학습 내용

3부. 데이터 분석 라이브러리 활용

11장. N차원 배열 다루기

12장. 데이터프레임과 시리즈

- 1절. 판다스 패키지
- 2절. 데이터프레임 만들기
- 3절. 이름 지정하기
- 4절. 부분 데이터 조회
- 5절. 데이터 삭제 및 추가
- 6절. 정렬
- 7절. 기초 통계 분석
- 8절. 데이터 그룹화 및 집계

13장. 데이터 시각화

14장. 웹 데이터 수집

- https://pypi.python.org/pypi/pandas (package index)
- http://pandas.pydata.org/pandas-docs/stable/api.html (API reference)

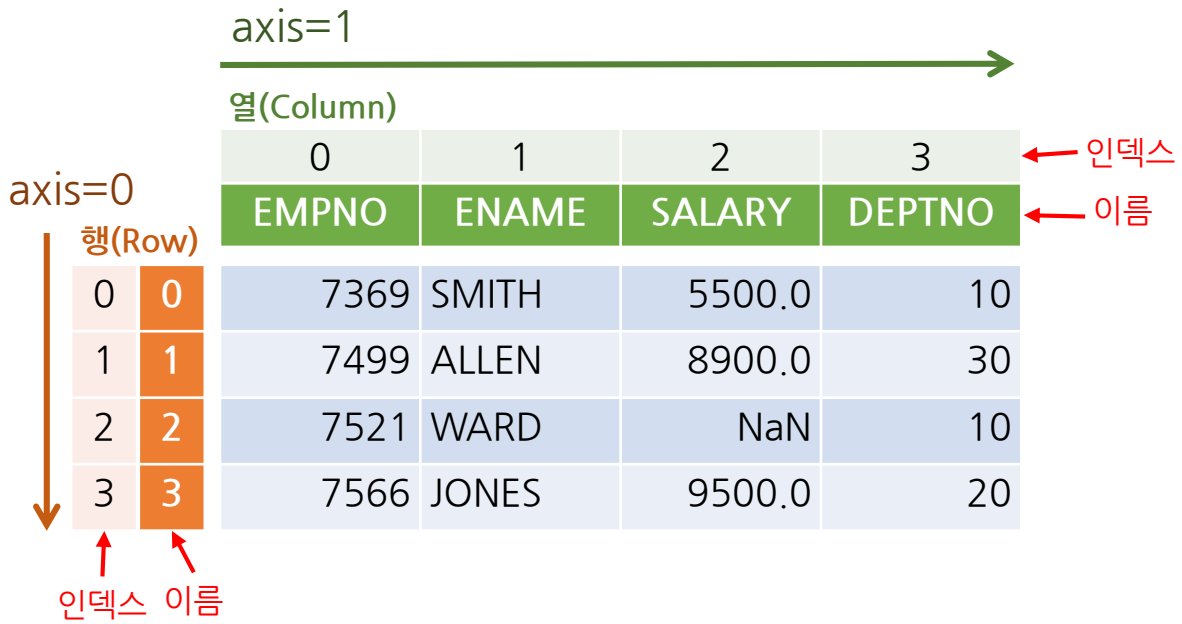
1.1. 판다스 소개

1절. 판다스 패키지

- 1차원 구조를 갖는 시리즈(Series)와 2차원 구조를 갖는 데이터프레임(DataFrame)을 제공
- 데이터프레임은 테이블 형식이고 이종모음들로 구조화 된 데이터를 말하는데 엑셀의 시트 또는 스프레드시트 형식의 데이터
- 시리즈는 시계열 데이터를 표현하기 위한 데이터 구조.
  - 시리즈는 데이터프레임에서 열(Column) 하나를 의미
- 판다스의 데이터프레임과 시리즈 데이터 구조는 재무, 통계, 사회 과학 등 다양한 분야의 데이터를 처리하기 위해 사용
- 판다스는 데이터의 부분집합 조회, 열 추가 및 제거, 병합, 데이터 구조 변경 등 데이터 전처리를 위한 많은 기능을 제공
- 다음처럼 다양한 종류의 데이터를 처리하기에 적합
  - SQL 테이블 또는 Excel 스프레드시트에서와 같이 열(column) 단위로 데이터의 타입이 지정된 테이블 형식
  - 순서가 있고 정렬되지 않은 시계열 데이터.
  - 행 및 열 레이블이 포함 된 임의의 행렬 데이터(동종 유형 또는 이종 유형)

DataFrame

- 2차원(행, 열) 구조(엑셀 시트 구조)
  - 행(Row)
    - 1개 행은 각각 다른 데이터를 갖는 튜플
    - 행의 이름과 인덱스(위치)를 가짐
    - 행의 이름은 인덱스와 같을 경우가 많음
  - 열(Column)
    - 열 내의 모든 데이터는 같은 타입
    - 열의 이름과 인덱스(위치)를 가짐



	0	1	2	3
	EMPNO	ENAME	SALARY	DEPTNO
0	0	7369	SMITH	5500.0	10
1	1	7499	ALLEN	8900.0	30
2	2	7521	WARD	NaN	10
3	3	7566	JONES	9500.0	20

1.2. 판다스 장점

1절. 판다스 패키지

- 결측치(Missing Value) 처리 : 부동 소수점 데이터뿐만 아니라 누락 된 데이터(NaN으로 표시됨)를 손쉽게 처리할 수 있음
- 크기 변경 : 데이터프레임 및 상위 차원 개체에서 열을 삽입하고 삭제할 수 있음
- 데이터 정렬 : 개체를 레이블 세트에 명시적으로 정렬하거나 사용자가 레이블을 무시하고 시리즈, 데이터프레임 등으로 자동으로 데이터를 정렬에 사용할 수 있음
- 데이터 분할 및 병합 : 데이터를 집계 및 변환하기 위해 데이터 세트에 분할 및 병합 작업을 수행 할 수 있는 강력하고 유연한 그룹 별 기능을 제공합니다.
- 데이터프레임 생성 : 다른 파이썬 및 넘파이 데이터 구조의 비정형 색인 데이터를 데이터프레임 객체로 쉽게 변환 할 수 있습니다.
- 부분 데이터 셋 추출 : 지능형 레이블 기반 슬라이싱, 고급 인덱싱 및 대용량 데이터 세트의 하위 집합을 사용할 수 있습니다.
- 피벗과 언피벗 : 데이터 세트의 피벗 및 언피벗 기능을 제공합니다.
- 레이블링 : 축의 계층적 레이블링(다중 레이블을 가질 수 있음)을 제공합니다.
- 파일 입출력 : CSV 파일 또는 구분자에 의한 플랫 파일, Excel 파일, 데이터베이스 및 초고속 HDF5 형식의 데이터 저장/로드를 위한 입출력 도구를 제공합니다.
- 시계열 관련 기능 : 날짜 범위 생성 및 빈도 변환, 통계, 선형 회귀 등을 사용할 수 있습니다.

2.1. 딕셔너리를 이용해서 데이터프레임 만들기

2절. 데이터프레임 만들기

- 딕셔너리를 이용해 데이터프레임을 만들면 키가 열 이름이 됨

```
DataFrame(data=None, index=None, columns=None, dtype=None,
          copy=False)
```

```
1  import pandas as pd
```

```
1  d = {'col1': [1, 2], 'col2': [3, 4]}
2  df = pd.DataFrame(data=d)
3  df
```

	col1	col2
0	1	3
1	2	4

```
1  d = [{'col1': 1, 'col2': 3}, {'col1': 2, 'col2': 4}]
2  df = pd.DataFrame(data=d)
3  df
```

	col1	col2
0	1	3
1	2	4

```
1  d = [{'col1': 1, 'col2': 3}, {'col1': 2, 'col2': 4},
2       {'col1': 2}]
3  df = pd.DataFrame(data=d)
4  df
```

	col1	col2
0	1	3.0
1	2	4.0
2	2	NaN

2.2. 리스트를 이용해 데이터프레임 만들기

- 리스트를 이용하려면 딕셔너리의 값으로 지정하거나

```
1  a = [1,2,3,4,5]
2  b = [6,7,8,9,10]
3  df = pd.DataFrame({"col1": a, "col2": b})
4  df
```

	col1	col2
0	1	6
1	2	7
2	3	8
3	4	9
4	5	10

1) CSV 파일 불러오기

- read_csv()

```
pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None,
                header='infer', ...)
```

```
1  import pandas as pd
2  member_df = pd.read_csv("member_data.csv", sep=",")
```

```
1  member_df
```

	Name	Age	Email	Address
0	홍길동	20	kildong@hong.com	서울시 강동구
1	홍길서	25	kilseo@hong.com	서울시 강서구
2	홍길남	26	south@hong.com	서울시 강남구
3	홍길북	27	book@hong.com	서울시 강북구

샘플 데이터(member_data.csv)

```
Name,Age,Email,Address
홍길동,20,kildong@hong.com,서울시 강동구
홍길서,25,kilseo@hong.com,서울시 강서구
홍길남,26,south@hong.com,서울시 강남구
홍길북,27,book@hong.com,서울시 강북구
```

2.4. sklearn.datasets 나 statsmodels.api.datasets 모듈 데이터를 데이터프레임으로

- Scikit-learn 패키지에는 학습을 위한 많은 데이터셋이 제공
- Scikit-learn에서 제공하는 데이터셋은 딕셔너리 형식

```
import statsmodels.api as sm
iris = sm.datasets.get_rdataset("iris", package="datasets").data
iris
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

3.1. 열 이름 지정하기

3절. 이름 지정하기

- columns 속성 이용 열 이름 지정

```
1  member_df.columns = ["이름", "나이", "이메일", "주소"]
2  member_df
```

	이름	나이	이메일	주소
0	홍길동	20	kildong@hong.com	서울시 강동구
1	홍길서	25	kilseo@hong.com	서울시 강서구
2	홍길남	26	south@hong.com	서울시 강남구
3	홍길북	27	book@hong.com	서울시 강북구

```
1  member_df.columns
```

```
Index(['이름', '나이', '이메일', '주소'], dtype='object')
```

3.2. 행 이름 지정하기

3절. 이름 지정하기

- index 속성 이용 행 이름 지정

```
1  member_df.index = ["동", "서", "남", "북"]
2  member_df
```

	이름	나이	이메일	주소
동	홍길동	20.0	kildong@hong.com	서울시 강동구
서	홍길서	25.0	kilseo@hong.com	서울시 강서구
남	홍길남	26.0	south@hong.com	서울시 강남구
북	홍길북	27.0	book@hong.com	서울시 강북구

4.1. 단일 열 조회

4절. 부분 데이터 조회

- 데이터프레임.*열이름*
- 데이터프레임*["열이름"]*
  - 열 이름에 . 또는 공백 등이 포함되어 있을 경우 사용

```
1  member_df.Name
```

```
0    홍길동
1    홍길서
2    홍길남
3    홍길북
Name: Name, dtype: object
```

```
1  member_df["Name"]
```

```
0    홍길동
1    홍길서
2    홍길남
3    홍길북
Name: Name, dtype: object
```

4.2. loc를 이용한 이름으로 조회

4절. 부분 데이터 조회

- loc[행이름, 열이름]
- 열이름 생략 가능

```
1  member_df.loc[0:2]
```

열 이름을 생략하면 행 이름으로 찾음

	Name	Age	Email	Address
0	홍길동	20	kildong@hong.com	서울시 강동구
1	홍길서	25	kilseo@hong.com	서울시 강서구
2	홍길남	26	south@hong.com	서울시 강남구

```
1  member_df.loc["Name":"Email"] # nothing
```

열 이름을 생략하면 행 이름으로 찾음

	Name	Age	Email	Address

슬라이싱으로 찾기

```
1  member_df.loc[0:2, "Name":"Email"]
```

	Name	Age	Email
0	홍길동	20	kildong@hong.com
1	홍길서	25	kilseo@hong.com
2	홍길남	26	south@hong.com

```
1  member_df.loc[[0,2], ["Name","Email"]]
```

리스트로 찾기

	Name	Email
0	홍길동	kildong@hong.com
2	홍길남	south@hong.com

4.3. iloc를 이용한 인덱스로 조회

4절. 부분 데이터 조회

- iloc[행인덱스, 열인덱스]

```
1  member_df.iloc[1:3, 1:3]
```

	Age	Email
1	25	kilseo@hong.com
2	26	south@hong.com

```
1  member_df.iloc[0:3, 0:3]
```

	Name	Age	Email
0	홍길동	20	kildong@hong.com
1	홍길서	25	kilseo@hong.com
2	홍길남	26	south@hong.com

```
1  member_df.iloc[::-1]
```

	Name	Age	Email	Address
3	홍길북	27	book@hong.com	서울시 강북구
2	홍길남	26	south@hong.com	서울시 강남구
1	홍길서	25	kilseo@hong.com	서울시 강서구
0	홍길동	20	kildong@hong.com	서울시 강동구

```
1  member_df.iloc[0::2,[1,3]]
```

	Age	Address
0	20	서울시 강동구
2	26	서울시 강남구

4.4 조건으로 조회하기

4절. 부분 데이터 조회

- 예제에 사용할 데이터 : 붓꽃 데이터 https://bruders.tistory.com/82

```
import seaborn as sns
iris_df = sns.load_dataset('iris')
iris_df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

4.4 조건으로 조회하기

4절. 부분 데이터 조회

- loc[행조건] : 행 조건에 맞는 모든 열을 반환

```
1  iris_df.loc[iris_df['Species']=='versicolor'].head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
50	7.0	3.2	4.7	1.4	versicolor
51	6.4	3.2	4.5	1.5	versicolor
52	6.9	3.1	4.9	1.5	versicolor
53	5.5	2.3	4.0	1.3	versicolor
54	6.5	2.8	4.6	1.5	versicolor

4.4 조건으로 조회하기

4절. 부분 데이터 조회

- loc[행조건, 열리스트] : 행 조건에 맞는 지정한 열을 반환

```
1  iris_df.loc[iris_df['Species']=='versicolor',
2              ['Sepal.Length', 'Species']].head()
```

	Sepal.Length	Species
50	7.0	versicolor
51	6.4	versicolor
52	6.9	versicolor
53	5.5	versicolor
54	6.5	versicolor

4.4 조건으로 조회하기

4절. 부분 데이터 조회

- loc[중복조건]

```
1  iris_df.loc[(iris_df['Species']=='versicolor') &
2              (iris_df['Sepal.Length'].astype(float) > 6.5)].head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
50	7.0	3.2	4.7	1.4	versicolor
52	6.9	3.1	4.9	1.5	versicolor
58	6.6	2.9	4.6	1.3	versicolor
65	6.7	3.1	4.4	1.4	versicolor
75	6.6	3.0	4.4	1.4	versicolor

시리즈를 문자 함수로 쓰기 위해서 :
https://pandas.pydata.org/pandas-docs/stable/reference/series.html#string-handling

5.1. 데이터프레임의 항목 삭제


```
DataFrame.drop(labels=None, axis=0, inplace=False)
```

구문에서...

- *labels* : 삭제할 index 또는 컬럼의 이름을 지정합니다.
- *axis* : int 타입 또는 축의 이름입니다. (0 또는 'index') 와 (1 또는 'columns') 중 하나를 갖습니다. 1이면 열을 삭제합니다.
- *inplace* : bool 타입이며, False(기본값)이면 삭제된 결과 데이터프레임을 리턴하며, True 이면 현재 데이터프레임에서 데이터를 삭제하고 None을 반환합니다.

1) 단일 행 삭제하기

- axis=0

```
1  member_df = member_df.drop('북') #axis=0(기본값)이면 행에서 찾아 삭제
2  member_df
```

	이름	나이	이메일	주소
동	홍길동	20	kildong@hong.com	서울시 강동구
서	홍길서	25	kilseo@hong.com	서울시 강서구
남	홍길남	26	south@hong.com	서울시 강남구

2) 단일 열 삭제하기

- axis=1

```
1  member_df = member_df.drop('주소', axis=1) #axis=1이면 열에서 찾아 삭제
2  member_df
```

	이름	나이	이메일
동	홍길동	20	kildong@hong.com
서	홍길서	25	kilseo@hong.com
남	홍길남	26	south@hong.com

3) 복수일 열 삭제하기

- labels = [삭제할_열_리스트,]

```
1  member_df.drop(labels=["Email", "Address"], axis=1)
```

	Name	Age
0	홍길동	20
1	홍길서	25
2	홍길남	26
3	홍길북	27

axis=1과 axis='columns'와 동일

```
1  member_df.drop(labels=["Email", "Address"], axis="columns")
```

	Name	Age
0	홍길동	20
1	홍길서	25
2	홍길남	26
3	홍길북	27

4) 열 삭제와 재 할당

- inplace=True

```
1  member_df.drop("Address", axis=1, inplace=True)
2  member_df
```

	Name	Age	Email
0	홍길동	20	kildong@hong.com
1	홍길서	25	kilseo@hong.com
2	홍길남	26	south@hong.com
3	홍길북	27	book@hong.com

1) 열 추가

5절. 데이터 추가 및 삭제 > 5.2. 데이터프레임의 항목 추가

- 데이터프레임["새로운_열_이름"] = 값

예제에 사용할 데이터

```
1  member_df = pd.read_csv("member_data.csv", comment='#')
2  member_df["BirthYear"] = 2000
3  member_df
```

	Name	Age	Email	Address	BirthYear
0	홍길동	20	kildong@hong.com	서울시 강동구	2000
1	홍길서	25	kilseo@hong.com		
2	홍길남	26	south@hong.com		
3	홍길북	27	book@hong.com		

```
1  member_df = pd.read_csv("member_data.csv", comment='#')
2  member_df["BirthYear"] = [2001, 2002, 2003, None]
3  member_df
```

	Name	Age	Email	Address	BirthYear
0	홍길동	20	kildong@hong.com	서울시 강동구	2001.0
1	홍길서	25	kilseo@hong.com	서울시 강서구	2002.0
2	홍길남	26	south@hong.com	서울시 강남구	2003.0
3	홍길북	27	book@hong.com	서울시 강북구	NaN

2) 시리즈를 이용한 열 추가

- 인덱스를 포함하는 시리즈 객체를 이용해 추가

```
1  member_df = pd.read_csv("member_data.csv", comment='#')
2  member_df["BirthYear"] = pd.Series([2001, 2002, 2004], index=[0,1,3])
3  member_df
```

	Name	Age	Email	Address	BirthYear
0	홍길동	20	kildong@hong.com	서울시 강동구	2001.0
1	홍길서	25	kilseo@hong.com	서울시 강서구	2002.0
2	홍길남	26	south@hong.com	서울시 강남구	NaN
3	홍길북	27	book@hong.com	서울시 강북구	2004.0

6절. 정렬

6절. 정렬

행 또는 이름으로 정렬

```
DataFrame.sort_index(axis=0, level=None, ascending=True,
                     inplace=False, kind='quicksort',
                     na_position='last', sort_remaining=True,
                     by=None)
```

값으로 정렬

```
DataFrame.sort_values(by, axis=0, ascending=True, inplace=False,
                      kind='quicksort', na_position='last')
```

```
1  member_df = pd.read_csv("member_data.csv", comment='#')
2  member_df.index = ["동", "서", "남", "북"]
3  member_df
```

	Name	Age	Email	Address
동	홍길동	20	kildong@hong.com	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

예제에 사용할 데이터

6.1. 행 이름으로 정렬

6절. 정렬

- sort_index() 함수는 데이터프레임의 행 이름을 이용해서 정렬

```
1  member_df.sort_index()
```

	Name	Age	Email	Address
남	홍길남	26	south@hong.com	서울시 강남구
동	홍길동	20	kildong@hong.com	서울시 강동구
북	홍길북	27	book@hong.com	서울시 강북구
서	홍길서	25	kilseo@hong.com	서울시 강서구

6.2. 열 이름으로 열 순서 바꾸기

6절. 정렬

- axis=1
- 열의 이름순으로 열의 순서를 바꿈

```
1  member_df.sort_index(axis=1)
```

	Address	Age	Email	Name
동	서울시 강동구	20	kildong@hong.com	홍길동
서	서울시 강서구	25	kilseo@hong.com	홍길서
남	서울시 강남구	26	south@hong.com	홍길남
북	서울시 강북구	27	book@hong.com	홍길북

6.3. 값으로 정렬

6절. 정렬

- 데이터프레임의 값을 기준으로 정렬하려면 sort_values()를 이용

```
1  member_df.sort_values(by=["Email"])
```

	Name	Age	Email	Address
북	홍길북	27	book@hong.com	서울시 강북구
동	홍길동	20	kildong@hong.com	서울시 강동구
서	홍길서	25	kilseo@hong.com	서울시 강서구
남	홍길남	26	south@hong.com	서울시 강남구

7절. 기초 통계 분석

- 판다스에서 제공하는 통계분석은 기본적인 기술통계 및 데이터 요약

함수	설명
count	NA를 제외한 개수
min	최소값
max	최대값
sum	합
cumprod	누적합
mean	평균
median	중앙값
quantile	분위수
corr	상관관계
var	표본분산
std	표본 정규분산

예제에 사용할 데이터

```
1  import statsmodels.api as sm
2  iris = sm.datasets.get_rdataset("iris", package="datasets")
3  iris_df = iris.data
4  iris_df.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

7.1. 최소값, 최대값, 평균, 중위수

7절. 기초 통계 분석

- min(), max(), mean(), median()

```
1  iris_df.min()
```

```
Sepal.Length        4.3
Sepal.Width           2
Petal.Length          1
Petal.Width         0.1
Species          setosa
dtype: object
```

```
1  iris_df.max()
```

```
Sepal.Length          7.9
Sepal.Width           4.4
Petal.Length          6.9
Petal.Width           2.5
Species         virginica
dtype: object
```

7.2. 요약 통계량

7절. 기초 통계 분석

```
DataFrame.describe(percentiles=None, include=None, exclude=None)
```

- 구문에서...
  - *percentiles* : 출력에 포함될 백분위 수를 0~1사이의 값으로 지정. 기본 값은 [.25, .5, .75]. 25%, 50%, 75% 위치 데이터를 출력
  - *include* : 출력에 포함될 데이터의 유형을 지정합니다. None(기본값) 이면 모든 숫자 타입 열들을 출력에 포함시킵니다. “all”이면 모든 열을 포함합니다. 정수형이면 “int64”, 논리형이면 “bool”, 실수형이면 “float64” 등으로 지정합니다.
  - *exclude* : 출력에서 제외할 데이터의 유형을 지정합니다. None(기본값) 이면 아무것도 제외시키지 않습니다.

7.2. 요약 통계량

- 숫자 데이터
  - 결과의 인덱스에는 count, mean, std, min, max 및 하위 백분위 수, 상위 백분위 수 및 상위 백분율이 포함
  - 기본적으로 하위 백분위 수는 25이고 상위 백분위 수는 75입니다. 50 백분위 수는 중앙값과 같음
- 객체 데이터(예 : 문자열 또는 타임 스탬프)
  - 결과 색인에 count, unique, top 그리고 freq가 포함
  - top가 가장 일반적인 값.
  - 여러 오브젝트 값이 가장 높은 count를 갖는 경우, count와 top 결과는 가장 높은 count를 갖는 오브젝트 값 중에서 임의로 선택.
- DataFrame을 통해 제공되는 혼합 데이터 유형
  - 기본값은 숫자 열의 분석만 반환
  - 데이터프레임이 숫자 열이 없는 개체 및 범주 데이터로만 구성된 경우 기본값은 개체 열과 범주 형 열 모두의 분석을 반환
  - include='all' 매개변수가 제공되면 결과에는 각 유형의 속성이 결합됨

1) 기본 요약 통계량

7절. 기초 통계 분석 > 7.2. 요약 통계량

- iris 데이터의 요약 통계량에는 종(Species) 정보는 출력되지 않음
- 기본적으로 숫자 데이터의 요약 통계량이 출력

```
1  iris_df.describe()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
1  iris_df.Species.describe()
```

```
count          150
unique           3
top      virginica
freq            50
Name: Species, dtype: object
```

2) include와 exclude

- include 및 exclude 매개 변수를 사용하여 DataFrame에서 출력용으로 분석되는 열을 포함 또는 제한

```
1  import pandas as pd
2  df = pd.DataFrame({'a': [1, 2] * 3,
3                     'b': [True, False] * 3,
4                     'c': [2.0, 4.0] * 3})
```

```
1  df.describe()
```

	a	c
count	6.000000	6.000000
mean	1.500000	3.000000
std	0.547723	1.095445
min	1.000000	2.000000
25%	1.000000	2.000000
50%	1.500000	3.000000
75%	2.000000	4.000000
max	2.000000	4.000000

```
1  df.describe(include=["int64"])
```

	a
count	6.000000
mean	1.500000
std	0.547723
min	1.000000
25%	1.000000
50%	1.500000
75%	2.000000
max	2.000000

```
1  df.describe(include='all')
```

	a	b	c
count	6.000000	6	6.000000
unique	NaN	2	NaN
top	NaN	True	NaN
freq	NaN	3	NaN
mean	1.500000	NaN	3.000000
std	0.547723	NaN	1.095445
min	1.000000	NaN	2.000000
25%	1.000000	NaN	2.000000
50%	1.500000	NaN	3.000000
75%	2.000000	NaN	4.000000
max	2.000000	NaN	4.000000

```
1  df.describe(exclude=["bool", "float64"])
```

7.3. 분산, 표준편차

7절. 기초 통계 분석

```
DataFrame.var(axis=None, skipna=None, level=None, ddof=1,
              numeric_only=None, **kwargs)
```

```
DataFrame.std(axis=None, skipna=None, level=None, ddof=1,
              numeric_only=None, **kwargs)
```

```
1  iris_df.var()
```

```
Sepal.Length    0.685694
Sepal.Width     0.189979
Petal.Length    3.116278
Petal.Width     0.581006
dtype: float64
```

```
1  iris_df.std()
```

```
Sepal.Length    0.828066
Sepal.Width     0.435866
Petal.Length    1.765298
Petal.Width     0.762238
dtype: float64
```

7.4. 공분산, 상관계수

7절. 기초 통계 분석

```
DataFrame.cov(min_periods=None)
```

```
DataFrame.corr(method='pearson', min_periods=1)
```

```
1  iris_df.corr()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.000000	-0.117570	0.871754	0.817941
Sepal.Width	-0.117570	1.000000	-0.428440	-0.366126
Petal.Length	0.871754	-0.428440	1.000000	0.962865
Petal.Width	0.817941	-0.366126	0.962865	1.000000

```
1  iris_df.cov()
```

	Sepal.Length	Sepal.Width	Pe	
Sepal.Length	0.685694	-0.042434	1.274315	0.516271
Sepal.Width	-0.042434	0.189979	-0.329656	-0.121639
Petal.Length	1.274315	-0.329656	3.116278	1.295609
Petal.Width	0.516271	-0.121639	1.295609	0.581006

8.1. groupby

8절. 데이터 그룹화 및 집계

- groupby()는 데이터를 구분 할 수 있는 열(column)의 값들을 이용하여 데이터를 여러 기준에 의해 구분하여 그룹화 한 후 기초 통계 함수 등을 적용 할 수 있도록 함

```
DataFrame.groupby(by=None, axis=0, level=None, as_index=True,
                  sort=True, group_keys=True, squeeze=False,
                  observed=False, **kwargs)
```

```
1 import statsmodels.api as sm
2 iris = sm.datasets.get_rdataset("iris", package="datasets")
3 iris_df = iris.data
4 iris_df.head()
```

예제에 사용할 데이터

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

1) 단일 열로 그룹화

- groupby 함수의 인수로 그룹화 할 열을 지정

```
1  iris_grouped = iris_df.groupby(iris_df.Species)
2  iris_grouped
```

```
<pandas.core.groupby.groupby.DataFrameGroupBy object at 0x0000014F76F0BEF0>
```

```
1  iris_grouped.mean()
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

Pandas가 제공하는 파일 형식

11절

파일 형식	설명	Read	Write
csv	✓ text 형태로 데이터 저장 ✓ 데이터와 데이터 사이에 구분자를 이용해 저장 ✓ 메모장에서도 작성 가능	read_csv	to_csv
excel	✓ 엑셀 프로그램 필요	read_excel	to_excel
Json	✓ Javascript 객체 저장 형식으로 데이터 저장 ✓ 웹 등을 이용해 데이터를 주고받기 위해 사용하는 형식	read_json	to_json
Html	✓ 웹 페이지 파일 형식	read_html	to_html
hd5	✓ 딥러닝에서의 모델 저장 시 사용하는 형식	read_hd5	to_hd5

연습문제 - 실습형

```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.sample(1)
```

1. iris 데이터에서 처음 다섯개 행만 출력하세요
2. iris 데이터를 데이터프레임 변수인 독립변수 X와 종속변수 y로 나누세요. hint: y = iris.loc[:, 'species'].to_frame()
3. iris 데이터에서 처음 50개행을 빼내서 temp변수에 저장하세요
4. 3번에서 선택한 데이터프레임의 요약정보를 출력하세요. 모든 열에 대해 요약정보가 출력되어야 합니다.
5. versicolor종의 데이터만 iris_versicolor변수에 저장하세요

연습문제 - 실습형

6. 2번의 2번의 X와 y변수를 합해서 iris_df데이터 프레임으로 만드세요

   hint : X와 y합하기 : pd.concat([X, y], axis=1)

7. iris 데이터의 각 열 평균값을 출력하세요.

8. iris 데이터의 각 열들 사이의 상관계수를 출력하세요

9. iris 데이터의 종별 평균을 출력하세요