500000    0.000000      ...      0.000000    7.910400
50%     446.000000    0.000000      ...      0.000000   14.454200
75%     668.500000    1.000000      ...      0.000000   31.000000
max     891.000000    1.000000      ...      6.000000  512.329200

[8 rows x 7 columns]
```

Section 03 데이터 프레임 생성

❑ 시리즈 생성

- 시리즈는 이름이 붙여진 1차원적인 배열이나 마찬가지이다.
- 가장 기본적인 방법은 파이썬의 리스트에서 생성하는 것이다.

```
>>> import pandas as pd
>>> data = ['Kim', 'Park', 'Lee', 'Choi']
>>> ser = pd.Series(data)
>>> ser
0     Kim
1    Park
2     Lee
3    Choi
dtype: object
>>>
```

Section 03 데이터 프레임 생성

❑ 데이터 프레임 생성

- **데이터 프레임은 행과 열에 이름이 붙여진 2차원 배열이다.**
- **가장 기초적인 생성 방법은 딕셔너리에서 데이터를 읽는 것이다.**
- index**가 주어지지 않으면** 0, 1, 2, .. **등의 숫자가 행에 붙여진다.**

```
>>> data = {'Name':["Kim", "Park", "Lee", "Choi"],
                'Age':[20, 23, 21, 26]}
>>> df = pd.DataFrame(data)
>>> df
   Name  Age
0   Kim   20
1  Park   23
2   Lee   21
3  Choi   26
>>>
```

- **데이터 프레임에** index**를 붙이려면 다음과 같이** index **매개 변수를 사용할 수 있다.**

```
>>> df = pd.DataFrame(data, index=['학번 1', '학번 2', '학번 3', '학번 4'])
>>> df
       Name  Age
학번 1   Kim   20
학번 2  Park   23
학번 3   Lee   21
학번 4  Choi   26
```

Section 03 데이터 프레임 생성

❑ csv 파일을 읽어서 데이터 프레임 생성

- 판다스에서 데이터를 읽는 메소드는 항상 read _xxx()와 같은 형태이고, 반대로 데이터를 파일에 쓰는 메소드는 to_xxx()의 형태를 가진다.
- 예를 들어서 csv 파일을 읽는 메소드는 read_csv()이고 csv 파일을 쓰는 메소드는 to_csv()이다.
- 우리는 이미 앞 절에서 CSV 파일에서 데이터를 읽어서 데이터 프레임에 저장한 바 있다.

```
titanic = pd.read_csv('./data/titanic.csv')
```

Section 03 데이터 프레임 생성

❑ csv 파일을 읽어서 데이터 프레임 생성

- dtypes **속성을 요청하여 각 열의 데이터 유형을 확인할 수 있다.**

```
>>> titanic = pd.read_csv('./data/titanic.csv')
>>> titanic.dtypes

PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Section 03 데이터 프레임 생성

❑ csv 파일을 읽어서 데이터 프레임 생성

- **파일에서 읽을 때** index**를 변경할 수 있다.**
- **예를 들어서 첫 번째 열을** index **객체로 사용할 수도 있다.**

```
>>> titanic = pd.read_csv("./data/titanic.csv", index_col=0)
>>> titanic
             Survived  Pclass   ...            Cabin Embarked
PassengerId                     ...
1                   0       3   ...              NaN        S
2                   1       1   ...              C85        C
3                   1       3   ...              NaN        S
4                   1       1   ...             C123        S
5                   0       3   ...              NaN        S
6                   0       3   ...              NaN        Q
7                   0       1   ...              E46        S
8                   0       3   ...              NaN        S
9                   1       3   ...              NaN        S
10                  1       2   ...              NaN        C
```

컬럼 0을 인덱스로 하겠다는 의미이다.

인덱스가 PassengerId가 됨

Section 03 데이터 프레임 생성

❑ 데이터 프레임의 몇 개 행을 보려면?

- **데이터 프레임의 첫 번째** 8**행을 보려면** head(8)**을 호출한다**.
- **마지막** 8**개의 행을 보려면** tail(8)**을 호출한다**.

```
>>> titanic.head(8)
             Survived  Pclass    ...     Cabin Embarked
PassengerId                      ...
1                   0       3    ...       NaN        S
2                   1       1    ...       C85        C
3                   1       3    ...       NaN        S
4                   1       1    ...      C123        S
5                   0       3    ...       NaN        S
6                   0       3    ...       NaN        Q
7                   0       1    ...       E46        S
8                   0       3    ...       NaN        S

[8 rows x 11 columns]
```

Section 03 데이터 프레임 생성

❑ 데이터 프레임을 엑셀 파일로 저장하려면?

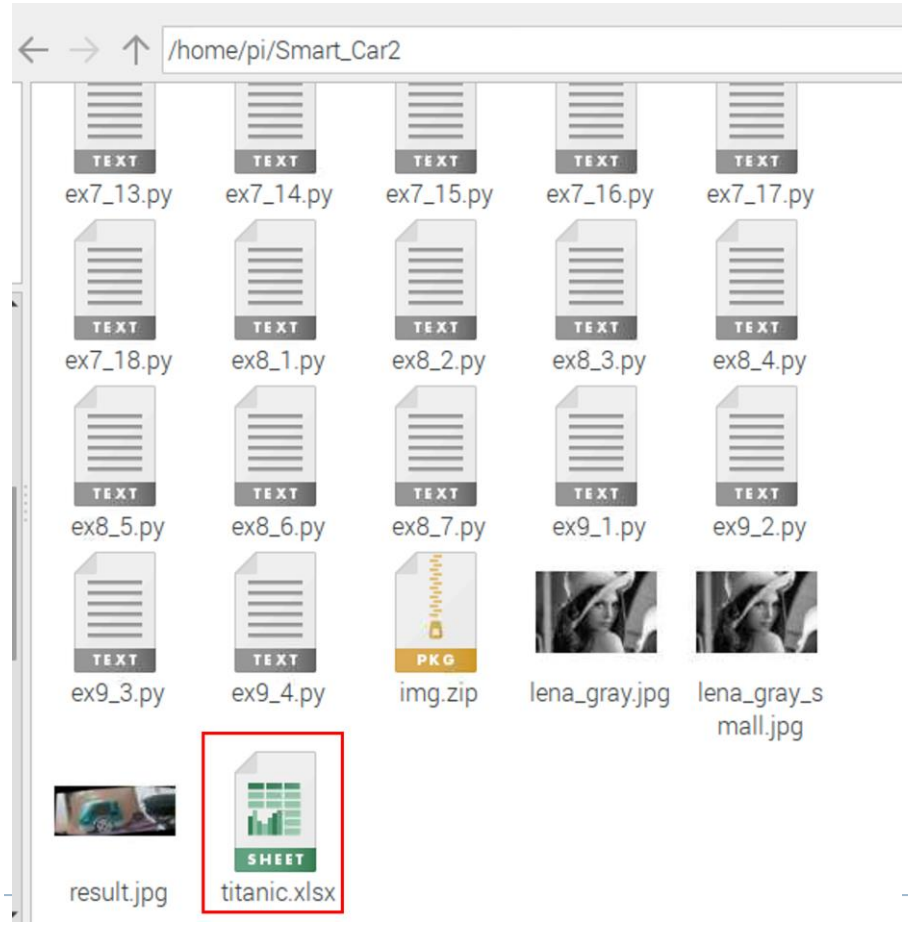
- to_xxx() **메소드는 데이터를 파일로 저장하는 데 사용된다**.
- to_excel()**은 데이터를 엑셀 파일로 저장한다**.
- **여기 예제에서는 기본** Sheet **대신** passengers **이름이 지정된다**.
- index=False**로 하면 데이터 프레임의 인덱스가 스프레드 시트에는 저장되지 않는다**.

```
>>> titanic.to_excel('titanic.xlsx', sheet_name='passengers', index=False)
>>>
```

Section 03 데이터 프레임 생성

❑ 데이터 프레임을 엑셀 파일로 저장하려면?

- /home/pi/Smart_Car2 **디렉토리에 들어가면 생성된 엑셀 파일을 확인할 수 있다.**



Section 03 데이터 프레임 생성

❑ 난수로 데이터 프레임 채우기

- **넘파이의 난수 함수들을 사용하여 데이터 프레임을 채울 수 있다.**

```
>>> import numpy as np
>>> df = pd.DataFrame(np.random.randint(0, 100, size=(5, 4)), columns=list('ABCD'))
>>> df
    A   B   C   D
0  54  46   4  16
1  87  59  15  80
2  17   5  90  77
3  14  39   5  84
4  32  97  45   1
```

- **위의 코드에서** np.random.randint(0, 100, size=(5, 4))**을 호출하여서 크기가** 5 x 4**인** 2**차원 배열을 생성하고 이 배열을** 0**에서** 100 **사이의 난수로 채웠다.**
- **데이터 프레임의** columns **객체는 문자열 'ABCD'를 리스트로 변환한** ['A', 'B', 'C', 'D']**로 지정되었다.**

Section 03 데이터 프레임 생성

❑ 난수로 데이터 프레임 채우기

- **아래와 같이, 국가에 대한 정보가 저장된 파일** countries.csv**를 텍스트 파일로 생성해보자**.
- **이 파일에는 다음과 같이 몇 개 국가에 대한 정보가 저장되어 있다**.
- **우리는 이 파일도 이번 장 내내 사용할 것이다**.

```
code,country,area,capital,population
KR,Korea,98480,Seoul,48422644
US,USA,9629091,Washington,310232863
JP,Japan,377835,Tokyo,127288000
CN,China,9596960,Beijing,1330044000
RU,Russia,17100000,Moscow,140702000
```

- CSV**는 콤마 분리값**(comma separated values)**을 의미한다**.
- **첫 번째 줄은 열의 이름이다**.
- CSV **파일은 데이터셋을 저장하는 데에 많이 사용된다**.

Section 03 데이터 프레임 생성

❑ 난수로 데이터 프레임 채우기

- **이제 데이터를 불러와 보자.**
- **먼저** pandas **패키지를 "pd'라는 이름으로 가져온다.**
- read_csv() **함수를 호출하고** csv **파일의 경로를 전달한다.**

```
1  import pandas as pd
2
3  countries = pd.read_csv('./data/countries.csv')
4  print(countries)
```

```
>>> %Run ex9_5.py
   code country      area     capital  population
0    KR   Korea     98480       Seoul    48422644
1    US     USA   9629091  Washington   310232863
2    JP   Japan    377835       Tokyo   127288000
3    CN   China   9596960     Beijing  1330044000
4    RU  Russia  17100000      Moscow   140702000
```

Section 04 원하는 데이터 선택

❑ 타이타닉 데이터에서 승객의 나이만 추출하려면?

- **데이터 프레임에서 어떤 조건을 주어서 우리에게 필요한 일부 데이터만을 선택할 수 있다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  ages = titanic["Age"]
6  print(ages.head())
```

Shell

```
>>> %Run ex9_6.py
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: Age, dtype: float64
```

Section 04 원하는 데이터 선택

❑ 타이타닉 데이터에서 승객의 나이만 추출하려면?

- 데이터 프레임에서 특정한 열을 선택하려면 관심있는 열의 열 이름을 대괄호 안에 적으면 된다.
- 단일 열을 선택하면 반환되는 객체는 시리즈 형식이 된다.
- shape 속성을 보면 반환되는 시리즈의 크기를 알 수 있다.

```
ages.shape
```

- shape 속성은 행과 열의 개수를 반환한다.
- 시리즈는 1차원이므로 행의 개수만 반환한다.

Section 04 원하는 데이터 선택

❑ 타이타닉 탑승객의 이름, 나이, 성별을 동시에 알고 싶으면?

- 여러 개의 열을 동시에 선택하려면 [...] 안에 열 이름 리스트를 사용한다.
- 내부 대괄호는 파이썬 리스트로서 열 이름들을 저장하고 있지만 외부 대괄호는 이전 예에서 볼 수 있듯이 판다스에서 데이터를 선택하는 데 사용된다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic[["Name", "Age", "Sex"]])
```

Shell

```
>>> %Run ex9_7.py
                                                 Name   Age     Sex
0                             Braund, Mr. Owen Harris  22.0    male
1   Cumings, Mrs. John Bradley (Florence Briggs Th...  38.0  female
2                              Heikkinen, Miss. Laina  26.0  female
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0  female
4                            Allen, Mr. William Henry  35.0    male
5                                    Moran, Mr. James   NaN    male
6                             McCarthy, Mr. Timothy J  54.0    male
7                      Palsson, Master. Gosta Leonard   2.0    male
```

Section 04 원하는 데이터 선택

❑ 20세 미만의 승객만 추리려면?

- **조건을 주어서 특정한 행을 골라내는 것을 필터링(filtering)이라고 한다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4  below_20 = titanic[titanic["Age"] < 20]
5
6  print(below_20.head())
```

Shell

```
>>> %Run ex9_8.py
     PassengerId  Survived  Pclass     ...        Fare Cabin  Embarked
7              8         0       3     ...     21.0750   NaN         S
9             10         1       2     ...     30.0708   NaN         C
10            11         1       3     ...     16.7000    G6         S
14            15         0       3     ...      7.8542   NaN         S
16            17         0       3     ...     29.1250   NaN         Q

[5 rows x 12 columns]
```

Section 04 원하는 데이터 선택

❑ 20세 미만의 승객만 추리려면?

- **조건식을 기반으로 행을 선택하려면 괄호 안에 조건을 사용한다.**
- **괄호 안의 조건** titanic["Age"] < 20**은 열 "Age"의 값이** 20**보다 작은 행이라는 의미이다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4  print(titanic["Age"] < 20)
```

Shell

```
  [5 rows x 12 columns]
>>> %Run ex9_9.py
  0      False
  1      False
  2      False
  3      False
  4      False
  5      False
  6      False
  7       True
```

Section 04 원하는 데이터 선택

❑ 1등석이나 2등석에 탑승한 승객들을 출력하려면?

- **조건식과 유사하게** isin() **함수는 제공된 리스트에 있는 값들이 들어 있는 각 행에 대하여** True**를 반환한다**.
- titanic[" Pclass"].isin([1, 2])**은** Pclass **열이** 1 **또는** 2**인 행을 확인한다**.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic[titanic["Pclass"].isin([1, 2])])
```

Shell

```
>>> %Run ex9_10.py
     PassengerId  Survived  Pclass  ...     Fare  Cabin  Embarked
1              2         1       1  ...  71.2833    C85         C
3              4         1       1  ...  53.1000   C123         S
6              7         0       1  ...  51.8625    E46         S
9             10         1       2  ...  30.0708    NaN         C
11            12         1       1  ...  26.5500   C103         S
15            16         1       2  ...  16.0000    NaN         S
17            18         1       2  ...  13.0000    NaN         S
20            21         0       2  ...  26.0000    NaN         S
21            22         1       2  ...  13.0000    D56         S
```

Section 04 원하는 데이터 선택

❑ 20세 미만의 승객 이름에만 관심이 있다면?

- **판다스는 데이터 프레임에서 어떤 조건을 주어서 행을 선택한 후에 특정 열만 추출할 수 있다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.loc[titanic["Age"] < 20, "Name"])
```

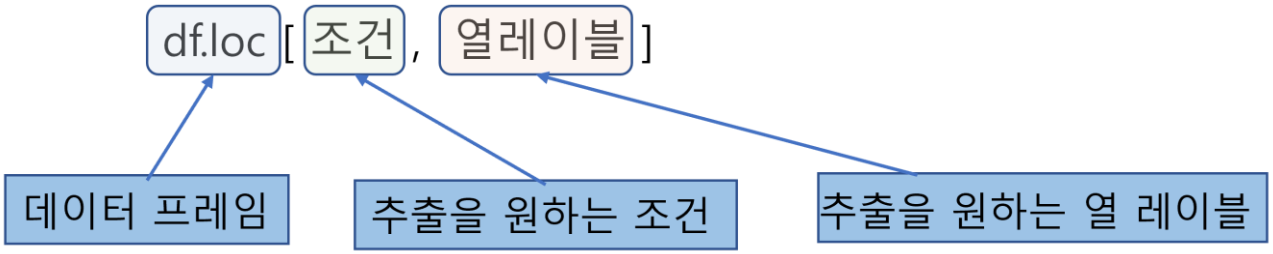
Shell

```
>>> %Run ex9_11.py
  7                     Palsson, Master. Gosta Leonard
  9                  Nasser, Mrs. Nicholas (Adele Achem)
  10                     Sandstrom, Miss. Marguerite Rut
  14                 Vestrom, Miss. Hulda Amanda Adolfina
  16                             Rice, Master. Eugene
  22                        McGowan, Miss. Anna "Annie"
  24                      Palsson, Miss. Torborg Danira
  27                     Fortune, Mr. Charles Alexander
  38                   Vander Planke, Miss. Augusta Maria
```

Section 04 원하는 데이터 선택

❑ 20세 미만의 승객 이름에만 관심이 있다면?

- **이 경우 행과 열의 부분 집합이 한 번에 만들어지므로 괄호만으로는 충분하지 않다.**
- **따라서 이런 경우에는** loc **연산자나** iloc **연산자를 사용할 수 있다.**
- Loc **연산자가 선택 괄호** [ ] **앞에 필요하다.**
- loc**를 사용할 때** loc **쉼표 앞의 부분은 원하는 조건이고 쉼표 뒤의 부분은 선택하려는 열 레이블이다.**



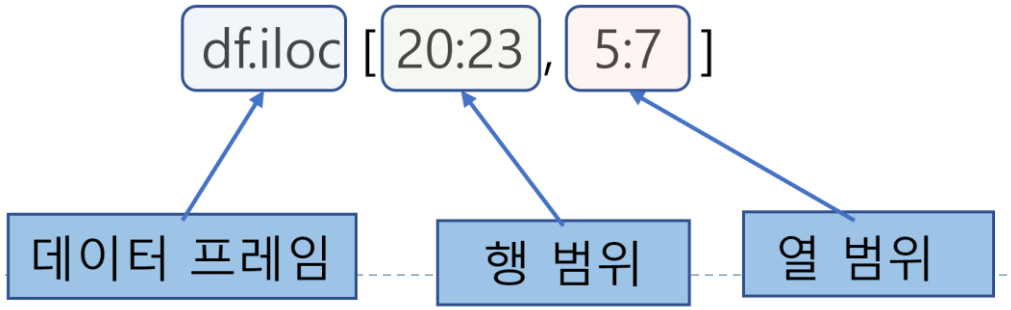
Section 04 원하는 데이터 선택

❑ 20행에서 23행, 5열에서 7열에만 관심이 있다면?

- iloc **연산자는 정수로 된 인덱스 값을 받는다**.
- iloc **연산자를 사용할 때, 쉼표 앞뒤로 슬라이싱을 사용할 수 있다**.
- **슬라이싱을 사용하면 원하는 행 또는 열을 정확하게 선택할 수 있다**.
- **예를 들면 다음과 같다**.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.iloc[20:23, 5:7])
```

```
>>> %Run ex9_12.py
      Age  SibSp
20   35.0      0
21   34.0      0
22   15.0      0
```



Section 04 원하는 데이터 선택

❑ 데이터를 정렬하는 방법

- **승객의 나이에 따라 타이타닉 데이터를 정렬하려면 어떻게 해야 하는가?**

```
1  import pandas as pd
2
3  titanic = pd.read_csv(“./data/titanic.csv”)
4
5  print(titanic.sort_values(by="Age").head()))
```

Shell

```
>>> %Run ex9_13.py
     PassengerId  Survived  Pclass     ...        Fare Cabin  Embarked
803          804         1       3     ...      8.5167   NaN         C
755          756         1       2     ...     14.5000   NaN         S
644          645         1       3     ...     19.2583   NaN         C
469          470         1       3     ...     19.2583   NaN         C
78            79         1       2     ...     29.0000   NaN         S

[5 rows x 12 columns]
```

Section 04 원하는 데이터 선택

❑ 데이터를 정렬하는 방법

- sort_values()를 사용하면 테이블의 행이 지정된 열에 따라 정렬된다.
- 이번에는 승객 클래스와 연령에 따라 타이타닉 데이터를 내림차순으로 정렬해보자.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.sort_values(by=['Pclass', 'Age'], ascending=False).head())
```

Shell

```
>>> %Run ex9_14.py
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
851          852         0       3  ...   7.7750   NaN         S
116          117         0       3  ...   7.7500   NaN         Q
280          281         0       3  ...   7.7500   NaN         Q
483          484         1       3  ...   9.5875   NaN         S
326          327         0       3  ...   6.2375   NaN         S

[5 rows x 12 columns]
```

Section 05 행과 열의 추가나 삭제

❑ 열 추가

- **판다스를 이용하면 다른 열의 정보를 토대로 새로운 열을 생성할 수도 있다.**
- **앞에서 생성하였던** countries.csv **파일을 읽어서 인구 밀도를 나타내는 열을 생성해보자.**
- **인구를 면적으로 나눠주면 된다.**

```
1  import pandas as pd
2
3  countries = pd.read_csv("./data/countries.csv")
4  countries["destiny"] = countries["population"] / countries["area"]
5
6  print(countries)
```

```
>>> %Run ex9_15.py
   code country      area     capital  population     destiny
0    KR   Korea     98480       Seoul    48422644  491.700284
1    US     USA   9629091  Washington   310232863   32.218292
2    JP   Japan    377835       Tokyo   127288000  336.887795
3    CN   China   9596960     Beijing  1330044000  138.590137
4    RU  Russia  17100000      Moscow   140702000    8.228187
```

Section 05 행과 열의 추가나 삭제

❑ 열 추가

- 위의 코드는 인구 밀도를 계산하고 이것을 기존의 데이터 프레임에 추가하였다.
- 결과에서 보다시피 5개국 중에서 우리나라가 가장 인구 밀도가 높다.
- 제곱 킬로미터 당 491명으로 계산되었다.
- 러시아는 제곱 킬로미터 당 8명으로 계산된다.
- 위의 코드에서 알 수 있듯이, 새로운 열을 추가하려면 데이터 프레임에 대괄호를 하고 새로운 열의 이름을 써주면 된다.
- 이제 데이터 프레임은 5개의 열을 가지게 된다.
- 우리는 각 행을 반복하기 위해 루프를 사용할 필요가 없다.
- 판다스 객체에 산술 연산을 하면 값 계산은 요소마다 적용된다.
- Countries["population"]/countries["area"]는 "population" 열의 모든 값을 "area" 열로 나눈 것을 의미한다.

Section 05 행과 열의 추가나 삭제

❑ 행 추가

- **데이터 프레임에 행을 추가하려면** append()**를 호출한다.**
- **예를 들어서 우리의 "**countries.csv"**에 캐나다에 대한 정보를 추가하려면 다음과 같이 한다.**

```
1  import pandas as pd
2
3  countries = pd.read_csv("./data/countries.csv")
4  df = pd.DataFrame({"code":["CA"], "country":["Canada"], "area":[9984670], "capital":["
   Ottawa"], "population":[34300000]})
5  df2 = countries.append(df, ignore_index = True)
6
7  print(df2)
```

```
>>> %Run ex9_16.py
   code country       area     capital  population
0    KR   Korea      98480       Seoul    48422644
1    US     USA    9629091  Washington   310232863
2    JP   Japan     377835       Tokyo   127288000
3    CN   China    9596960     Beijing  1330044000
4    RU  Russia   17100000      Moscow   140702000
5    CA  Canada    9984670      Ottawa    34300000
```

Section 05 행과 열의 추가나 삭제

❑ 행 삭제

- **행을 삭제하려면** drop()**을 호출한다**.
- drop(index=2)**하면 번호가** 2**인 행이 삭제된다**.
- **인덱스가 문자열로 되어 있으면 문지열로 지정하여야 한다**.
- **여기서** inplace=True**는 현재 객체를 수정하고 반환하지는 말라는 의미이다**.

```
1  import pandas as pd
2
3  countries = pd.read_csv("./data/countries.csv")
4  df = pd.DataFrame({"code":["CA"], "country":["Canada"], "area":[9984670], "capital":["
   Ottawa"], "population":[34300000]})
5  df2.drop(index=2, axis=0, inplace=True)
6
7  print(df2)
```

```
>>> %Run ex9_17.py
   code country      area     capital  population
0    KR   Korea     98480       Seoul    48422644
1    US     USA   9629091  Washington   310232863
3    CN   China   9596960     Beijing  1330044000
4    RU  Russia  17100000      Moscow   140702000
5    CA  Canada   9984670      Ottawa    34300000
```

Section 05 행과 열의 추가나 삭제

❑ 열 삭제

- **열을 삭제하려면** axis=1**로 지정하고 삭제하고자 하는 열의 이름들을 리스트로 만들어서 전달한다**.
- **예를 들어서** "countries.csv"**에서** "capital" **열을 삭제하려면 다음과 같이 호출한다**.

```
1  import pandas as pd
2
3  countries = pd.read_csv("./data/countries.csv")
4  df = pd.DataFrame({"code":["CA"], "country":["Canada"], "area":[9984670], "capital":["Ottawa"],
   "population":[34300000]})
5  df2.drop(["capital"], axis=1, inplace=True)
6
7  print(df2)
```

```
>>> %Run ex9_18.py
   code country      area  population
0    KR   Korea     98480    48422644
1    US     USA   9629091   310232863
2    JP   Japan    377835   127288000
3    CN   China   9596960  1330044000
4    RU  Russia  17100000   140702000
5    CA  Canada   9984670    34300000
```

Section 06 데이터 통계

❑ 타이타닉 승객의 평균 연령은 얼마인가?

- **타이타닉 데이터에서 "Age" 열을 추출한 후에** mean() **함수를 적용하면 된다.**
- Mean()**은 평균 값을 계산하는 메소드로서 숫자 데이터가 있는 열에만 적용할 수 있다.**
- **일반적으로 누락된 데이터를 제외하고 기본적으로 행 전체에서 작동한다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic["Age"].mean())
```

```
>>> %Run ex9_19.py
  29.69911764705882
```

Section 06 데이터 통계

❑ 타이타닉 승객 연령과 탑승권 요금의 중간 값은 얼마일까?

- **여러 열에 대한 통계를 계산할 수도 있다.**
- **예를 들어서 타이타닉 승객 연령의 중간값과 탑승권 요금의 중간값을 알고 싶으면** ["'Age", "Fare"]**으로 선택한 후에** median()**을 호출한다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic[["Age", "Fare"]].median())
```

```
>>> %Run ex9_20.py
Age      28.0000
Fare     14.4542
dtype: float64
```

Section 06 데이터 통계

❑ 카테고리별로 그룹화된 통계

- 우리는 어떤 기준에 의하여 데이터들을 그룹화하고 이 그룹에 대한 통계값을 계산할 수도 있다.
- 예를 들면 타이타닉 데이터에서 남성 대 여성 승객의 평균 연령을 알고 싶은 경우가 있다.
- 이때는 groupby() 함수를 사용한다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv(“./data/titanic.csv”)
4
5  print(titanic[["Sex", “Age”]].groupby(“Sex”).mean())
```

```
>>> %Run ex9_21.py
Age     29.699118
dtype: float64
```

Section 06 데이터 통계

❑ 카테고리별로 그룹화된 통계

- **우리의 관심은 각 성별의 평균 연령이므로** titanic[["Sex", "Age"]]**에 의하여 이 두 열의 선택이 먼저 이루어진다**.
- **다음으로**, groupby() **메소드가 "Sex" 열에 적용되어 "Sex" 값에 따라서 그룹을 만든다**.
- **이어서 각 성별의 평균 연령이 계산되어 반환된다**.
- **이전 예에서는** 2**개의 열을 먼저 명시적으로 선택했다**.
- **아무 열도 선택하지 않으면** groupby()**는 숫자 열이 포함된 각 열에 적용된다**.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.groupby("Sex").mean())
```

```
>>> %Run ex9_22.py
        PassengerId  Survived  ...      Parch       Fare
Sex                            ...
female   431.028662  0.742038  ...   0.649682  44.479818
male     454.147314  0.188908  ...   0.235702  25.523893

[2 rows x 7 columns]
```

Section 06 데이터 통계

❑ 카테고리별로 그룹화된 통계

- 생각해보면 Pclass의 평균값을 얻는 것은 의미가 없다.
- 각 성별의 평균 연령에만 관심이 있는 경우, 그룹화된 데이터에서도 열 선택이 지원된다.
- 평소처럼 [ ]를 사용하면 된다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.groupby("Sex")["Age"].mean())
```

```
>>> %Run ex9_23.py
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```

Section 06 데이터 통계

❑ 성별 및 승객 등급 조합의 평균 탑승권 요금은 얼마인가?

- **판다스는 여러 열을 동시에 그룹화할 수 있다.**
- **이때는 열 이름을** groupby()**에 리스트 형태로 제공한다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4
5  print(titanic.groupby(["Sex", "Pclass"])['Fare'].mean())
```

```
>>> %Run ex9_24.py
Sex     Pclass
female  1         106.125798
        2          21.970121
        3          16.118810
male    1          67.226127
        2          19.741782
        3          12.661633
Name: Fare, dtype: float64
```

Section 06 데이터 통계

❑ 각 승객 등급의 수는 몇 명인가?

- value_counts() **메서드는 열의 각 카테고리에 대한 행의 수를 계산한다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv(“./data/titanic.csv”)
4
5  print(titanic["Pclass"].value_counts())
```

```
>>> %Run ex9_23.py
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

Section 07 데이터로 차트 그리기

❑ 개요

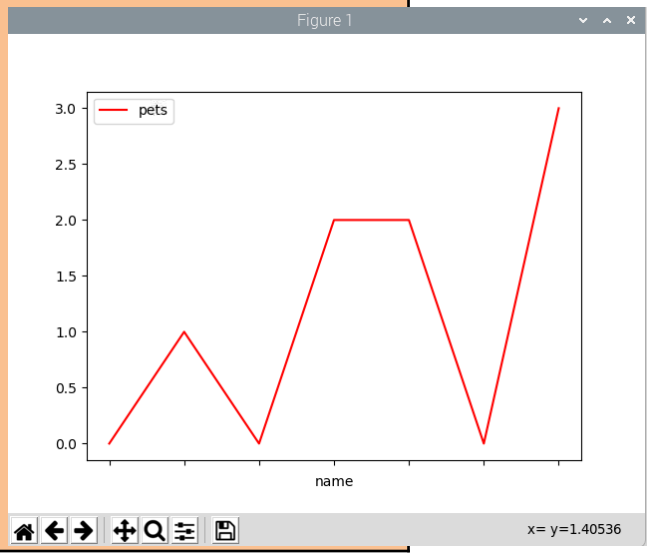
- **판다스에서는** matplotlib **라이브러리를 이용하여 여러 가지 형태의 차트를 그릴 수 있다**.
- **판다스에서 데이터 프레임에 대하여** plot()**만 호출하면 자동으로 차트가 그려진다**.
- **판다스에서** plot()**을 사용할 때, 다음과 같은 경우가 있다**.
  - df.plot()**와 같이 호출하면 인덱스에 대하여 모든 열을 그린다**.
  - df.plot(x='col1')**와 같이 호출하면 하나의 열만을 그린다**.
  - df.plot(x='col1', y='col2')**와 같이 호출하면 특정 열에 대하여 다른 열을 그리게 된다**.

Section 07 데이터로 차트 그리기

❑ 개요

- **딕셔너리를 사용하여 다음과 같은 데이터를 생성한다.**
- **데이터 프레임에 들어 있는** plot() **메소드를 사용하여서 데이터를 직접 그릴 수 있다.**
- Plot()**은** plt.plot()**의 단순한 래퍼 함수이다.**

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  df.plot(kind='line', x = 'name', y = 'pets', color='red')
7  plt.show()
```

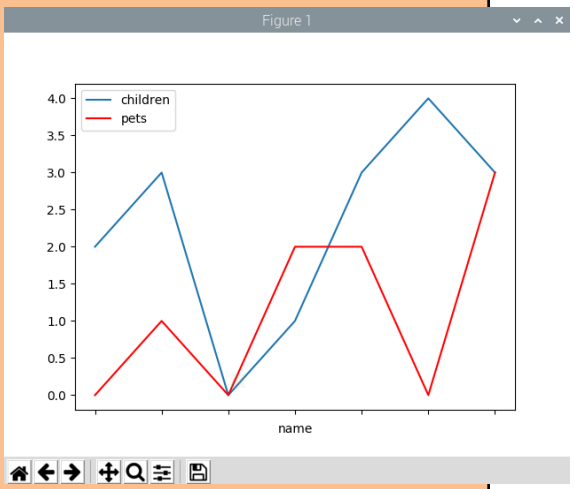


Section 07 데이터로 차트 그리기

❑ 중첩 차트 그리기

- **하나의 차트에 여러 개의 데이터를 중첩해서 그리려면** gca() **함수를 사용한다.**

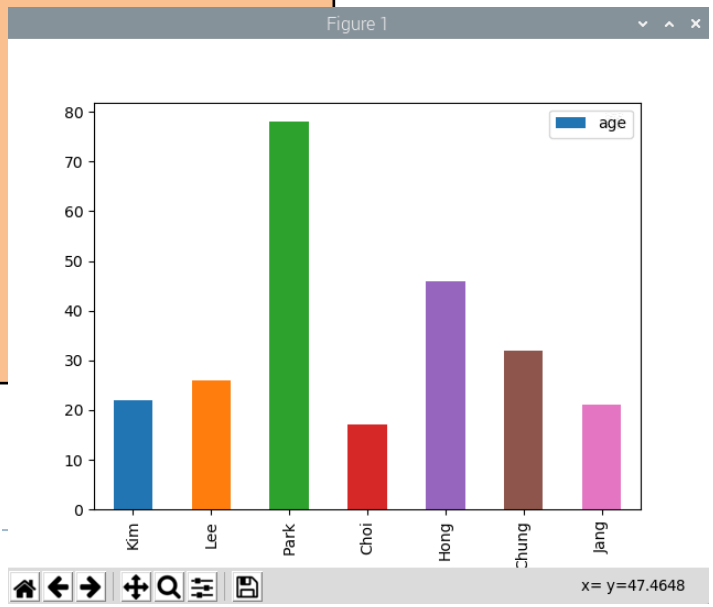
```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  ax = plt.gca()
7
8  df.plot(kind='line', x = 'name', y = 'children', ax=ax)
9  df.plot(kind='line', x = 'name', y = 'pets', color="red", ax=ax)
10
11 plt.show()
```



Section 07 데이터로 차트 그리기

❑ 막대 그래프 그리기

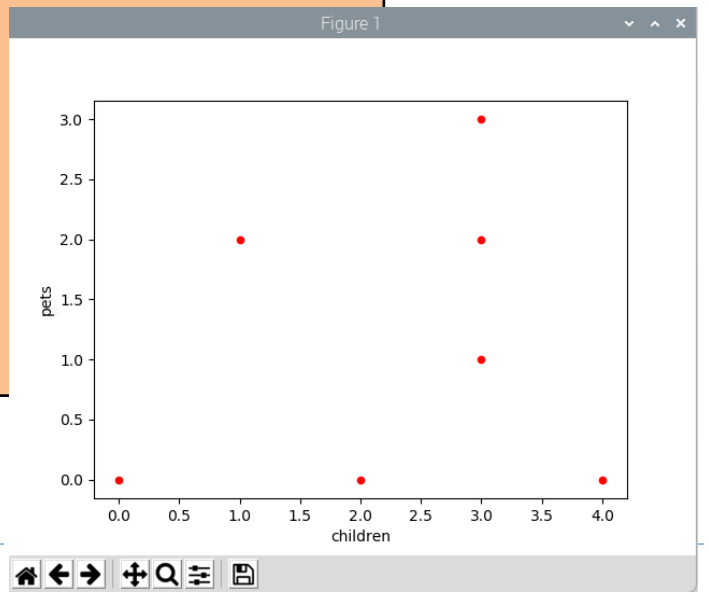
```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  df.plot(kind='bar', x = 'name', y = 'age')
7
8  plt.show()
```



Section 07 데이터로 차트 그리기

❑ 산포도 그리기

```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  df.plot(kind='scatter', x = children', y = 'pets', color=’red’)
7
8  plt.show()
```

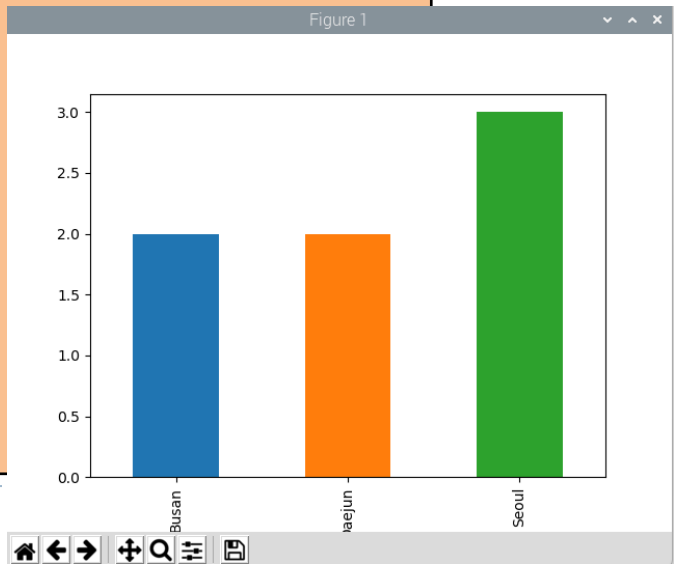


Section 07 데이터로 차트 그리기

❑ 그루핑하여 그리기

- **이번에는 groupby('city')를 사용하여서 도시 별로 그룹핑한 후에 유일한 이름의 개수를 계산하여서 막대 그래프로 그려보자.**

```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  df.groupby('city')['name'].nunique().plot(kind='bar')
7
8  plt.show()
```

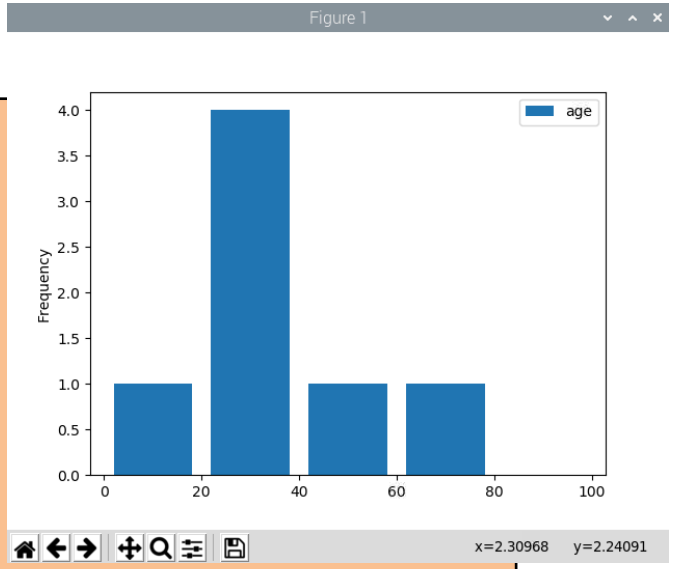


Section 07 데이터로 차트 그리기

❑ 히스토그램 그리기

- 데이터 프레임에서 나이를 선택한 후에 나이별로 몇 명이나 있는지를 그려보자.
- 히스토그램 기능을 사용하면 쉽다.

```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4  df = pd.DataFrame({
       'name':['Kim','Lee', 'Park', 'Choi', 'Hong', 'Chung', 'Jang'],
       'age':[22, 26, 78, 17, 46, 32, 21],
       'city':['Seoul', 'Busan', 'Seoul', 'Busan', 'Seoul', 'Daejun', 'Daejun'],
       'children':[2, 3, 0, 1, 3, 4, 3],
       'pets':[0, 1, 0, 2, 2, 0, 3]
   })
5
6  df[['age']].plot(kind='hist', bins=[0, 20, 40, 60, 80, 100], rwidth=0.8)
7
8  plt.show()
```



Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 피벗 테이블

- 피벗 테이블을 이용하면 데이터를 아주 빨리 분석할 수 있다.
- 즉 마우스 클릭 한 번으로 특정 제품 카테고리에 대한 높은 수준의 개요를 얻을 수 있다.
- 피벗 테이블은 데이터 과학자한테 아주 중요한 도구이다.
- 데이터 과학자들은 프로젝트의 탐색적 데이터 분석 단계에서 피벗 테이블을 많이 사용한다.
- 엑셀 사용자는 이러한 피벗 테이블에 대해 잘 알고 있다.
- 피벗 테이블은 엑셀에서 가장 많이 사용되는 기능이다.
- 판다스에서도 이러한 피벗 테이블을 만들 수 있다.
- 판다스 라이브러리는 값을 깔끔한 2차원 테이블로 요약한 pivot_table() 이라는 함수를 제공한다.

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 피벗 테이블

- **다음 예제는 파이썬에서 피벗 테이블을 만드는 코드이다.**

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
5
6  print(titanic.head())
```

```
>>> %Run ex9_31.py
   Survived  Pclass     Sex   Age  SibSp  Parch     Fare Cabin Embarked
0         0       3    male  22.0      1      0   7.2500   NaN        S
1         1       1  female  38.0      1      0  71.2833   C85        C
2         1       3  female  26.0      0      0   7.9250   NaN        S
3         1       1  female  35.0      1      0  53.1000  C123        S
4         0       3    male  35.0      0      0   8.0500   NaN        S
```

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 피벗 테이블에서 인덱스를 사용하여 데이터를 그룹화하자

- pivot_table(data, index)**에는** data**와** index **매개 변수가 필요하다**.
- data**는 함수에 전달하는 데이터 프레임이다**.
- **매개 변수** index**는 데이터를 그룹화할 수 있는 열이다**.
- Index**는 결과 테이블에 인덱스로 나타난다**.
- **우리는 성별을 나타내는** "Sex" **열을** index**로 사용해보자**.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
5  table = pd.pivot_table(data=titanic, index=['Sex'])
6
7  print(table)
```

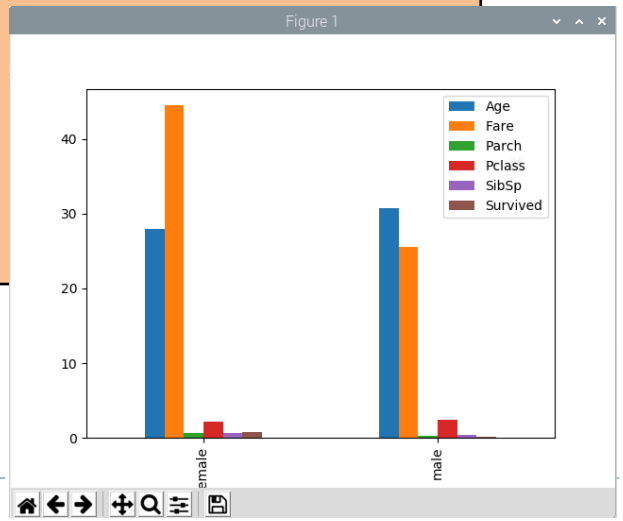
```
>>> %Run ex9_32.py
              Age       Fare     Parch    Pclass     SibSp  Survived
Sex
female  27.915709  44.479818  0.649682  2.159236  0.694268  0.742038
male    30.726645  25.523893  0.235702  2.389948  0.429809  0.188908
```

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 피벗 테이블에서 인덱스를 사용하여 데이터를 그룹화하자

- 우리는 이들 특징값들을 남성과 여성으로 분리하여서 비교할 수 있다.
- 이것을 그래프로 그리려면 다음과 같이 plot()만 붙이면 된다.

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  titanic = pd.read_csv("./data/titanic.csv")
5  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
6  table = pd.pivot_table(data=titanic, index=['Sex'])
7
8  table.plot(kind='bar')
9  plt.show()
```



Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 다중 인덱스로 피봇

- 하나 이상의 열을 인덱스로 사용하여 데이터를 그룹화할 수도 있다.
- 결과 테이블의 세분성 수준이 높아지고 결과를 보다 구체적으로 알 수 있다.

```
1  import pandas as pd
2
3  titanic = pd.read_csv("./data/titanic.csv")
4  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
5  table = pd.pivot_table(data=titanic, index=['Sex', 'Pclass'])
6
7  print(table)
```

```
>>> %Run ex9_34.py
                      Age        Fare     Parch     SibSp  Survived
Sex    Pclass
female 1        34.611765  106.125798  0.457447  0.553191  0.968085
       2        28.722973   21.970121  0.605263  0.486842  0.921053
       3        21.750000   16.118810  0.798611  0.895833  0.500000
male   1        41.281386   67.226127  0.278689  0.311475  0.368852
       2        30.740707   19.741782  0.222222  0.342593  0.157407
       3        26.507589   12.661633  0.224784  0.498559  0.135447
```

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 특징별로 다른 집계 함수 적용

- **피봇 테이블에 표시된 값은** aggfunc**가 데이터에 적용한 요약 결과이다**.
- aggfunc**는** pivot_table**이 그룹화된 데이터에 적용하는 집계 함수이다**.
- **기본적으로** np.mean() **이지만 특정 데이터마다 다른 집계 함수를 사용할 수 있다**.
- **특정 이름을 키로 사용하고 해당 집계 함수를 값으로 사용하여** aggfunc **매개 변수에 대한 입력으로 딕셔너리를 생성하여 전달한다**.
- "Age" **데이터에는** np.mean**을 적용하고**, "Survived" **데이터에는** np. Sum()**을 적용한다**.

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 특징별로 다른 집계 함수 적용

```
1  import pandas as pd
2  import numpy as np
3
4  titanic = pd.read_csv("./data/titanic.csv")
5  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
6  table = pd.pivot_table(data=titanic, index=['Sex', 'Pclass'],
                          aggfunc={'Age':np.mean, 'Survived':np.sum})
7
8  print(table)
```

```
>>> %Run ex9_35.py
                      Age  Survived
Sex    Pclass
female 1        34.611765        91
       2        28.722973        70
       3        21.750000        72
male   1        41.281386        45
       2        30.740707        17
       3        26.507589        47
```

Section 08 테이블의 레이아웃을 바꾸는 방법

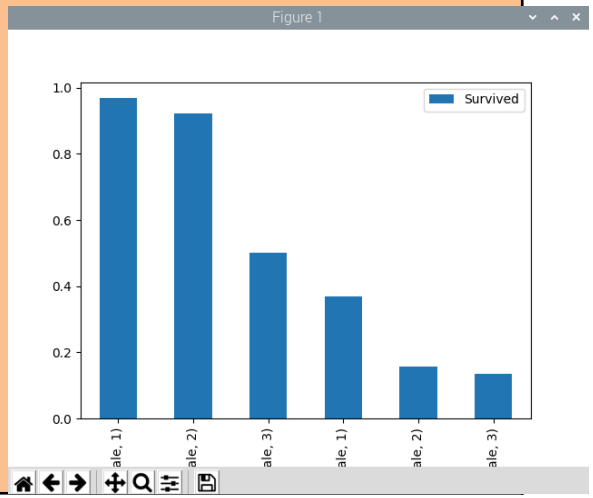
❑ value 매개 변수를 사용하며 특정한 데이터에 대한 집계

- value **매개 변수에서 판다스에게 집계 함수를 적용할 데이터를 지정할 수 있다.**
- value**는 선택 필드이며 이 값을 지정하지 않으면 집계 함수는 데이터셋의 모든 숫자 데이터에 대해 집계된다.**

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  titanic = pd.read_csv("./data/titanic.csv")
6  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
7  table = pd.pivot_table(data=titanic, index=['Sex', 'Pclass'],
                          values=['Survived'], aggfunc=np.mean)
8
9  print(table)
10 table.plot(kind='bar')
11 Plt.show()
```

```
>>> %Run ex9_36.py

                 Survived
Sex    Pclass
female 1         0.968085
       2         0.921053
       3         0.500000
male   1         0.368852
       2         0.157407
       3         0.135447
```



Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 데이터 간의 관계 찾기

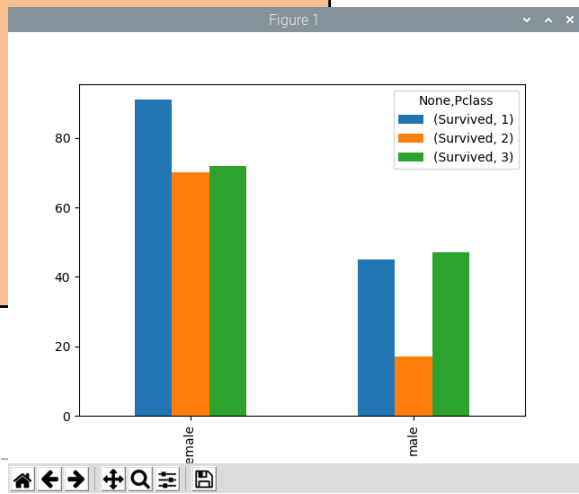
- **일부 특징값을 열로 사용하면 특징 간의 관계를 직관적으로 이해하는 데 도움이 된다.**
- Columns **매개 변수는 선택 사항이며 결과 테이블 상단에 값을 가로로 표시한다.**
- colums**와** index **매개 변수는 선택 사항이지만 이를 효과적으로 사용하면 직관적으로 특징값 사이의 관계를 이해하는 데 도움이 된다.**

Section 08 테이블의 레이아웃을 바꾸는 방법

❑ 데이터 간의 관계 찾기

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  titanic = pd.read_csv("./data/titanic.csv")
6  titanic.drop(['PassengerId', 'Ticket', 'Name'], inplace=True, axis=1)
7  table = pd.pivot_table(data=titanic, index=['Sex'],
                          columns=['Pclass'], values=['Survived'], aggfunc=np.sum)
8
9  print(table)
10 table.plot(kind='bar')
11 Plt.show()
```

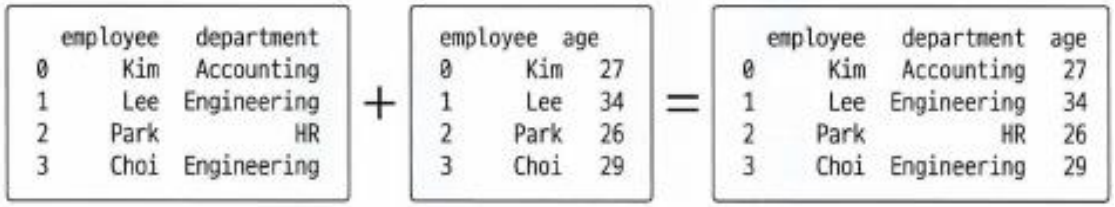
```
>>> %Run ex9_37.py
        Survived
Pclass         1   2   3
Sex
female        91  70  72
male          45  17  47
```



Section 09 데이터 병합

❑ Merge()

- Merge()는 3가지 방법 중에서도 가장 유연한 방법이다.
- Merge()를 사용하면 데이터 베이스의 조인(join) 연산을 할 수 있다.
- 예를 들어보자.
- 어떤 회사의 정보가 다음과 같이 2개의 데이터 프레임에 나누어서 저장되어 있다고 하자.
- 이 2개의 테이블을 병합해보자.



	employee	department
0	Kim	Accounting
1	Lee	Engineering
2	Park	HR
3	Choi	Engineering

+

	employee	age
0	Kim	27
1	Lee	34
2	Park	26
3	Choi	29

=

	employee	department	age
0	Kim	Accounting	27
1	Lee	Engineering	34
2	Park	HR	26
3	Choi	Engineering	29

- 이 2개의 테이블은 공통적인 열(employee)을 가지고 있다.

Section 09 데이터 병합

❑ Merge()

- 2개의 데이터 프레임을 딕셔너리로부터 생성하여서 하나로 합쳐보자.
- Merge() 함수를 사용한다.

```
1  import pandas as pd
2
3  df1 = pd.DataFrame({'employee':['Kim', 'Lee', 'Park', 'Choi'],
       'department':['Accounting', 'Engineering', 'HR', 'Engineering']})
4  df2 = pd.DataFrame({'employee':['Kim', 'Lee', 'Park', 'Choi'],
                       'age':[27, 34, 26, 29]})
5  df3 = pd.merge(df1, df2)
6  print(df3)
```

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run ex9_38.py
   employee   department  age
0       Kim   Accounting   27
1       Lee  Engineering   34
2      Park           HR   26
3      Choi  Engineering   29
```

Section 10 데이터 정제

❑ 결손값 삭제하기

- 실제 데이터셋들은 완벽하지 않다.
- 즉 상당한 수의 결손값을 가지고 있거나 의심스러운 값을 가지고 있다.
- 따라서 데이터를 처리하기 전에 반드시 거쳐야 하는 절차가 데이터 정제이다.
- 판다스에서는 결손값을 NaN으로 나타낸다.
- 판다스는 결손값을 탐지하고 수정하는 함수를 제공한다.
- 왜 결손값이 발생할까?
- 데이터가 아예 수집되지 않았을 수도 있다.
- 예를 들어서 6.25 사변 때는 데이터를 수집할 수 없을 것이다.
- 또 수집은 되었지만 적절하지 않아서 버렸을 수도 있다.
- 데이터를 처리하기 전에 결손값을 처리하지 않으면 어떠한 데이터 분석도 불가능하다.
- 따라서 우리는 결손값을 삭제하거나 다른 값으로 교체하여야 한다.

Section 10 데이터 정제

❑ 결손값 삭제하기

- 결손값이 있는 데이터셋을 다음과 같이 생성하자.
- 인도의 면적 데이터가 누락되어 있다.

```
code,country,area,capital,population
KR,Korea,98480,Seoul,48422644
US,USA,9629091,Washington,310232863
JP,Japan,377835,Tokyo,127288000
CN,China,9596960,Beijing,1330044000
RU,Russia,17100000,Moscow,140702000
IN,India,,New Delhi,1368737513
```

면적 데이터가 누락되어 있다.

Section 10 데이터 정제

❑ 결손값 삭제하기

- 다음과 같이 CSV파일을 읽는다.
- 결손값을 다루는 가장 간단한 방법은 결손값을 가진 행을 삭제하는 것이다.
- 판다스에서 dropna() 함수를 이용하여 삭제할 수 있다.
- how="any"이라고 하면 결손 데이터를 하나라도 가지고 있으면 해당 행을 삭제하고 정제된 데이터 프레임의 복사본을 반환한다.
- inplace=True이라고 하면 원본 데이터 프레임이 수정된다.

```
1  import pandas as pd
2
3  df = pd.read_csv('./data/countries1.csv', index_col=0)
4
5  print(df.dropna(how="any"))
```

```
>>> %Run ex9_39.py
      country        area     capital  population
code
KR      Korea     98480.0       Seoul    48422644
US        USA   9629091.0  Washington   310232863
JP      Japan    377835.0       Tokyo   127288000
CN      China   9596960.0     Beijing  1330044000
RU     Russia  17100000.0      Moscow   140702000
```

Section 10 데이터 정제

❑ 결손값 보정하기

- 결손값을 다루는 또 하나의 방법은 결손값를 다른 깨끗한 값으로 교체하는 것이다.
- fillna() 함수를 사용하면 특정한 값을 다른 값으로 교체한다.
- 예를 들어서 앞의 csv 파일에서 누락된 값을 0으로 교체해보자.

```
1  import pandas as pd
2
3  df = pd.read_csv('./data/countries1.csv', index_col=0)
4  df_0 = df.fillna(0)
5
6  print(df_0)
```

```
>>> %Run ex9_40.py
      country        area     capital  population
code
KR      Korea     98480.0       Seoul    48422644
US        USA   9629091.0  Washington   310232863
JP      Japan    377835.0       Tokyo   127288000
CN      China   9596960.0     Beijing  1330044000
RU     Russia  17100000.0      Moscow   140702000
IN      India         0.0   New Delhi  1368737513
```

Q&A