

Chapter 3. Tabular data

데이터

- 다양한 센서를 통해 어떤 현상을 관측/측정하여 기록
- 한번의 관측을 통해 1개의 샘플을 얻을 수 있음.
- 1개의 샘플은 d차원의 벡터, $x \in \mathbb{R}^d$
- 즉, d개의 변수에 대한 정보를 측정 (대부분 수치적인 값)

| 온도 | 습도 | 풍속 | 풍향 |
|------|-----|------|----|
| 28°C | 30% | 5m/s | SW |

데이터셋

- N회의 관측을 통해 N개의 샘플을 얻을 수 있음.
- 1개 샘플이 d차원 벡터일때 N개 샘플은?
- $X = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$
- $N \times d$ matrix의 형태로 표현가능
- N : 샘플의 수, 관측 횟수, 행의 개수
- d : 특성(feature)의 수, 열의 개수

| 온도 | 습도 | 풍속 | 풍향 |
|------|------|------|----|
| 28°C | 30% | 5m/s | SW |
| 27°C | 10% | 0m/s | - |
| 23°C | 100% | 1m/s | NE |
| 30°C | 60% | 2m/s | W |

정형 데이터

- 정형 데이터(Tabular data)란?
- 각 샘플이 동일한 수의 특성(Feature)으로 구성된 고정 구조의 데이터
- 관측 결과로 얻어진 샘플들이 열(column)과 행(row)로 구성되는 표의 형식으로 표현될 수 있는 데이터를 의미함
- 열(column): 특정 열은 특정 특성에 대응함. 따라서 컬럼별로 동일한 자료형
- 행(row): 각 행은 하나의 관측값(샘플)에 대응함.

정형 데이터의 예시 - iris

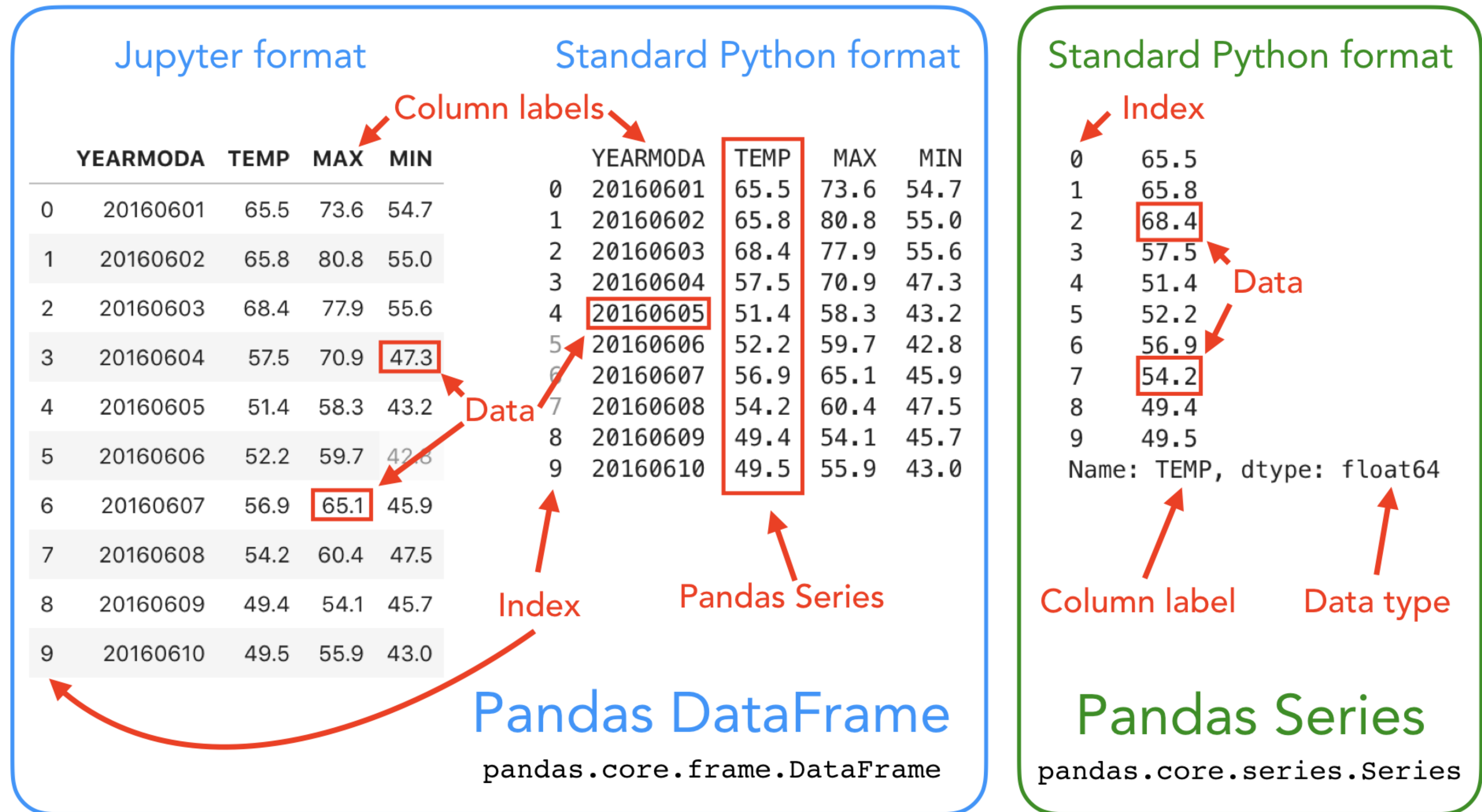
| SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---------------|--------------|---------------|--------------|-------------|
| 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |

Iris dataset

정형 데이터의 예시 - Titanic

| | Survived | Pclass | Sex | Age | SibSp | Parch | Fare |
|---|----------|--------|-----|------|-------|-------|---------|
| 0 | 0 | 3 | 1 | 22.0 | 1 | 0 | 7.2500 |
| 1 | 1 | 1 | 0 | 38.0 | 1 | 0 | 71.2833 |
| 2 | 1 | 3 | 0 | 26.0 | 0 | 0 | 7.9250 |
| 3 | 1 | 1 | 0 | 35.0 | 1 | 0 | 53.1000 |
| 4 | 0 | 3 | 1 | 35.0 | 0 | 0 | 8.0500 |

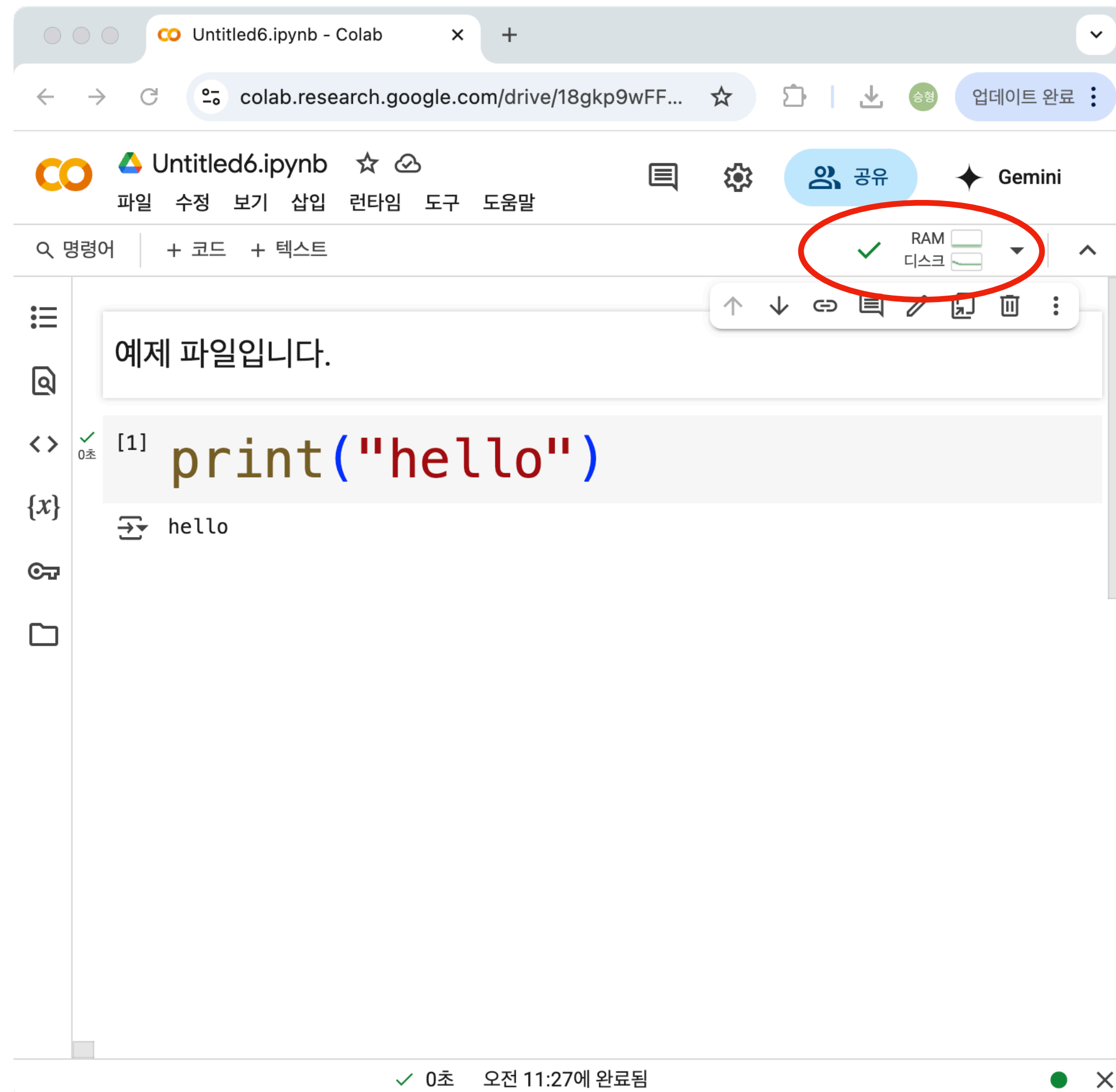
Pandas dataframe



정형 데이터 살펴보기 - 코랩

- 구글 코랩 로그인
- collab.google.com -> 새 노트
- 노트북은 코드 셀과 마크다운 셀로 구분되어있음
- 코드셀 : python code 작성 및 실행 결과 확인
- 마크다운셀: 마크다운 문법을 통한 문서 작성 가능

정형 데이터 살펴보기 - 코랩



- 코드셀 (회색배경영역)에 코드 작성후 ctrl + enter로 해당 코드 블록 실행
- GPU 사용 필요시에는
런타임 - 런타임 유형변경
- GPU/TPU 설정 - 저장
- 화면 우측 상단에 런타임 유형 확인

정형 데이터 살펴보기 - 초기설정

```
▶ import pandas as pd  
  
!wget https://raw.githubusercontent.com/shryu8902/KIRD_AUTOML/main/Iris.xlsx  
!wget https://raw.githubusercontent.com/shryu8902/KIRD_AUTOML/main/Iris.csv
```

- Pandas 라이브러리의 구현된 클래스들을 pd라는 약어를 통해 접근가능.
- 두개의 샘플 데이터파일 다운로드 (xlsx, csv 형식)
- Python code 내에서 리눅스 명령어를 사용하기 위해 !사용 (e.g., !wget, !pip)
- 왼쪽 메뉴바에서 폴더 아이콘 클릭시 iris.csv 파일과 iris.xlsx파일 확인 가능

정형 데이터 살펴보기 - 파일 읽기

- `pd.read_excel('파일경로')` : 엑셀파일을 읽어와 데이터 프레임 객체로 변환
- `pd.read_csv('파일경로')` : csv 파일을 읽어와 데이터 프레임 객체로 변환
- `type('변수이름')` 명령어로 해당 변수명으로 접근가능한 객체의 유형을 살펴보면?

```
▶ df_xlsx = pd.read_excel('Iris.xlsx')  
df_csv = pd.read_csv('Iris.csv')
```

```
print(type(df_csv.SepalLengthCm))  
print(type(df_csv.iloc[0]))
```

Dataframe과 Series

- Series: 자료형들의 연속체
- Dataframe: Series들의 모음

```
print(type(df_xlsx))  
print(type(df_csv))
```

```
print(type(df_csv.SepalLengthCm))  
print(type(df_csv.iloc[0]))
```

정형 데이터 살펴보기 - 기본 형태

- 데이터프레임의 기본 메서드
- `.head(n)` : 처음 n 개의 행 출력
- `.tail(n)` : 마지막 n 개의 행 출력
- `.columns` : 데이터 프레임 컬럼명

```
print(df_csv.columns)
print(df_csv.head()) |
print(df_csv.tail())
```


```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
      'Species'],
      dtype='object')
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0   1             5.1           3.5           1.4           0.2  Iris-setosa
1   2             4.9           3.0           1.4           0.2  Iris-setosa
2   3             4.7           3.2           1.3           0.2  Iris-setosa
3   4             4.6           3.1           1.5           0.2  Iris-setosa
4   5             5.0           3.6           1.4           0.2  Iris-setosa
145 146             6.7           3.0           5.2           2.3  Iris-virginica
146 147             6.3           2.5           5.0           1.9  Iris-virginica
147 148             6.5           3.0           5.2           2.0  Iris-virginica
148 149             6.2           3.4           5.4           2.3  Iris-virginica
149 150             5.9           3.0           5.1           1.8  Iris-virginica
```

정형 데이터 살펴보기 - 간단요약

- 다음 메서드를 통해 요약정보 확인가능
- `.info()` : 각 컬럼의 데이터 타입, 누락 데이터 등 데이터 프레임의 요약 정보 출력
- `.describe()` : 데이터 프레임의 통계량 (평균, 표준편차, 최대, 최소 값등) 정보 출력

정형 데이터 살펴보기 - 간단요약


▶ df_csv.info()

 <class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):

| # | Column | Non-Null Count | Dtype |
|---|---------------|----------------|---------|
| 0 | Id | 150 non-null | int64 |
| 1 | SepalLengthCm | 150 non-null | float64 |
| 2 | SepalWidthCm | 150 non-null | float64 |
| 3 | PetalLengthCm | 150 non-null | float64 |
| 4 | PetalWidthCm | 150 non-null | float64 |
| 5 | Species | 150 non-null | object |

dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB

▶ df_csv.describe()

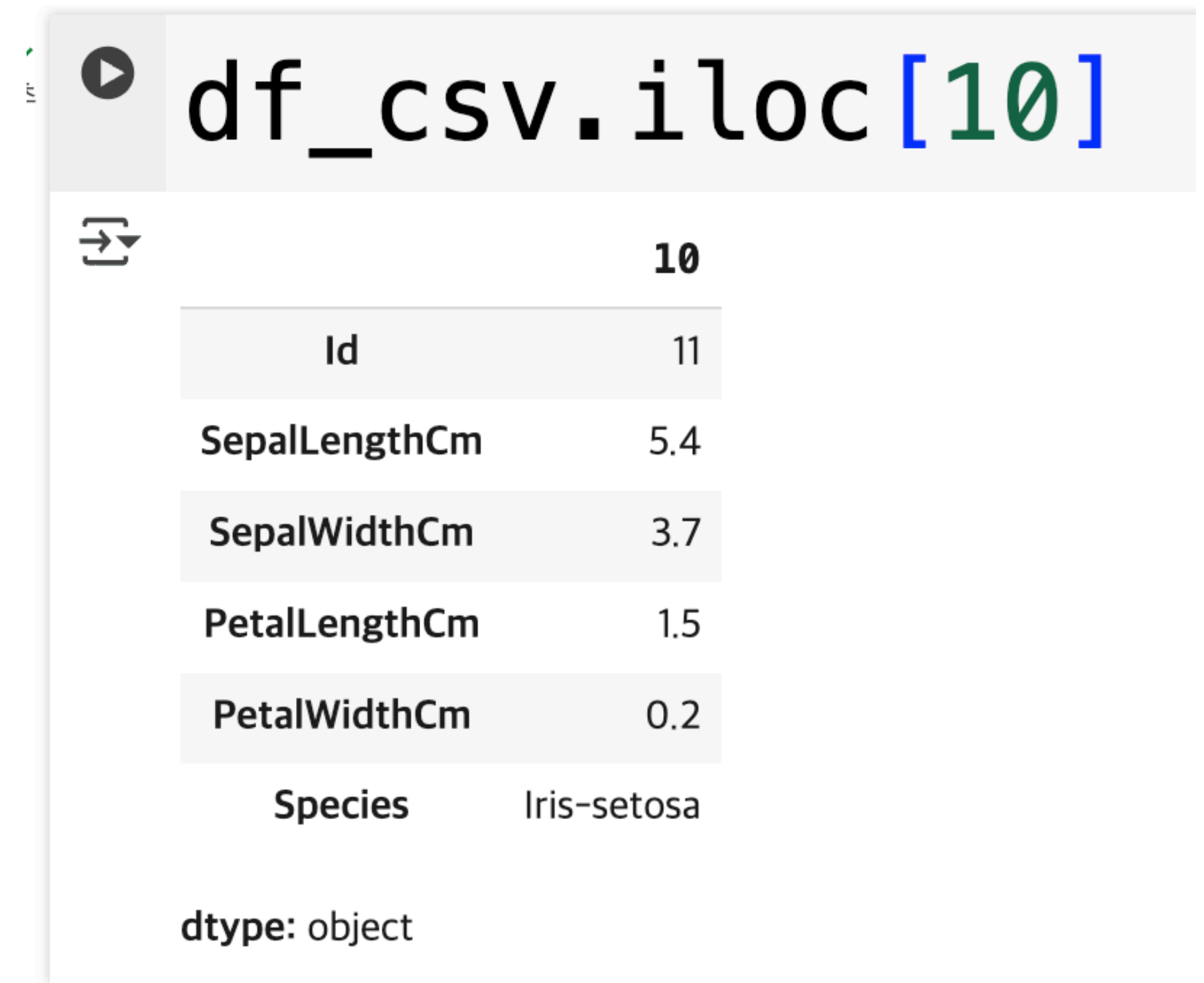


| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|-------|------------|---------------|--------------|---------------|--------------|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |



특정 행을 선택하는 방법

- 데이터 프레임의 각 행은 하나의 샘플을 의미함.
- 특정 행을 선택하는 방법: 인덱스 명, 또는 순서
- `.index` : index 이름 확인
- `.loc['인덱스 이름']` : 입력된이름을가진 행 선택
- `.iloc['숫자']` : i번째 행 선택

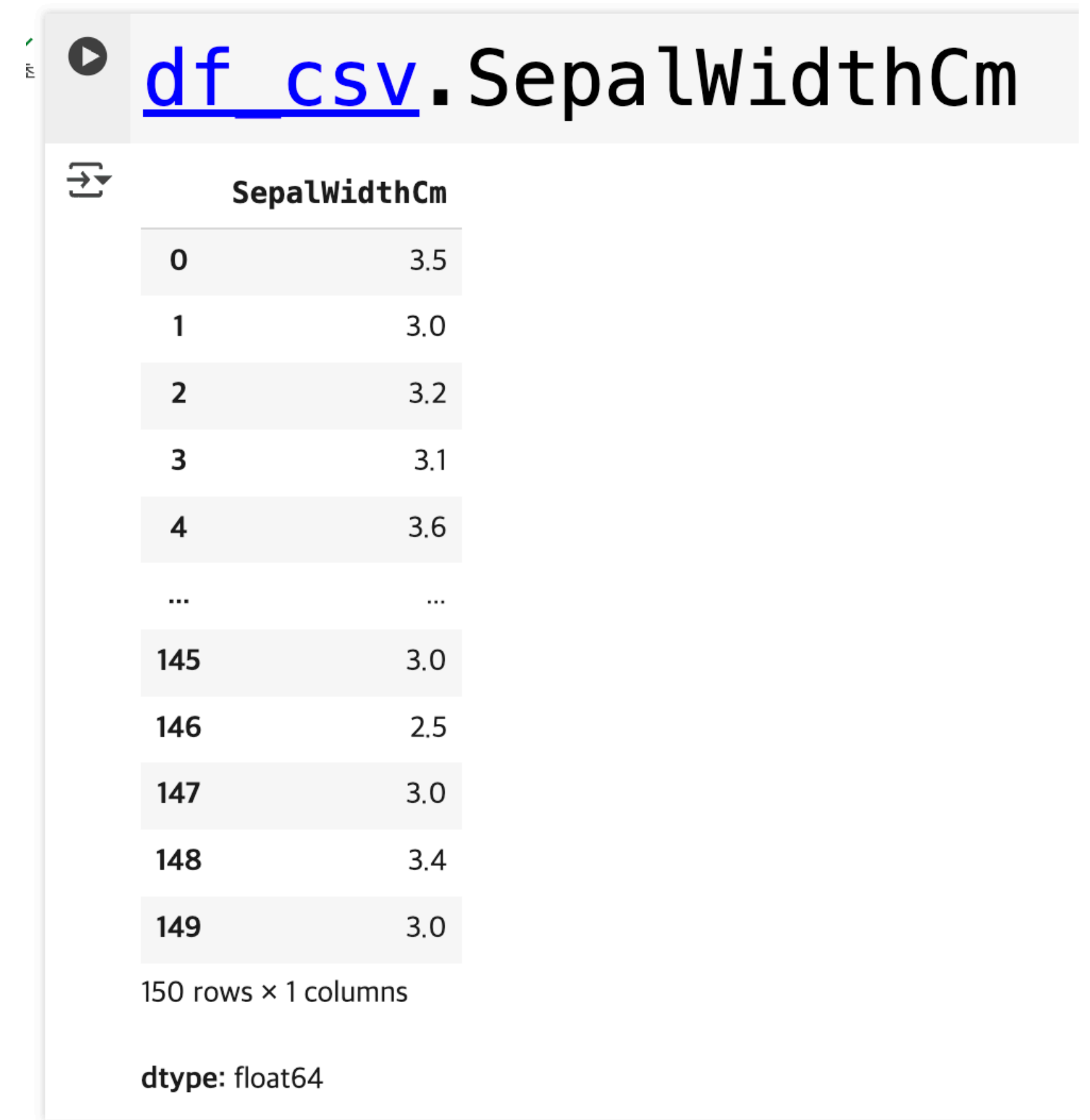


A screenshot of a Jupyter Notebook cell. The code `df_csv.iloc[10]` is entered, with the index `10` highlighted in green. Below the code, a table representing the selected row is displayed. The table has two columns: the feature names and their corresponding values. The values are: 11 for Id, 5.4 for SepalLengthCm, 3.7 for SepalWidthCm, 1.5 for PetalLengthCm, 0.2 for PetalWidthCm, and Iris-setosa for Species. At the bottom, it shows `dtype: object`.

| | |
|---------------|-------------|
| | 10 |
| Id | 11 |
| SepalLengthCm | 5.4 |
| SepalWidthCm | 3.7 |
| PetalLengthCm | 1.5 |
| PetalWidthCm | 0.2 |
| Species | Iris-setosa |
| dtype: object | |

특정 열을 선택하는 방법

- 데이터 프레임의 각 열은 특정 feature를 나타냄.
- 특정 열을 선택하는 방법: 컬럼 명
- `.columns` : column 이름 확인
- `.'컬럼 이름'` : 입력된 이름을 가진 열 선택
- `['컬럼 이름']` 또는 `[['컬럼이름1','컬럼이름2']]`



The screenshot shows a Jupyter Notebook interface. At the top, there is a code cell with the text `df_csv.SepalWidthCm`. Below the code cell, there is a table view of the resulting pandas Series. The table has two columns: an index column and a data column labeled 'SepalWidthCm'. The index values range from 0 to 149, with some rows highlighted in grey. The data values are floating-point numbers. At the bottom of the table view, it says '150 rows x 1 columns' and 'dtype: float64'.

| | SepalWidthCm |
|-----|--------------|
| 0 | 3.5 |
| 1 | 3.0 |
| 2 | 3.2 |
| 3 | 3.1 |
| 4 | 3.6 |
| ... | ... |
| 145 | 3.0 |
| 146 | 2.5 |
| 147 | 3.0 |
| 148 | 3.4 |
| 149 | 3.0 |

150 rows x 1 columns

dtype: float64

특정 행과 열을 선택하는 방법

- 인덱스 기반 : loc
- e.g., `df.loc[10, 'SepalLengthCm']` -> 10번 인덱스의 SepalLengthCm 컬럼
- 순서 기반 : iloc
- e.g., `df.iloc[10, 1]` -> 10번째 인덱스의 1번째 컬럼

```
[14] df_csv.loc[10, 'SepalLengthCm']
```

↪ np.float64(5.4)

```
df_csv.loc[:, 'SepalLengthCm']
```

```
df_csv.iloc[10, 1]
```

↪ np.float64(5.4)

특정 행과 열을 선택하는 방법

- 이를 확장해서 특정컬럼의 모든 행을 선택하는식으로 컬럼을 선택

- `df_csv.loc[:, 'SepalLengthCm']`

- 또는 특정 행의 모든 컬럼을 선택하는식도 가능


- `df_csv.loc[10, :]`

조건에 맞는 행만 선택하는법

- Boolean indexing 활용

- `a = df_csv.SepalLengthCm >= 6`

- `df_csv.loc[a, ['SepalLengthCm', 'SepalWidthCm']]`



| | SepalLengthCm | SepalWidthCm |
|-----|---------------|--------------|
| 50 | 7.0 | 3.2 |
| 51 | 6.4 | 3.2 |
| 52 | 6.9 | 3.1 |
| 54 | 6.5 | 2.8 |
| 56 | 6.3 | 3.3 |
| ... | ... | ... |
| 144 | 6.7 | 3.3 |
| 145 | 6.7 | 3.0 |
| 146 | 6.3 | 2.5 |
| 147 | 6.5 | 3.0 |
| 148 | 6.2 | 3.4 |

67 rows × 2 columns

랜덤 인덱스를 통한 데이터셋 분할

- 특정 인덱스를 갖는 행들의 데이터를 모아서 새로운 서브 데이터셋을 구성할 수 있음.
- 학습 기반 방법론 적용시 training, validation, test set 세개의 데이터셋으로 분할
- Training set : 모델 학습에 사용
- Validation set : 하이퍼파라미터 설정시 사용
- Test set : 일반화 성능 평가시 사용

트레이닝 셋

밸리데이션셋

테스트 셋

7:3, 6:2:2

```
#numpy 라이브러리
import numpy as np

#전체 샘플 수 계산
num_rows = len(df_csv)

# 트레이닝 세트와 테스트 세트 비율 설정
train_ratio = 0.8 # 트레이닝 세트 비율
test_ratio = 0.2 # 테스트 세트 비율

# 트레이닝 세트와 테스트 세트에 해당하는 인덱스 생성
train_size = int(num_rows * train_ratio)
# replace : 값을 중복해서 추출할지에 대한 값,
# 같은 샘플이 학습과 평가에 모두 사용되지 않도록함.
train_indices = np.random.choice(num_rows, train_size, replace=False)
# 나머지 샘플을 테스트셋의 인덱스로 할당
test_indices = np.setdiff1d(np.arange(num_rows), train_indices)

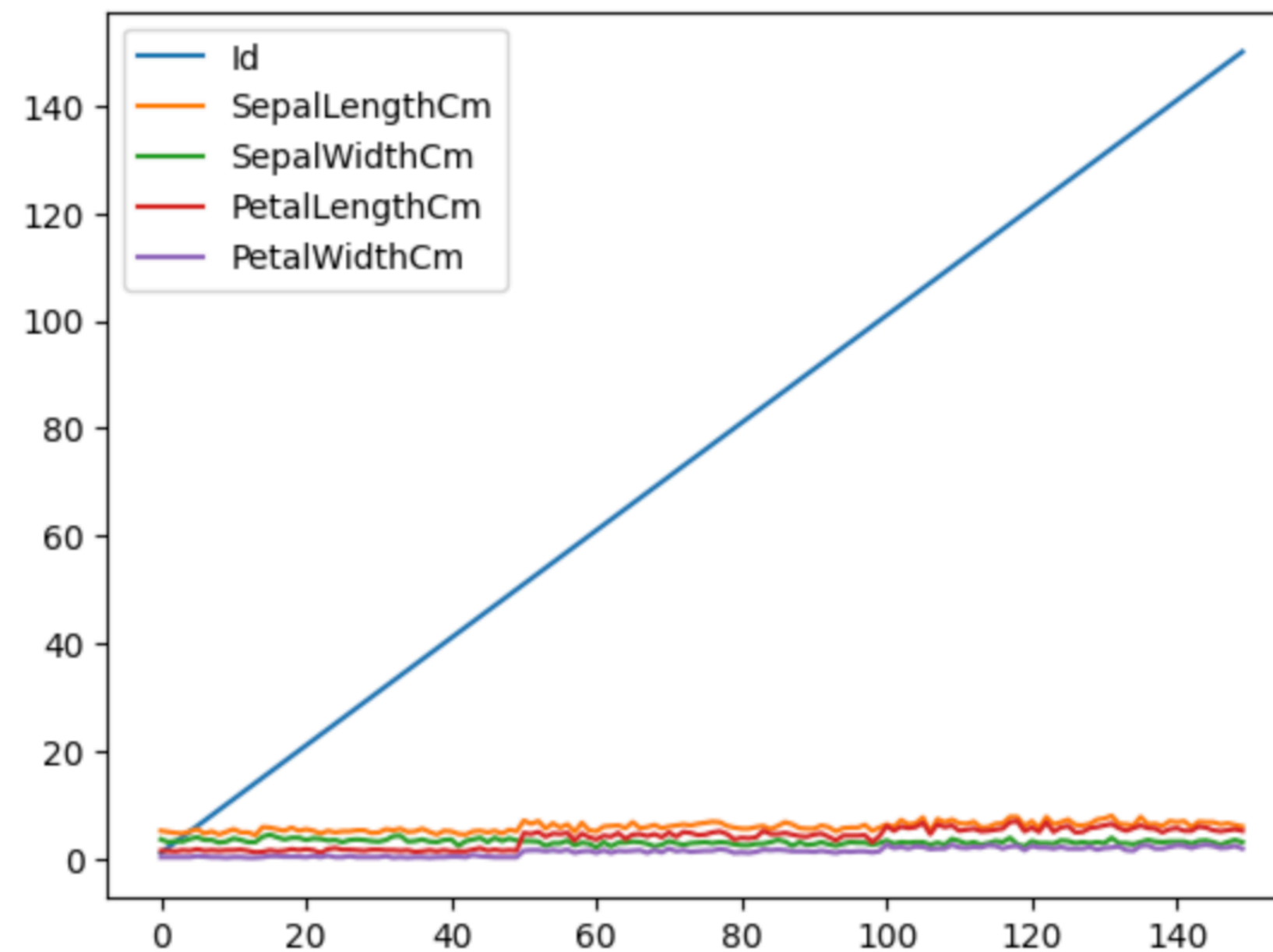
# 분리된 데이터프레임 생성
train_df = df_csv.iloc[train_indices]
test_df = df_csv.iloc[test_indices]
```

EDA를 위한 시각화

- Dataframe 또는 Series 객체에 대해 plot() 메서드를 활용하여 간단한 시각화 가능

```
df_csv.plot()
```

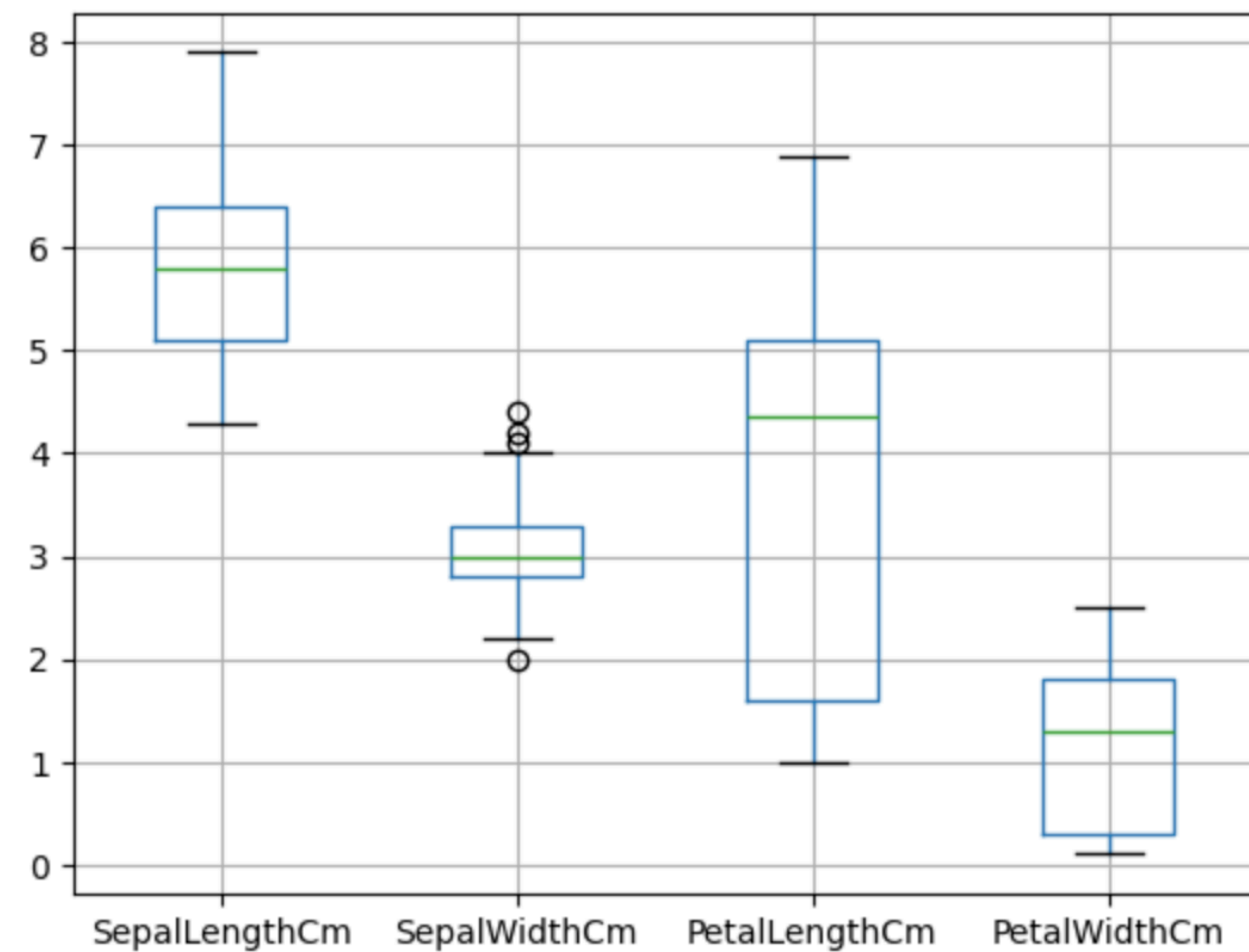
<Axes: >



EDA를 위한 시각화

```
df_csv.drop(columns=['Id']).boxplot()
```

<Axes: >

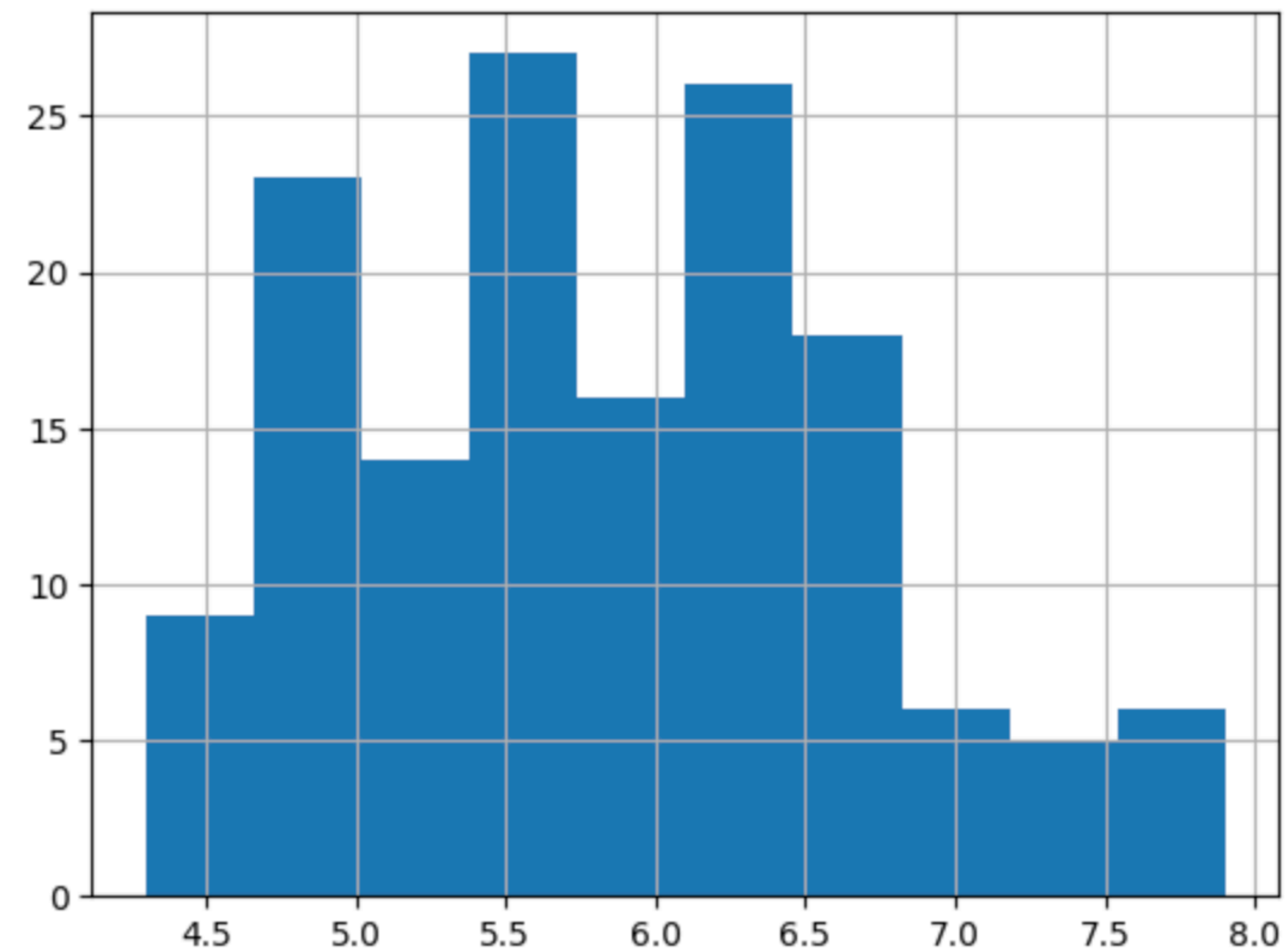


EDA를 위한 시각화

- Dataframe 또는 Series 객체에 대해 plot() 메서드를 활용하여 간단한 시각화 가능

```
df_csv.SepalLengthCm.hist()
```

<Axes: >

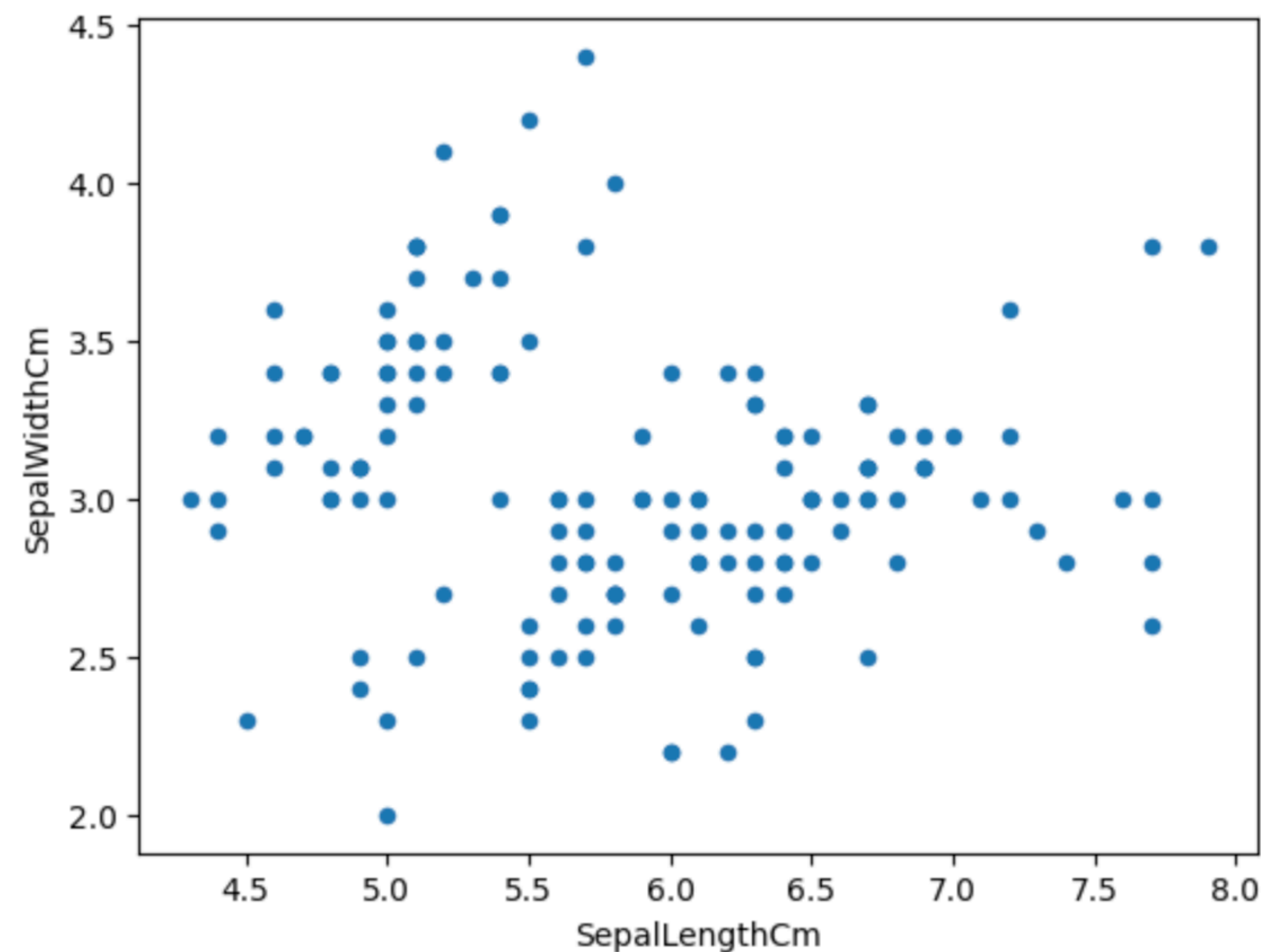


EDA를 위한 시각화

- Dataframe 또는 Series 객체에 대해 plot() 메서드를 활용하여 간단한 시각화 가능

```
df_csv.drop(columns=['Id']).plot.scatter(x='SepalLengthCm', y='SepalWidthCm')
```

<Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>



EDA를 위한 시각화

- Matplotlib.pyplot 에서 지원하는 다양한 그래프 시각화 메서드 지원
- `plot.kde()` : kernel density estimation
- `plot.bar()` : bar plot
- `plot.pie()` : pie plot
- Etc