기초 인공지능 프로그래밍

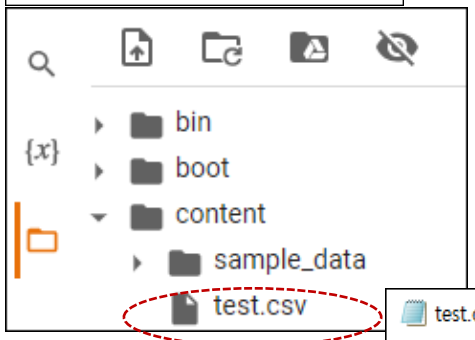
12장 Pandas 라이브러리2

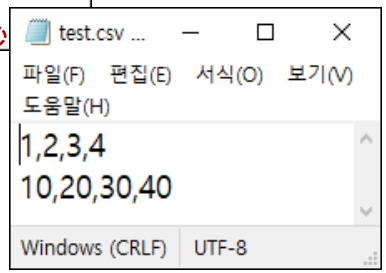
외부 데이터 파일 읽기(CSV 파일)

- pandas는 CSV 파일, 엑셀 파일등 다양한 형식의 데이터 파일을 불러 올 수 있음
- CSV 파일(확장자 csv)을 읽어 DataFrame 객체를 만들어 반환하는 함수
  - **read_csv("파일이름")**
  - CSV 파일은 쉼표(,)로 데이터를 구분

```
%%writefile test.csv
1,2,3,4
10,20,30,40
Writing test.csv
```

구글 서버 접속 위치인 content 하위에 test.csv 파일 생성






```
import pandas as pd
pf1 = pd.read_csv("test.csv")
print(pf1)
print(pf1["3"])
print()
pf2 = pd.read_csv("test.csv", header = None)
print(pf2)
print(pf2[3])
```

```
    1   2   3   4
0  10  20  30  40
0    30
Name: 3, dtype: int64

    0   1   2   3
0   1   2   3   4
1  10  20  30  40
0     4
1    40
Name: 3, dtype: int64
```

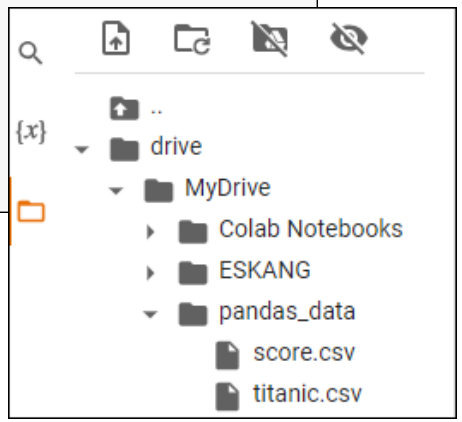
pandas 함수

- DataFrame의 앞 뒤 일부 데이터를 확인할 때
  - df.head(n) : 처음 n개의 행 데이터 반환 (n 지정하지 않으면 5개)
  - df.tail(n) : 끝에서 n개의 데이터 반환 (n 지정하지 않으면 5개)
- DataFrame의 데이터 정보 확인할 때
  - df.info()
  - 열(column) 정보, 몇 개의 값이 있는지, 어떤 형태로 저장되어 있는지 확인 가능
- DataFrame의 데이터 숫자 세기
  - df[“column name”].value_counts()

pandas 함수

```
import pandas as pd
pf = pd.read_csv("/content/drive/MyDrive/pandas_data/score.csv",index_col = 0)
print(pf, "₩n")
print(pf.info(), "₩n")
print(pf.head(3), "₩n")
print(pf.tail(), "₩n")
print(pf["Jessy"].value_counts())
```

0번째 열을 인덱스로 설정

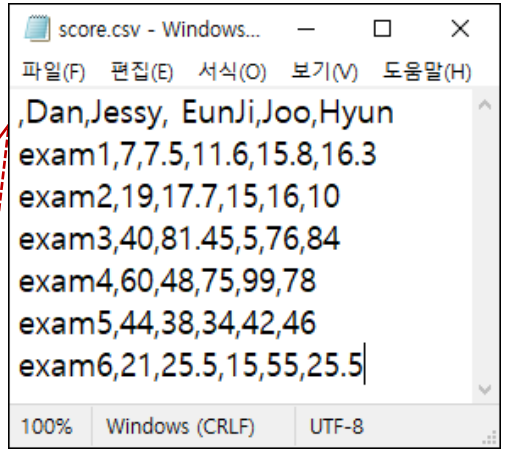


첫번째 행을 컬럼 이름으로 설정

첫번째 열을 인덱스로 설정

```
      Dan  Jessy  EunJi   Joo  Hyun
exam1   7   7.50   11.6  15.8  16.3
exam2  19  17.70   15.0  16.0  10.0
exam3  40  81.45    5.0  76.0  84.0
exam4  60  48.00   75.0  99.0  78.0
exam5  44  38.00   34.0  42.0  46.0
exam6  21  25.50   15.0  55.0  25.5
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, exam1 to exam6
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Dan     6 non-null      int64
 1   Jessy   6 non-null      float64
 2    EunJi  6 non-null      float64
 3   Joo     6 non-null      float64
 4   Hyun    6 non-null      float64
dtypes: float64(4), int64(1)
memory usage: 288.0+ bytes
None
```

```
      Dan  Jessy  EunJi   Joo  Hyun
exam1   7   7.50   11.6  15.8  16.3
exam2  19  17.70   15.0  16.0  10.0
exam3  40  81.45    5.0  76.0  84.0

      Dan  Jessy  EunJi   Joo  Hyun
exam2  19  17.70   15.0  16.0  10.0
exam3  40  81.45    5.0  76.0  84.0
exam4  60  48.00   75.0  99.0  78.0
exam5  44  38.00   34.0  42.0  46.0
exam6  21  25.50   15.0  55.0  25.5

7.50     1
17.70    1
81.45    1
48.00    1
38.00    1
25.50    1
Name: Jessy, dtype: int64
```



첫번째 행, 첫번째 열을 인덱스로 사용.
따라서 첫번째 행의 첫번째 데이터 생략

pandas 함수(데이터 변환)

- 데이터는 항상 원하는 자료형으로 저장되어 있지 않음
- 데이터 정보를 확인해서 원하는 자료형으로 변환을 해야하는 경우가 있음
- pf["column name"] = pf["column name"].astype(toDataType)
  - 해당 열의 자료형을 원하는 자료형으로 변환
  - astype(int)이면 해당 열의 데이터들을 정수형으로 변환된 DataFrame(Series)을 반환
- pf["column name"] = pf.to_numeric(df["column name"])
  - 해당 열의 자료형을 숫자 자료형으로 변환

```
import pandas as pd
pf = pd.read_csv("/content/drive/MyDrive/pandas_data/score.csv",index_col = 0)
pf["Dan"] = pf["Dan"].astype(float)
print(pf, "₩n")
for c in pf.columns :
    pf[c] = pf[c].astype(int)
print(pf, "₩n")
```

반복문을 사용해서 열들의 자료형을 한번에 바꿀 수도 있음

```
       Dan  Jessy  EunJi   Joo  Hyun
exam1  7.0   7.50   11.6  15.8  16.3
exam2 19.0  17.70   15.0  16.0  10.0
exam3 40.0  81.45    5.0  76.0  84.0
exam4 60.0  48.00   75.0  99.0  78.0
exam5 44.0  38.00   34.0  42.0  46.0
exam6 21.0  25.50   15.0  55.0  25.5

       Dan  Jessy  EunJi  Joo  Hyun
exam1    7      7     11   15    16
exam2   19     17     15   16    10
exam3   40     81      5   76    84
exam4   60     48     75   99    78
exam5   44     38     34   42    46
exam6   21     25     15   55    25
```

pandas 함수(데이터 변환)

- DataFrame의 각 열은 Series이며 연산이 가능
- 열에 대한 다양한 연산을 활용해서 데이터를 변환, 또는 새로운 열을 생성할 수 있음
- pf["column name"] = pf.sum(axis = 0 (or 1))
  - 인자 axis에 따라서 합을 구해서 해당 컬럼에 저장
- pf["column name"].apply(funcName, axis = 0 (or 1))
  - 해당 열의 데이터를 funcName으로 지정된 함수의 인자로 전달
  - 함수의 결과 값을 사용

```
import pandas as pd
def grade(n):
    if n >= 300 :
        return "Excellent"
    elif n >= 200 :
        return "Good"
    else :
        return "Not bad"

pf = pd.read_csv("/content/drive/MyDrive/pandas_data/score.csv",index_col = 0)
pf["Total"] = pf.sum(axis = 1)
pf["Grade"] = pf["Total"].apply(grade)
print(pf, "₩n")
```

	Dan	Jessy	EunJi	Joo	Hyun	Total	Grade
exam1	7	7.50	11.6	15.8	16.3	58.20	Not bad
exam2	19	17.70	15.0	16.0	10.0	77.70	Not bad
exam3	40	81.45	5.0	76.0	84.0	286.45	Good
exam4	60	48.00	75.0	99.0	78.0	360.00	Excellent
exam5	44	38.00	34.0	42.0	46.0	204.00	Good
exam6	21	25.50	15.0	55.0	25.5	142.00	Not bad

pandas 함수(데이터 요약)

- 데이터의 정보를 요약해서 확인/파악하여 분석/판단에 사용 가능
- pf.describe()
  - 간단한 통계 정보를 요약
  - 평균(mean), 표준편차(std), 최소값(min), 25/50/75% 분위 수, 최대값(max) 출력
- 각 열 별로 평균을 구할 수도 있음
- pf.loc[“row name”] = pf.mean(axis = 1)
  - 인자 axis가 1인 경우는 열 별로 평균을 구해서 지정한 행에 저장, 존재하지 않는 행 이름이면 행이 추가됨

```
import pandas as pd
pf = pd.read_csv("/content/drive/MyDrive/pandas_data/score.csv",index_col = 0)
print(pf.describe(), "₩n")
pf.loc["Averge"] = pf.mean(axis = 0)
print(pf, "₩n")
```

	Dan	Jessy	EunJi	Joo	Hyun
count	6.000000	6.000000	6.000000	6.000000	6.000000
mean	31.833333	36.358333	25.933333	50.633333	43.300000
std	19.528611	26.345102	25.904954	33.133769	31.694164
min	7.000000	7.500000	5.000000	15.800000	10.000000
25%	19.500000	19.650000	12.450000	22.500000	18.600000
50%	30.500000	31.750000	15.000000	48.500000	35.750000
75%	43.000000	45.500000	29.250000	70.750000	70.000000
max	60.000000	81.450000	75.000000	99.000000	84.000000

	Dan	Jessy	EunJi	Joo	Hyun
exam1	7.000000	7.500000	11.600000	15.800000	16.3
exam2	19.000000	17.700000	15.000000	16.000000	10.0
exam3	40.000000	81.450000	5.000000	76.000000	84.0
exam4	60.000000	48.000000	75.000000	99.000000	78.0
exam5	44.000000	38.000000	34.000000	42.000000	46.0
exam6	21.000000	25.500000	15.000000	55.000000	25.5
Averge	31.833333	36.358333	25.933333	50.633333	43.3

pandas 함수(데이터 그룹화)

- 데이터를 그룹으로 묶어 분석 가능
- pf.groupby("column name1")["column name2].aggregate_function
  - "column name1" 열 기준으로 그룹화한 후, "column name2" 열에 집계 함수를 적용
  - 2 개 이상의 열을 기준으로 그룹화 할 수 있음
- pf.groupby("column name1").aggregate_function
  - 특정 열을 지정하지 않고 전체 열에 대해서 집계 함수를 적용할 수도 있음

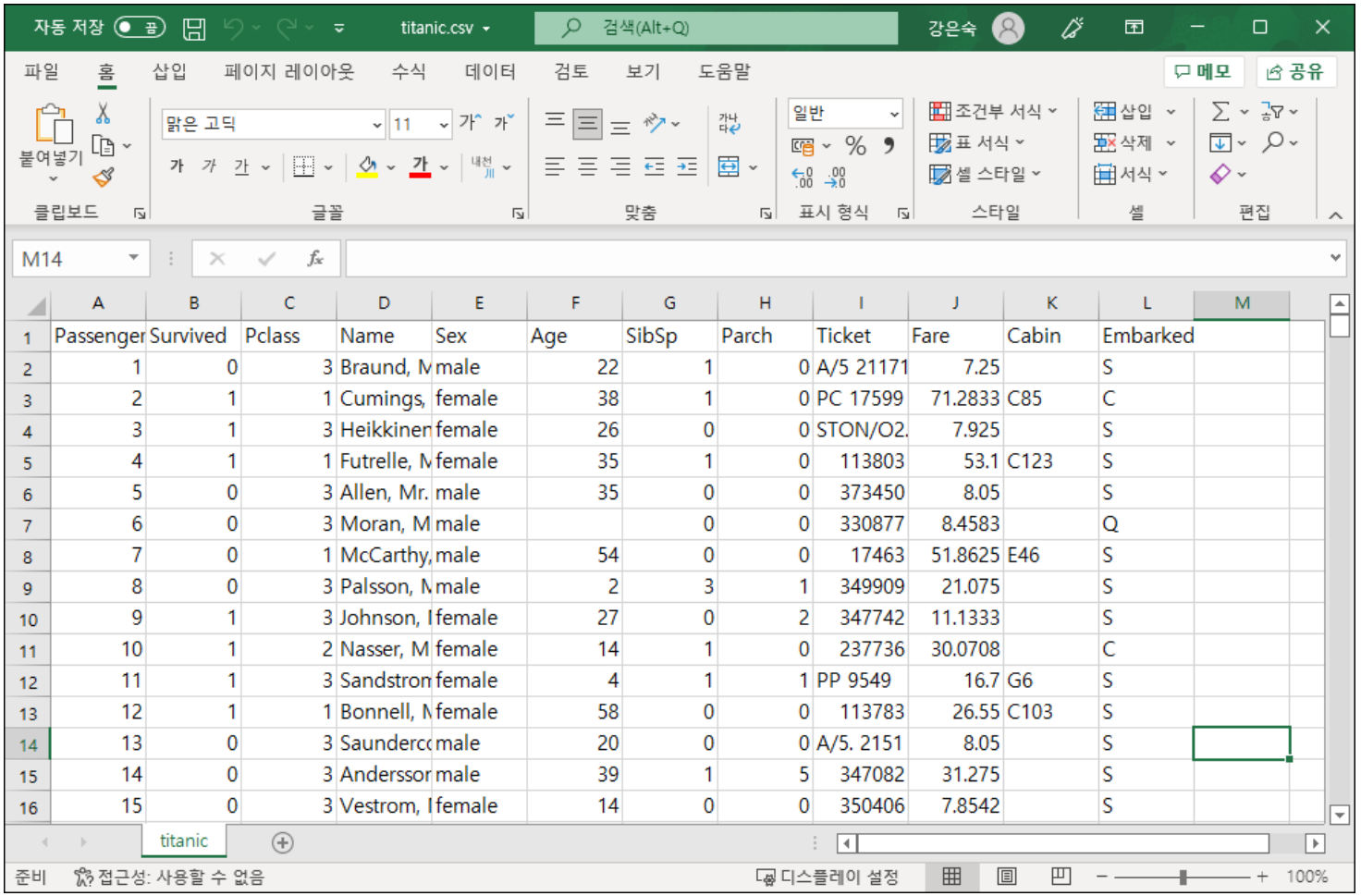
```
import numpy as np
import pandas as pd
students = {"Name": ["Bob", "Jessica", "Mary", "John", "Kate"],
            "Score" : [90, np.nan, 99, 83, 81],
           "Grade" :["A", "F", "A", "B", "B"] }
df = pd.DataFrame(students)
print(df, "₩n")
print(df.groupby("Grade").mean(numeric_only = True), "₩n")
```

"Grade" 열 기준으로 그룹화 한 후, 각 그룹의 평균을 구함
(numeric_only = True는 숫자 값을 가지는 열에 적용한다는 의미)

```
   Name  Score Grade
0   Bob   90.0     A
1  Jessica   NaN     F
2  Mary   99.0     A
3  John   83.0     B
4  Kate   81.0     B

       Score
Grade
A       94.5
B       82.0
F        NaN
```

타이타닉 승객 데이터 분석

- https://pandas.pydata.org/docs/getting_started/index.html
- Getting started tutorials 참조
- 데이터셋 titanic.csv 파일 (891명의 데이터)



titanic.csv

	Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
2	2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
3	3	1	3	Heikkinen	female	26	0	0	STON/O2.	7.925		S
4	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	6	0	3	Moran, M	male		0	0	330877	8.4583		Q
7	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
8	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
9	9	1	3	Johnson, I	female	27	0	2	347742	11.1333		S
10	10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
11	11	1	3	Sandstron	female	4	1	1	PP 9549	16.7	G6	S
12	12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
13	13	0	3	Saunderc	male	20	0	0	A/5. 2151	8.05		S
14	14	0	3	Anderssor	male	39	1	5	347082	31.275		S
15	15	0	3	Vestrom, I	female	14	0	0	350406	7.8542		S

타이타닉 승객 데이터 분석

- titanic.csv 파일 데이터 필드 (승객 1명의 각 필드 의미)
  - PassengerId : 승객 ID
  - Survived : 생존 여부
  - Pclass : 탑승 등급(클래스 1, 2, 3)
  - Name : 이름
  - Sex : 성별
  - Age : 나이
  - SibSp : 타이타닉호에 탑승한 형제/배우자 수
  - Parch : 타이타닉호에 탑승한 부모/자녀 수
  - Ticket : 티켓 번호
  - Fare : 운임
  - Cabin : 객실 번호
  - Embarked : 탑승한 위치
    (C=Cherbourg/Q=Queenstown/S=Southampton)

타이타닉 승객 데이터 분석

- 파일에서 데이터 읽어 오기
- 데이터 내용 출력하기
- 데이터 내용 출력하기
- 데이터에 관한 더 자세한 사항 확인
- 원하는 열의 숫자 데이터 분석

타이타닉 승객 데이터 분석

- titanic.csv 파일
  - 데이터 내용 출력하기
  - 각 열의 자료형 확인

```
1 import pandas as pd
2
3 titanic = pd.read_csv("titanic.csv")
4 print(titanic, "₩n")  # 행의 수가 많으면, 앞뒤 5개의 행을 출력
5 print(titanic.dtypes, "₩n") # pandas의 속성 dtypes는 각 열의 자료형, object는 문자열
6 print(titanic.info(), "₩n")   # more technical information
7 print(titanic[["Age", "Fare"]].describe())   # describe() 함수는 숫자 데이터 분석
```

실행 결과

```
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0              1         0       3  ...   7.2500   NaN         S
1              2         1       1  ...  71.2833   C85         C
2              3         1       3  ...   7.9250   NaN         S
3              4         1       1  ...  53.1000  C123         S
4              5         0       3  ...   8.0500   NaN         S
..           ...       ...     ...  ...      ...   ...       ...
886          887         0       2  ...  13.0000   NaN         S
887          888         1       1  ...  30.0000   B42         S
888          889         0       3  ...  23.4500   NaN         S
889          890         1       1  ...  30.0000  C148         C
890          891         0       3  ...   7.7500   NaN         Q

[891 rows x 12 columns]
```

```
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
```

타이타닉 승객 데이터 분석

- titanic.csv 파일
  - 데이터에 관한 더 자세한 사항 확인
  - 원하는 열의 숫자 데이터 분석

```python
import pandas as pd

titanic = pd.read_csv("titanic.csv")
print(titanic, "₩n")  # 행의 수가 많으면, 앞뒤 5개의 행을 출력
print(titanic.dtypes, "₩n") # pandas의 속성 dtypes는 각 열의 자료형, object는 문자열
print(titanic.info(), "₩n")   # more technical information
print(titanic[["Age", "Fare"]].describe())   # describe() 함수는 숫자 데이터 분석
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

	Age	Fare
count	714.000000	891.000000
mean	29.699118	32.204208
std	14.526497	49.693429
min	0.420000	0.000000
25%	20.125000	7.910400
50%	28.000000	14.454200
75%	38.000000	31.000000
max	80.000000	512.329200

타이타닉 승객 데이터 분석

- 나이가 35세 초과하는 승객 정보 추출

```
# DataFrame 데이터 중에 특정 조건을 만족하는 데이터 추출
above_35 = titanic[titanic["Age"] > 35]
print(above_35.head(), "₩n") # 디폴트로 앞 5개 행 출력
```

```
    PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
1             2         1       1  ...  71.2833   C85         C
6             7         0       1  ...  51.8625   E46         S
11           12         1       1  ...  26.5500  C103         S
13           14         0       3  ...  31.2750   NaN         S
15           16         1       2  ...  16.0000   NaN         S

[5 rows x 12 columns]
```

타이타닉 승객 데이터 분석

- 클래스 2와 3에 탑승한 승객들의 이름 정보 추출

```
14 # 클래스 2와 3에 탑승한 승객들의 이름 추출
15 name_class23 = titanic.loc[titanic["Pclass"].isin([2,3]), "Name"]
16 print(name_class23, "₩n")
```

```
0                                Braund, Mr. Owen Harris
2                                 Heikkinen, Miss. Laina
4                               Allen, Mr. William Henry
5                                       Moran, Mr. James
7                         Palsson, Master. Gosta Leonard
                                 ...
884                               Sutehall, Mr. Henry Jr
885                 Rice, Mrs. William (Margaret Norton)
886                                Montvila, Rev. Juozas
888             Johnston, Miss. Catherine Helen "Carrie"
890                                   Dooley, Mr. Patrick
Name: Name, Length: 675, dtype: object
```

타이타닉 승객 데이터 분석

- 나이 정보가 NaN이 아닌 여자 승객들의 정보 추출

```
18 # 나이가 알려진 여자 승객들의 명단 추출
19 man_age = titanic[(titanic["Age"].notna()) & (titanic["Sex"] == "female")]
20 # notna() 함수는 null 값이 아니면 True
21 print(man_age, "₩n")
```

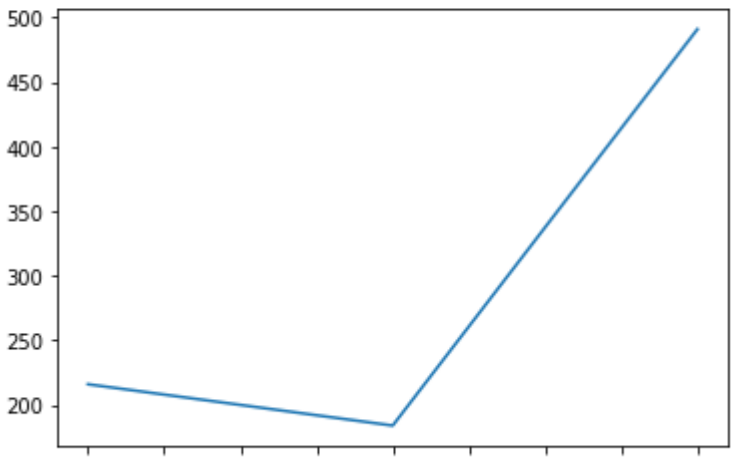
```
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
1              2         1       1  ...  71.2833   C85         C
2              3         1       3  ...   7.9250   NaN         S
3              4         1       1  ...  53.1000  C123         S
8              9         1       3  ...  11.1333   NaN         S
9             10         1       2  ...  30.0708   NaN         C
..           ...       ...     ...  ...      ...   ...       ...
879          880         1       1  ...  83.1583   C50         C
880          881         1       2  ...  26.0000   NaN         S
882          883         0       3  ...  10.5167   NaN         S
885          886         0       3  ...  29.1250   NaN         Q
887          888         1       1  ...  30.0000   B42         S

[261 rows x 12 columns]
```

타이타닉 승객 데이터 분석

- 클래스 별로 탑승 승객의 인원 수 정보를 추출하여 그래프 그리기

```
26 # 클래스별 탑승 인원
27 pf1 = titanic.groupby("Pclass")["Pclass"].count()
28 print(pf1, "₩n")
29 pf1.plot()
30 plt.show()
```

```
Pclass
1    216
2    184
3    491
Name: Pclass, dtype: int64
```

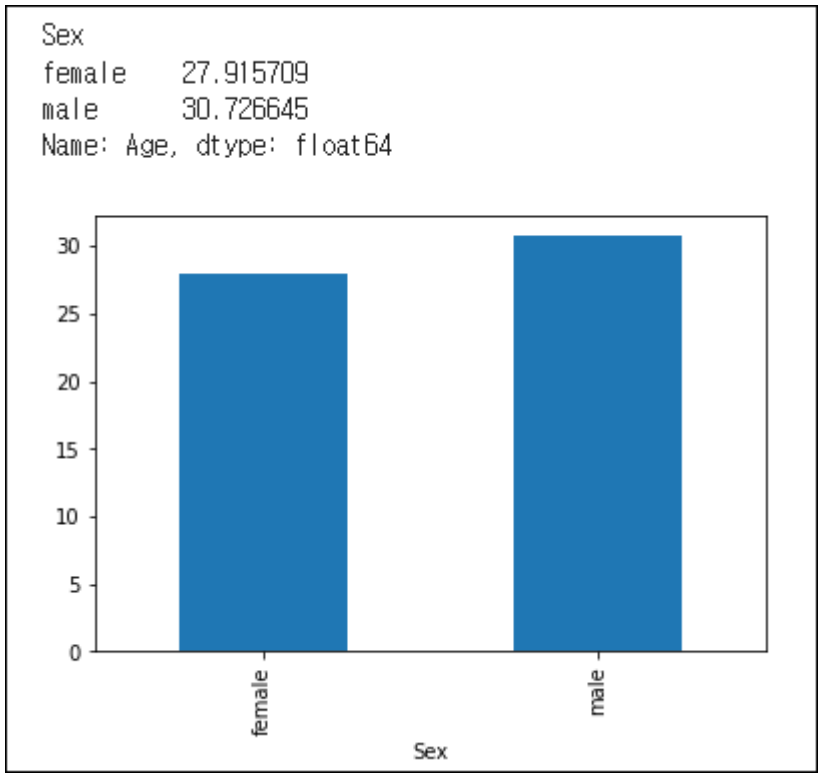


타이타닉 승객 데이터 분석

- 성별로 구분하여 성별 평균 나이 추출하여 그래프 그리기

```
32 # 성별 평균 나이
33 pf1 = titanic.groupby("Sex")["Age"].mean()
34 print(pf1, "₩n")
35 pf1.plot(kind = "bar")
36 plt.show()
```

```
Sex
female    27.915709
male      30.726645
Name: Age, dtype: float64
```



타이타닉 승객 데이터 분석

- 각 클래스별 성별 생존 인원 추출하여 그래프 그리기

```
37 # 클래스와 성별 기준 생존율 생존 인원
38 pf1 = titanic[["Survived", "Pclass", "Sex"]].groupby(["Pclass", "Sex"]).sum()
39 # rename() 함수는 index와 열의 이름을 변경
40 # 인자로 mapper가 필요. 예를 들면, index = {"a" : "A", "b":"B"}와 같이 old_name, new_name
41 # inplace = True는 변경된 사항이 해당 프레임 변수에 반영, 디폴트는 False
42 pf1.rename(columns = {"Survived" :"Survived numbers" }, inplace = True)
43 print(pf1, "₩n")
44 pf1["Survived numbers"].plot(kind = "bar")
45 plt.show()
```

		Survived numbers
Pclass	Sex	
1	female	91
	male	45
2	female	70
	male	17
3	female	72
	male	47