



Introduction To Data Analysis

# 데이터분석인문

## Lecture 27. 데이터프레임을 위한 pandas 패키지

인공지능소프트웨어학과 강환수 교수

## 학습개요

- ✓ 테이블 형태를 다루는 pandas 개요
- ✓ pandas 주요 자료형 Series와 DataFrame
- ✓ DataFrame의 index와 columns



## 학습목표

- ✓ pandas 주요 자료형인 Series와 DataFrame을 이해할 수 있다.
- ✓ 행 제목과 열 제목을 지정해 DataFrame을 생성할 수 있다.
- ✓ DataFrame의 열을 생성하거나 삭제할 수 있다.

LESSON 01

# 테이블 형태의 데이터를 쉽게 다루는 pandas 패키지



## → pandas 라이브러리 (1/9)

- panel data system의 약자로 파이썬을 활용한 데이터 분석에서 많이 활용되고 있음
  - numpy를 기반으로 만들어졌으며 데이터 분석을 위한 효율적인 데이터 구조를 제공하고 있음
- ✓ <https://pandas.pydata.org>



## → pandas 라이브러리 (2/9)

📄 사용자 가이드(User guide)를 클릭하세요.

**pandas**

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

**Getting started**

- [Install pandas](#)
- [Getting started](#)

**Documentation**

- [User guide](#)
- [API reference](#)
- [Contributing to pandas](#)
- [Release notes](#)

**Community**

- [About pandas](#)
- [Ask a question](#)
- [Ecosystem](#)

**Latest version: 1.5.0**

- [What's new in 1.5.0](#)
- [Release date: Sep 19, 2022](#)
- [Documentation \(web\)](#)
- [Download source code](#)

**Follow us**

[Telegram](#) [Twitter](#)

**Get the book**

[Python for Data Analysis](#)

**With the support of:**

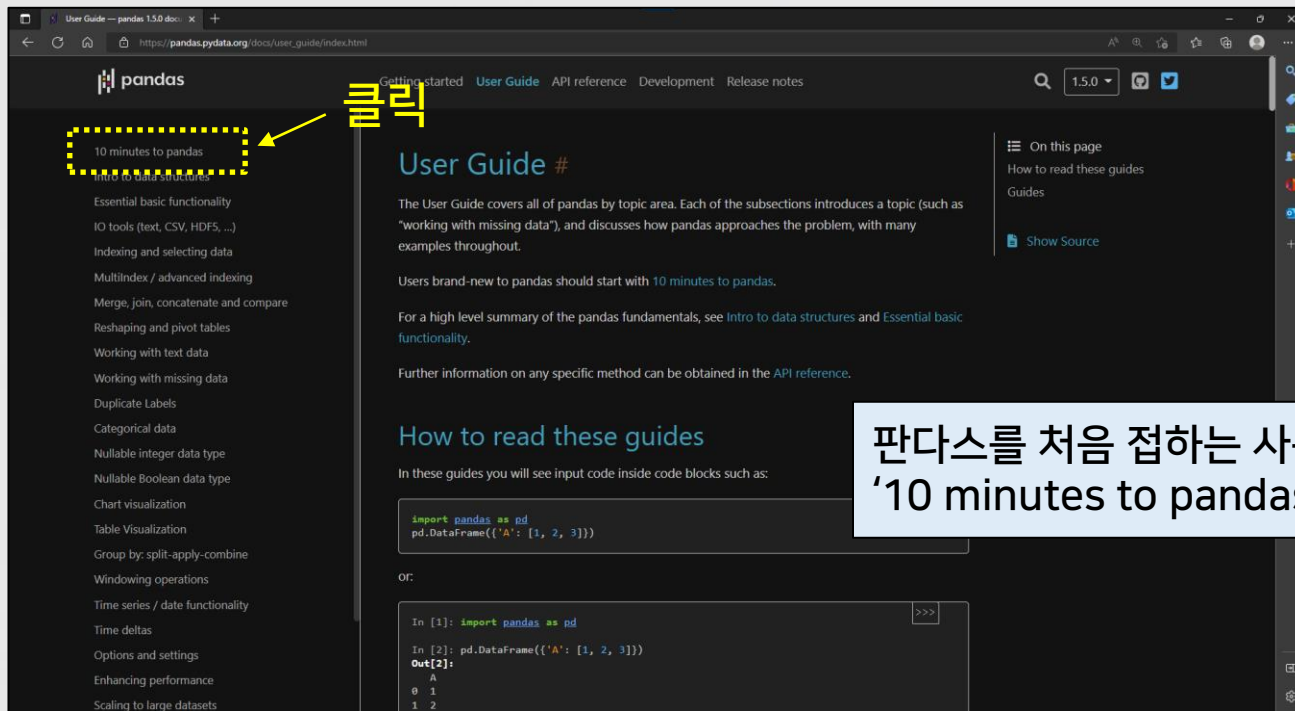
NUMFOCUS TWO SIGMA VOLTRON DATA d-fine Quansight Labs NVIDIA

**Previous versions**

- 1.4.4 (Aug 31, 2022)  
[changelog](#) | [docs](#) | [code](#)
- 1.4.3 (Jun 23, 2022)  
[changelog](#) | [docs](#) | [code](#)
- 1.4.2 (Apr 02, 2022)

## — pandas 라이브러리 (3/9)

 pandas 라이브러리 사용자를 위한 설명을 확인할 수 있습니다.



클릭

10 minutes to pandas

User Guide #

The User Guide covers all of pandas by topic area. Each of the subsections introduces a topic (such as "working with missing data"), and discusses how pandas approaches the problem, with many examples throughout.

Users brand-new to pandas should start with [10 minutes to pandas](#).

For a high level summary of the pandas fundamentals, see [Intro to data structures](#) and [Essential basic functionality](#).

Further information on any specific method can be obtained in the [API reference](#).

How to read these guides

In these guides you will see input code inside code blocks such as:

```
import pandas as pd
pd.DataFrame({'A': [1, 2, 3]})
```

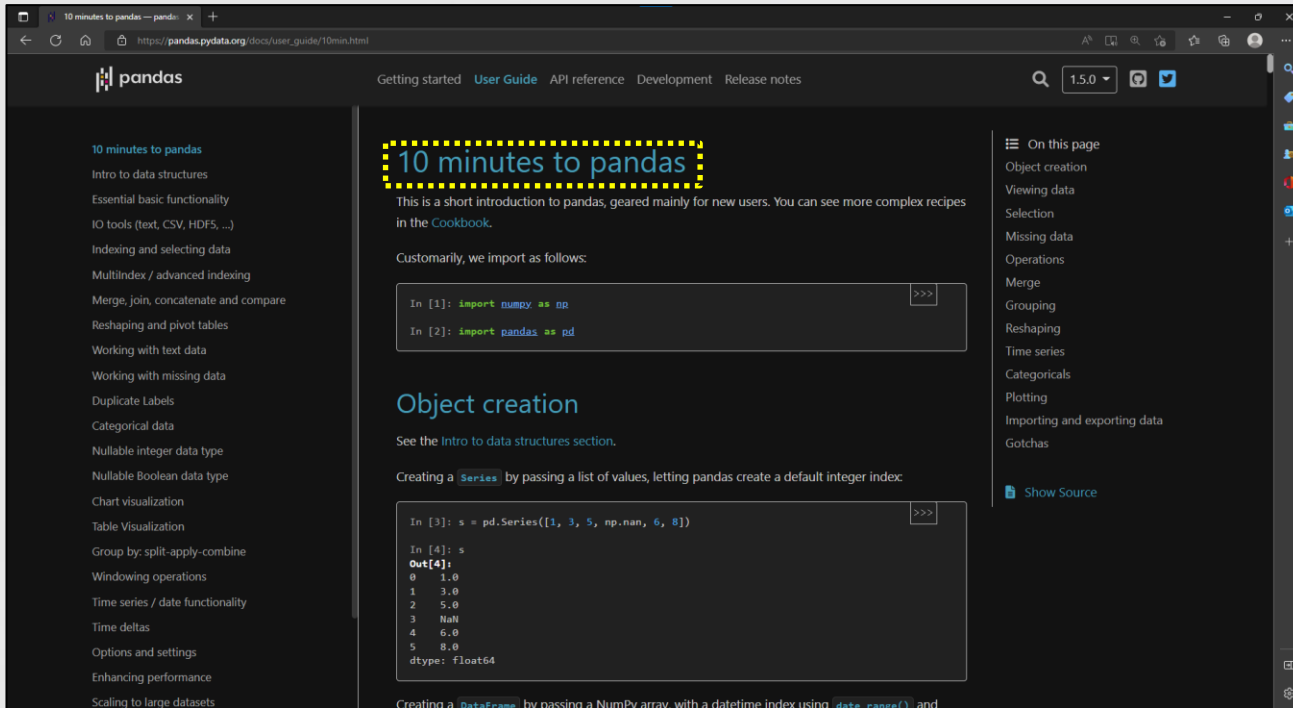
or:

```
In [1]: import pandas as pd
In [2]: pd.DataFrame({'A': [1, 2, 3]})
Out[2]:
   A
0  1
1  2
```

판다스를 처음 접하는 사용자는  
'10 minutes to pandas'를 먼저 보면 좋습니다.

## — pandas 라이브러리 (4/9)

 pandas 라이브러리에 대한 간략한 소개 내용을 살펴볼 수 있습니다.



The screenshot shows the pandas website's '10 minutes to pandas' guide. The page is titled '10 minutes to pandas' and is part of the 'Getting started' section. It includes a table of contents on the left, a main content area with code snippets, and a right sidebar with a table of contents.

**10 minutes to pandas**

This is a short introduction to pandas, geared mainly for new users. You can see more complex recipes in the Cookbook.

Customarily, we import as follows:

```
In [1]: import numpy as np
In [2]: import pandas as pd
```

**Object creation**

See the Intro to data structures section.

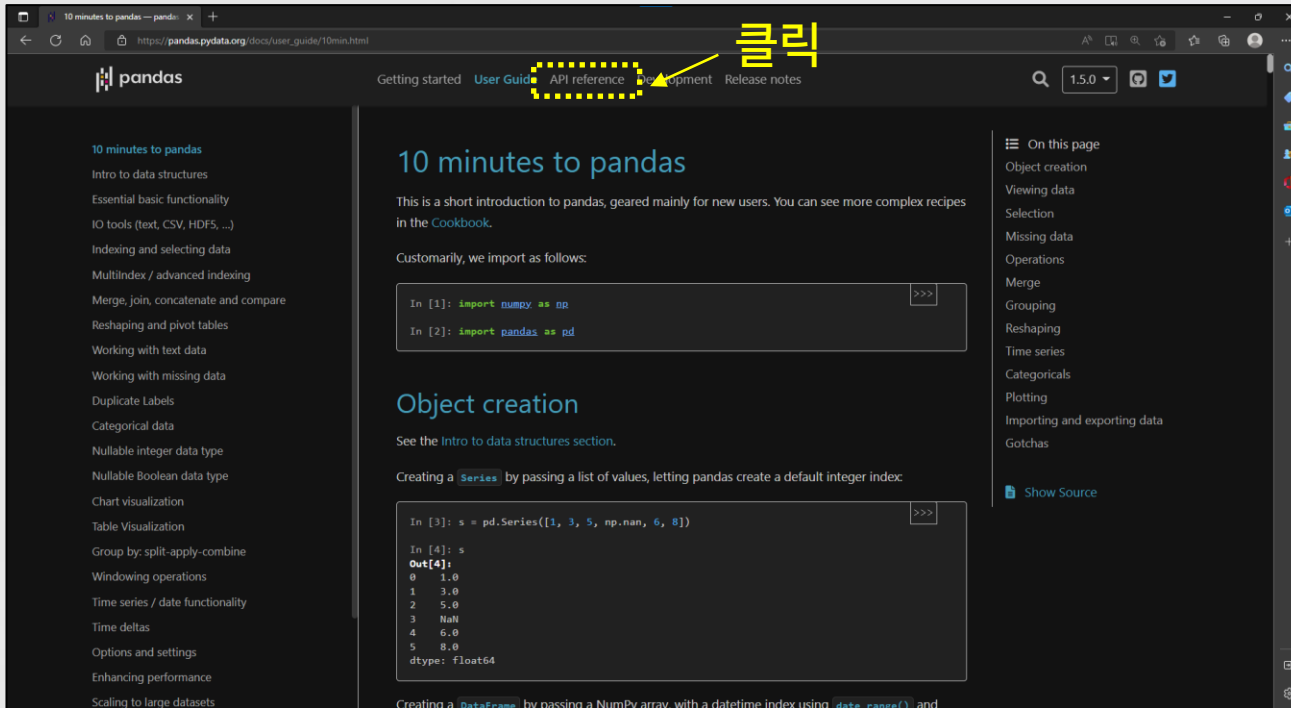
Creating a **Series** by passing a list of values, letting pandas create a default integer index

```
In [3]: s = pd.Series([1, 3, 5, np.nan, 6, 8])
Out[3]:
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Creating a **DataFrame** by passing a NumPy array, with a datetime index using `date_range()` and

## — pandas 라이브러리 (5/9)

상단에 위치한 메뉴 중에서 API reference를 클릭하세요.



The screenshot shows the pandas website with the 'API reference' menu item highlighted by a yellow dashed box. A yellow arrow points to this box with the Korean word '클릭' (Click) next to it. The website content includes a sidebar with a list of topics, a main section titled '10 minutes to pandas', and a right sidebar with a table of contents.

**API reference**

Getting started User Guide **API reference** New development Release notes

10 minutes to pandas

This is a short introduction to pandas, geared mainly for new users. You can see more complex recipes in the Cookbook.

Customarily, we import as follows:

```
In [1]: import numpy as np
In [2]: import pandas as pd
```

**Object creation**

See the Intro to data structures section.

Creating a **Series** by passing a list of values, letting pandas create a default integer index

```
In [3]: s = pd.Series([1, 3, 5, np.nan, 6, 8])
Out[3]:
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Creating a **DataFrame** by passing a NumPy array, with a datetime index using `date_range()` and

On this page

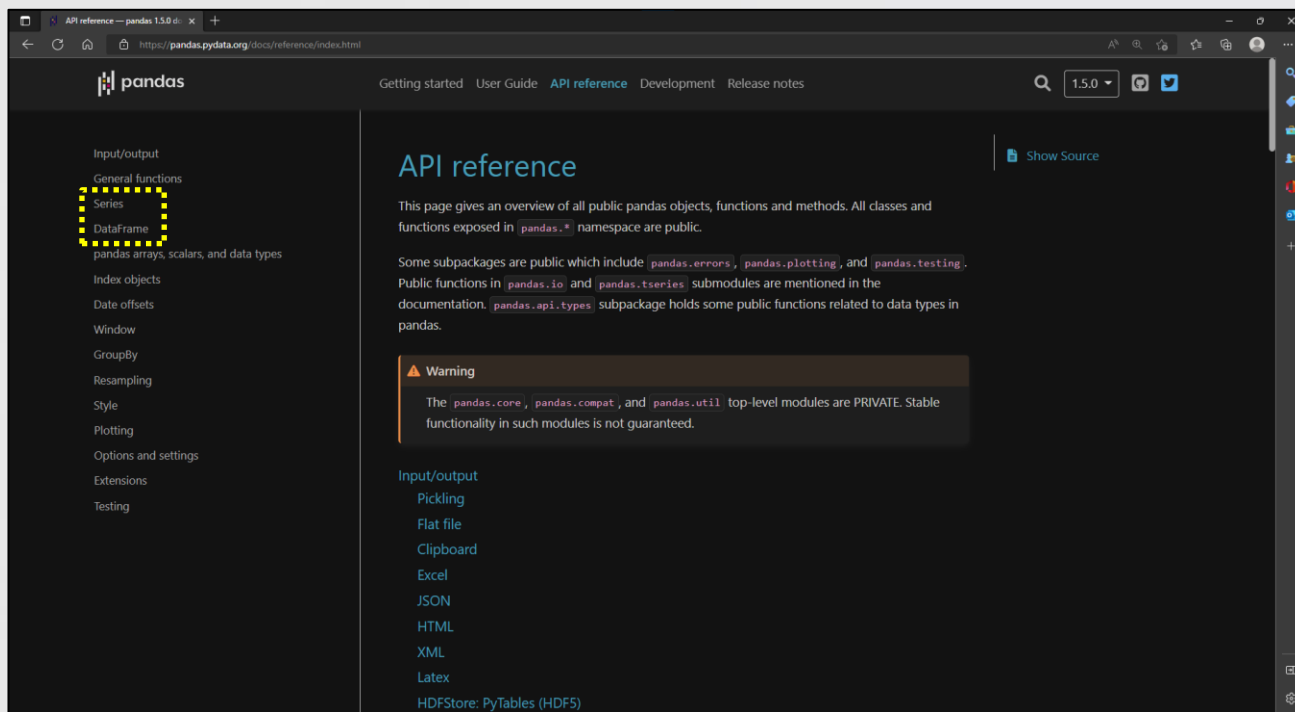
- Object creation
- Viewing data
- Selection
- Missing data
- Operations
- Merge
- Grouping
- Reshaping
- Time series
- Categoricals
- Plotting
- Importing and exporting data
- Gotchas

Show Source



## — pandas 라이브러리 (6/9)

판다스에서 지원하는 데이터 구조인 Series(시리즈), DataFrame(데이터 프레임)을 확인할 수 있습니다.



## — pandas 라이브러리 (7/9)

Series(시리즈)에 대한 내용을 확인할 수 있습니다.

The screenshot shows the pandas 1.5.0 documentation page for the Series object. The page is titled "Series" and is part of the "API reference" section. The left sidebar lists various pandas components, with "Series" highlighted. The main content area is divided into sections: "Constructor", "Attributes", and "Axes". The "Constructor" section describes Series as a "One-dimensional ndarray with axis labels (including time series)". The "Attributes" section lists attributes like .index, .array, .values, .dtype, .shape, and .nbytes. The "Axes" section lists attributes like .index, .array, .values, .dtype, .shape, and .nbytes. The right sidebar lists topics "On this page", including Constructor, Attributes, Conversion, Indexing, iteration, Binary operator functions, Function application, GroupBy & window, Computations / descriptive stats, Reindexing / selection / label manipulation, Missing data handling, Reshaping, sorting, Combining / comparing / joining / merging, Time Series-related, Accessors, Plotting, and Serialization / IO / conversion. A "Show Source" link is also present.

Series

### Constructor

`Series` ([data, index, dtype, name, copy, ...]) One-dimensional ndarray with axis labels (including time series).

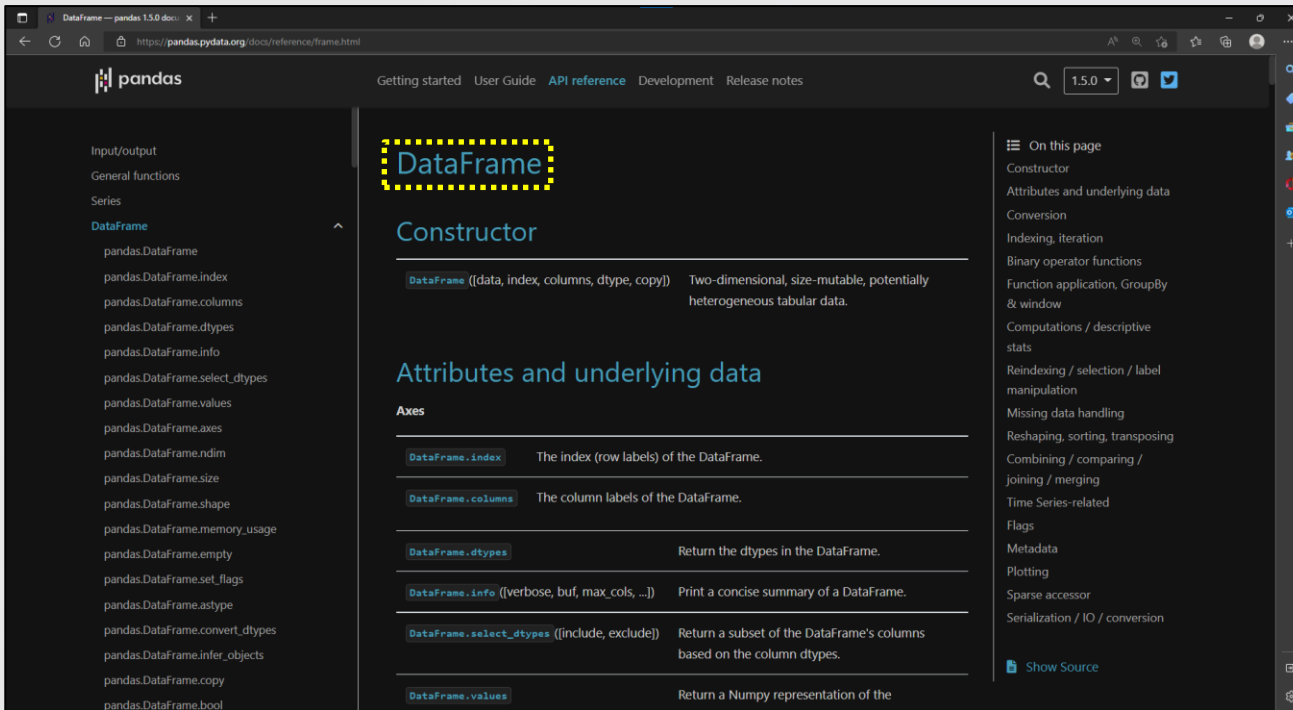
### Attributes

#### Axes

<code>Series.index</code>	The index (axis labels) of the Series.
<code>Series.array</code>	The ExtensionArray of the data backing this Series or Index.
<code>Series.values</code>	Return Series as ndarray or ndarray-like depending on the dtype.
<code>Series.dtype</code>	Return the dtype object of the underlying data.
<code>Series.shape</code>	Return a tuple of the shape of the underlying data.
<code>Series.nbytes</code>	Return the number of bytes in the underlying data.

## → pandas 라이브러리 (8/9)

 DataFrame(데이터 프레임)에 대한 내용을 확인할 수 있습니다.



The screenshot shows the pandas DataFrame documentation page. The left sidebar lists various pandas objects, with 'DataFrame' selected. The main content area is titled 'DataFrame' and includes a 'Constructor' section with the definition: 'DataFrame ([data, index, columns, dtype, copy]) Two-dimensional, size-mutable, potentially heterogeneous tabular data.' Below this is the 'Attributes and underlying data' section, which lists several attributes: 'DataFrame.index' (The index (row labels) of the DataFrame), 'DataFrame.columns' (The column labels of the DataFrame), 'DataFrame.dtypes' (Return the dtypes in the DataFrame), 'DataFrame.info' ([verbose, buf, max\_cols, ...]) (Print a concise summary of a DataFrame), 'DataFrame.select\_dtypes' ([include, exclude]) (Return a subset of the DataFrame's columns based on the column dtypes), and 'DataFrame.values' (Return a Numpy representation of the DataFrame).

## — pandas 라이브러리 (9/9)

### Series(시리즈)

- ✓ 1차원 배열 형태의 데이터 구조

--	--	--	--	--	--	--	--	--	--

### DataFrame(데이터 프레임)

- ✓ 2차원 배열 형태의 데이터 구조


## Series 개요

### Series(시리즈)

- ✓ 1차원 배열 형태의 데이터 구조

#### Series Index

	A
1	1
2	2
3	3
4	4

**Series Name** (points to 'A')

**Series Values** (points to the column of values)

	Data
p	3.0
q	2.0
r	2.0
s	1.0
t	1.0

**Index** (points to the column of labels)

**Series** (points to the column of values)

dtype: float64

## ◦ DataFrame 개요

### DataFrame(데이터 프레임)

- ✓ 2차원 배열 형태의 데이터 구조

Series 1		Series 2		Series 3		DataFrame			
Mango		Apple		Banana		Mango	Apple	Banana	
0	4	0	5	0	2	0	4	5	2
1	5	1	4	1	3	1	5	4	3
2	6	2	3	2	5	2	6	3	5
3	3	3	0	3	2	3	3	0	2
4	1	4	2	4	7	4	1	2	7

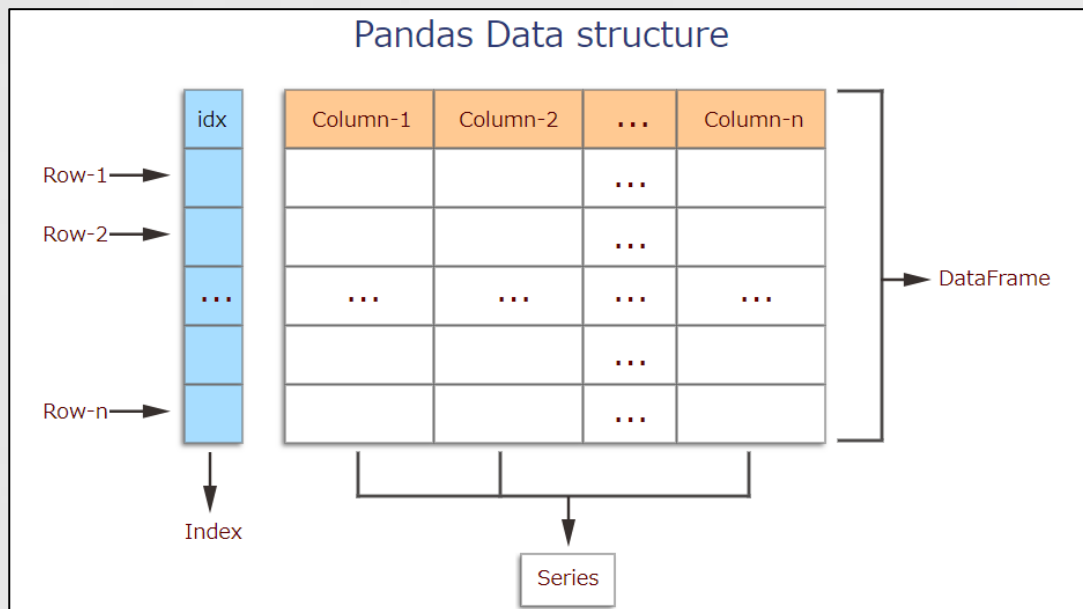
LESSON 02

# 데이터프레임 기초



## ◦ DataFrame 개요

- 행(Row)과 열(Column)로 구성
- 행을 구분해주는 인덱스, 열을 구분해주는 컬럼이 있음
- 인덱스와 열은 별도로 지정을 하지 않으면 정수로 지정됨





## → DataFrame의 index 생성

- 행(Row)과 열(Column)로 구성
- 행을 구분해주는 인덱스, 열을 구분해주는 컬럼이 있음
- 인덱스와 열은 별도로 지정을 하지 않으면 정수로 지정됨
- 인덱스 만들기 예제

```
import pandas as pd
```

```
index = pd.date_range('1/1/2024', periods=8)  
index
```

✓ 2.0s

Python

```
DatetimeIndex(['2024-01-01', '2024-01-02', '2024-01-03', '2024-01-04',  
              '2024-01-05', '2024-01-06', '2024-01-07', '2024-01-08'],  
              dtype='datetime64[ns]', freq='D')
```

## ◦ DataFrame 생성 예제

```
import numpy as np
import pandas as pd

index = pd.date_range('1/1/2024', periods=8)
df = pd.DataFrame(np.random.rand(8, 3), index=index, columns= list('ABC'))
df
```

✓ 0.0s

Python

	A	B	C
2024-01-01	0.850061	0.481933	0.400766
2024-01-02	0.376727	0.847266	0.716762
2024-01-03	0.340956	0.123893	0.396765
2024-01-04	0.682324	0.371941	0.759515
2024-01-05	0.790759	0.012287	0.129102
2024-01-06	0.112983	0.900028	0.575334
2024-01-07	0.292861	0.268015	0.162694
2024-01-08	0.551867	0.330099	0.817889

## ◦ DataFrame의 특정 열에 접근하기

```
import pandas as pd
```

```
index = pd.date_range('1/1/2024', periods=8)
```

```
df = pd.DataFrame(np.random.rand(8, 3), index=index, columns= list('ABC'))
```

```
df['B']
```

✓ 0.0s

Python

```
2024-01-01    0.377473
```

```
2024-01-02    0.117435
```

```
2024-01-03    0.510446
```

```
2024-01-04    0.190878
```

```
2024-01-05    0.687397
```

```
2024-01-06    0.483472
```

```
2024-01-07    0.216247
```

```
2024-01-08    0.666448
```

```
Freq: D, Name: B, dtype: float64
```

“B”라는 이름(컬럼 이름)을 통해서  
DataFrame에 저장된 특정 열에 접근할 수 있습니다.

```
df['B'][0]
```

✓ 0.0s

Python

```
0.3774726543845809
```

## 특정 열에 마스크(Mask) 생성하기

```
import pandas as pd
import numpy as np

index = pd.date_range('1/1/2000', periods=8)

df = pd.DataFrame(np.random.rand(8, 3), index=index, columns=['A', 'B', 'C'])
print(df['B'] > 0.4)
```

```
2000-01-01    False
2000-01-02    False
2000-01-03     True
2000-01-04     True
2000-01-05    False
2000-01-06    False
2000-01-07    False
2000-01-08     True
Freq: D, Name: B, dtype: bool
```

마스크는 특정한 조건을 만족하는지에 따라 참(True) 또는 거짓(False)을 반환하여 원하는 데이터를 골라내는데 유용하게 사용할 수 있습니다.

## 마스크가 적용된 결과를 데이터 프레임으로 저장하기

```
import pandas as pd
import numpy as np

index = pd.date_range('1/1/2000', periods=8)

df = pd.DataFrame(np.random.rand(8, 3), index=index, columns=['A', 'B', 'C'])
df
```

	A	B	C
2000-01-01	0.738730	0.910045	0.965540
2000-01-02	0.632368	0.532855	0.779480
2000-01-03	0.106017	0.202848	0.992850
2000-01-04	0.157887	0.202096	0.211844
2000-01-05	0.318404	0.555418	0.861561
2000-01-06	0.267227	0.070146	0.699629
2000-01-07	0.406687	0.970594	0.019432
2000-01-08	0.780414	0.457065	0.980875



```
df2 = df[df['B'] > 0.4]
df2
```

	A	B	C
2000-01-01	0.738730	0.910045	0.965540
2000-01-02	0.632368	0.532855	0.779480
2000-01-05	0.318404	0.555418	0.861561
2000-01-07	0.406687	0.970594	0.019432
2000-01-08	0.780414	0.457065	0.980875

## 행과 열 바꾸기

df2.T

T는 행과 열을 바꾼다는 의미의 단어인 Transpose를 의미합니다.

	A	B	C
2000-01-01	0.738730	0.910045	0.965540
2000-01-02	0.632368	0.532855	0.779480
2000-01-05	0.318404	0.555418	0.861561
2000-01-07	0.406687	0.970594	0.019432
2000-01-08	0.780414	0.457065	0.980875



	2000-01-01	2000-01-02	2000-01-05	2000-01-07	2000-01-08
A	0.738730	0.632368	0.318404	0.406687	0.780414
B	0.910045	0.532855	0.555418	0.970594	0.457065
C	0.965540	0.779480	0.861561	0.019432	0.980875

데이터 프레임 뒤에 .T만 붙여주면  
행과 열을 쉽게 바꿀 수 있습니다.

## → A열의 값을 B열의 값으로 나눈 결과를 D열로 추가하기

```
import pandas as pd
import numpy as np

index = pd.date_range('1/1/2000', periods=8)

df = pd.DataFrame(np.random.rand(8, 3), index=index, columns=['A', 'B', 'C'])
df
```

	A	B	C
2000-01-01	0.905683	0.029957	0.026522
2000-01-02	0.533392	0.119775	0.011812
2000-01-03	0.590626	0.189633	0.068469
2000-01-04	0.048945	0.760138	0.543997
2000-01-05	0.612580	0.847152	0.044991
2000-01-06	0.097279	0.160584	0.529673
2000-01-07	0.273220	0.979850	0.178016
2000-01-08	0.936792	0.921132	0.528050



```
df['D'] = df['A'] / df['B']
df
```

	A	B	C	D
2000-01-01	0.905683	0.029957	0.026522	30.232433
2000-01-02	0.533392	0.119775	0.011812	4.453275
2000-01-03	0.590626	0.189633	0.068469	3.114577
2000-01-04	0.048945	0.760138	0.543997	0.064390
2000-01-05	0.612580	0.847152	0.044991	0.723104
2000-01-06	0.097279	0.160584	0.529673	0.605786
2000-01-07	0.273220	0.979850	0.178016	0.278838
2000-01-08	0.936792	0.921132	0.528050	1.017001

## 같은 행에 있는 데이터의 합을 구하고, 결과를 E열에 추가하기

```
df['E'] = np.sum(df, axis=1)
df
```

	A	B	C	D	E
2000-01-01	0.905683	0.029957	0.026522	30.232433	31.194595
2000-01-02	0.533392	0.119775	0.011812	4.453275	5.118255
2000-01-03	0.590626	0.189633	0.068469	3.114577	3.963304
2000-01-04	0.048945	0.760138	0.543997	0.064390	1.417471
2000-01-05	0.612580	0.847152	0.044991	0.723104	2.227827
2000-01-06	0.097279	0.160584	0.529673	0.605786	1.393322
2000-01-07	0.273220	0.979850	0.178016	0.278838	1.709924
2000-01-08	0.936792	0.921132	0.528050	1.017001	3.402975



## 새 열 추가하기

```
df['F'] = np.ones(8)  
df.head(3) # head(n)은 많은 데이터 중 처음 n개 데이터만 보여줍니다.
```

	A	B	C	D	E	F
2000-01-01	0.905683	0.029957	0.026522	30.232433	31.194595	1.0
2000-01-02	0.533392	0.119775	0.011812	4.453275	5.118255	1.0
2000-01-03	0.590626	0.189633	0.068469	3.114577	3.963304	1.0

## 특정 열 제거하기

```
df1 = df.drop(['E', 'F'], axis='columns') # axis=1 이라고 적어도 됩니다.
df1.head() # 괄호 ()안에 특정 숫자를 지정하지 않으면, 처음 5개만 보여줍니다.
```

	A	B	C	D
2000-01-01	0.905683	0.029957	0.026522	30.232433
2000-01-02	0.533392	0.119775	0.011812	4.453275
2000-01-03	0.590626	0.189633	0.068469	3.114577
2000-01-04	0.048945	0.760138	0.543997	0.064390
2000-01-05	0.612580	0.847152	0.044991	0.723104

```
del df1['D'] # del로는 한 번에 하나의 열만 삭제할 수 있습니다.
df1.head()
```

	A	B	C
2000-01-01	0.905683	0.029957	0.026522
2000-01-02	0.533392	0.119775	0.011812
2000-01-03	0.590626	0.189633	0.068469
2000-01-04	0.048945	0.760138	0.543997
2000-01-05	0.612580	0.847152	0.044991

df.pop('열이름') 메소드를 이용해서도  
열을 제거할 수도 있습니다.

## ◦ (전체) 열 이름 변경하기

```
df1.columns = ['a', 'b', 'c']
df1
```

columns를 이용할 경우,  
열 개수를 정확하게 일치시켜 주어야 합니다.  
따라서 열의 개수가 많을 경우에는 불편할 수 있겠죠.

	a	b	c
2000-01-01	0.905683	0.029957	0.026522
2000-01-02	0.533392	0.119775	0.011812
2000-01-03	0.590626	0.189633	0.068469
2000-01-04	0.048945	0.760138	0.543997
2000-01-05	0.612580	0.847152	0.044991
2000-01-06	0.097279	0.160584	0.529673
2000-01-07	0.273220	0.979850	0.178016
2000-01-08	0.936792	0.921132	0.528050

## — (특정) 열 이름 변경하기

```
df1.rename(columns = {'a': '2020', 'b': '2021'}, inplace=True)
df1.tail()      # 전체 데이터 중 뒤에서 5개의 데이터를 보여줍니다.
```

inplace=True를 적은 경우와  
적지 않은 경우를 비교해 볼까요?

	2020	2021	c
2000-01-04	0.048945	0.760138	0.543997
2000-01-05	0.612580	0.847152	0.044991
2000-01-06	0.097279	0.160584	0.529673
2000-01-07	0.273220	0.979850	0.178016
2000-01-08	0.936792	0.921132	0.528050

## 전체 데이터에 대해서 '2020'열의 값을 빼기

```
df1 = df1.sub(df1['2020'], axis='index') # axis=0 이라고 적어도 됩니다.  
df1.tail()
```

	2020	2021	c
2000-01-04	0.0	0.711193	0.495052
2000-01-05	0.0	0.234573	-0.567588
2000-01-06	0.0	0.063304	0.432394
2000-01-07	0.0	0.706630	-0.095204
2000-01-08	0.0	-0.015660	-0.408742

## → 전체 데이터에 대해서 'c'열의 값으로 나누기

```
df1 = df1.div(df1['c'], axis='index')  
df1.tail()
```

	2020	2021	c
2000-01-04	0.0	1.436602	1.0
2000-01-05	-0.0	-0.413280	1.0
2000-01-06	0.0	0.146405	1.0
2000-01-07	-0.0	-7.422253	1.0
2000-01-08	-0.0	0.038313	1.0

## 데이터 프레임을 CSV 파일로 저장하기

```
df1.to_csv('test.csv')
df1
```

	2020	2021	c
2000-01-01	-0.0	0.996092	1.0
2000-01-02	-0.0	0.793007	1.0
2000-01-03	-0.0	0.767955	1.0
2000-01-04	0.0	1.436602	1.0
2000-01-05	-0.0	-0.413280	1.0
2000-01-06	0.0	0.146405	1.0
2000-01-07	-0.0	-7.422253	1.0
2000-01-08	-0.0	0.038313	1.0

test - Excel

파일 홈 삽입 그리기 페이지 레이아웃 수식 데이터 검토 보기 도움말

맑은 고딕 11

붙여넣기 글꼴 맞춤 표시 형식

**데이터가 손실될 수 있음** 이 통합 문서를 실효로 구분된 형식(.csv)으로 저장하면 일부 기능이 손실될 수 있습니다. 기능을 유지하려면 Excel 파일 형식으로 저장하세요.

O16

	A	B	C	D	E	F	G	H
1		2020	2021 c					
2	2000-01-01	0	0.996092	1				
3	2000-01-02	0	0.793007	1				
4	2000-01-03	0	0.767955	1				
5	2000-01-04	0	1.436602	1				
6	2000-01-05	0	-0.41328	1				
7	2000-01-06	0	0.146405	1				
8	2000-01-07	0	-7.42225	1				
9	2000-01-08	0	0.038313	1				
10								
11								

SUMMARY

# 학습정리





## ⚙ pandas 주요 자료형인 Series와 DataFrame

- `s = pd.Series([1, 3, 5, np.nan, 6, 8])`
- `df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list("ABCD"))`

## ⚙ DataFrame의 행 제목과 열 제목

- `pd.DataFrame(np.random.rand(8, 3), index=index, columns= ['A', 'B', 'C'])`
- `df1.columns = ['a', 'b', 'c']`
- `df1.rename(columns={'a': '2020', 'b': '2021'}, inplace=True)`

## ⚙ DataFrame의 열 생성과 삭제

- `df['D'] = df['A'] / df['B']` # A열의 값을 B열의 값으로 나눈 값을 D열에 저장
- `df1 = df.drop(['E', 'F'], axis='columns')`
- `del df1['D']`

