Unit 36.

Pandas DataFrame for Data Processing

Unit learning objective (1/3)

Learning objectives

- ✓ Be able to explain the structure in terms of a DataFrame when a two-dimensional data structure is given.
- ✓ Be able to change to DataFrames stably when a dictionary object is given.
- ✓ Be able to rename rows, columns, and indexes in a DataFrame.
- ✓ Be able to create a DataFrame by importing csv, excel, json in the form of an external file.
- ✓ Be able to create DataFrames by importing existing datasets through the API.
- ✓ Be able to edit row and column information for the created DataFrame.

Unit learning objective (2/3)

Learning overview

- ✓ Understand the basic structure of a DataFrame.
- ✓ Learn how to edit the names of rows, columns, and indexes in a DataFrame.
- ✓ Learn how to converts various data objects into DataFrames.
- ✓ Learn how to obtain data from remote data service using DataReader API.
- ✓ Learn how to deletes, adds, and merges rows, columns, and data elements of a DataFrame.

Concepts you will need to know from previous units

- ✓ Understanding Pandas Series Objects
- ✓ Understanding of two types of data structures in Pandas
- ✓ How to use Matplotlib library

Keywords

- DataFrame
- Pandas I/O
- Rows in DataFrame
- Columns in DataFrame
- DataReader API
- Data Elements in DataFrame

Mission

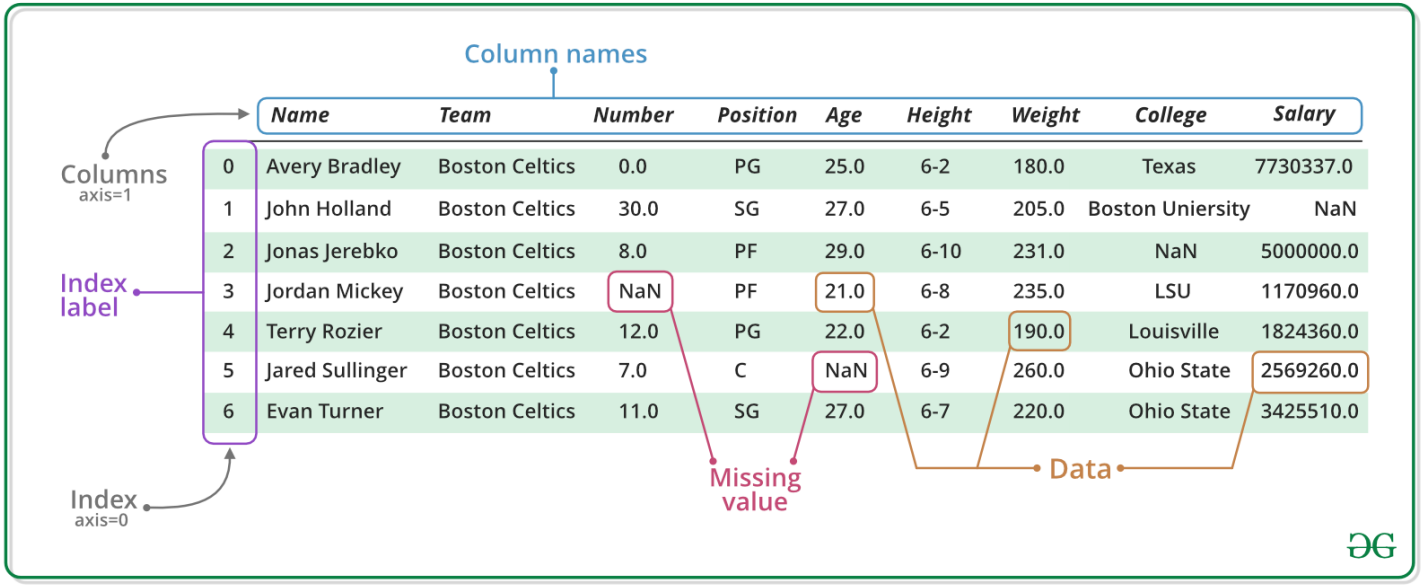
"Import external data to find and sort data elements you want."

- Search and download the 2019 Most Streamed Tracks data set from the global streaming service Spotify from Kaggle. Find out which artist has the most popular songs based on this data set!
- Also, check whether the tempo of music is correlated with popularity through scatter plot data visualization!
- Finally, create a tracklist of your favorite artist and share it with your learning colleagues by creating a playlist in Excel!

Key concept

1. DataFrame

- DataFrame is a two-dimensional label data structure made up of rows and columns. (Series is different from DataFrame in that it is a one-dimensional array.)
- Simply put, tabular data and Excel spreadsheets that are commonly encountered in data analysis are in the DataFrame format.
- Unlike a Series that can have only one value per index, a DataFrame can have multiple values per index label. At this time, each Series becomes a column of the DataFrame.



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

https://www.geeksforgeeks.org/creating-a-pandas-dataframe/

1. DataFrame

- Terms commonly used in data statistics and data science fields are summarized below.

DataFrame	The most basic spreadsheet-like data structure in statistics and machine learning models.
Feature	Typically, each column in the table is a feature. Similar terms include attributes and predictors.
Outcome	The goal of most data science projects is to predict some outcome. The feature is used for prediction. Similar terms include dependent variable, response, goal, and output.
Record	Typically, each row in a table represents one record. Similar terms include recorded value, case, case, example, observation, pattern, sample, etc.

- In other words, to define a DataFrame using the above terms, it can be basically said that it is a two-dimensional matrix consisting of a row representing each record (case) and a column representing a feature (variable).
- In order to effectively process the DataFrame object, each column is designated as an index. In the case of pandas, multi/hierarchical indexes can be set, so more complex data processing can be done effectively.
- Please visit https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html#dataframe to read more about DataFrame definitions and uses from a Python and pandas' perspective.

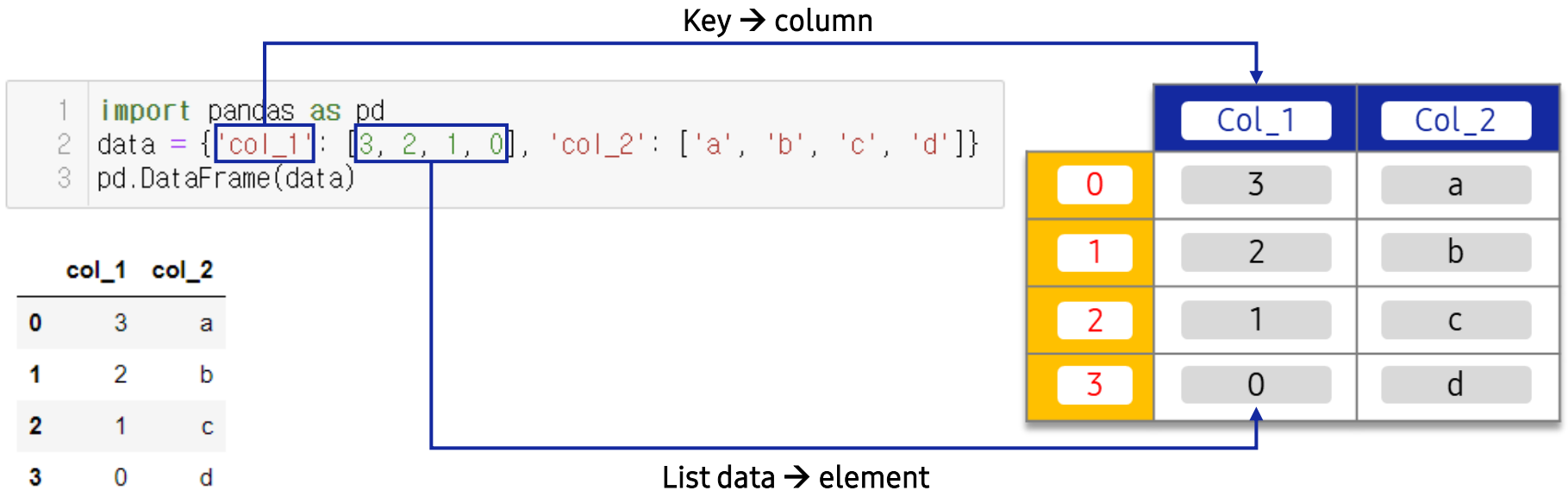
2. Creating a DataFrame

- Multiple one-dimensional arrays of the same length are combined to form a DataFrame. The same length means that the number of data elements is the same. In other words, multiple Series can be combined to create one DataFrame.
- Also, since an object that combines several Series is a dictionary, it can also be understood as "convert a dictionary into a DataFrame".

2. Creating a DataFrame

2.1. DataFrame creation basics

pandas.Dataframe[The dictionary object you want to convert]



2. Creating a DataFrame

2.1. DataFrame creation basics

pandas.Dataframe[The dictionary object you want to convert]

```
1 import pandas as pd
2 data = {'col_1': [3, 2, 1, 0], 'col_2': ['a', 'b', 'c', 'd']}
3 pd.DataFrame(data)
```

	col_1	col_2
0	3	a
1	2	b
2	1	c
3	0	d

Line 3

- Convert a dictionary called data to a DataFrame

2. Creating a DataFrame

2.1. DataFrame creation basics

pandas.Dataframe[The dictionary object you want to convert]

```
1 import pandas as pd
2 s= pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"])
3 s2= pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"])
4
5 data = {"one":s, "two":s2}
6
7 pd.DataFrame(data)
8
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

Line 2, 3

- 2: Store in series object s
- 3: Store in another series object s2

2. Creating a DataFrame

2.1. DataFrame creation basics

pandas.Dataframe[The dictionary object you want to convert]

```
1 import pandas as pd
2 s= pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"])
3 s2= pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"])
4
5 data = {"one":s, "two":s2}
6
7 pd.DataFrame(data)
8
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

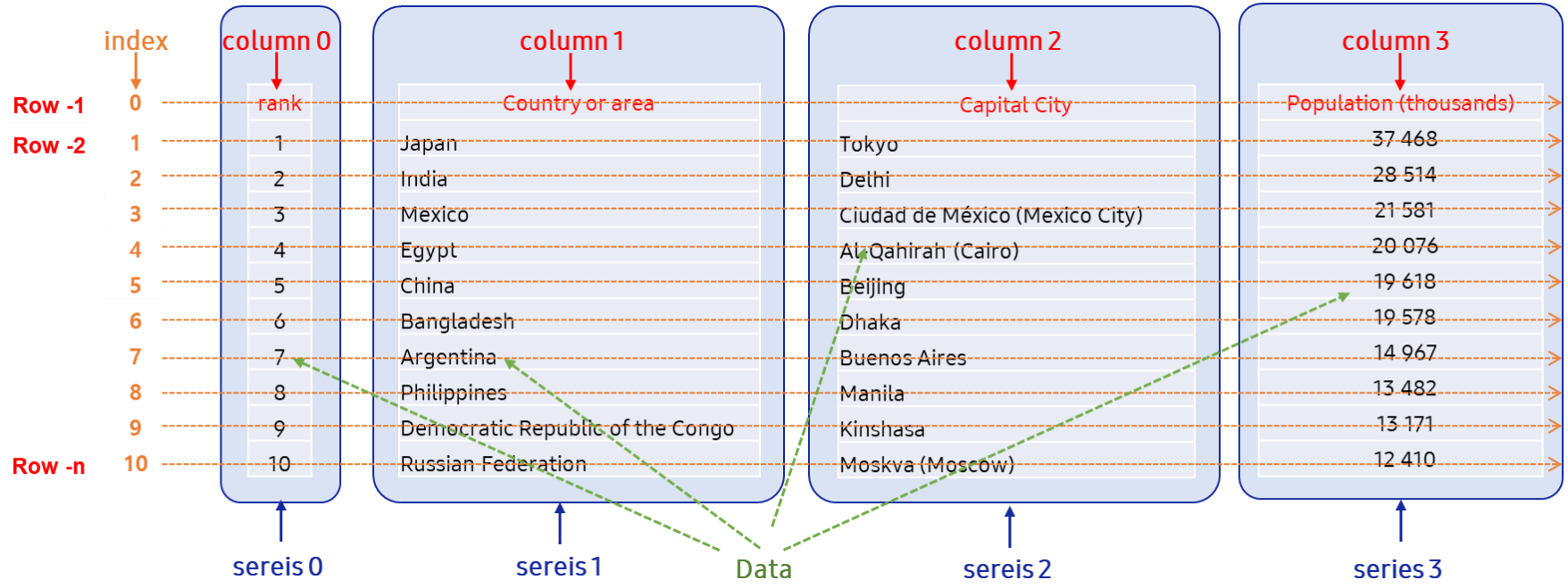
Line 5, 7, 8

- 5: Import two Series and assign key names one, two
- 7: When the DataFrame is output, the key name of the called dictionary becomes the column name.
- 8: NaN occurs because the number of elements in the data does not match when two Series are imported.

2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

- You can set the row and column names separately when creating a DataFrame or after creating it.
- First, understand the structure and naming rules of DataFrames through the images below.
- As described in the image, each column of a DataFrame is a series object, and these series objects have a matrix structure where they are combined based on the index of the same row.



2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

- Change all row indexes: **DataFrame object name**.index = **Array of row indexes to change**
- Change all column names (columns name): **DataFrame object name**.columns = **Array of new column names**

```
1  import pandas as pd
2  d = {'col1': [1, 2], 'col2': [3, 4]}
3  df = pd.DataFrame(data=d)
4  print(df)
5
6  df.index = ["new index0", "new index1"]
7  print(df)
8
9  df.columns = ["new_col1", "new_col2"]
10
11 print(df)
```

```
   col1  col2
0     1     3
1     2     4
            col1  col2
new index0     1     3
new index1     2     4
            new_col1  new_col2
new index0         1         3
new index1         2         4
```

Line 2, 3, 4, 6, 7, 9, 11

- 2: It is a dictionary data type.
- 3: Create a DataFrame object with the name df by assigning a dictionary.
- 4: Check the DataFrame created before the name change.
- 6: Change to new index name
- 7: Check the result of index change
- 9: Change to new column name
- 11: Check the result of column name change

As you can see from the previous code execution result, when you change the index or column name, you can see that the DataFrame before the change is not maintained, but the data frame is changed with the changed index and column name.

2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

- Select part of a row and rename it: **DataFrame object name.rename(index={existing index: index to be replaced with new one, ...})**
- Select part of a column and rename it: **DataFrame object name.rename(columns={Existing column name: new column name, ...})**

2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

```
1 import pandas as pd
2 d = {'col1': [1, 2], 'col2': [3, 4]}
3 df = pd.DataFrame(data=d)
4 print(df)
5
6 df.rename(index= {0:"new index0"})
```

```
   col1  col2
0     1     3
1     2     4
```

	col1	col2
new index0	1	3
1	2	4

Line 4, 6

- 4: Check the DataFrame created before the name change.
- 6: Change index 0 to a new name, new index0.

2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

- Select part of a row and rename it: **DataFrame object name.rename(index={existing index: index to be replaced with new one, ...})**
- Select part of a column and rename it: **DataFrame object name.rename(columns={Existing column name: new column name, ...})**

```
1 df.rename(columns ={"col2":"new col2"})
```

	col1	new col2
0	1	3
1	2	4

Line 1

- Change the column name col2 to the new name new col2.

If you see the result of executing the code above, it is different from when the entire index and column name were changed. You can see that it is returning a new DataFrame object rather than changing the original itself. You must use inplace = True to change the original object.

2. Creating a DataFrame

2.2. Set the name of the row (index) / column (columns) of the DataFrame

- Select part of a row and rename it: **DataFrame object name.rename(index={existing index: index to be replaced with new one, ...})**
- Select part of a column and rename it: **DataFrame object name.rename(columns={Existing column name: new column name, ...})**

```
1 import pandas as pd
2 d = {'col1': [1, 2], 'col2': [3, 4]}
3 df = pd.DataFrame(data=d)
4 df.rename(index= {0:"new index0"}, inplace=True)
5 df.rename(columns ={"col2":"new col2"})
```

	col1	new col2
new index0	1	3
1	2	4

Line 4, 5

- 4: Make changes to the original DataFrame object itself through the inplace = True option.
- 5: Although only the column name has been changed, you can see that the original object itself has been changed through the code above.

One More Step

- Another alternative to converting a dictionary to a DataFrame is to use .from_dic().
  - A dictionary of array-like objects
  - dictionary

```
pd.DataFrame.from_dict(dictionary object to convert, orient = 'columns')
```

- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.from_dict.html#pandas-dataframe-from-dict

One More Step

```python
import pandas as pd
s= pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"])
s2= pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"])

dic_data = {"one":s, "two":s2}

type(dic_data)

pd.DataFrame(dic_data)

pd.DataFrame.from_dict(dic_data, orient = 'columns')
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

One More Step

```
1  pd.DataFrame.from_dict(dic_data, orient = 'index')
```

	a	b	c	d
one	1.0	2.0	3.0	NaN
two	1.0	2.0	3.0	4.0

```
1  data = {'row_1': [3, 2, 1, 0], 'row_2': ['a', 'b', 'c', 'd']}
2  pd.DataFrame.from_dict(data, orient='index', columns=['A', 'B', 'C', 'D'])
```

	A	B	C	D
row_1	3	2	1	0
row_2	a	b	c	d

3. Converting various data objects to DataFrame

3.1. The pandas I/O API

It is sometimes expressed that the conversion of other data types such as a dictionary into a DataFrame is input to the DataFrame. Data is input and converted using various IO tools (text, CSV, HDF5, ...) of pandas.

Ex Use the top-level "read" functions that convert a file to a DataFrame object, such as pandas.read_csv().

3. Converting various data objects to DataFrame

3.1. The pandas I/O API

The list of IO tools organized in the panda's official documentation is as follows.

(https://pandas.pydata.org/docs/user_guide/io.html)

Format Type	Data Description	Reader	Writer	Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv	binary	Feather Format	read_feather	to_feather
text	Fixed-Width Text File	read_fwf		binary	Parquet Format	read_parquet	to_parquet
text	JSON	read_json	to_json	binary	ORC Format	read_orc	
text	HTML	read_html	to_html	binary	Stata	read_stata	to_stata
text	LaTeX		Styler.to_latex	binary	SAS	read_sas	
text	XML	read_xml	to_xml	binary	SPSS	read_spss	
text	Local Clipboard	read_clipboard	to_clipboard	binary	Python Pickle Format	read_pickle	to_pickle
binary	MS Excel	read_excel	to_excel	SQL	SQL	read_sql	to_sql
binary	OpenDocument	read_excel		SQL	Google BigQuery	read_gbq	to_gbq
binary	HDF5 Format	read_hdf	to-hdf				

3. Converting various data objects to DataFrame

3.2. About file paths

- One of the many mistakes beginners make while learning Python for data processing in the first place is entering the wrong path to the files to load data.
- You can leave it as the basics of computing, but it's helpful to make sure you clean up the file paths once. Paths of files in the local computer can be expressed in two ways.
  - **Absolute path**: This method uses all paths from the first starting point (start of OS) to the file.

    Ex The OS uses the Windows system. If you find "sample.txt" on the desktop, it is C:\Users\UserID\Desktop\sample.txt. No matter which operating system (OS) is used as well as Windows, it is to find the file with an absolute path that contains all the paths passed through from the top-level root.

  - **Relative path**: It is the "location of the file you want to find" based on "the current location".

```
/ : Move to the top-level directory (root)
./ : Current same directory, can be deleted
../ : Parent directory from my current location (file being created)
../../ : Directory two levels higher
```

3. Converting various data objects to DataFrame

3.2. About file paths

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current page
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

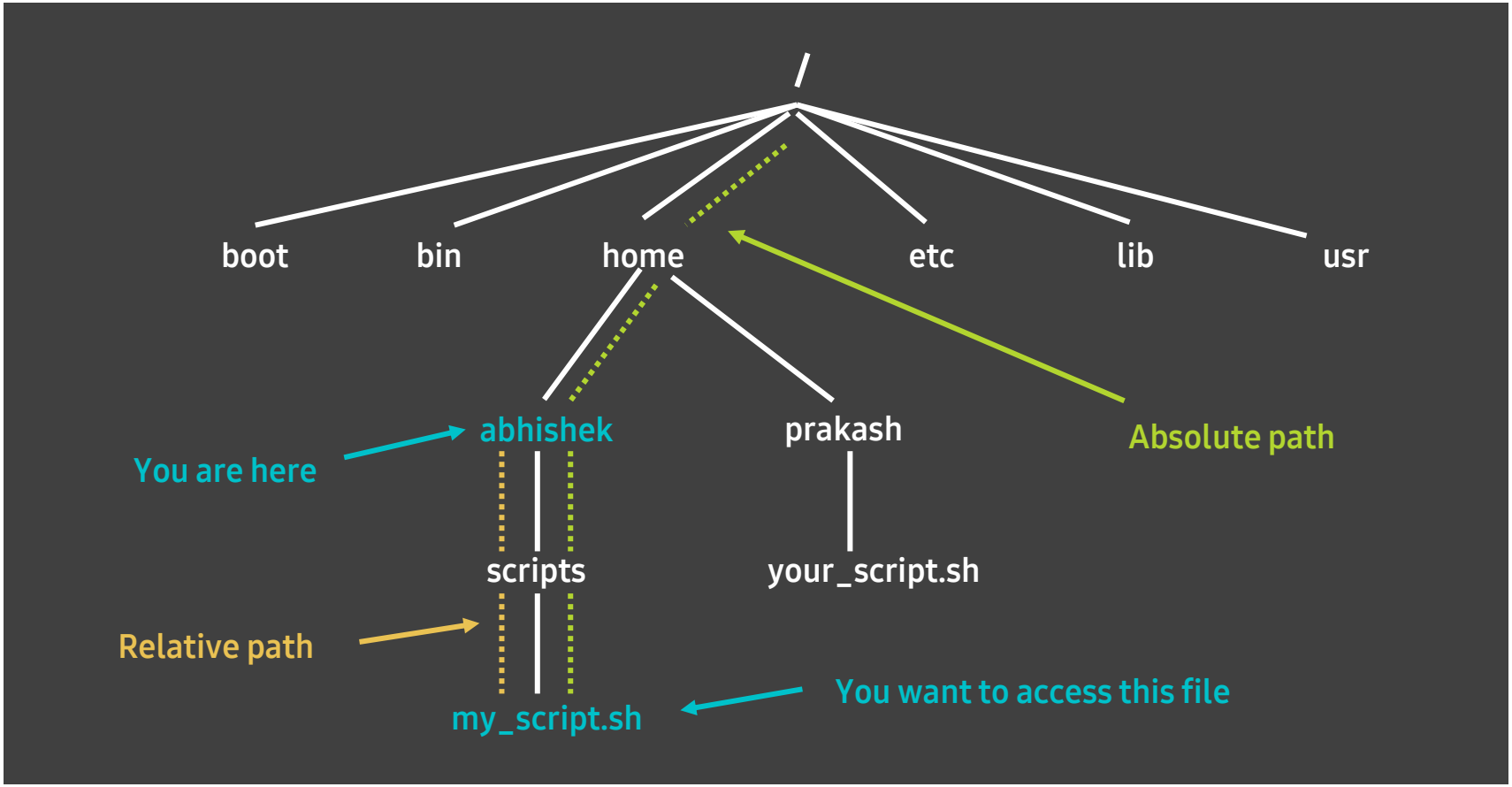
Why do we need relative paths?

- n absolute path is a static string that tells you exactly where a file is located on a specific computer. However, when dealing with paths, this static feature may come as a disadvantage.

  Ex What if the path of test.txt is always changed frequently, or if the root directory covers different OSs? Because of the static characteristics, the former needs to rewrite all documents written with absolute paths, and the latter has to create and manage absolute paths for each OS.

3. Converting various data objects to DataFrame

3.2. About file paths



3. Converting various data objects to DataFrame

3.3. dataframe.head() for data review

- It returns the row (rows) from the first to n items of the DataFrame. The main reason for using this is to check the result of the DataFrame currently being processed while data is being processed.
- We will deal with it now because it will be most used when handling the DataFrame that we will learn soon.

3. Converting various data objects to DataFrame

3.3. dataframe.head() for data review

DataFrame.head(n)

```
1  import pandas as pd
2
3  df = pd.DataFrame({'animal': ['alligator', 'bee', 'falcon', 'lion',
4                     'monkey', 'parrot', 'shark', 'whale', 'zebra']})
5
6  df
```

	animal
0	alligator
1	bee
2	falcon
3	lion
4	monkey
5	parrot
6	shark
7	whale
8	zebra

3. Converting various data objects to DataFrame

3.3. dataframe.head() for data review

DataFrame.head(n)

```
1 df.head()
```

	animal
0	alligator
1	bee
2	falcon
3	lion
4	monkey

Line 1

- Return data from top to index 5 in a DataFrame named df. If no number is entered, only 5 are output.

3. Converting various data objects to DataFrame

3.3. dataframe.head() for data review

DataFrame.head(n)

```
1 df.head(3)
```

	animal
0	alligator
1	bee
2	falcon

Line 1

- Return data from the top as many rows as the number is entered in head().

3. Converting various data objects to DataFrame

3.3. dataframe.head() for data review

DataFrame.head(n)

```
1 df.head(-3)
```

	animal
0	alligator
1	bee
2	falcon
3	lion
4	monkey
5	parrot

Line 1

- If "-" is input, data is returned after excluding 3 rows from the bottom.

TIP

- A DataFrame is a data object consisting of a two-dimensional array. If you use df.shape, you can check how many dimensions the data frame consists of.
- It returns the dimensionality of the data frame in the form of a tuple.

```
1 df.shape
```

```
(9, 1)
```

TIP

- Other functions for reviewing data

```
1 len(df)
```

```
9
```

Line 1

- Use the len() function to know the length of a row.

```
1 df.columns
```

```
Index(['animal'], dtype='object')
```

Line 1

- Print the columns.

TIP

- Other functions for reviewing data

```
1 df.values
```

```
array([['alligator'],
       ['bee'],
       ['falcon'],
       ['lion'],
       ['monkey'],
       ['parrot'],
       ['shark'],
       ['whale'],
       ['zebra']], dtype=object)
```

Line 1

- Only the values of the data frame are output. Actually, it isn't used much.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

In csv, data is separated by a comma (,) so the name is csv (comma -separated values).

- Separate columns with commas
- Separate rows with newlines

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(”Path")
```

1) Default

- Download the Titanic Survivor data file from https://www.kaggle.com/c/titanic/data

```
1 import pandas as pd
2 titanic = pd.read_csv("./data//titanic/train.csv")
```

Line 2

- Enter the path of the downloaded file as a relative path.

```
1 type(titanic)
```

```
pandas.core.frame.DataFrame
```

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

1) Default

```
1 titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Line 1

- Output only the data in the 5th row from the top and review.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

pandas.read_csv("Path")

1) Default

TIP

- Dataframe can search not only data but also metadata information such as column type, number of null data, and data distribution. You can use info() and describe().

```
1 titanic.info()
```

Line 1

- This is a method that can check the total number of DataFrames and data types of the imported DataFrame. When preprocessing data, it is recommended to check through this method first as a habit.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(”Path")
```

1) Default

TIP

```
<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
```

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

1) Default

TIP

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Rangeindex: You can see the total number of rows (417) and number of columns (11) in the range of the DataFrame Index.

Data type of each column

(object can be thought of as a string)

A summary of all column information

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(Path")
```

1) Default

TIP

```
1 titanic.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(”Path")
```

1) Default

TIP

Line 1

- You can roughly check the data distribution of the imported DataFrame.

Ex When using this data in machine learning, it is very important to start by knowing the distribution of the data to improve performance. It is recommended that you use this method to get a rough distribution diagram as a habit.

mean	Average value of all data
std	Standard Deviation
min	Minimum value
max	Maximum value
25%	25 percentible value
50%	50 percentible value
75%	75 percentible value

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(”Path")
```

1) Default

Notes when loading a csv file into a DataFrame

- Be sure to open the file and check the structure before loading the file into the code: Check how the data set is structured and set the appropriate parameters.
- Most csv files use , as a delimiter to separate data. However, sometimes there are files that are separated by tabs.

→ In this case, use sep = '\t' through the sep parameter. If the file delimiter is '|', add sep='|'.

Enter the number of rows to read in the nrows parameter.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

2) skiprow

- When importing a file, it specifies whether to skip the first few lines and import. It can also be set as a list containing the number of lines to skip.

  Ex [1, 4, 7]

```
1 titanic = pd.read_csv("./data//titanic/test.csv", skiprows= 2)
2
3 titanic.head(5)
```

	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	Unnamed: 9	S
0	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
1	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
2	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
3	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
4	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
1 titanic = pd.read_csv("./data//titanic/test.csv", skiprows= 2)
2
3 titanic.head(5)
```

	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	Unnamed: 9	S
0	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
1	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
2	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
3	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
4	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q

Line 1

- You can specify a row to be the column name.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

3) header

- Check the csv file before loading it in the code, and then set the row you want to specify as the column name. By default, the file is loaded as it is, and the row at index 0 of the csv becomes the column header.

```
1 titanic = pd.read_csv("./data//titanic/test.csv", header= 2)
2
3 titanic.head(5)
```

	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	Unnamed: 9	S
0	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
1	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
2	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
3	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
4	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
1 titanic = pd.read_csv("./data//titanic/test.csv", header= 2)
2
3 titanic.head(5)
```

	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	Unnamed: 9	S
0	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
1	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
2	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
3	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
4	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q

Line 1

- You can specify a row to be the column name.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

3) header

```
titanic = pd.read_csv("./data//titanic/test.csv", header=0)
```

header = 0 (default)

header = 2

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	Unnamed: 9	S
0	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
1	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
2	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

3) header

header = None →

	0	1	2	3	4	5	6	7	8	9	10
0	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
2	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	NaN	S
3	894	2	Myles, Mr. Thomas Francis	male	62	0	0	240276	9.6875	NaN	Q
4	895	3	Wirz, Mr. Albert	male	27	0	0	315154	8.6625	NaN	S

headheader= None ,
names= ['a','b','c','d','e','f','g','h','i','j', 'k'] →

	a	b	c	d	e	f	g	h	i	j	k
0	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

pandas.read_csv("Path")

3) header

- When should "header = None" be used? After checking the csv, it is used when starting from data rather than a separate name from the first row. And in this case, you can specify the header with the name parameter.
  - names = [List to use as column names]

```
1 titanic = pd.read_csv("./data//titanic/test.csv", header=None  , names= ['a','b','c','d','e','f','g','h','i','j','k'])
2 titanic.head(2)
```

	a	b	c	d	e	f	g	h	i	j	k
0	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q

Line 1

- You can specify a row to be the column name.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

pandas.read_csv("Path")

3) header

```
1 titanic = pd.read_csv("./data//titanic/test.csv", header= None , names= ['a','b','c','d','e','f','g','h','i','j','k','l', 'm', 'n'])
2 titanic.head(2)
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n
0	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	NaN	NaN	NaN
1	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	NaN	NaN	NaN

Line 1, 2

- 1: You can specify a row that becomes a column name.
- 2: Return NaN when naming more columns than the data object has.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv(”Path")
```

4) encoding

- It can be used both to read or write to the file.
- It is easy to understand that encoding is a process of transforming languages other than the Ascii-series string (which can be expressed as 0 to 127), such as Hangul by adding bytes to the computer for use. However, there are several ways for this encoding.
- It functions to specify the encoding type of text when loading a csv file.

  Ex encoding = 'utf-8'
- There may be cases where the imported text looks broken even with this option. In this case, the easiest solution is to specify the data format as utf-8 in Excel and save it.
  - See https://docs.python.org/3/library/codecs.html#standard-encodings for Python standard encoding information.

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

```
pandas.read_csv("Path")
```

5) index_col

- Specifies the column to be used as the row.

```
1 titanic = pd.read_csv("./data//titanic/test.csv")
2 titanic.head(2)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S

4. Converting a csv file to a DataFrame

4.1. Reading a file and creating a DataFrame object

pandas.read_csv("Path")

5) index_col

- Specifies the column to be used as the row.

```
1 titanic = pd.read_csv("./data//titanic/test.csv", index_col='Name')
2 titanic.head(2)
```

Name	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Kelly, Mr. James	892	3	male	34.5	0	0	330911	7.8292	NaN	Q
Wilkes, Mrs. James (Ellen Needs)	893	3	female	47.0	1	0	363272	7.0000	NaN	S

5. Converting Excel file to DataFrame

5.1. Reading a file and creating a DataFrame object

Rows and columns in Excel have a one-to-one correspondence between rows and columns in DataFrame.

```
pandas.read_excel(Path, sheet_name = 0 , header= 1)
```

- Download data on NBA players in the american professional basketball from https://www.kaggle.com/justinas/nba-players-data
- Due to the characteristics of the Excel file, you can select and load a specific sheet of the file to be imported.
- The sheet_name parameter is used, and a string consisting of the name of the actual sheet or a list of integers starting from 0 can be used.

  Ex If sheet_name = [0,1,2,"names"] is specified, the first, second, third, and sheet names of "names" are all loaded when Excel is imported.

5. Converting Excel file to DataFrame

5.1. Reading a file and creating a DataFrame object

pandas.read_excel(Path, sheet_name = 0 , header= 1)

```
1 import pandas as pd
2 df= pd.read_excel("./data//NBA/Players.xls", sheet_name = 0, header =0)
3
4 df.head()
```

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

Line 2, 4

- 2: header specifies the order of the columns to be specified as columns.
- 4: Output only the data of 5 rows from the top and review.

5. Converting Excel file to DataFrame

5.1. Reading a file and creating a DataFrame object

```
pandas.read_excel(Path, sheet_name = 0 , header= 1)
```

Warning

- To use the read_excel function, the optional dependency xlrd package must be installed in advance.

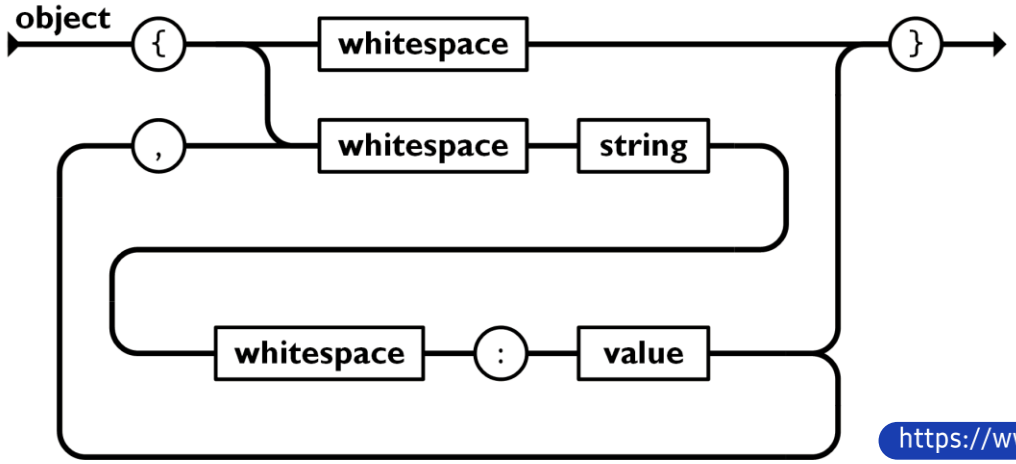
  cond install xlrd

You can check various parameters that can be used when loading a file like csv from
https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html?highlight=read_excel

6. Converting JSON File to DataFrame

6.1. JSON

- JSON (JavaScript Object Notation) is a format created for sharing data. It consists of attribute-value pairs or key-value pairs. attribute–value pairs and array data types (or any other serializable value))
- How to use : "Dataname": value
- It contains strings, numbers, arrays, booleans, and other objects, which are the basic data types of JavaScript. In this way, the data layer can be configured as shown in the example picture below.
  - If created as a file, the file extension is ".json"
  - Content-Type of HTTP request is "application/json"



https://www.json.org/json-en.html

6. Converting JSON File to DataFrame

6.2. JSON object

JSON data consists of name-value pairs. JSON data is listed with commas (,).

```
1 {
2     "name": "cherry",
3     "family": "Shih Tzu",
4     "age": 14,
5     "weight": 4.5
6 }
```

```
{'name': 'cherry', 'family': 'Shih Tzu', 'age': 14, 'weight': 4.5}
```

Line 1

- JSON object

6. Converting JSON File to DataFrame

6.3. JSON array

- It includes multiple json data using commas.
- An object is expressed by enclosing it in curly braces ({}). An array is expressed by enclosing it in square brackets ([]).

```
1 [
2     {"name": "cherry","family": "Shih Tzu","age": 14,"weight": 4.5},
3     {"name": "cola","family": "Terrier","age": 1,"weight": 2.5},
4     {"name": "baby","family": "Maltese","age": 14,"weight": 4.5}
5 ]
```

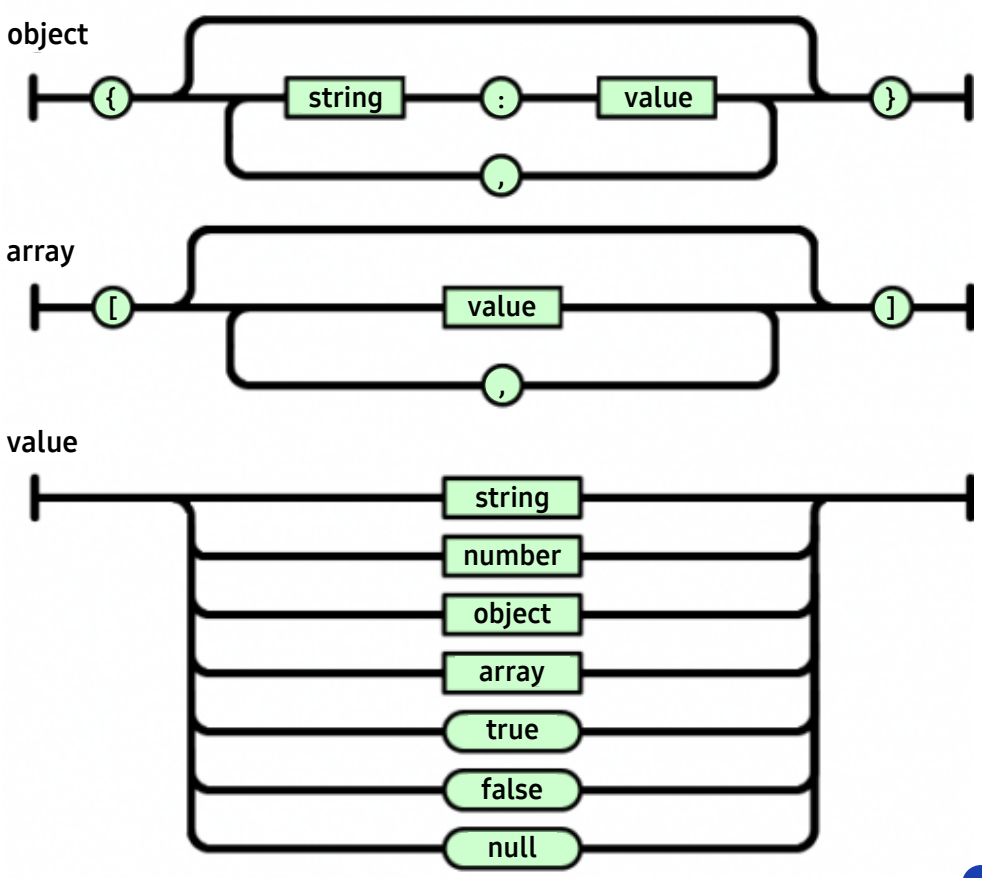
```
[{'name': 'cherry', 'family': 'Shih Tzu', 'age': 14, 'weight': 4.5},
 {'name': 'cola', 'family': 'Terrier', 'age': 1, 'weight': 2.5},
 {'name': 'baby', 'family': 'Maltese', 'age': 14, 'weight': 4.5}]
```

Line 1

- JSON array

6. Converting JSON File to DataFrame

6.3. JSON array



https://www.json.org/json-en.html

6. Converting JSON File to DataFrame

6.4. Reading a file and creating a DataFrame object

pandas.read_json(Path, orient=None)

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
5
6 df.head()
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
0	1	University of Oxford	United Kingdom	20,774	11.1	41%	46 : 54
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61

Line 4

- A url can also be used in the path.

6. Converting JSON File to DataFrame

6.4. Reading a file and creating a DataFrame object

Orient parameter

- The characteristic of JSON object is JSON object or its complex form, JSON array form. Also, it can be composed of various hierarchical structures.
- Just like we checked the structure by opening csv or excel file before loading the file, we also open JSON to check the structure of the string first. And then, select the option that suits the structure and use them. Rather than memorizing all the conditions and using them, it is a good way to experiment while substituting them one by one. Default value is None.

'**columns**': {columns: {index: value, ...}, ...}

'**split**': {"index" : [index, ...], "columns" : [column, ...], "data": [value, ...]}

'**records**': [{column:value}, .. ,{column:value}]

'**index**': {index " {column: value, ...}, ...}

'**values**': [values, ...] just the values array

Warning

- You should remember that what we are learning here is loading the json object into a DataFrame using pandas. It is different from encoding to load json into Python and convert it to a Python string. Hope you don't get confused.

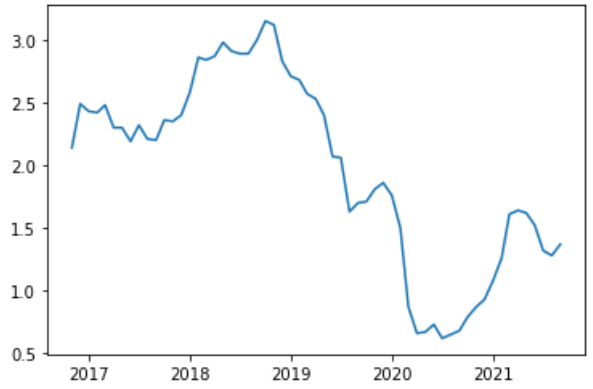
7. Reading data from remote data service using DataReader API

- To use a module, install it first.
  - **pip install pandas-DataReader**
- The DataReader package supports access to a variety of useful data sources. The most useful thing is that it can be processed directly into a DataFrame.

  Ex With just two lines of code, you can get data on 5-years of 10-year constant maturity yields on U.S. government bonds.

7. Reading data from remote data service using DataReader API

```
1 import matplotlib.pyplot as plt
2 import pandas_datareader as pdr
3
4 df = pdr.get_data_fred('GS10')
5
6 plt.plot(df)
7
8 plt.show()
```

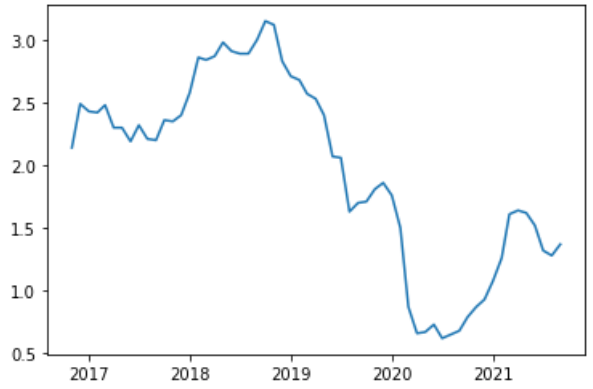


Line 2, 4

- 2: Import the package.
- 4: Save the data of sysmbol called GS10 in Federal Reserve Economic Data (FRED) data as a DataFrame.

7. Reading data from remote data service using DataReader API

```
1  import matplotlib.pyplot as plt
2  import pandas_datareader as pdr
3
4  df = pdr.get_data_fred('GS10')
5
6  plt.plot(df)
7
8  plt.show()
```

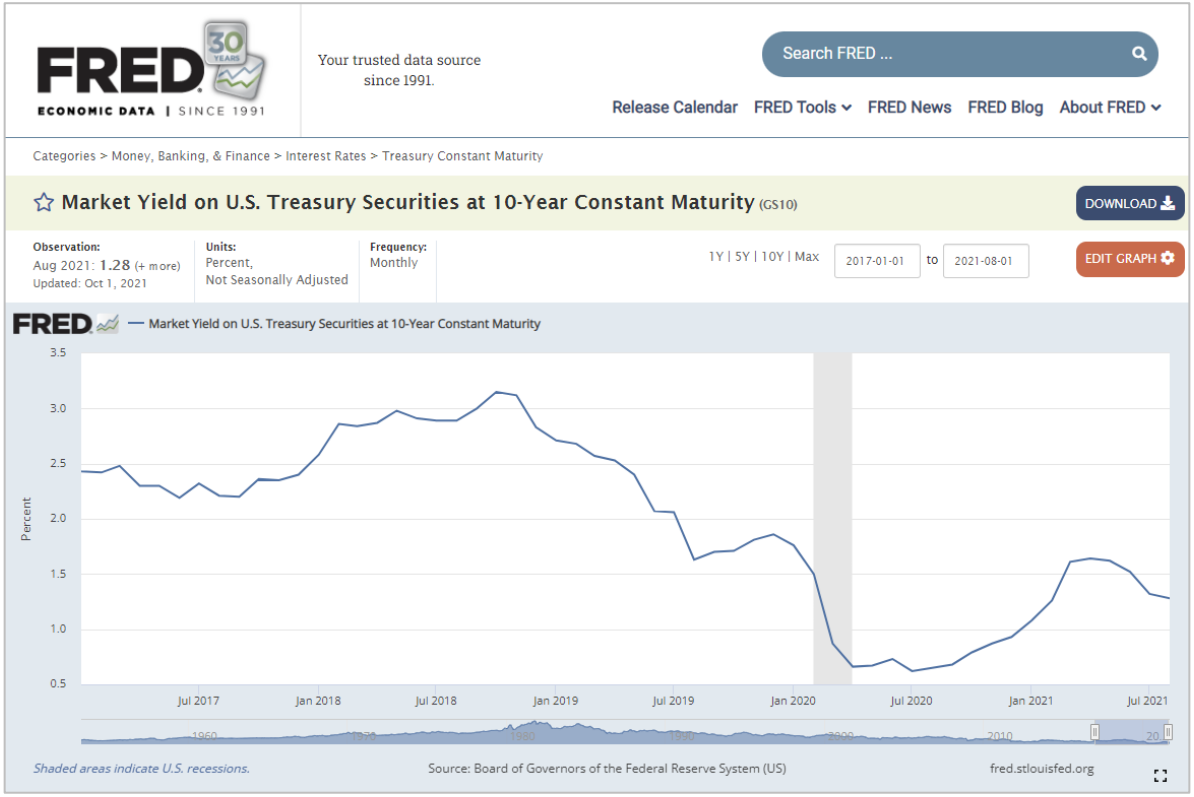


Line 6

- Visualize the DataFrame as a line graph.

7. Reading data from remote data service using DataReader API

- Same data set as https://fred.stlouisfed.org/series/GS10. Since the period is 5 years of yield, if you want to compare only data from 2017, you can specify the graph search period from January 1, 2017. Compare the graph we processed with the official graph of FRED. Is it the same?



7. Reading data from remote data service using DataReader API

- Remote data access is possible as follows. (The following data feeds are available:) However, depending on the type of data, you need to register an API Key, so be sure to check the usage for each specification.

  Ex Tiingo is a tracking platform that provides a data API by the closing prices for stocks, mutual funds, and ETFs. A free registration is required to obtain an API key. Free accounts have limited fees and access to a limited number of symbols (500 at the time of writing).

- For technical specifications of each data, visit https://pandas-datareader.readthedocs.io/en/latest/remote_data.html#remote-data-access to check detailed information.

- Tiingo
- IEX
- Alpha Vantage
- Econdb
- Enigma
- Quandl
- St.Louis FED (FRED)
- Kenneth French's data library
- World Bank
- OECD
- Eurostat
- Thrift Savings Plan
- Nasdaq Trader symbol definitions
- Stooq
- MOEX
- Naver Finance
- Yahoo Finance

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

- World Bank supports thousands of datasets through DataReader. You can see the World Bank's data catalog at https://datacatalog.worldbank.org/
- The World Bank dataset is identified through an indicator. Currently, there are about 1,897 identifiers. You can check it with the wb.get_indicators() function.

TIP

- It provides guidance on how to use DataReader based on World Bank data, but you can use it sufficiently for other data by finding the link below in the official document. We will deal with indicators as a sample, but I want you to learn while thinking, "I can find and use it this way" while comparing it with the official documentation. https://pandas-datareader.readthedocs.io/en/latest/remote_data.html#indicators

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 import pandas_datareader as pdr
2 from pandas_datareader import wb  #
3
4 all_indicators = pdr.wb.get_indicators() #
5
6 all_indicators.iloc[:5, :2]
```

	id	name
0	1.0.HCount.1.90usd	Poverty Headcount ($1.90 a day)
1	1.0.HCount.2.5usd	Poverty Headcount ($2.50 a day)
2	1.0.HCount.Mid10to50	Middle Class ($10-50 a day) Headcount
3	1.0.HCount.Ofcl	Official Moderate Poverty Rate-National
4	1.0.HCount.Poor4uds	Poverty Headcount ($4 a day)

Line 2, 4

- 2: Import the package.
- 4: The wb.get_indicators() function can get the full list of indicators.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 import pandas_datareader as pdr
2 from pandas_datareader import wb  #
3
4 all_indicators = pdr.wb.get_indicators() #
5
6 all_indicators.iloc[:5, :2]
```

	id	name
0	1.0.HCount.1.90usd	Poverty Headcount ($1.90 a day)
1	1.0.HCount.2.5usd	Poverty Headcount ($2.50 a day)
2	1.0.HCount.Mid10to50	Middle Class ($10-50 a day) Headcount
3	1.0.HCount.Ofcl	Official Moderate Poverty Rate-National
4	1.0.HCount.Poor4uds	Poverty Headcount ($4 a day)

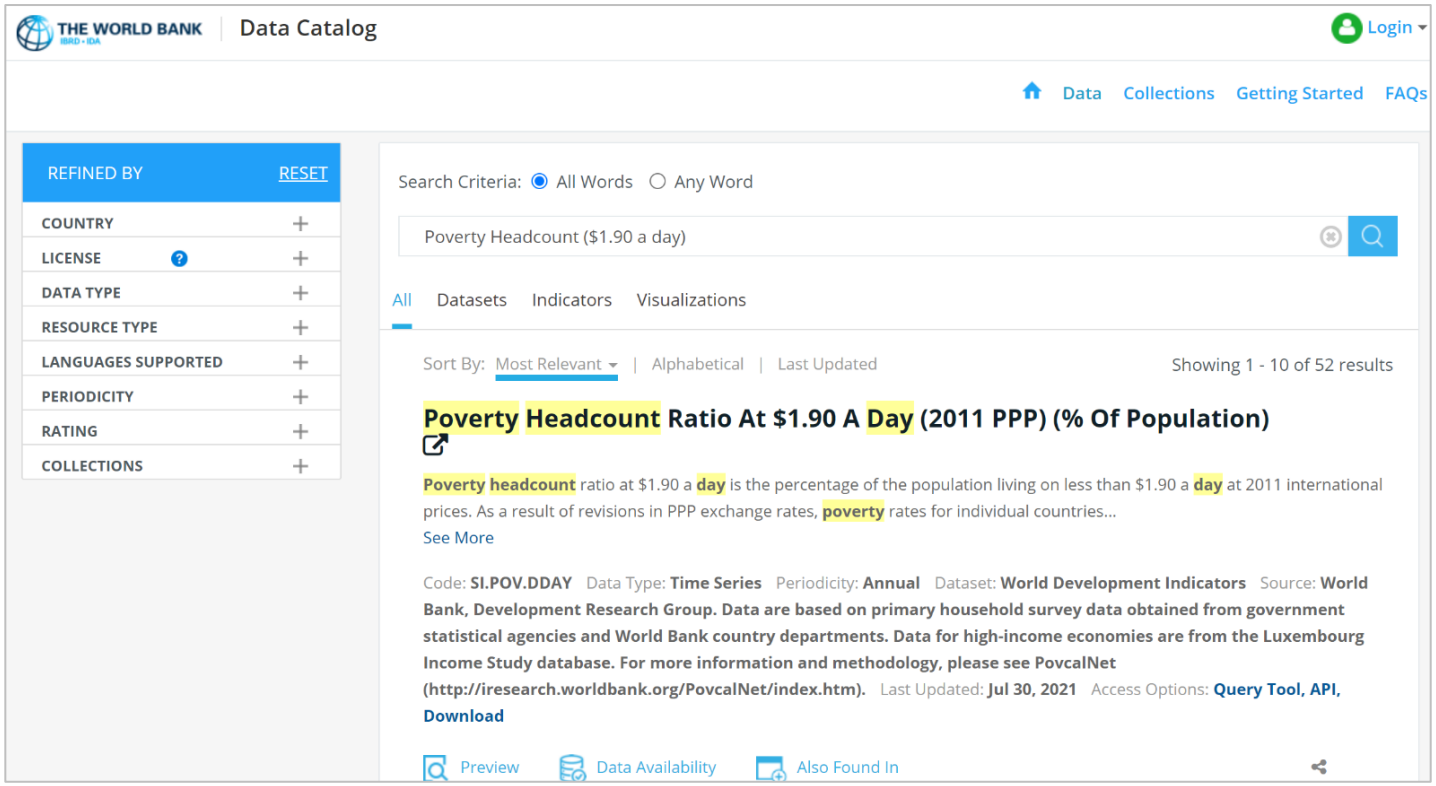
Line 6

- We will study iloc in the DataFrame row and column selection section. This way you can only get 5 indicators from the beginning of the whole list.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

If you want to know what data the indicator means, you can search the World Bank data catalog as shown in the image below.



7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

- Conversely, you can also search for the necessary indicators to get the data you want.
- You can search for an indicator with the wb.search( ) function. Let's practice searching for the indicator and downloading the corresponding data.

```
1 from pandas_datareader import wb
2
3 indicator_search = pdr.wb.search("life expectancy")
4
5 indicator_search.iloc[:5, :2]
```

	id	name
11955	SE.SCH.LIFE	School life expectancy, primary to tertiary, b...
11956	SE.SCH.LIFE.FE	School life expectancy, primary to tertiary, f...
11957	SE.SCH.LIFE.MA	School life expectancy, primary to tertiary, m...
13477	SP.DYN.LE00.FE.IN	Life expectancy at birth, female (years)
13478	SP.DYN.LE00.IN	Life expectancy at birth, total (years)

Line 3

- Use wb.search() function to search indicator to get data on life expectancy.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

- Conversely, you can also search for the necessary indicators to get the data you want.
- You can search for an indicator with the wb.search( ) function. Let's practice searching for the indicator and downloading the corresponding data.

```
1 from pandas_datareader import wb
2
3 indicator_search = pdr.wb.search("life expectancy")
4
5 indicator_search.iloc[:5, :2]
```

	id	name
11955	SE.SCH.LIFE	School life expectancy, primary to tertiary, b...
11956	SE.SCH.LIFE.FE	School life expectancy, primary to tertiary, f...
11957	SE.SCH.LIFE.MA	School life expectancy, primary to tertiary, m...
13477	SP.DYN.LE00.FE.IN	Life expectancy at birth, female (years)
13478	SP.DYN.LE00.IN	Life expectancy at birth, total (years)

Line 5

- If you print all the results, you get a lot of search results. For now, let's print only 5 results. Each indicator is country-specific.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 all_countries = pdr.wb.get_countries()
2
3 all_countries.loc[0:5, ['name', 'capitalCity', 'iso2c']]
```

	name	capitalCity	iso2c
0	Aruba	Oranjestad	AW
1	Africa Eastern and Southern		ZH
2	Afghanistan	Kabul	AF
3	Africa		A9
4	Africa Western and Central		ZI
5	Angola	Luanda	AO

Line 1, 3

- 1: Use the .wb.get_countries() function to get data for all countries.
- 2: Among all data, only the data in the 'name', 'capitalCity', and 'iso2c' columns are extracted from the above.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 result_data = pdr.wb.download(indicator = "SP.DYN.LE00.IN", start = '1980', end = '1999')
2
3 result_data
```

		SP.DYN.LE00.IN
country	year	
Canada	1999	78.885366
	1998	78.634146
	1997	78.431707
	1996	78.180488
	1995	78.029268
	1994	77.826829
	1993	77.824390
	1992	77.724390
	1991	77.621951
	1990	77.421951
	1989	77.119512

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 result_data = pdr.wb.download(indicator = "SP.DYN.LE00.IN", start = '1980', end = '1999')
2
3 result_data
```

Line 1, 3

- 1: You can download by putting the indicator and period you want to search in wb.download().
- 3: If you see the output results, you can find that only the data of Canada, the United States, and Mexico are output, not all countries. To download the entire country, you need to use the parameter country='all'.

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 result_data = pdr.wb.download(indicator = "SP.DYN.LE00.IN", start = '1980', end = '1999', country = 'all')
2
3 result_data
```

		SP.DYN.LE00.IN
country	year	
Africa Eastern and Southern	1999	51.044191
	1998	50.897607
	1997	50.820614
	1996	50.796157
	1995	50.808484
...	...	...
Zimbabwe	1984	61.280000
	1983	61.025000
	1982	60.650000
	1981	60.203000
	1980	59.731000

7. Reading data from remote data service using DataReader API

Ex How to use DataReader: Process life expectancy data by country around the world.

```
1 result_data = pdr.wb.download(indicator = "SP.DYN.LE00.IN", start = '1980', end = '1999', country = 'all')
2
3 result_data
```

Line 3

- You can get data for all countries by using the parameter country='all'.

One More Step

Let's find out which country recorded the lowest average life expectancy by year among the standard life expectancy data by country.

- First, through pivoting, the data is restructured with year index and country column.

```
1  from pandas_datareader import wb
2
3  result_data = pdr.wb.download(indicator = "SP.DYN.LE00.IN", start = '1980', end = '1999', country='all')
4
5  pivot_result = result_data.reset_index().pivot(index = 'country', columns = 'year')
6
7  pivot_result
```

Line 3, 5, 7

- 3: The indicator to be used for the search uses the average lifespan data used in the previous indicator search practice.
- 5: Create a new DataFrame by pivoting.
- 7: If you check the pivoted result, the data is restructured into a country index and a column for each year as planned.

One More Step

- Let's find out which country recorded the lowest average life expectancy by year among the standard life expectancy data by country.
  - First, through pivoting, the data is restructured with year index and country column.

	SP.DYN.LE00.IN												
year	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992
country													
Afghanistan	43.244000	43.923000	44.617000	45.324000	46.040000	46.761000	47.486000	48.211000	48.930000	49.640000	50.331000	50.999000	51.6
Africa Eastern and Southern	49.870693	50.115886	50.363456	50.610385	50.848335	51.058057	51.214281	51.299781	51.308595	51.251763	51.154113	51.048414	50.9
Africa Western and Central	46.366104	46.798292	47.188618	47.533985	47.830536	48.079381	48.284581	48.454128	48.597421	48.719231	48.816999	48.885934	48.9
Albania	70.208000	70.416000	70.635000	70.876000	71.134000	71.388000	71.605000	71.760000	71.843000	71.860000	71.836000	71.803000	71.8
Algeria	58.198000	59.519000	60.813000	62.029000	63.130000	64.087000	64.884000	65.545000	66.097000	66.554000	66.938000	67.270000	67.5
...	...	...	...	...	...	...	...	...	...	...	...	...	
West Bank and Gaza	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	68.048000	68.433000	68.7

One More Step

DataFrame.idxmin(axis=0, skipna=True)

- Return index of first occurrence of minimum over requested axis.
- https://pandas.pydata.org/pandas-docs/dev/reference/api/pandas.DataFrame.idxmin.html#pandas-dataframe-idxmin

```
1 result_contry = pivot_result.idxmin(axis=0)
2
3 print(result_contry)
```

Line 3

- You can check the results of the countries with the lowest life expectancy by year.

One More Step

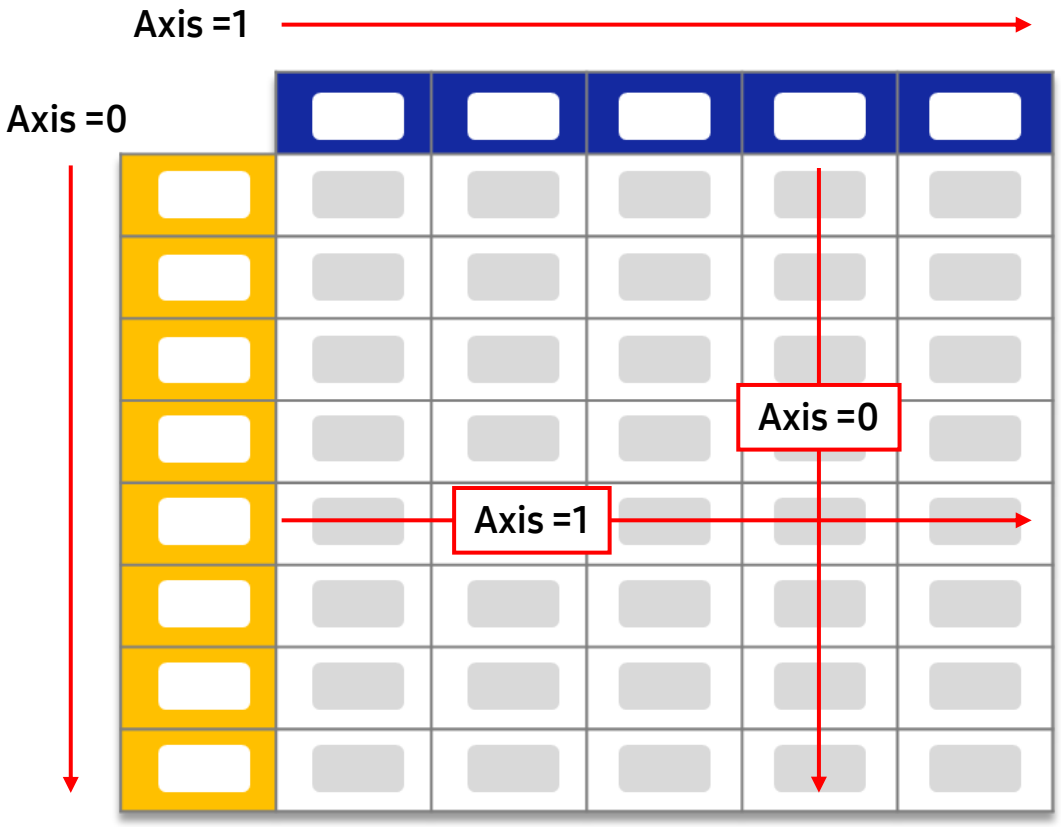
DataFrame.idxmin(axis=0, skipna=True)

	year	
SP.DYN.LE00.IN	1980	Cambodia
	1981	Cambodia
	1982	Timor-Leste
	1983	South Sudan
	1984	South Sudan
	1985	Sierra Leone
	1986	Sierra Leone
	1987	Sierra Leone
	1988	Sierra Leone
	1989	Rwanda
	1990	Rwanda
	1991	Rwanda
	1992	Rwanda
	1993	Rwanda
	1994	Rwanda
	1995	Rwanda
	1996	Rwanda
	1997	Sierra Leone
	1998	Sierra Leone

What do you think of the results? If you search the history of each country to find out what happened in that year, you can understand what seriously affects the average life expectancy in each country.

8. Manipulating Rows, Columns, and Elements in a DataFrame

It is necessary to understand the structure of the DataFrame by adding the concept of the axis of the array.



- axis = 0 (index)
  - It works in the row direction. The result of the work appears as a row. It is easy to understand if you imagine that the books are stacked on top of each other.
- axis = 1 (columns)
  - It works in the column direction. The result of the work appears as a column. It's like putting a book aside.

8. Manipulating Rows, Columns, and Elements in a DataFrame

Let's prepare the data for practice first. In this lesson, we will use what we used in the previous practice of converting JSON to a DataFrame.

```
1  import pandas as pd
2
3  rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
4
5  rank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1526 entries, 0 to 1525
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ranking              1526 non-null   int64
 1   title                1526 non-null   object
 2   location             1526 non-null   object
 3   number students      1526 non-null   object
 4   students staff ratio 1526 non-null   float64
 5   perc intl students   1526 non-null   object
 6   gender ratio         1526 non-null   object
dtypes: float64(1), int64(1), object(5)
memory usage: 83.6+ KB
```

Line 3

- Using the 2021 Worldwide University Links data downloaded from Kaggle.

8. Manipulating Rows, Columns, and Elements in a DataFrame

Let's prepare the data for practice first. In this lesson, we will use what we used in the previous practice of converting JSON to a DataFrame.

```
1  import pandas as pd
2
3  rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
4
5  rank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1526 entries, 0 to 1525
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ranking               1526 non-null   int64
 1   title                 1526 non-null   object
 2   location              1526 non-null   object
 3   number students       1526 non-null   object
 4   students staff ratio  1526 non-null   float64
 5   perc intl students    1526 non-null   object
 6   gender ratio          1526 non-null   object
dtypes: float64(1), int64(1), object(5)
memory usage: 83.6+ KB
```

Line 5

- Check the contents of the data frame. There are 1526 indexes from 0 to 1525, and it consists of 7 columns.

8. Manipulating Rows, Columns, and Elements in a DataFrame

Let's prepare the data for practice first. In this lesson, we will use what we used in the previous practice of converting JSON to a DataFrame.

```
1 rank.head()
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
0	1	University of Oxford	United Kingdom	20,774	11.1	41%	46 : 54
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61

Line 1

- Check the data structure by printing only 5 indexes.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.1. drop() to delete row, column

- Delete row: **DataFrame object name.drop(row index or array (list-like), axis=0, inplace=False)**
- Delete column: **DataFrame object name.drop(column name or array(list-like), axis=0, inplace=False)**

As described in the rename() method, use the inplace parameter to determine whether to change the original data object when a row or column is deleted. If this parameter is not specified, the default value is false (the original data is not changed).

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.1. drop() to delete row, column

```
1 rank.drop([0,1,2,3,4,5,6,7,8,9,10], axis=0, inplace=False)
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
11	12	Johns Hopkins University	United States	16,432	4.4	27%	52 : 48
12	13	University of Pennsylvania	United States	20,771	6.4	21%	52 : 48
13	14	ETH Zurich	Switzerland	19,632	13.1	40%	32 : 68
14	15	University of California, Los Angeles	United States	41,673	10.0	17%	55 : 45
15	16	UCL	United Kingdom	34,590	10.8	55%	57 : 43
...	...	...	...	...	...	...	...
1521	1522	Yuan Ze University	Taiwan	8,188	19.7	7%	42 : 58
1522	1523	Yuriy Fedkovych Chernivtsi National University	Ukraine	12,616	10.7	0%	57 : 43
1523	1524	Zagazig University	Egypt	156,270	24.4	2%	54 : 46
1524	1525	University of Zagreb	Croatia	59,336	15.3	3%	59 : 41
1525	1526	University of Žilina	Slovakia	7,136	11.7	2%	34 : 66

1515 rows × 7 columns

Line 1

- Delete indexes 0 through 15.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.1. drop() to delete row, column

```
1 rank.drop(["students staff ratio", "gender ratio"], axis=1, inplace=False)
```

	ranking	title	location	number students	perc intl students
0	1	University of Oxford	United Kingdom	20,774	41%
1	2	Stanford University	United States	16,223	23%
2	3	Harvard University	United States	21,261	25%
3	4	California Institute of Technology	United States	2,238	33%
4	5	Massachusetts Institute of Technology	United States	11,276	34%
...	...	...	...	...	...
1521	1522	Yuan Ze University	Taiwan	8,188	7%
1522	1523	Yuriy Fedkovych Chernivtsi National University	Ukraine	12,616	0%
1523	1524	Zagazig University	Egypt	156,270	2%
1524	1525	University of Zagreb	Croatia	59,336	3%
1525	1526	University of Žilina	Slovakia	7,136	2%

1526 rows × 5 columns

Line 1

- Delete two columns.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.1. drop() to delete row, column

rank.drop([0,1,2,3,4,5,6,7,8,9,10], axis=0, inplace=False)

Axis =0

Axis =1

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
0	1	University of Oxford	United Kingdom	20,774	11.1	41%	46 : 54
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
5	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
6	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
7	8	Yale University	United States	12,910	6.0	20%	50 : 50
8	9	Princeton University	United States	8,091	8.0	23%	46 : 54
9	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54
10	11	Imperial College London	United Kingdom	17,176	11.6	58%	39 : 61
11	12	Johns Hopkins University	United States	16,432	4.4	27%	52 : 48
12	13	University of Pennsylvania	United States	20,771	6.4	21%	52 : 48
13	14	ETH Zurich	Switzerland	19,632	13.1	40%	32 : 68
14	15	University of California, Los Angeles	United States	41,673	10.0	17%	55 : 45
15	16	UCL	United Kingdom	34,590	10.8	55%	57 : 43
16	17	Columbia University	United States	27,384	5.7	39%	n/a
17	18	University of Toronto	Canada	74,502	20.0	22%	59 : 41
18	19	Cornell University	United States	23,016	10.2	25%	50 : 50
19	20	Duke University	United States	15,489	4.3	21%	49 : 51

rank.drop(["students staff ratio", "gender ratio"], axis=1, inplace=False)

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

- Select by index name (index label): loc
- Select by integer position index (integer position): iloc

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

```
1  import pandas as pd
2
3  rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
4
5  index_position = rank.iloc[1:10]
6
7  index_position
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
5	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
6	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
7	8	Yale University	United States	12,910	6.0	20%	50 : 50
8	9	Princeton University	United States	8,091	8.0	23%	46 : 54
9	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54

Line 5

- Select numbers 1 to 9 by the integer index position and store them in a new DataFrame. (except 10)

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

rank.iloc[1:10]

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
0	1	University of Oxford	United Kingdom	20,774	11.1	41%	46 : 54
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
5	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
6	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
7	8	Yale University	United States	12,910	6.0	20%	50 : 50
8	9	Princeton University	United States	8,091	8.0	23%	46 : 54
9	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54
10	11	Imperial College London	United Kingdom	17,176	11.6	58%	39 : 61
11	12	Johns Hopkins University	United States	16,432	4.4	27%	52 : 48
12	13	University of Pennsylvania	United States	20,771	6.4	21%	52 : 48
13	14	ETH Zurich	Switzerland	19,632	13.1	40%	32 : 68
14	15	University of California, Los Angeles	United States	41,673	10.0	17%	55 : 45
15	16	UCL	United Kingdom	34,590	10.8	55%	57 : 43
16	17	Columbia University	United States	27,384	5.7	39%	n/a
17	18	University of Toronto	Canada	74,502	20.0	22%	59 : 41
18	19	Cornell University	United States	23,016	10.2	25%	50 : 50
19	20	Duke University	United States	15,489	4.3	21%	49 : 51

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

```
1  df = index_position.copy()
2
3  df.rename(index= {1:"a", 2:"b", 3:"c", 4:"d", 5:"e", 6:"f", 7:"g", 8:"h", 9:"i"}, inplace=True)
4
5  df
6
7  index_label = df.loc [["a","b"]]
8
9  index_label
10
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
a	2	Stanford University	United States	16,223	7.4	23%	44 : 56
b	3	Harvard University	United States	21,261	9.3	25%	49 : 51

Line 1, 3

- 1: Use rename() to change the name. When saving the original DataFrame with the inplace=True option, the DataFrame sometimes gives an error message. This is because of memory management. To prevent this, pandas recommends copying the original to a new DataFrame with the copy() method and then working on it.
- 3: For the loc practice, change the integer indexes to string index labels.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

```
1  df = index_position.copy()
2
3  df.rename(index= {1:"a", 2:"b", 3:"c", 4:"d", 5:"e", 6:"f", 7:"g", 8:"h", 9:"i"}, inplace=True)
4
5  df
6
7  index_label = df.loc [["a","b"]]
8
9  index_label
10
```

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
a	2	Stanford University	United States	16,223	7.4	23%	44 : 56
b	3	Harvard University	United States	21,261	9.3	25%	49 : 51

Line 5, 7

- 5: You can see that the integer index has been changed to an index label made of a string such as a,b,c..
- 7: Select the rows with index labels a and b and store them in a new DataFrame.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.2. Select row

df.loc [["a","b"]]

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
a	2	Stanford University	United States	16,223	7.4	23%	44 : 56
b	3	Harvard University	United States	21,261	9.3	25%	49 : 51
c	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
d	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
e	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
f	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
g	8	Yale University	United States	12,910	6.0	20%	50 : 50
h	9	Princeton University	United States	8,091	8.0	23%	46 : 54
i	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

- When selecting a column,
  - **DataFrame object name["column name"]**
  - **DataFrame object name.column name:** Only possible when the column name must be a string.
- When selecting multiple (n) columns,
  - **DataFrame object name["column name", "column name", "column name" ..]**

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

```
import pandas as pd

rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')

index_position = rank.iloc[1:10]

a = index_position.title

b = index_position[["title"]]

print(a)
print('\n')
print(b)

c = index_position[["title", "location", "students staff ratio" ]]
print('\n')
print(c)
```

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

```
1                     Stanford University
2                      Harvard University
3       California Institute of Technology
4    Massachusetts Institute of Technology
5                  University of Cambridge
6       University of California, Berkeley
7                           Yale University
8                     Princeton University
9                 The University of Chicago
Name: title, dtype: object


                                title
1                  Stanford University
2                   Harvard University
3     California Institute of Technology
4  Massachusetts Institute of Technology
5                University of Cambridge
6     University of California, Berkeley
7                       Yale University
```

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

```
1  import pandas as pd
2
3  rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
4
5  index_position = rank.iloc[1:10]
6
7  a = index_position.title
8
9  b = index_position[["title"]]
```

Line 7, 9

- 7: Select only one column named title and save it as a new DataFrame
- 9: Select only one column named title and save it as a new DataFrame

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

```
10
11 print(a)
12 print('\n')
13 print(b)
14
15 c = index_position[["title", "location", "students staff ratio" ]]
16 print('\n')
17 print(c)
```

Line 11, 13

- 11: A format that selects one column but compare the output results of both formats. The column name of the selected column is not printed.
- 13: This is a form of selecting one column, and the selected column name is printed.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.3. Select column

index_position[["title", "location", "students staff ratio"]]

	ranking	title	location	number students	students staff ratio	perc intl students	gender ratio
1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
5	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
6	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
7	8	Yale University	United States	12,910	6.0	20%	50 : 50
8	9	Princeton University	United States	8,091	8.0	23%	46 : 54
9	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54

index_position[[“gender ratio]]

index_position.gender ratio

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.4. Select data element

Select an element within the DataFrame by specifying the position of the element as if inputting [row. column] coordinates.

- DataFrame object name.loc[row name, column name]
- DataFrame object name.iloc[row number, column number]

```
1  import pandas as pd
2
3  rank = pd.read_json('./data/World University Rankings 2021/universities_ranking.json', orient='records')
4
5  df = rank.iloc[1:10]
6
7  d = df.iloc[7,1]
8
9  d
```

```
'Princeton University'
```

Line 5

- Select the data element in row number 7, column number 1.

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.4. Select data element

		0	1	2	3	4	5	6
	ranking		title	location	number students	students staff ratio	perc intl students	gender ratio
0	1	2	Stanford University	United States	16,223	7.4	23%	44 : 56
1	2	3	Harvard University	United States	21,261	9.3	25%	49 : 51
2	3	4	California Institute of Technology	United States	2,238	6.3	33%	36 : 64
3	4	5	Massachusetts Institute of Technology	United States	11,276	8.4	34%	39 : 61
4	5	6	University of Cambridge	United Kingdom	19,370	11.0	38%	47 : 53
5	6	7	University of California, Berkeley	United States	39,918	19.8	17%	51 : 49
6	7	8	Yale University	United States	12,910	6.0	20%	50 : 50
7	8	9	Princeton University	United States	8,091	8.0	23%	46 : 54
8	9	10	The University of Chicago	United States	14,292	5.9	31%	46 : 54

df[7,1]

8. Manipulating Rows, Columns, and Elements in a DataFrame

8.4. Select data element

```
1  e = df.iloc[0:1,0:2]
2
3  f = df.iloc[0:4, 3:]
4
5  print(e)
6  print('\n')
7  print(f)
```

```
   ranking                title
1        2  Stanford University


  number students  students staff ratio perc intl students gender ratio
1          16,223                   7.4                 23%      44 : 56
2          21,261                   9.3                 25%      49 : 51
3           2,238                   6.3                 33%      36 : 64
4          11,276                   8.4                 34%      39 : 61
```

Line 1, 3

- 1: Select 2 or more elements
- 3: You can also specify the range of column selection.

Let's code

Step 1

Think about what libraries are necessary to solve the mission. Find the necessary libraries, install those that are not, and import the module through import in the code.

- Pandas for creating DataFrame objects
- Matplotlib for visualizing data
- Seaborn, one of the data visualization tools

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
```

Step 1

After searching the target data and downloading, create a DataFrame.

- Download Spotify's 2019 Top Songs list data file from https://www.kaggle.com/prasertk/spotify-global-2019-moststreamed-tracks

```
1 song = pd.read_csv("./data/spotify/spotify_global_2019_most_streamed_tracks_audio_features.csv")
```

Line 1

- Load the downloaded csv file into the song DataFrame.

Step 1

```
1 song.head(10)
```

	Country	Rank	Track_id	Streams	Track Name	Artist	URL	acousticness	danceal
0	global	1.0	25sgk305KZfyuqVBQIahim	1166185736	Sweet but Psycho	Ava Max	https://open.spotify.com/track/25sgk305KZfyuqV...	0.0691	0
1	global	2.0	2FxmhksObxGSBdJ92vM42m	1052358787	bad guy	Billie Eilish	https://open.spotify.com/track/2FxmhksObxGSBdJ...	0.3280	0
2	global	3.0	6ocbgoVGwYJhOv1GgI9NsF	789094044	7 rings	Ariana Grande	https://open.spotify.com/track/6ocbgoVGwYJhOv1...	0.5920	0
3	global	4.0	1rgnBhdG2JDFTbYkYRZAku	764208309	Dance Monkey	Tones and I	https://open.spotify.com/track/1rgnBhdG2JDFTbY...	0.6880	0
4	global	5.0	6v3KW9xbzN5yKLt9YKDYA2	763064359	Señorita	Shawn Mendes	https://open.spotify.com/track/6v3KW9xbzN5yKLt...	0.0392	0
5	global	6.0	5w9c2J52mkdntKOmRLeM2m	715027898	Con Calma	Daddy Yankee	https://open.spotify.com/track/5w9c2J52mkdntKO...	0.1100	0
6	global	7.0	2VxeLyX666F8uXCJ0dZF8B	714097238	Shallow	Lady Gaga	https://open.spotify.com/track/2VxeLyX666F8uXC...	0.3710	0

Line 1

- Print only 10 DataFrames on the screen.

Step 1

```
1 song.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1717 entries, 0 to 1716
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Country           1717 non-null   object
 1   Rank              1717 non-null   float64
 2   Track_id          1717 non-null   object
 3   Streams           1717 non-null   int64
 4   Track Name        1717 non-null   object
 5   Artist            1717 non-null   object
 6   URL               1717 non-null   object
 7   acousticness      1717 non-null   float64
 8   danceability      1717 non-null   float64
 9   energy            1717 non-null   float64
 10  instrumentalness  1717 non-null   float64
 11  liveness          1717 non-null   float64
```

Line 1

- This is a method that can check the total number of DataFrames and data types of the imported DataFrame. It is a DataFrame with a total of 1717 rows and 24 columns.

Step 1

```
1 song.describe()
```

	Rank	Streams	acousticness	danceability	energy	instrumentalness	liveness	loudness	s
count	1717.000000	1.717000e+03	1717.000000	1717.000000	1717.000000	1717.000000	1717.000000	1717.000000	1
mean	859.000000	5.166175e+07	0.257652	0.675438	0.624354	0.014577	0.174317	-6.477624	
std	495.799523	1.047532e+08	0.256975	0.154163	0.172298	0.085988	0.135992	2.777252	
min	1.000000	5.242300e+05	0.000037	0.151000	0.013700	0.000000	0.019700	-25.166000	
25%	430.000000	1.763026e+06	0.051900	0.584000	0.525000	0.000000	0.097200	-7.595000	
50%	859.000000	9.623926e+06	0.171000	0.694000	0.646000	0.000000	0.123000	-5.991000	
75%	1288.000000	4.829352e+07	0.385000	0.787000	0.747000	0.000022	0.198000	-4.696000	
max	1717.000000	1.166186e+09	0.979000	0.974000	0.978000	0.956000	0.959000	-1.339000	

Line 1

- A method to check data distribution

Step 2

Selects only the necessary columns and recreates the DataFrame.

```
1 df = song[["Rank","Artist","Track Name", "Artist_popularity", "Streams","tempo", "danceability", "energy"]]
2 df.head()
```

	Rank	Artist	Track Name	Artist_popularity	Streams	tempo	danceability	energy
0	1.0	Ava Max	Sweet but Psycho	87	1166185736	133.002	0.719	0.704
1	2.0	Billie Eilish	bad guy	98	1052358787	135.128	0.701	0.425
2	3.0	Ariana Grande	7 rings	97	789094044	140.048	0.778	0.317
3	4.0	Tones and I	Dance Monkey	92	764208309	98.078	0.825	0.593
4	5.0	Shawn Mendes	Señorita	94	763064359	116.967	0.759	0.548

Line 1

- Select several columns you want and load them into a new DataFrame called df.

Step 3

- Only data with Artist_popularity of 95 or higher is filtered.
- In this case, we can use the data filtering method to filter rows by using logical operators on column values. A single logical operator or multiple logical operators can be used at the same time. Here, we use one logical operator to filter only the data with Artist_popularity of 95 or higher.
- The comparison value of 95 can be changed as you want in the practice. However, the first thing to do when importing an external file was to open the data directly and check the data structure. Since this data itself is only from Spotify's popular tracks in 2019, most of the artists are over 90.
- It is recommended to set it to 90 or higher for discrimination.

Step 3

```
1 pop_song = df[df["Artist_popularity"]>90]
2 pop_song.head()
```

	Rank	Artist	Track Name	Artist_popularity	Streams	tempo	danceability	energy
1	2.0	Billie Eilish	bad guy	98	1052358787	135.128	0.701	0.425
2	3.0	Ariana Grande	7 rings	97	789094044	140.048	0.778	0.317
3	4.0	Tones and I	Dance Monkey	92	764208309	98.078	0.825	0.593
4	5.0	Shawn Mendes	Señorita	94	763064359	116.967	0.759	0.548
5	6.0	Daddy Yankee	Con Calma	95	715027898	93.989	0.737	0.860

Line 1

- Create a new DataFrame with a value of 90 or higher in the Artist_popularity column and the index pop_song by using logical operators.

Step 3

Who is the artist with the most songs on the list?

```
1 rank = pop_song["Artist"].value_counts()
```

Line 1

- Counts the total number of non-duplicate unique data elements (values) and returns them as a series object.

Step 3

```
1 rank
```

```
Post Malone                  45
Juice WRLD                   31
Ariana Grande                28
Billie Eilish                27
Taylor Swift                 27
                             ..
Nicki Minaj                   1
YoungBoy Never Broke Again    1
Bass Santana                  1
J. Cole                       1
Rihanna                       1
Name: Artist, Length: 62, dtype: int64
```

Line 1

- An artist named Post Malone has the most songs with 45 songs.

Step 3

```
1 type(rank)
```

```
pandas.core.series.Series
```

Line 1

- If you check the returned data type, it is Series.

Step 3

TIP

- If **pop_song["Artist"].unique()** is used, all non-overlapping unique values are found in the corresponding DataFrame column.

```
1 pop_song["Artist"].unique()
```

```
array(['Billie Eilish', 'Ariana Grande', 'Tones and I', 'Shawn Mendes',
       'Daddy Yankee', 'Lewis Capaldi', 'Post Malone', 'Halsey',
       'Sam Smith', 'Ed Sheeran', 'Anuel AA', 'Bad Bunny', 'J. Cole',
       'Travis Scott', 'Juice WRLD', 'Maroon 5', 'Khalid', 'Sech',
       'Drake', 'XXXTENTACION', 'Lauv', 'J Balvin',
       'A Boogie Wit da Hoodie', 'Imagine Dragons', 'Queen', 'Ozuna',
       'Nicky Jam', 'Maluma', 'Chris Brown', 'Lil Baby', 'DaBaby', 'BTS',
       'Selena Gomez', 'Cardi B', 'Dua Lipa', 'Camila Cabello', 'Dalex',
       'The Chainsmokers', 'Young Thug', 'Tyga', 'Taylor Swift',
       'Justin Quiles', 'KAROL G', 'Myke Towers', 'Lil Uzi Vert',
       'Harry Styles', 'The Weeknd', 'Kanye West', 'Michael Bublé',
       'Justin Bieber', 'Roddy Ricch', 'Kendrick Lamar', 'Future',
       'YoungBoy Never Broke Again', 'Coldplay', 'Eminem', 'Nicki Minaj',
       'Trippie Redd', 'Gunna', 'Rihanna', 'Bass Santana', 'Bruno Mars'],
      dtype=object)
```

Step 4

- Is BTS Really Popular?
- If you are not a fan of BTS, you can filter data based on the English name of your favorite artist.
- First, search whether the name of the artist you want to find exists in the current dataset.

```
1 'BTS'in rank
```

```
True
```

Line 1

- If the return result is true, it means that the data exists. Let's not forget! rank is a series object made up of non-overlapping artist names and total numbers.

Step 4

We saw that the songs of the corresponding artist exist. Let's filter the data to see how many songs and which songs are included.

```
1 bts = pop_song[pop_song["Artist"]=="BTS"]
2
3 bts.shape
```

```
(11, 8)
```

Line 1, 3

- 1: pop_song is the DataFrame created in step3. Create a new DataFrame by collecting only the rows with BTS in the Artists column
- 3: You can see that there are 11 songs in total.

Step 4

```
1 bts
```

	Rank	Artist	Track Name	Artist_popularity	Streams	tempo	danceability	energy
86	87.0	BTS	Boy With Luv (feat. Halsey)	93	261504425	119.991	0.645	0.862
464	465.0	BTS	Make It Right (feat. Lauv)	93	42873724	97.551	0.584	0.685
676	677.0	BTS	Mikrokosmos	93	18801822	174.039	0.580	0.858
678	679.0	BTS	Dream Glow (BTS World Original Soundtrack) - P...	93	18648089	141.948	0.735	0.740
681	682.0	BTS	Make It Right	93	18423901	105.766	0.638	0.703
710	711.0	BTS	HOME	93	16442038	142.991	0.633	0.799
725	726.0	BTS	Dionysus	93	15134946	176.084	0.502	0.910
778	779.0	BTS	Jamais Vu	93	12992565	81.000	0.608	0.470
883	884.0	BTS	Intro : Persona	93	8775556	86.622	0.469	0.870
1057	1058.0	BTS	A Brand New Day (BTS World Original Soundtrack...	93	4383131	105.008	0.804	0.498
1265	1266.0	BTS	All Night (BTS World Original Soundtrack) [Pt. 3]	93	1921011	120.042	0.725	0.706

Line 1

- Song list

Step 5

If you see the result of step3, an American rapper named Post Malone has 45 songs in the dataset. He is the artist with the most hit songs. Let's make a playlist by making a separate list only for Post Malone.

```
play_list = pop_song[pop_song["Artist"] =="Post Malone"]

Malone= play_list.reset_index(drop=True, inplace=False)

Malone
```

	Rank	Artist	Track Name	Artist_popularity	Streams	tempo	danceability	energy
0	10.0	Post Malone	Wow.	100	615519053	99.947	0.833	0.539
1	14.0	Post Malone	Goodbyes (Feat. Young Thug)	100	501130662	150.231	0.580	0.653
2	26.0	Post Malone	Circles	100	428712946	120.042	0.695	0.762
3	32.0	Post Malone	Better Now	100	405953336	145.038	0.680	0.578
4	34.0	Post Malone	rockstar (feat. 21 Savage)	100	395449434	159.801	0.585	0.520
5	113.0	Post Malone	Sunflower - Spider-Man: Into the Spider-Verse	100	209280692	89.960	0.755	0.522
6	126.0	Post Malone	Psycho (feat. Ty Dolla $ign)	100	195603663	140.060	0.750	0.560
7	174.0	Post Malone	Take What You Want (feat. Ozzy Osbourne & Trav...	100	153232549	139.919	0.499	0.800
8	192.0	Post Malone	Congratulations	100	142006864	123.146	0.630	0.804
9	228.0	Post Malone	Saint-Tropez	100	118147209	132.113	0.617	0.684
10	230.0	Post Malone	I Fall Apart	100	116832184	143.950	0.556	0.538
11	235.0	Post Malone	Wow.	100	114072916	99.960	0.829	0.539

Step 5

```
1 play_list = pop_song[pop_song["Artist"] =="Post Malone"]
2
3 Malone= play_list.reset_index(drop=True, inplace=False)
4
5 Malone
```

Line 3

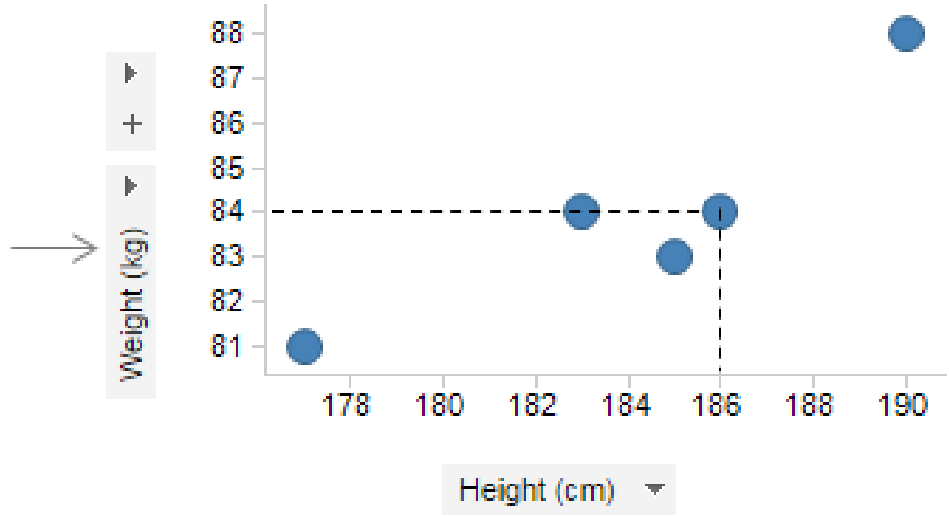
- Initialize the index with reset_index.
  https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reset_index.html

Step 6

Expressing the tempo by rank as a scatter plot graph

- A scatterplot is a visual representation of the relationship between two variables.
- A scatterplot is used to understand the relationship between two variables x and y. It is a graph in the form of a linear function in which the point (x, y) with x and y as an ordered pair is shown on the coordinate plane.

Name	Height (cm)	Weight (kg)
Sean	185	83
Roger	177	81
Timothy	190	88
Pierce	183	84
Daniel	186	84



https://docs.tibco.com/pub/spotfire_server/7.10.0/doc/html/en-US/TIB_sfire-bauthor-consumer_usersguide/GUID-A8DC822E-35B3-4289-94CB-642DFFE5E88F.html

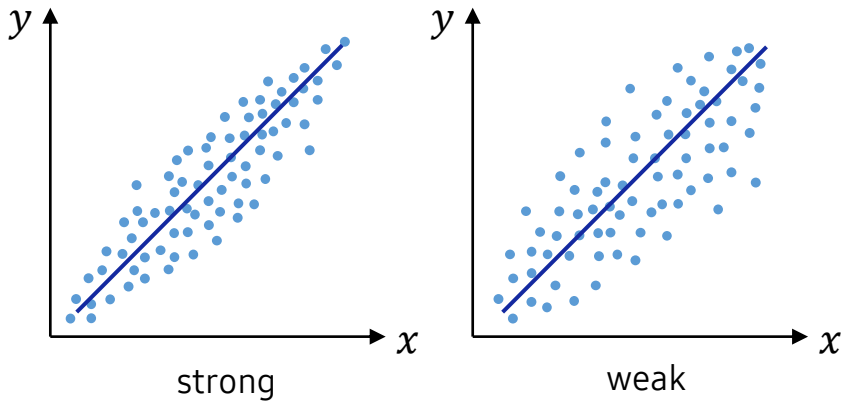
Step 6

- In a graph, you can see the relationship between two variables if one of them tends to increase (or decrease) as the other increases.
- The relationship is evaluated as large or small depending on the degree to which the distributed form of each point converges to the center based on the linear function.
- What is the positive correlation in the figure below?

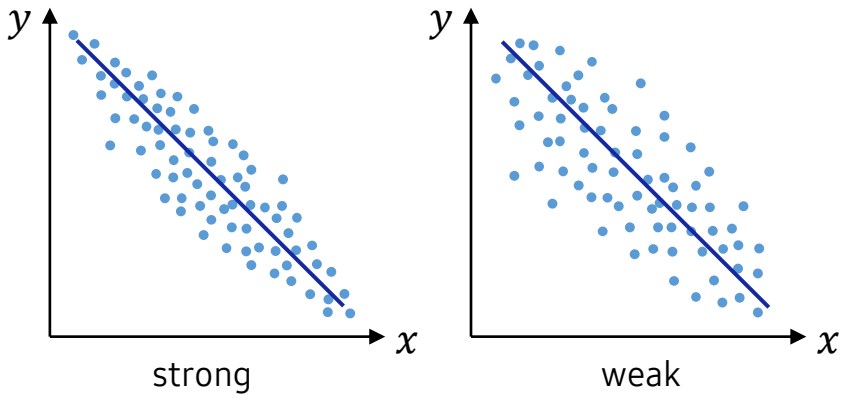
  Ex If a company produces more of a product, the price of the product falls, then production and price have a negative correlation. If sales increase as marketing investment increases, these two relationships are positively correlated.

- Don't get confused! A positive or negative correlation does not evaluate the degree of correlation between two variables.

Positive Correlation



Negative Correlation

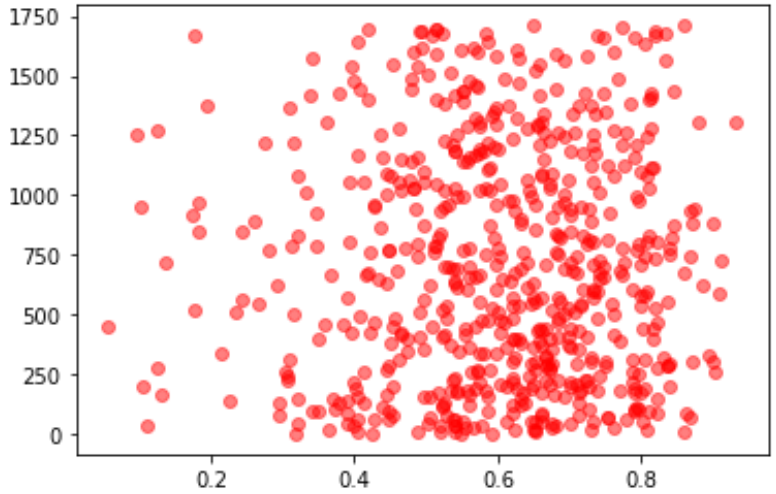


Step 6

You can easily plot correlation using Matplotlib's scatter.

- https://matplotlib.org/stable/gallery/shapes_and_collections/scatter.html#scatter-plot

```
1 x= pop_song["energy"]
2 y= pop_song["Rank"]
3 plt.scatter(x,y, color="red", alpha=0.5)
4
5 plt.show()
```



- When visually checking the graph, there is no clear correlation between the two variables.

Step 6

```
pandas.DataFrame.corr(method='pearson')
```

- Pandas supports a function that makes it easy to calculate the correlation coefficient.
- This is a function that calculates the pairwise correlation of columns excluding NA/Null values. The following three methods can be used for this function. Among the data of most pearson-used columns, numerical data is found and compared, and the string-type data is naturally subtracted.
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html
  - pearson : standard correlation coefficient
  - kendall : Kendall Tau correlation coefficient
  - spearman : Spearman rank correlation
- The result is returned as a DataFrame, and the interpretation of the values is as follows.
  - Closer to 1, both increase equally
  - Closer to -1, increase by one / decrease by one
  - Closer to 0, there is no relationship between the two

Step 6

```
1 cor = pop_song.corr(method ='pearson')
2 cor
```

	Rank	Artist_popularity	Streams	tempo	danceability	energy
Rank	1.000000	-0.125771	-0.667177	0.055495	-0.014811	0.015078
Artist_popularity	-0.125771	1.000000	0.076414	0.033521	-0.084745	0.019909
Streams	-0.667177	0.076414	1.000000	-0.034935	0.084925	-0.061912
tempo	0.055495	0.033521	-0.034935	1.000000	0.040506	0.245615
danceability	-0.014811	-0.084745	0.084925	0.040506	1.000000	0.116359
energy	0.015078	0.019909	-0.061912	0.245615	0.116359	1.000000

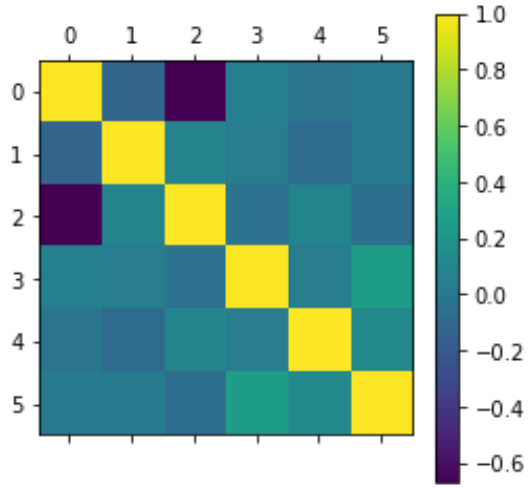
Line 2

- If you print the correlation coefficient result, you can't see a variable that is very close to 1. However, if the closest value is found among them, tempo appears to have the least effect.

Step 6

```
1 plt.matshow(cor)
2 plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x22ccd8fabe0>
```

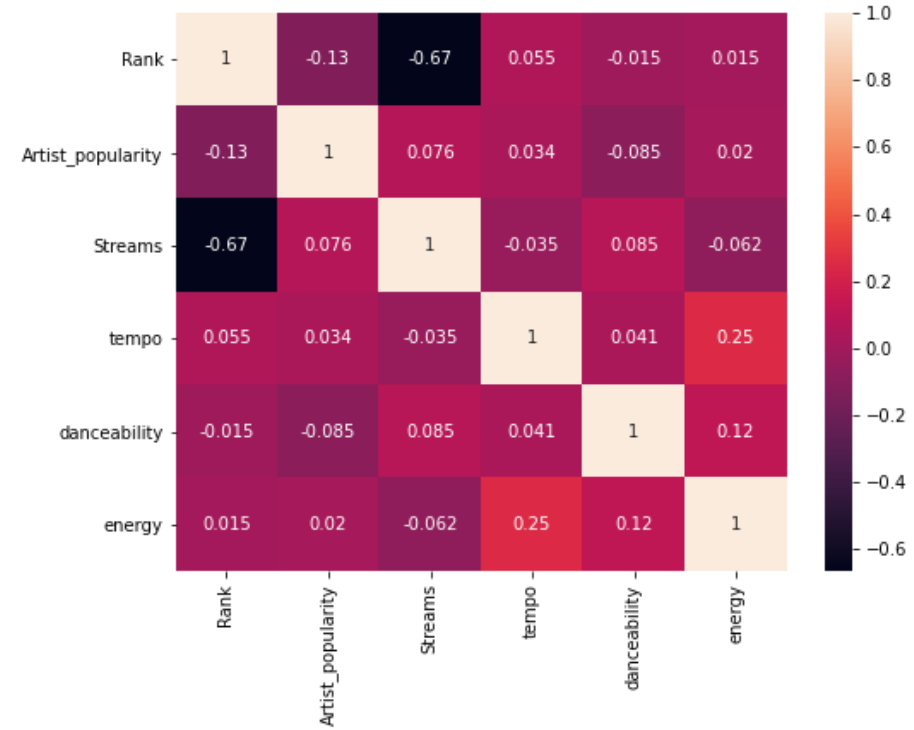


Line 2

- It is displayed in the form of a heatmap in a matrix.
- Display the color bar.

Step 6

```
1 plt.figure(figsize=(8, 6))
2 sns.heatmap(cor, annot=True)
3 plt.show()
```



Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor's question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students "divide and conquer." When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Q1. In the Spotify dataset, there are various column values in addition to the artist's name, rank, and popularity we used in our mission. Discuss with your learning colleagues to create special playlists using different column values.

Ex A playlist of only upbeat music: use the "tempo" column to select an appropriate tempo.

TIP

- If you have made a playlist, save it as an Excel file and share it with your learning colleagues.
- Saving a DataFrame as an Excel file is very simple.

```
pandas.DataFrame.to_excel()
```

  - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_excel.html