



Samsung Innovation Campus

| Coding and Programming Course

Together for Tomorrow!
Enabling People

Education for Future Generations

Chapter 7.

Data Processing, Descriptive Statistics, and Data Visualization

C&P Course

Chapter Description

◆ Chapter objectives

- ✓ Learners will be able to collect various types of large amounts of data and organize them in a form that can be analyzed.
- ✓ Learners will be able to generate various descriptive statistics for organized data using Pandas.
- ✓ Learners can visualize data using the Python Visualization Library.

◆ Chapter contents

- ✓ Unit 34. Using Python Modules
- ✓ Unit 35. Pandas Series for Data Processing
- ✓ Unit 36. Pandas DataFrame for Data Processing
- ✓ Unit 37. Data Tidying
- ✓ Unit 38. Time Series Data

Unit 34.

Using Python Modules

Learning objectives

- ✓ Learners will be able to explain why modules are grouped into classes, functions, variables, execution codes, etc., and why they are needed.
- ✓ Learners will be able to look at documents on how to use the module and decrypt how to separate and use the functions, classes, and parameters, etc. that the module contains.
- ✓ Learners can select and use the import, from, and as statements in modules appropriately according to need.
- ✓ Learners will be able to generate as many integers and real data as they want using the random function when they encounter situations where unspecified test data is needed.
- ✓ Learners will be able to convert and generate values for a specific date and time using a data module.

Learning overview

- ✓ Be able to use Python's standard module and external module.
- ✓ Be able to use the most commonly used standard library among many standard modules.
- ✓ Be able to create the necessary modules yourself and learn how to use them in other codes.
- ✓ Be able to bring up the entire module, classes, for functions in the module, and learn how to use it in your code.
- ✓ Although we haven't covered the concept of time series yet, be able to generate date and time data using datetime.

Concepts you will need to know from previous units

- ✓ Know how to use Python Functions.
- ✓ Know how to use Python class.

Keywords

import

Standard Module

External Module

random

pip install

datetime

Mission

1. Make a Dice Game!

| What do we need to know about making a dice game?

- ▶ Today, we are going to make a dice game in which the computer and the user take turns to roll a fixed number of dice and see who gets the higher total. Each of them will get one chance to roll again and can decide which of the dices will be held or rerolled.
- ▶ The dice game problem involves a concept of randomness and an element of chance. Instead of deciding on an item, we will let the computer to pick something at random.
- ▶ Through this mission, we will learn how to make our own Python functions, which will allow us to name a block of code and then reuse it later by calling the name.

2. Your Dice Game will look like this!

※ To view the video clip, put the mouse on the box above, and the play button appears. Click it to watch.



| Key concept

1. Modules

1.1. What is a Module?

- | A module enables the Python code to be logically grouped, managed, and used. Normally, one Python .py file becomes one module. Functions, classes, or variables may be defined in the module, and may include an execution code.
- | Simply put, it is a code file.
- | It is divided into a standard module and an external module. The standard module refers to what is basically built into Python. In addition, modules created by other 3rd parties are called external modules.
 - ▶ Standard Module: Built-in modules within Python
 - ▶ External Module: Other modules made by a 3rd party.

1. Modules

1.1. What is a Module?

- | To use these modules, the module may be imported and used, and the import statement may invoke one or more modules within the code as shown below.

```
1 | import Module Name
```

 Line 1

- In case only one module is brought in for use.

```
1 | import Module Name1, [Module Name2, Module Name3, ...]
```

 Line 1

- For cases of multiple modules.

1. Modules

1.1. What is a Module?

| After importing a module using 'import', use the following method when using a specific function or variable of the module.

- ▶ ModuleName.Variable
- ▶ ModuleName.FunctionName()
- ▶ ModuleName.Class()

1. Modules

1.2. Standard Module, Standard Library

Standard modules are installed when installing Python. There is no need to memorize what is in the standard module, as it can be searched through a search engine or through Python's standard library at <https://docs.python.org/3/library/>.

Typical Functions of Standard Libraries

- ▶ Date and Time Modules
- ▶ Numbers and Math Modules
- ▶ File System Modules
- ▶ Operating System Modules
- ▶ Reading and Writing Data Format Modules such as HTML, XML, and JSON
- ▶ Internet Protocol Modules such as HTTP, SMTP, and FTP
- ▶ Multimedia Data Modules such as sound and video
- ▶ Localized Information Modules such as calls and dates

1. Modules

1.2. Standard Module, Standard Library

Since the standard library is built-in, it does not require a separate installation process and can be used immediately by simply importing.

```
1 import math
```

Line 1

- A math module, one of the standard libraries with math-related functions

```
1 # It is used in the form of ModuleName.FunctionName()  
2 math.sin(1)
```

0.8414709848078965

Line 1~2

- 1: It is used in the form of ModuleName.FunctionName()
- 2: An example of using a sin(x) function that obtains the sine value among several functions of the math module.

To find out what other functions the Math module has, such as math.sin(x), visit <https://docs.python.org/3/library/math.html>. In this way, you can learn its basic use and use what you need through searching.

1. Modules

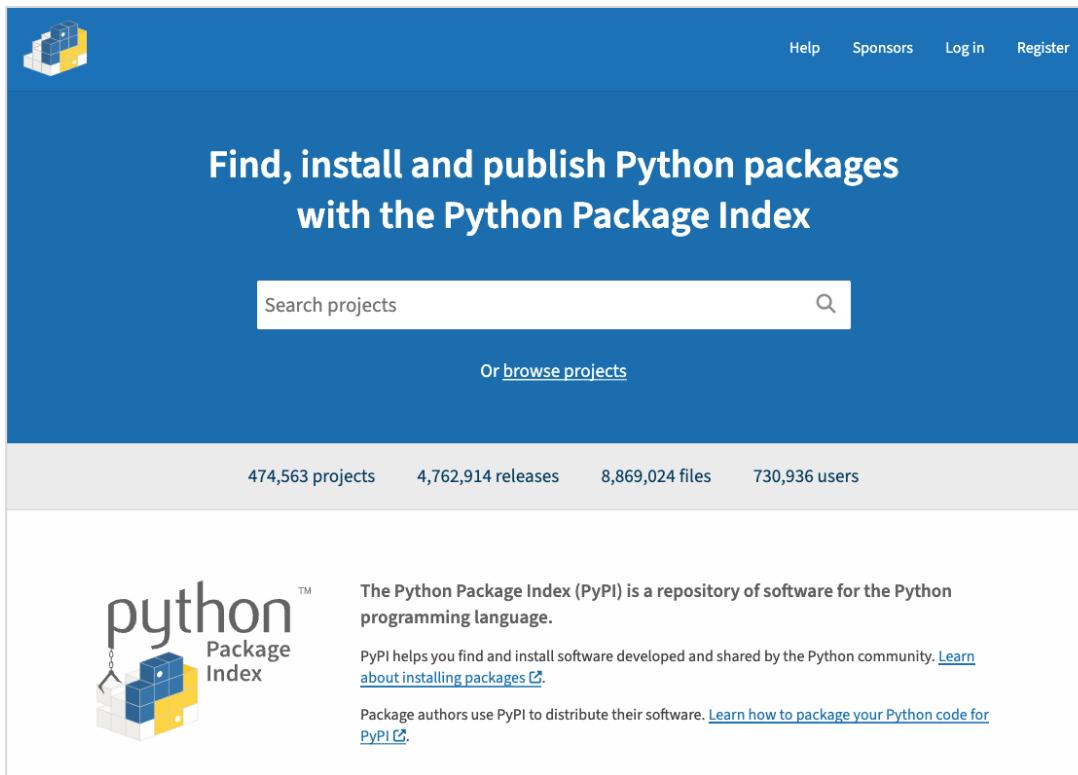
1.3. External Module, External Library

- █ Since external modules were created by other people (such as open-source libraries), the process of installing modules is required in order to use them in code.
- █ The most recommended and safe way to install an external library is to use pip. After Python 3.4, it is basically included in the Python binary installation program and can be easily used.
- █ Pip is a utility that allows for access of a widely used Python package index called PyPI(Python Package Index).
Ex If you want to install a module that has a special function, you can use the pip to install the available candidate library after searching in PyPI.

1. Modules

1.3. External Module, External Library

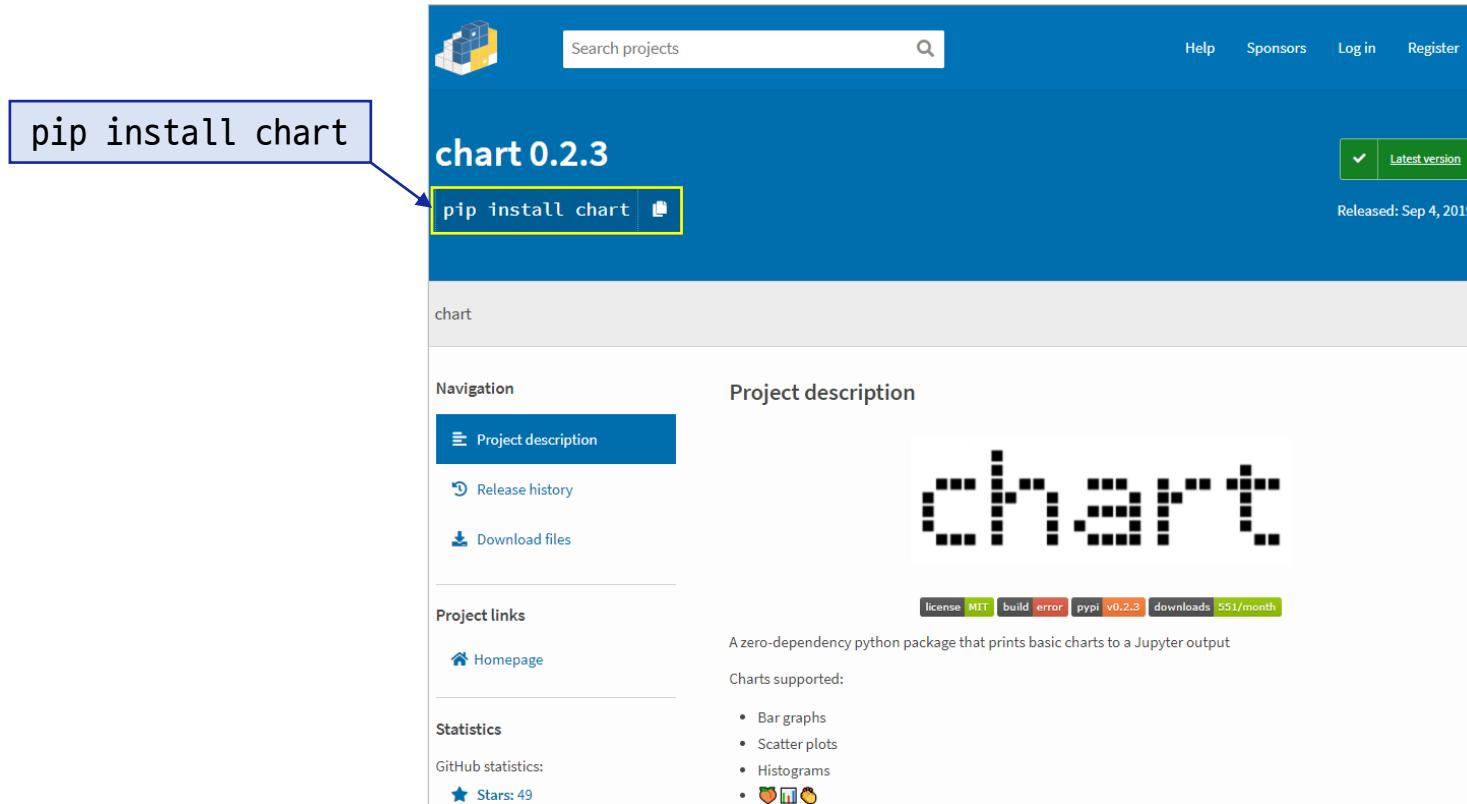
Visit <https://pypi.org/> to search for the necessary functions.



1. Modules

1.3. External Module, External Library

Examine the detail page in the library and copy the pip install under the module name if it is the desired function.



1. Modules

1.3. External Module, External Library

- Run Anaconda prompt and install the library after moving to the virtual environment you are currently using. The library installation instruction used as an example is a pip install chart.

```
(sic) PS C:\Users\User> pip install chart
Collecting chart
  Downloading chart-0.2.3.tar.gz (5.5 kB)
Building wheels for collected packages: chart
  Building wheel for chart (setup.py) ... done
    Created wheel for chart: filename=chart-0.2.3-py3-none-any.whl size=6950 sha256=b3bdfb412e18b8102c607fbbcfbd3a5f648083
4189951c7509cf2db2706339d9
  Stored in directory: c:\users\user\appdata\local\pip\cache\wheels\0a\aa\2\c44a91336651c0d310cdbe63a718edb2f79b3b79
8ac4d2a
Successfully built chart
Installing collected packages: chart
Successfully installed chart-0.2.3
(sic) PS C:\Users\User>
```

1. Modules

1.3. External Module, External Library

```
1 import chart  
2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]  
3 chart.histogram(x)
```



Line 1, 3

- 1: Import the external library you installed through the import command.
- 3: Use histogram(x), which outputs a histogram chart (one of the chart library functions).

1. Modules

1.3. External Module, External Library

| If you run the code and get an error message like the one below, read the message carefully. It is a message showing that the module cannot be found, and in this case, it is an error caused by the library not being installed. This is a mistake that is made more often at the beginner level than expected, and there are cases where a library is not installed in the virtual environment.

```
1 import chart  
2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]  
3 chart.histogram(x)
```

```
ModuleNotFoundError  
Cell In[3], line 1  
----> 1 import chart  
      2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]  
      3 chart.histogram(x)
```

Traceback (most recent call last)

| **ModuleNotFoundError:** No module named 'chart'

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

- In addition to being able to import and use the Python module .py file, the entire script in the module file can be executed immediately. Let's practice bringing up modules through the practice of calculating the hospital funds.

We sell event tickets to raise funds for hospitals.

Each individual participating in the event has to pay $t + 3$.

T is the number of tickets purchased by one person.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1 # Save the name of this file as fund_cal.py.  
2  
3 def fund(t):  
4     return 5*t + 3
```

 Line 1

- Save the name of this file as fund_cal.py.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1 import fund_cal  
2  
3 def main():  
4     t = int(input("Enter the total number of people who participated in the hospital donation event"))  
5     total = fund_cal.fund(t)  
6  
7     print("Total Donation Amount :", total)  
8  
9 main()
```

Line 1

- The plan is to bring and use the fund_cal.py file, a module created and stored above. Be careful not to write the extension for the filename.py.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

Ex An example of when an external file (*.py) is retrieved and the module selects and calls only specific functions.

```
1 # Store this file under the name cal_n.py
2
3 def sum_n(n):
4     return n * (n+1) // 2
5
6 def sum_n2(n):
7     t = 0
8     for i in range(1, n+1):
9         t = t + i
10    return t
```

 Line 1, 3, 4, 6

- 1: Store this file under the name cal_n.py.
- 3: An algorithm that calculates the value obtained by adding all the numbers from 1 to the input n.
- 4: Two slashes are divided into two
- 6: An algorithm to find the sum of consecutive numbers from 1 to input n.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1 import cal_n
2
3 def main():
4     n = int(input("Enter a desired number for calculation : "))
5
6     sum_v = cal_n.sum_n(n)
7     print ("The sum of adding from 1 until", n, "is: ", sum_v)
8
9     sum_vv =cal_n.sum_n2(n)
10    print ("The sum of adding consecutively from 1 until", n, "is: ", sum_vv)
11
12 main()
```

Line 1

- Note that you do not use the .py extension when importing a file into a module.

1. Modules

1.5. Python Syntax: From

The module contains many variables and functions, all of which are extremely rare to find and use 100%. When you want to use only a specific function in the module, you can use the 'From' syntax in the following format.

- from Module import FunctionName
- from Module import ClassName
- from Module import VariableName

1. Modules

1.5. Python Syntax: From

If we apply it to the `math.sin(1)` we practiced earlier, it looks like the following.

```
1 from math import sin  
2 sin(1)
```

0.8414709848078965

 Line 2

- We can use it by only using the function name, as opposed to adding the module name 'math' in the front.

1. Modules

1.5. Python Syntax: From

- Several variables or functions that you want to use in the module can also be used in the following format.

```
1 from math import sin, cos, tan  
2 sin(1)
```

0.8414709848078965

Line 1

- Several functions names can be called at once using ','.

```
1 cos(1)
```

0.5403023058681398

```
1 tan(1)
```

1.557407724654902

1. Modules

1.5. Python Syntax: From

However, if it is inconvenient to use the module name in front of it, then you can code using only the function name. The entire function of the module can be brought using the form 'from ModuleName import*'

```
1 from math import *
```

```
1 sin(1)
```

```
0.8414709848078965
```

Line 1

- Even though the function sin is not written after import, it can be used only with the function name.

1. Modules

1.5. Python Syntax: From

```
1 floor(3.2)
```

3



Line 1

- You can round down without writing the function ‘floor’ after import.

```
1 ceil(4.6)
```

5



Line 1

- You can round up without writing the function ‘ceil’ after import.

1. Modules

1.6. Python Syntax: As

- The name of the module is long, so it is sometimes cumbersome to write the code. And sometimes the names overlap when installing and using an external library. In this case, change the name of the library using the as syntax. It can be used as a short word.

```
import ModuleName as DesiredModuleName(Identifier)
```

```
1 import math as m
```

```
1 m.sin(1)
```

```
0.8414709848078965
```

```
1 m.floor(2.7)
```

```
2
```

1. Modules

1.6. Python Syntax: As

- | When describing ‘from’, it was explained that various variables or function names can be retrieved at once using ‘,’ when importing.

```
from Module import Variable as Name1, Function as Name2, Class as Name3
```

```
1 from math import sin, cos, tan
```

 Line 1

- It can be called one after another using ‘as’ even while changing it using abbreviations.

```
1 from math import sin as s, cos as c
```

```
1 s(1)
```

0.8414709848078965

```
1 c(1)
```

0.5403023058681398

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

- | This module is used to make random numbers. And it can be used to sample and extract some parts from the list.
 - | Random means that the results appear unpredictably every time.
- Ex** When you roll a dice, one number is selected randomly from 1 to 6.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

1) Example of Generating Random Numbers

- ▶ random() is a function belonging to the random module which randomly generates floats between 0 and 1.

```
1 import random
2 i = 0
3 for i in range(10):
4     a = random.random()
5     print(a)
```

```
0.2939696655770577
0.9045712624577867
0.16411869204093776
0.9493556922164117
0.14146141493121367
0.6002837337885266
0.008548281185505324
0.23803173721103665
0.5498142899956122
0.9667998991030945
```



Line 4, 5

- 4: random() randomly generates floats among numbers greater than or equal to 0 and less than 1.
- 5: Each time it is executed, a number between 0 and 1 is randomly returned. Another number returns randomly when 10 loops are executed.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

2) An example of sampling a part of a list or a tuple and extracting randomly

- ▶ `sample` (list name, number of samples) is a function that randomly extracts as many samples as the number of samples from the list. It is used for random sampling without duplication.
- ▶ Instead of the list, a tuple and set may also be used as a collection of data extraction targets.

```
1 import random
2
3 data = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
4 a = random.sample(data, 6)
5
6 print(a)
```

[3, 15, 12, 5, 14, 10]

Line 4

- We randomly extract the six samples from a collection called ‘data’ and store them in the variable a.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

2) An example of sampling a part of a list or a tuple and extracting randomly

- ▶ The choice() function randomly extracts any element from the collection without specifying the number of samples.

```
1 import random
2
3 dog_list = ('Labrador Retriever', 'German Shepherd', 'Bulldog', 'Beagle', 'Yorkshire Terrier')
4 my_lovely_dog = random.choice(dog_list)
5
6 print(my_lovely_dog)
```

Labrador Retriever



Line 3, 4

- 3: Tuple collection made of dog names
- 4: A random element is extracted and stored in the variable my_lovely_dog.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

```
random.random()
```

| Return the next arbitrary floating-point number in the section [0.0, 1.0]

```
1 random.random()
```

```
0.8198361770392332
```

2. Using Typical Standard Libraries

2.2. Functions of the Random Module that Generates an Integer Distribution

random.uniform(a,b)

- | Returns an arbitrary float N that satisfies the condition if $a \leq b$, then $a \leq N \leq b$, and if $b < a$, then $b \leq N \leq a$.
- | The termination value b may or may not be included in the range according to the float position of $a + (b-a) * \text{random}()$.
- | Simply put, you can set it to return any float in the range where a is the minimum value and b is the maximum value.

```
1 import random
2
3 x = random.uniform(1.5, 10)
4 y = random.uniform(10, 1.5)
5
6 print("If a <=b, then a <= N <=b", x)
7 print("If b < a, then b <= N <=a", y)
8 print("Set to return any float in the range where a is the minimum value and b is the maximum value.")
```

If a <=b, then a <= N <=b 3.371571679725208

If b < a, then b <= N <=a 8.393597178289754

Set to return any float in the range where a is the minimum value and b is the maximum value.

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

`random.randint(a, b)`

>Returns any integer ($a \leq N \leq b$) between a and b as the minimum and maximum value, respectively.

```
1 import random  
2  
3 x = random.randint(1, 100)  
4 print(x)
```

45

 Line 3

- Returns an integer between 1 and 100 and stores it in variable x.

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

`random.randint(a, b)`

| Returns any integer N that satisfies $a \leq N \leq b$.

```
1 import random
2
3 roll = random.randint (1, 10)
4 print(f'You rolled {roll}.')
```

You rolled 7.

 Line 3

- Returns an integer between 1 and 10 and stores it in the variable roll.

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

```
random.randrange(start, stop, step )
```

| Returns an arbitrary integer as a step from the start value to the stop value.

```
1 import random  
2  
3 x = random.randrange(1, 100, 2)  
4 print(x)
```

41



TIP

- A module that generates random numbers should not be used for security purposes. For security or encryption, it is recommended to use the secrets module.

2. Using Typical Standard Libraries

2.4. Time Module

```
time.sleep(secs)
```

- | It is the most commonly used function among time module functions.
- | It functions to pause the execution of the thread called for a given second.

```
1 import time
2
3 for i in range(10):
4     print("Hello", i+1)
5     time.sleep(2)
```

```
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
```

Line 5

- All executions are paused for 2 seconds at this part while the loop is in execution.

2. Using Typical Standard Libraries

2.4. Time Module

```
time.time(), time.ctime()
```

```
1 import time
2
3 a = time.time()
4 b = time.ctime(a)
5
6 print(a)
7 print (b)
```

```
1692108918.286222
Tue Aug 15 23:15:18 2023
```

Line 3, 4

- 3: It is a function of finding the current time. As of 0h 0m 0s on January 1st, 1970, it informs you of the past time in seconds. However, the return value is returned to a real value that is difficult to read.
- 4: It is a function for returning the form of time that we can understand.

Unix timestamp

- ▶ It is also called Epoch time. The elapsed time from 00:00:00(UTC) on January 1st, 1970 is converted into seconds and expressed as an integer.

2. Using Typical Standard Libraries

2.4. Time Module

```
time.sleep(secs), time.ctime()
```

| Let's make a simple electronic watch.

```
1 import time  
2  
3 for i in range(10):  
4     time.sleep(1)  
5     print(time.ctime())
```

```
Tue Aug 15 23:18:53 2023  
Tue Aug 15 23:18:54 2023  
Tue Aug 15 23:18:55 2023  
Tue Aug 15 23:18:56 2023  
Tue Aug 15 23:18:58 2023  
Tue Aug 15 23:18:59 2023  
Tue Aug 15 23:19:00 2023  
Tue Aug 15 23:19:01 2023  
Tue Aug 15 23:19:02 2023  
Tue Aug 15 23:19:03 2023
```

2. Using Typical Standard Libraries

2.4. Time Module

```
time.strftime('Format', Time object)
```

- I When the %y format code is provided, a two-digit year can be parsed. Values 69 to 99 are mapped from 1969 to 1999, and values 0 to 68 are mapped from 2000 to 2068.

%a : Name of the Week

%b : Name of the Month

%d : Day of the Month in Decimal

%Y : Year in Decimal

- ▶ <https://docs.python.org/3/library/time.html?highlight=time#time.strftime>

2. Using Typical Standard Libraries

2.4. Time Module

```
time.strftime('Format', Time object)
```

```
1 from time import gmtime, strftime  
2 strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime ())
```

```
'Tue, 15 Aug 2023 14:23:32 +0000'
```

```
1 from time import localtime, strftime  
2 strftime ("%a, %d %b %Y %H:%M:%S +0000", localtime())
```

```
'Tue, 15 Aug 2023 23:23:32 +0000'
```

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Module

As a module related to date and time, it is often used to create date formats. Below is a summary of various cases that print out basic dates.

`datetime.date.today()`

- ▶ Returns the current date.

```
1 import datetime  
2  
3 today = datetime.date.today()  
4  
5 print(today)  
6 print(today.year, "year")  
7 print(today.month, "month")  
8 print(today.day, "day")
```

2023-08-15

2023 year

8 month

15 day



Line 3, 6

- 3: Returns the current date to the variable today and stores it
- 6: Separates the information about year

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Module

As a module related to date and time, it is often used to create date formats. Below is a summary of various cases that print out basic dates.

`datetime.date.today()`

- ▶ Returns the current date.

```
1 import datetime  
2  
3 today = datetime.date.today()  
4  
5 print(today)  
6 print(today.year, "year")  
7 print(today.month, "month")  
8 print(today.day, "day")
```

2023-08-15
2023 year
8 month
15 day



Line 7, 8

- 7: Separates the information about month
- 8: Separates the information about date

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Module

`datetime.datetime.now()`

- ▶ Returns the current date up to hours, minutes, and seconds.

```
1 import datetime  
2  
3 now = datetime.datetime.now()  
4  
5 print(now)  
6 print(now.year, "year")  
7 print(now.month, "month")  
8 print(now.day, "day")  
9 print(now.hour, "hour")  
10 print(now.minute, "minute")  
11 print(now.second, "second")
```

2023-08-15 23:29:26.534533

2023 year

8 month

15 day

23 hour

29 minute

26 second

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Module

```
1 import datetime  
2  
3 now = datetime.datetime.now()  
4  
5 print(now)  
6 print(now.year, "year")  
7 print(now.month, "month")  
8 print(now.day, "day")  
9 print(now.hour, "hour")  
10 print(now.minute, "minute")  
11 print(now.second, "second")
```



Line 3, 6, 7

- 3: Year, month, day, hour, minute, second
- 6: Separates the information about year
- 7: Separates the information about month

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Module

```
1 import datetime  
2  
3 now = datetime.datetime.now()  
4  
5 print(now)  
6 print(now.year, "year")  
7 print(now.month, "month")  
8 print(now.day, "day")  
9 print(now.hour, "hour")  
10 print(now.minute, "minute")  
11 print(now.second, "second")
```



Line 8, 9, 10, 11

- 8: Separates the information about day
- 9: Separates the information about hour
- 10: Separates the information about minute
- 11: Separates the information about second

2. Using Typical Standard Libraries

2.6. Using `datetime.timedelta` to Indicate the Difference Between Two Dates and Time

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
```

All factors are optional, not essentials, and the default value is zero. You can enter it as an integer or float. You can use both positive and negative numbers.

- ▶ Milliseconds are converted into 1000 microseconds.
- ▶ Minutes are converted into 60 seconds.
- ▶ Hours are converted into 3600 seconds.
- ▶ Weeks are converted into 7 days.

2. Using Typical Standard Libraries

2.6. Using `datetime.timedelta` to Indicate the Difference Between Two Dates and Time

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
```

```
1 from datetime import timedelta  
2  
3 delta = timedelta(  
4     days=50,  
5     seconds=27,  
6     microseconds=10,  
7     milliseconds=29000,  
8     minutes=5,  
9     hours=8,  
10    weeks=2)  
11  
12 print(delta)
```

```
64 days, 8:05:56.000010
```

2. Using Typical Standard Libraries

2.6. Using `datetime.timedelta` to Indicate the Difference Between Two Dates and Time

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
```

| Let's find the date that is 30 days from May 3rd, 2000.

```
1 from datetime import timedelta
2
3 d = datetime.datetime(2000, 5, 3)
4
5 delta = datetime.timedelta(days = 30)
6
7 print(d + delta)
```

```
2000-06-02 00:00:00
```



Line 3

- Objects can be created by adding years, months, days, hours, minutes, seconds, and microseconds to `datetime.datetime`.

2. Using Typical Standard Libraries

2.7. Using Replace to Replace Elements of a Specific Time

```
datetime.replace(year=, month= , day= , hour=, minute=, second=, microsecond= )
```

- ▶ You can change the elements of a specific time.

```
1 from datetime import datetime
2
3 now = datetime.now()
4
5 print(now)
6
7 replace_time = now.replace(month = 12, day = 30)
8
9 print(replace_time)
```

2023-08-15 23:38:34.948122

2023-12-30 23:38:34.948122



Line 7

- Change the month to December and the day to the 30th.

2. Using Typical Standard Libraries

2.8. OS Modules with Functions Related to the Operating System

| It is a module that has functions related to the operating system. You can create a new folder on our computer or look at the list of files inside the folder using the OS module.

- `os.mkdir("Folder Name")`: Creates a folder.
- `os.rmdir("Folder Name")`: Deletes the folder.
- `os.getcwd()`: Returns the current path.
- `os.listdir()`: Inquires the list of files and directories

2. Using Typical Standard Libraries

2.8. OS Modules with Functions Related to the Operating System

```
1 import os  
2 os.getcwd()
```

```
'C:\\\\Users\\\\User\\\\Documents\\\\sic_project'
```

```
1 import os  
2 os.listdir()
```

```
['__pycache__',  
'Ch07_Units34.ipynb',  
'cal_n.py',  
'.ipynb_checkpoints',  
'fund_cal.py']
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1. Randomly select three multiples of 5 from the range 0 to 100 using the random module and print them in the form of a list.



Write the entire code and the expected output results in the note.

Q2.. Use timedelta to create a program that prints a 100-day anniversary from a special day of yours. It doesn't have to be a 100-day anniversary, so feel free to make your own special anniversary calculator.



Write the entire code and the expected output results in the note.

| Let's code

Steps for Writing Python Code

STEP 1

- I Ask the user to enter the number of dice. Then, convert the data type of the number of dice from string to integer.

```
# step1 in main program area - start game
number_dice = input('Enter number of dice:')
number_dice = int(number_dice)
```

STEP 2

- I Ask the user to press any key to start the dice game.

```
ready = input('Ready to start? Hit any key to continue')
```

STEP 3

- I We need to generate random numbers for rolling a dice. We're going to import and use the random module from the Python library. Write the line on the very top of the code.

```
import random
```

Steps for Writing Python Code

STEP 4

- I Define a function named “roll_dice(n).” Make sure to create the function right after the import statement. It will use a parameter “n,” which is the number of dice. We are going to call this function in the main program for rolling a dice. Within the roll_dice(n) function, create an empty list that will store the dice numbers rolled.

```
def roll_dice(n):
    dice = [] # start with empty list of dice
```

STEP 5

- I Add random numbers that are between 1 and 6 to the list of dice. Create the numbers as much as the number of dice indicated as a parameter. When you’re done with appending all the random numbers required to the list, return the list of dice.

```
def roll_dice(n):
    dice = [] # start with empty list of dice
    # add random numbers between 1 to 6 to the list
    for i in range(n):
        dice.append(random.randint(1,6))
    return dice
```

Steps for Writing Python Code

STEP 6

- In the main programming area, call the roll_dice(n) function with number_dice as a parameter. Name the list of dice returned by the function as “user_rolls.” Display the user’s first roll on the screen.

```
# step 2 in main program area - roll dice
# User turn to roll
user_rolls = roll_dice(number_dice)
print('User first roll: ', user_rolls)
```

Steps for Writing Python Code

STEP 7

- I Get the user's choices for holding on or rerolling each dice. Check if the user enters the right number of choices that matches the number of dice. If the length of the user's input and the number of dice do not match, ask the user to re-enter the choices.

```
# step 3 - get user choices
user_choices = input("Enter - to hold or r to \
roll again :")
# check length of user input
while len(user_choices) != number_dice:
    print('You must enter', number_dice, \
        'choices')
    user_choices = input("Enter - to hold or r \
        to roll again :")
```

Steps for Writing Python Code

STEP 8

- We are going to use a `sleep()` function to pause the program for seconds. Import the time module from the Python library. Write the code below the statement for importing the random module.

```
import random  
  
import time
```

STEP 9

- Define a function named “`roll_again(choices, dice_list)`,” two lines after the function definition of `roll_dice(n)`. It will use parameters of either the user or computer’s choices and of the list of dices. Display a message, and pause the program for 3 seconds to wait for the computer to roll a dice. Refer to the side note on the next slide for using the `sleep()` function of the time module.

```
def roll_again(choices, dice_list):  
    print('Rolling again ...')  
    time.sleep(3)
```

Steps for Writing Python Code

STEP 10

- Step through the choices, and if a character within the string is “r” (roll again), replace the dice at the index of the dice list to a new random number between 1 and 6. When you’re done with the for loop, pause the program again for 3 seconds. Since the function is a void function, it does not return anything at the end.

```
def roll_again(choices, dice_list):
    print('Rolling again ...')
    time.sleep(3)
    for i in range(len(choices)):
        if choices[i] == 'r':
            dice_list[i] = random.randint(1,6)
    time.sleep(3)
```

Important

`time.sleep(secs)`

- ▶ pauses, stops, waits, or sleeps your Python program for secs. Here, secs is the number of seconds that the Python code should pause execution, and the argument should be either an integer or a float.

Steps for Writing Python Code

STEP 11

- In the main programming area, call the function roll_again(choices, dice_list) with user_choices and user_rolls as parameters. The list of dice will be updated through the execution of the function, based on the user's choices. Display the user's new roll on the screen.

```
# step 4 - roll again based on user choices
roll_again(user_choices, user_rolls)
print('Player new Roll: ', user_rolls)
```

STEP 12

- Now, it's the computer's turn to roll a dice. Call the function roll_dice(n) with number_dice as a parameter. Name the list of dice returned by the function as "computer_rolls." Display the computer's first roll on the screen.

```
# step 5 - computer's turn to roll
print('Computers turn ')
computer_rolls = roll_dice(number_dice)
print('Computer first roll: ', computer_rolls)
```

Steps for Writing Python Code

STEP 13

- | Define a function named “computer_strategy(n),” which will decide on the computer’s choices for whether to hold on or to re-roll each dice. It will use a parameter “n,” which is the number of dice. Display a message that the computer is thinking, and pause the program for 3 seconds. Also, create an empty list that will store the computer’s choices on each dice.

```
def computer_strategy1(n):
    # create computer choices : roll everything again
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
```



TIP

- You can create an empty list in three ways:
 - ① Name of an empty list = []
 - ② Name of an empty list = list[]
 - ③ Name of an empty list = ''

Steps for Writing Python Code

STEP 14

- Next, step through each element in the list of dice for the computer. If the dice in the computer's dice list is less than 5, append "r" (roll again) to the computer's choices list. If it is 5 or 6, append "-" (hold) to the computer's choices list. Lastly, return the list of choices that the computer has made.

```
def computer_strategy2(n):
    # create computer choices: roll if < 5
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        if computer_rolls[i] < 5:
            choices = choices + 'r'
        else:
            choices = choices + '-'
    return choices
```

Steps for Writing Python Code

STEP 15

- In the main programming area, call the function computer_strategy(n) with number_dice as a parameter. The list of dice will be updated through the execution of the function, based on the user's choices. Display the user's new roll on the screen.

```
# step 6
# decide on what choice - using one of the strategy functions
computer_choices = computer_strategy2(number_dice)
print('Computer Choice: ', computer_choices)
```

STEP 16

- Now, it's the computer's turn to roll again. Call the function roll_again(choices, dice_list) with computer_choices and computer_rolls as parameters. The list of dice will be updated through the execution of the function based on the computer's choices. Display the computer's new roll on the screen.

```
# Computer rolls again using the choices it made
roll_again(computer_choices, computer_rolls)
print('Computer new Roll: ', computer_rolls)
```

Steps for Writing Python Code

STEP 17

- I Define a function named “find_winner(cdice_list, udice_list),” which will determine the winner for the dice game. It will use the lists of dices for each computer and user as parameters. With the Python’s sum() function, calculate the totals of the dice numbers from each computer and user’s list of dices. Then, display the totals on the screen.

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total', user_total )
```

Important

sum(iterable, start)

- ▶ returns a number, the sum of all items in an iterable. Here, iterable is a required parameter, which is the sequence or a list of numbers to sum. On the other hand, start is an optional parameter, which is a value that is added to the return value.

Steps for Writing Python Code

STEP 18

- If the user has a higher total, display that the user is a winner. If the computer has a higher total, display that the computer is a winner. Otherwise, the user and the computer have the same total, so display that it is a tie. Since the function is a void function, it does not return anything.

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total', user_total)
    if user_total > computer_total:
        print('User wins')
    elif user_total < computer_total:
        print('Computer wins')
    else:
        print('It is a tie!')
```

Steps for Writing Python Code

STEP 19

- In the main programming area, call the `find_winner(cdice_list, udice_list)` with `computer_rolls` and `user_rolls` as parameters. The parameters, which are the lists of dices, have been updated through the previous functions. The program will end by determining the winner of the dice game in the `find_winner(cdice_list, udice_list)` function.

```
# final line in code - deciding who wins
find_winner(computer_rolls,user_rolls)
```

Finalized Code for a Dice Game

```
import random

import time

def roll_dice(n):
    dice = [] # start with empty list of dice
    # add random numbers between 1 to 6 to the list
    for i in range(n):
        dice.append(random.randint(1,6))
    return dice
```

Finalized Code for a Dice Game

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total', user_total )
    if user_total > computer_total:
        print('User wins')
    elif user_total < computer_total:
        print('Computer wins')
    else:
        print('It is a tie!')

def roll_again(choices, dice_list):
    print('Rolling again ...')
    time.sleep(3)
    for i in range(len(choices)):
        if choices[i] == 'r':
            dice_list[i] = random.randint(1,6)
    time.sleep(3)
```

Finalized Code for a Dice Game

```
def computer_strategy1(n):
    # create computer choices : roll everything again
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        choices = choices + 'r'
    return choices

def computer_strategy2(n):
    # create computer choices: roll if < 5
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        if computer_rolls[i] < 5:
            choices = choices + 'r'
        else:
            choices = choices + '-'
    return choices
```

Finalized Code for a Dice Game

```
# step1 in main program area - start game
number_dice = input('Enter number of dice:')
number_dice = int(number_dice)
ready = input('Ready to start? Hit any key to continue')

# step 2 in main program area - roll dice
# User turn to roll
user_rolls = roll_dice(number_dice)
print('User first roll: ', user_rolls)

# step 4 - get user choices
user_choices = input("Enter - to hold or r to \
roll again :")
# check length of user input
while len(user_choices) != number_dice:
    print('You must enter', number_dice, \
    'choices')
    user_choices = input("Enter - to hold or r \
    to roll again :")
```

Finalized Code for a Dice Game

```
# step 5 - roll again based on user choices
roll_again(user_choices, user_rolls)
print('Player new Roll: ', user_rolls)

# Computer's turn to roll
print('Computers turn ')
computer_rolls = roll_dice(number_dice)
print('Computer first roll: ', computer_rolls)

# step 6
# decide on what choice - using one of the strategy functions
computer_choices = computer_strategy2(number_dice)
print('Computer Choice: ', computer_choices)
# Computer rolls again using the choices it made
roll_again(computer_choices, computer_rolls)
print('Computer new Roll: ', computer_rolls)

# final line in code - deciding who wins
find_winner(computer_rolls,user_rolls)
```

| 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.

Complete the creative artwork by looking at the document with your colleagues.

- | The sample code below executes an artwork using cool turtle graphics. Turtle is also one of Python's standard libraries and is very easy to use.
- | Descriptions for the Python Turtle Graphics Module is in the link below
- | <https://docs.python.org/3/library/turtle.html?highlight=turtle#module-turtle>.

```
1 # make a geometric rainbow pattern
2 import turtle
3 colors = ['red', 'yellow', 'blue', 'orange', 'green', 'red']
4
5 aiden= turtle.Turtle()
6 turtle.bgcolor('black') # turn background black
7 # make 36 hexagons, each 10 degrees apart
8
9 for n in range(36):
10 # make hexagon by repeating 6 times
11     for i in range(6):
12         aiden.color(colors[i]) # pick color at position i
13         aiden.forward(100)
14         aiden.left(60)
15     # add a turn before the next hexagon
```

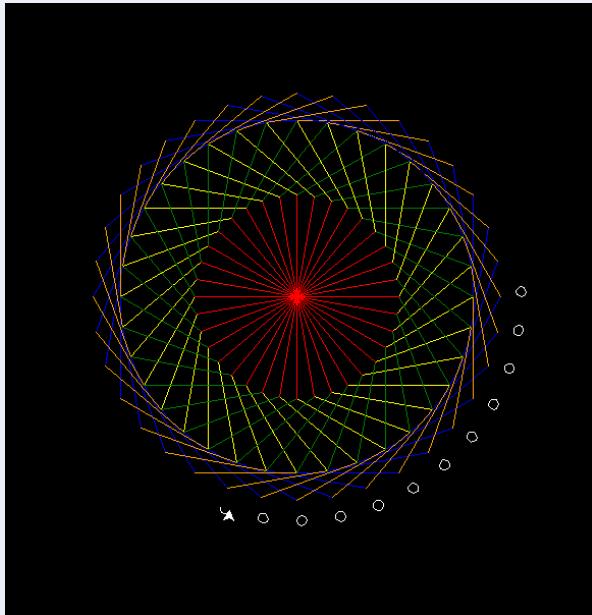
Q1.

Complete the creative artwork by looking at the document with your colleagues.

```
16    aiden.right(10)
17
18 # get ready to draw 36 circles
19 aiden.penup()
20 aiden.color('white')
21 # repeat 36 times to match the 36 hexagons
22 for i in range(36):
23     aiden.forward(220)
24     aiden.pendown()
25     aiden.circle(5)
26     aiden.penup()
27     aiden.backward(220)
28     aiden.right(10)
29 # hide turtle to finish the drawing
30 aiden.hideturtle()
```

Q1.

Complete the creative artwork by looking at the document with your colleagues.

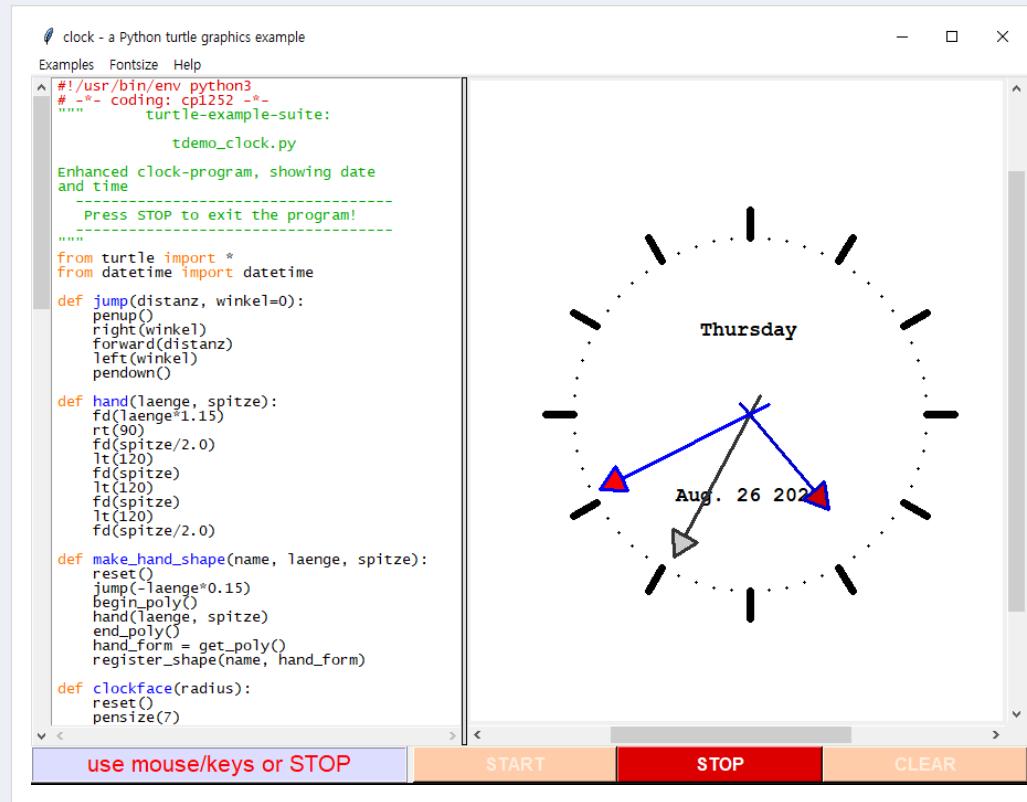


Line 1, 6, 7, 10, 12, 15, 18, 21, 29

- 1: make a geometric rainbow pattern
- 6: turn background black
- 7: make 36 hexagons, each 10 degrees apart
- 10: make hexagon by repeating 6 times
- 12: pick color at position i
- 15: add a turn before the next hexagon
- 18: get ready to draw 36 circles
- 21: repeat 36 times to match the 36 hexagons
- 29: hide turtle to finish the drawing

Q1.

Complete the creative artwork by looking at the document with your colleagues.



When you run the help → turtle menu in the menu selection in Python Idle, the demo window, as shown on the left, is executed. You can take a look at this one by one with your pair programming colleague and choose a demo you want to use to change the artwork.

Unit 35.

Pandas Series for Data Processing

Learning objectives

- ✓ Be able to install the latest version of pandas library to the current virtual environment.
- ✓ Be able to distinguish between series and dataframe when pandas-based data set is given.
- ✓ Be able to process list and dictionary data sets into series.
- ✓ Be able to select a specific data element from the series data set and perform arithmetic operation.
- ✓ Be able to differentiate data feature and draw a suitable graph.

Learning overview

- ✓ Learn how to install the pandas library and other basic dependent libraries to the virtual environment
- ✓ Learn about two different types of pandas data structures
- ✓ Learn how to created series by using the list, dictionary, numpy, and scalar values
- ✓ Learn how to find a specific element in the series
- ✓ Learn how to perform arithmetic operation within the series structure
- ✓ Learn to visualize the series data set by using the matplotlib

Concepts you will need to know from previous units

- ✓ How to install and import python modules
- ✓ How to perform arithmetic operation
- ✓ Python data structures including list, dictionary, etc.

Keywords

Pandas

Series

Data element

index

Line graph

Bar graph

Mission

What is population density?

- | Population density is the ratio of population in the unit area of a certain region. It is normally expressed as number of population in 1 km^2 .
- | Population density can be used to explain the location, growth, and movement of many organisms.
In the case of human, population density is often used for urbanization, immigration, and population statistics.
Statistics related to global population density is traced by the UN Bureau of Statistics.
- | Population density is easy to calculate.
 - ▶ $\text{Population density} = \text{Number of population} / \text{Area} (\text{km}^2)$

1. Convert the attached dictionary data into series objects.
2. Calculate the population fluctuation of each city in each year and express it as a bar graph.
3. Calculate the population density of each capital and find the cities with the maximum and minimum population density.
4. Visualize the population density calculation results.

- | Use two attached dictionary data for solving the mission.
- | Data source is <https://unstats.un.org/> which organized 15 cities with largest populations.

What is population density?

```
1 # population_2020 is dictionary data object that stores total population in 2020.
2 population_2020 = {'Tokyo': 37339804,
3                     'Delhi': 31181376,
4                     'Shanghai': 27795702,
5                     'Sao Paulo': 22237472,
6                     'Mexico City': 21918936,
7                     'Dhaka': 21741090,
8                     'Cairo': 21322750,
9                     'Karachi': 16459472,
10                    'Istanbul': 15415197,
11                    'Buenos Aires': 15257673,
12                    'Kinshasa': 14970460,
13                    'Lagos': 14862111,
14                    'Manila': 14158573,
15                    'Rio de Janeiro': 13544462,
16                    'Moscow': 12593252,
17                    'Bogota': 11167392,
18                    'Paris': 11078546,
19                    'Jakarta': 10915364,
20                    'Lima': 10882757}
```



Line 1

- population_2020 is dictionary data object that stores total population of each city in 2020.

What is population density?

```
1 # population_2021 is dictionary data object that stores total population in 2021.
2 population_2021 = {'Tokyo': 37393128,
3                     'Delhi': 30290936,
4                     'Shanghai': 27058480,
5                     'Sao Paulo': 22043028,
6                     'Mexico City': 21782378,
7                     'Dhaka': 21005860,
8                     'Cairo': 20900604,
9                     'Karachi': 16093786,
10                    'Istanbul': 15190336,
11                    'Buenos Aires': 15153729,
12                    'Kinshasa': 14342439,
13                    'Lagos': 14368332,
14                    'Manila': 13923452,
15                    'Rio de Janeiro': 13458075,
16                    'Moscow': 12537954,
17                    'Bogota': 10978360,
18                    'Paris': 11017230,
19                    'Jakarta': 10770487,
20                    'Lima': 10719188}
```



Line 1

- population_2021 is dictionary data object that stores total population of each city in 2021.

What is population density?

```
1 # area is the area data of each city, and the unit is km^2.
2 city_area = {'Tokyo': 2194,
3               'Delhi': 1484,
4               'Shanghai': 6340,
5               'Sao Paulo': 1521,
6               'Mexico City': 1485,
7               'Dhaka': 306.4,
8               'Cairo': 3085,
9               'Karachi': 3780,
10              'Istanbul': 5343,
11              'Buenos Aires': 203,
12              'Kinshasa': 9965,
13              'Lagos': 1171,
14              'Manila': 42.88,
15              'Rio de Janeiro': 1255,
16              'Moscow': 2511,
17              'Bogota': 1775,
18              'Paris': 105.4,
19              'Jakarta': 661.5,
20              'Lima': 2672}
```



Line 1

- area is the area data of each city, and the unit is km^2 .

| Key concept

1. Introduction of Pandas

- | Developed in 2008 by Wes McKinney, pandas has been an open source since 2009 as a library for data processing.
- | Pandas was originally designed for time series data operation and analysis in finance, especially stock price. To perform such operations, analytic tools including searching, indexing, refining, arranging, reshaping, and slicing were required, and pandas was developed as a solution.

1. Introduction of Pandas

Ever since switched to an open source, pandas supports effective data-related functions as follows due to the efforts made by many people. The list of functions provided below was complete by referring to <https://Pandas.pydata.org>.

- ▶ Quick and effective series and dataframe objects for operating data with integrated indexing
- ▶ Intelligent data arrangement by using index and label
- ▶ Finding omitted data and supporting integrated processing
- ▶ Converting unorganized data into organized data
- ▶ Built-in tool to read and write data in the in-memory file, database, and web service
- ▶ Be able to process data stored in csv, excel, and json formats
- ▶ Flexible pivoting and remodeling of data set
- ▶ Slicing with index values, indexing, and subsetting of data set
- ▶ Addition and deletion of seta set columns
- ▶ Split-apply-combine functions to combine or change data as a powerful data grouping tool
- ▶ Integration with high performance data set merge
- ▶ Performance optimization through the critical path written with Cpython or C

2. How to Use Pandas

- | It is most appropriate to use pandas for data operation. However, not all issues related to comprehensive data can be solved only with pandas.
- | Instead of being a comprehensive data analytic tool, pandas is a data operation tool that has some level of analytic function.
- | For more in-depth analysis, data operation, and data visualization in different fields, use libraries including Scipy, Numpy, scikit-learn, matplotlib, seaborn, ggplot along with pandas. Likewise, a great advantage of pandas is that there are many python-based libraries to connect with.

2. How to Use Pandas

I Advantages of learning pandas

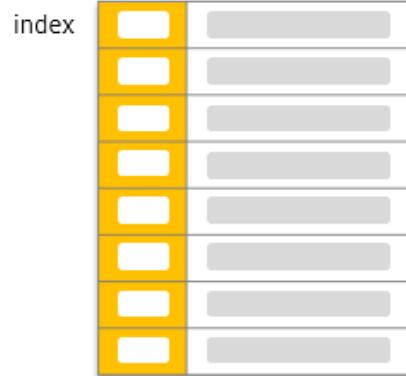
- ▶ It is necessary to learn data processing for machine learning or AI deep learning, and even if not working in the AI field, data processing and analysis that go further beyond excel in regular working environment will be practically great advantage.

 <p><i>Python Pandas Features</i></p>	Cleaning up Data	Python support	Visualize
	Great handling of data	Grouping	Optimized performance
	Alignment and indexing	Merging and joining of datasets	Unique data
	Multiple file formats supported	Input and output tools	Mask data
	Handling missing data	Lot of time Series	

3. Pandas Has Two Different Data Structures

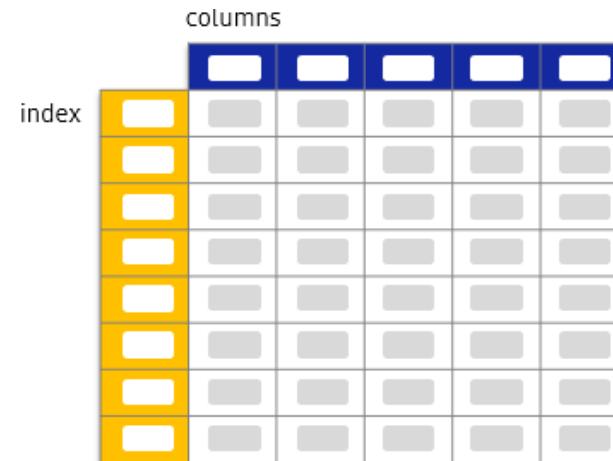
- Series and DataFrame are two different data structure types of Pandas. It is extremely important to fully understand those two data structures.
- The difference between the two data structures is that while Series has one-dimensional array data, DataFrame has two-dimensional array data.

Series



Sequentially arranged one-dimensional array
Index (yellow) and data (white) are 1:1.

DataFrame



Two-dimensional array consists of index and column (blue)
Adding multiple series can make one dataframe.

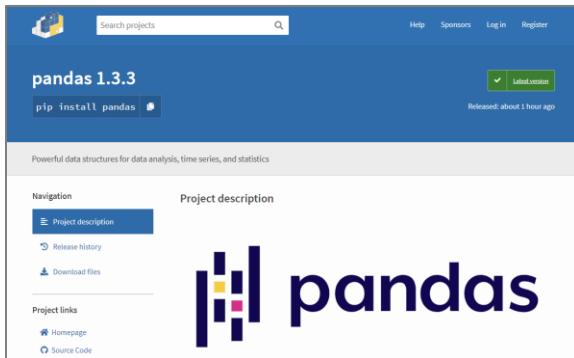
4. Install Pandas Library

1) The following are the official python versions for pandas installation.

Officially Python 3.8, 3.9, 3.10, and 3.11.

- ▶ https://Pandas.pydata.org/docs/getting_started/install.html#python-version-support for reference

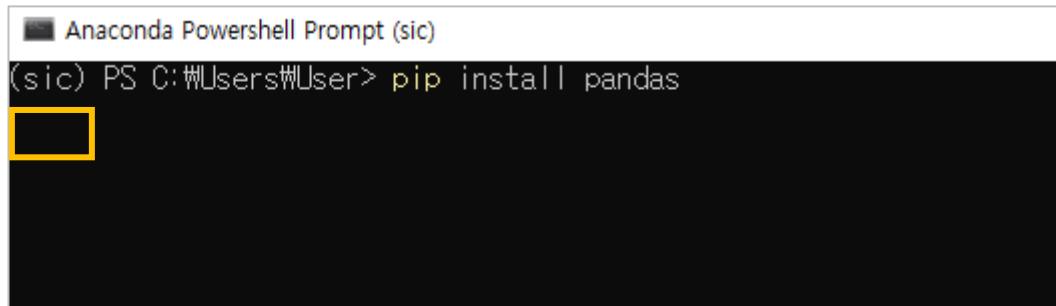
2) There are many different ways to install pandas, but this unit will only introduce pandas downloading from PyPI just like for external library installation introduced previously.



- ▶ Just like any other external libraries, pandas can be installed from PyPI by using pip.
- ▶ <https://pypi.org/project/Pandas/>

4. Install Pandas Library

- 3) Make sure to run pip install Pandas after executing anaconda prompt and moving to the virtual environment.
Installation must be done after moving to the currently used virtual environment. This is the most fundamental thing to remember for all of the other external library installation, but many beginners make mistakes.



- 4) After installation, the Pandas.version command checks the installed pandas library version to determine if the installation is done appropriately.

```
1 import pandas as pd  
2 pd.__version__
```

'2.0.2'

4. Install Pandas Library

5) Possible error after module installation (Solution for import ERROR).

- ▶ In general, ImportError means that it is impossible to find pandas from the python library list. Python has built-in directory list to find a package.
- ▶ Most of the time, error occurs because different versions of python are installed in the computer, and if the module is not installed to the currently used version of python installation environment but in python of a different version.
- ▶ Use the built-in python module sys to locate.

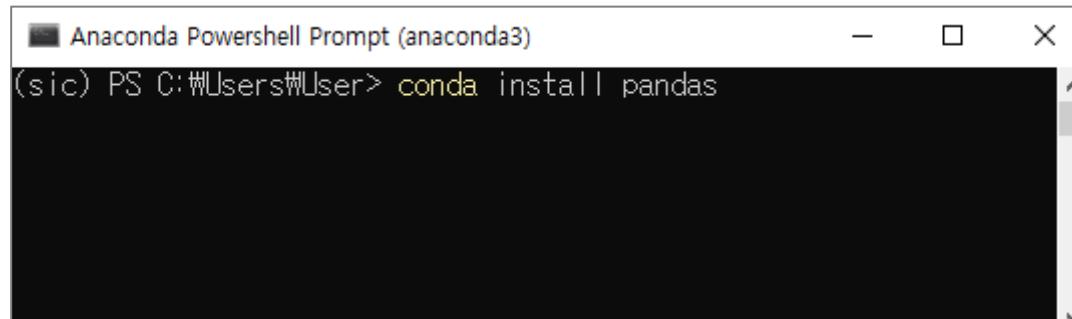
```
1 import sys
2 sys.path
['C:\\\\Users\\\\User\\\\Documents\\\\sic_project',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\python38.zip',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\DLLs',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib\\\\site-packages',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib\\\\site-packages\\\\win32',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib\\\\site-packages\\\\win32\\\\lib',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib\\\\site-packages\\\\Pythonwin',
'C:\\\\Users\\\\User\\\\anaconda3\\\\envs\\\\sic\\\\lib\\\\site-packages\\\\IPython\\\\extensions',
'C:\\\\Users\\\\User\\\\.ipython']
```

- ▶ Linux/Mac can run a specific python with terminal and show which python installation is used. It is not recommended to use “/user/bin/python” because it uses the basic python of the system.

4. Install Pandas Library

6) The most precise and reliable installation method is to use conda.

- ▶ Conda is an open source package and environment management system that is executive in Windows, macOS, and Linux. Coda can easily and quickly install, execute, and update the package and relevant dependency. It is originally developed for python programs, but software packaging and distribution are possible for any other languages (Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN).
- ▶ Similar to installing other modules from PyPI, conda can be installed through the pip install conda, the it is highly likely that it's not in the latest version.
- ▶ Because conda is included in all versions of Anaconda®, Miniconda and Anaconda Repository, it is recommended to install Anaconda or Miniconda first.



The screenshot shows a black terminal window titled "Anaconda Powershell Prompt (anaconda3)". The command "(sic) PS C:\Users\User> conda install pandas" is visible at the top of the window. A cursor is positioned at the end of the command line. Below the window, a text box contains the command "conda install Pandas(PKGNAME)==2.0.2(version)".

conda install Pandas(PKGNAME)==2.0.2(version)

5. Optional Dependencies

- | When using pandas, there are many cases where you'd need dependencies packages to use specific methods. If you do not have them, it would result in difficulties. When you experience import error even if you used methods as learned from a book or lecture, you need to check dependencies packages.
 - | It is recommended to first install dependencies packages before using pandas.
- Ex** While the `read_hdf()` requires `pytables` package, `DataFrame.to_markdown()` requires `tabulate` package.

5. Optional Dependencies

The following table shows some of the optional dependencies required for this unit. It is mandatory to install them before continuing this chapter.

For computation	- Scipy : Miscellaneous statistical functions - numba : Alternative execution engine for rolling operations
Excel operation	- xlrd : Reading Excel - xlwt : Writing Excel - xlsxwriter : Writing Excel - openpyxl : Reading / writing for xlsx files - pyxlsb : Reading for xlsb files
HTML operation	- BeautifulSoup4 : HTML parser for read_html - html5lib : HTML parser for read_html - lxml : HTML parser for read_html
XML operation	- lxml : XML parser for read_xml and tree builder for to_xml
Data visualization	- matplotlib : Plotting library - seaborn : Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
SQL database	- SQLAlchemy : SQL support for databases other than sqlite - psycopg2 : PostgreSQL engine for sqlalchemy - pymysql : MySQL engine for sqlalchemy

6. Series

- | Series is the basic data structure of pandas.
- | It is similar to Numpy array, but it is different that series has indices. As shown in the figure below, the index value and data have 1:1 ratio. Thus, the structure itself is similar to dictionary that has {key : value} structure.
- | Each index value functions as the address of each data.

```
1 import pandas as pd  
2 s = pd.Series([1, 2, 3, 4, 5])  
3  
4 print(s)  
  
0    1  
1    2  
2    3  
3    4  
4    5  
dtype: int64
```

	index	data
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5

- | From the sample code provided above, separate designation of an index is not required to create series. Pandas automatically generates integer type index from 0 that increases by 1 for each data item.

6. Series



TIP

- The series indices can be flexibly designated. It doesn't necessarily start from 0. Indices other than integer type can be created as well.
- When creating series, use index parameter to designate desirable index values.

```
1 import pandas as pd  
2 s = pd.Series([1,2,3,4,5], index = ['a', 1, 'cat', 'd', 'dog'])  
3  
4 print(s)
```

```
a    1  
1    2  
cat  3  
d    4  
dog  5  
dtype: int64
```

| Visit the link <https://pandas.pydata.org/docs/reference/series.html> to check available parameters for series.

7. Creating Series

- | There are many different ways to create series, but only four of them are provided this chapter.
 - ▶ Creating series by using python list
 - ▶ Creating series by using python dictionary
 - ▶ Creating series by using Numpy
 - ▶ Creating series by using scalar value (a value that has designated range similar to integers)

7. Creating Series

7.1. Creating series from python list or dictionary

Pandas.Series (list or dictionary)

```
1 import pandas as pd
2
3 marvel = ['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron', 'Ant-Man', 'Spider-Man']
4
5 s = pd.Series(marvel)
6
7 print(s)
```

```
0      Iron Man
1    Captain America
2          Thor
3    Winter Soldier
4        Ultron
5      Ant-Man
6    Spider-Man
dtype: object
```

7. Creating Series

7.1. Creating series from python list or dictionary

Pandas.Series (list or dictionary)

- The dictionary key pairs with the index of series, while the dictionary value becomes each element (data value) of series.

```
1 import pandas as pd  
2  
3 marvel_year = {'Iron Man': 2010, 'Captain America': 2011, 'Thor': 2013, 'Winter Soldier': 2014, 'Ultron': 2015}  
4  
5 s = pd.Series(marvel_year)  
6  
7 print(s)  
8 s.index
```

```
Iron Man      2010  
Captain America 2011  
Thor          2013  
Winter Soldier 2014  
Ultron        2015  
dtype: int64
```

```
Index(['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron'], dtype='object')
```

The diagram illustrates the creation of a Pandas Series from a dictionary. On the left, a code snippet shows how to create a dictionary `marvel_year` and a corresponding Series `s`. The dictionary contains superhero names as keys and their release years as values. On the right, a table represents this data. Red arrows indicate the mapping: one arrow labeled "Series → index" points from the superhero names in the dictionary to the indices in the Series; another arrow labeled "dictionary → key" points from the superhero names in the dictionary to the keys in the Series; a third arrow labeled "data" points from the year values in the dictionary to the data values in the Series; and a fourth arrow labeled "value" points from the year values in the dictionary to the values in the Series.

Iron Man	2010
Captain America	2011
Thor	2013
Winter Soldier	2014
Ultron	2015

7. Creating Series

7.1. Creating series from python list or dictionary

Pandas.Series (list or dictionary)

```
1 import pandas as pd
2
3 marvel_year = {'Iron Man': 2010, 'Captain America': 2011, 'Thor': 2013, 'Winter Soldier': 2014, 'Ultron': 2015}
4
5 s = pd.Series(marvel_year)
6
7 print(s)
8 s.index
```

```
Iron Man      2010
Captain America  2011
Thor          2013
Winter Soldier 2014
Ultron        2015
dtype: int64
```

```
Index(['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron'], dtype='object')
```

7. Creating Series

7.2. Creating series by using array data made with Numpy

- ▶ Series objects can be initialized by using different kinds of Numpy functions. Use array-creating functions and methods to create series.

```
1 import pandas as pd
2 import numpy as np
3
4 n = np.arange(10, 16)
5
6 s = pd.Series(n)
7
8 print(s)
```

```
0    10
1    11
2    12
3    13
4    14
5    15
dtype: int64
```



Line 4

- `arange()` creates an array of integers from 10 to 15

7. Creating Series

7.2. Creating series by using array data made with Numpy

- ▶ Series objects can be initialized by using different kinds of Numpy functions. Use array-creating functions and methods to create series.

```
1 import pandas as pd
2 import numpy as np
3
4 n = np.arange(10, 16)
5
6 s = pd.Series(n)
7
8 print(s)
```

```
0    10
1    11
2    12
3    13
4    14
5    15
dtype: int64
```



- Line 6
- The array created with numpy is converted into series object

7. Creating Series

7.2. Creating series by using array data made with Numpy

```
1 import pandas as pd
2 import numpy as np
3
4 np.random.seed(0)
5
6 r = np.random.random(size = 6)
7
8 s = pd.Series(r)
9
10 print(s)
```

```
0    0.548814
1    0.715189
2    0.602763
3    0.544883
4    0.423655
5    0.645894
dtype: float64
```

Line 6

- Creates 6 normally distributed random numbers and stores them to variable r.

7. Creating Series

7.3. Creating series by using scalar value

- ▶ Scalar value refers to a value that has designated range similar to integers.
- ▶ First, create a simple series object that has single data.

```
1 import pandas as pd
2 import numpy as np
3
4 s = pd.Series(4)
5
6 print(s)
```

```
0    4
dtype: int64
```

Line 4

- Create series object consists of one index that has data of 4.

7. Creating Series

7.3. Creating series by using scalar value

```
1 s = pd.Series([0, 1, 2, 3])
2
3 s * 2
```

```
0    0
1    2
2    4
3    6
dtype: int64
```

7. Creating Series

7.3. Creating series by using scalar value

```
1 arr = np.arange(0, 5)
2
3 ss = pd.Series(arr)
4
5 print (ss)
```

```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

    Line 1, 3

- 1: Creates an array of integers from 0 to 5 in order
- 3: Uses the array data to create series object

7. Creating Series

7.3. Creating series by using scalar value

```
1 s * ss
0    0.0
1    1.0
2    4.0
3    9.0
4    NaN
dtype: float64
```



TIP

- In general, series is used to express time series data with indices of data and time.
Use the [Pandas.date_range\(\)](#) function to create range of data for an index value.

```
1 import pandas as pd
2
3 marvel = ['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron', 'Ant-Man', 'Spider-Man']
4
5 s = pd.Series(marvel)
6
7 print(s)
```

```
0      Iron Man
1    Captain America
2          Thor
3    Winter Soldier
4        Ultron
5        Ant-Man
6    Spider-Man
dtype: object
```

Line 5, 7

- 5: Converts a list into series object to prepare data for practice.
- 7: When printed, the indices will consist of integers starting from 0.



TIP

- In general, series is used to express time series data with indices of data and time. Use the [Pandas.date_range\(\)](#) function to create range of data for an index value.

```
1 dates = pd.date_range('2023-01-01', '2023-01-07')
2
3 print(dates)
```

```
DatetimeIndex(['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',
                 '2023-01-05', '2023-01-06', '2023-01-07'],
                dtype='datetime64[ns]', freq='D')
```



Line 1,3

- 1: Creates a range of date in the form of Pandas.date_range('starting date', 'end data').
- 3: When printed, the special index data DatetimeIndex will be created.



TIP

- In general, series is used to express time series data with indices of data and time. Use the `Pandas.date_range()` function to create range of data for an index value.

```
1 new_s = pd.Series (marvel, index=dates)
2
3 print(new_s)
```

```
2023-01-01      Iron Man
2023-01-02    Captain America
2023-01-03        Thor
2023-01-04   Winter Soldier
2023-01-05       Ultron
2023-01-06      Ant-Man
2023-01-07    Spider-Man
Freq: D, dtype: object
```

Line 1~3

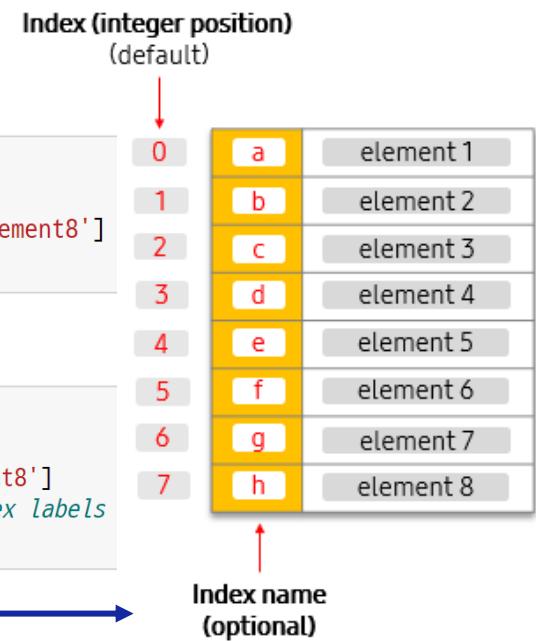
- 1, 2: Replaces the index of `marvel`, which is `series` object, to the data range that was previously created.
Of course, the data amount of newly created index and object numbers must meet to prevent an error.
- 3: Checks the `series` object with changed index.

8. Selecting a Specific Element (Data) from Series

- In series object, each index has data that makes a 1:1 pair. Consider the index as intrinsic address of each data and use it to find, arrange, and select each data.
- As explained earlier, there are two types of indices. There are integer type index that is automatically created from 0 when nothing is designated, and index label that you can specifically designate index names.

Series consists of integer type index starting from 0

```
1 import pandas as pd
2
3 s = ['element1', 'element2', 'element3', 'element4', 'elements', 'element6', 'element7', 'element8']
4 pd.Series(s) #creates series object consists of general integer type indices
```



Series consists of index label other than integers

```
1 import pandas as pd
2
3 s = ['element1', 'element2', 'element3', 'element4', 'elements', 'element6', 'element7', 'element8']
4 pd.Series(s, index = ('a','b','c','d','e','f','g','h')) # creates series object consists of index labels
# with designated names
5
```

8. Selecting a Specific Element (Data) from Series

- Series name.index: Inquires entire index values of the series data
- Series name.values: Inquires entire element values of the series data

```
1 import pandas as pd  
2  
3 marvel = ['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron', 'Ant-Man', 'Spider-Man']  
4  
5 s = pd.Series(marvel)
```

Line 4

- Converts the list into series object to prepare data for practice.

8. Selecting a Specific Element (Data) from Series

- Series name.index: Inquires entire index values of the series data
- Series name.values: Inquires entire element values of the series data

```
1 s.index
```

```
RangeIndex(start=0, stop=7, step=1)
```

 Line 1

- Use index property to inquire the index value of the series

```
1 s.values
```

```
array(['Iron Man', 'Captain America', 'Thor', 'Winter Soldier', 'Ultron',
       'Ant-Man', 'Spider-Man'], dtype=object)
```

 Line 1

- Use values property to inquire all of the data elements of the series

8. Selecting a Specific Element (Data) from Series

8.1. Selecting series element

- ▶ A method to select an element is different for integer type index or index label.
- ▶ How to select an element from integer type index

```
1 import pandas as pd
2
3 s = ['element1', 'element2', 'element3', 'element4', 'elements', 'element6', 'element7', 'element8']
4 sr = pd.Series(s)
5
6 print(sr[2])
7 print(sr[[4, 5]])
```

```
element3
4     elements
5     element6
dtype: object
```

Line 6, 7

- 6: For selecting data with a single index address
- 7: For selecting data in index number range

8. Selecting a Specific Element (Data) from Series

8.1. Selecting series element

- ▶ How to select an element from index label

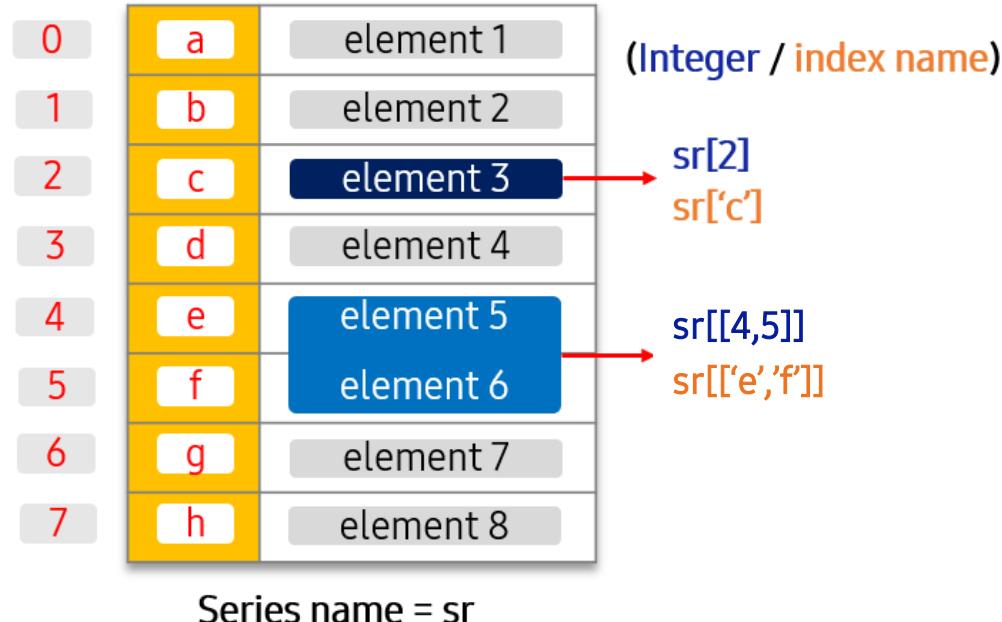
```
1 import pandas as pd  
2  
3 s = ['element1', 'element2', 'element3', 'element4', 'elements', 'element6', 'element7', 'element8']  
4 sr = pd.Series(s, index = ('a','b','c','d','e','f','g','h'))  
5  
6 print(sr['c'])  
7 print(sr[['e', 'f']])
```

```
element3  
e    elements  
f    element6  
dtype: object
```

8. Selecting a Specific Element (Data) from Series

8.1. Selecting series element

- ▶ How to select an element from index label



9. Basic Arithmetic Operation for Series Object (+, -, /, *, ...)

- Pandas perform three built-in stages of processes for arithmetic operation between data objects. Data are arranged in each row and column, and the elements on the same index (location) are paired up. If so, elements that are not paired up are classified as NaN.

9. Basic Arithmetic Operation for Series Object (+, -, /, *, ...)

9.1. Arithmetic operation for single series object

```
1 import pandas as pd  
2  
3 s = [10, 20, 30, 40, 50]  
4 sr = pd.Series(s)  
5  
6 print(sr)  
7 print(sr ** 2)
```

```
0    10  
1    20  
2    30  
3    40  
4    50  
dtype: int64  
0    100  
1    400  
2    900  
3   1600  
4   2500  
dtype: int64
```



Line 7

- All of the elements in the sr Series object are squared by 2.

9. Basic Arithmetic Operation for Series Object (+, -, /, *, ...)

9.2. Operation with index for 2 series objects

- If the elements of each index are not matched, the operation result returns NaN.

```
1 import pandas as pd
2
3 s = [10,20,30,40,50]
4 t = [23,345,123]
5
6 sr1 = pd.Series(s)
7 sr2 = pd.Series(t)
8
9 sr3 = sr1 + sr2
10
11 print(sr3)
```

```
0    33.0
1   365.0
2   153.0
3    NaN
4    NaN
dtype: float64
```



Line 8

- Check how each element value changes from the addition of s1 and s2.

10. Pandas Provides Many Accessible Descriptive Statistics

The descriptive statistics methods provided below are both used for Series objects and DataFrame objects.

<code>count()</code>	Number of non-null observations
<code>sum()</code>	Sum of values
<code>mean()</code>	Mean of Values
<code>median()</code>	Median of Values
<code>mode()</code>	Mode of values
<code>std()</code>	Standard Deviation of the Values
<code>min()</code>	Minimum Value
<code>max()</code>	Maximum Value
<code>abs()</code>	Absolute Value
<code>prod()</code>	Product of Values
<code>cumsum()</code>	Cumulative Sum
<code>cumprod()</code>	Cumulative Product

10. Pandas Provides Many Accessible Descriptive Statistics

```
1 import pandas as pd  
2  
3 s = [10, 20, 30, 40, 50]  
4 t = [23, 345, 123]  
5 sr1 = pd.Series(s)  
6 sr2 = pd.Series(t)  
7  
8 sr3 = sr1 + sr2  
9  
10 print(sr3)  
11  
12 print(sr3.count())  
13 print(sr3.sum())
```

```
0    33.0  
1    365.0  
2    153.0  
3    NaN  
4    NaN  
dtype: float64  
3  
551.0
```

Line 10

- When printing, the elements of indices 3 and 4 are NaN data.

10. Pandas Provides Many Accessible Descriptive Statistics

```
1 import pandas as pd  
2  
3 s = [10, 20, 30, 40, 50]  
4 t = [23, 345, 123]  
5 sr1 = pd.Series(s)  
6 sr2 = pd.Series(t)  
7  
8 sr3 = sr1 + sr2  
9  
10 print(sr3)  
11  
12 print(sr3.count())  
13 print(sr3.sum())
```

```
0    33.0  
1    365.0  
2    153.0  
3    NaN  
4    NaN  
dtype: float64  
3  
551.0
```

Line 12

- Returns the number of elements except for missing data.

10. Pandas Provides Many Accessible Descriptive Statistics

```
1 import pandas as pd  
2  
3 s = [10, 20, 30, 40, 50]  
4 t = [23, 345, 123]  
5 sr1 = pd.Series(s)  
6 sr2 = pd.Series(t)  
7  
8 sr3 = sr1 + sr2  
9  
10 print(sr3)  
11  
12 print(sr3.count())  
13 print(sr3.sum())
```

```
0    33.0  
1    365.0  
2    153.0  
3    NaN  
4    NaN  
dtype: float64  
3  
551.0
```

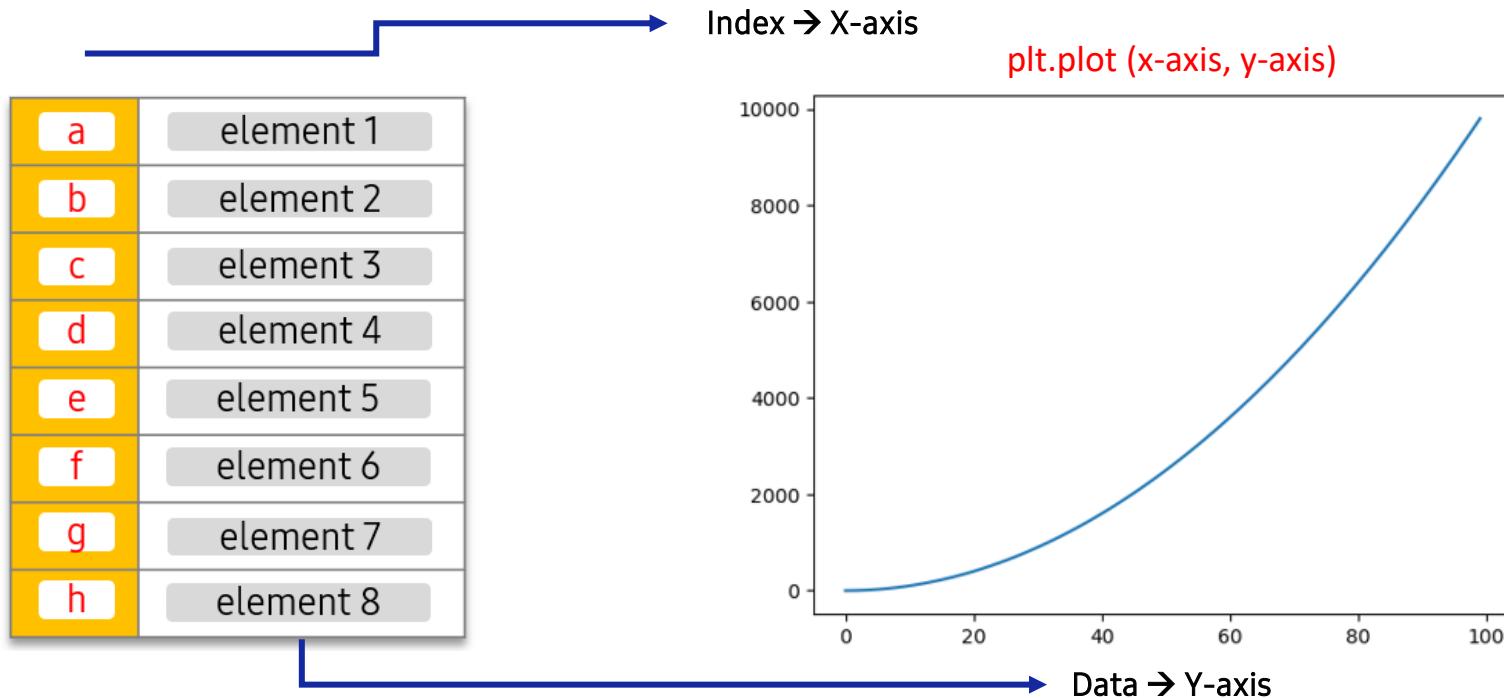
Line 13

- Returns sum of the elements except for missing data.

11. Basic of Data Visualization Tool Matplotlib

11.1. Drawing a line plot

- | Assigning the plt.plot (x-axis and y-axis data) passes series index to x-axis and data element of each index to y-axis.
- | Or, drawing a graph is also possible with the plt.plot (Series object or DataFrame object).



11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot

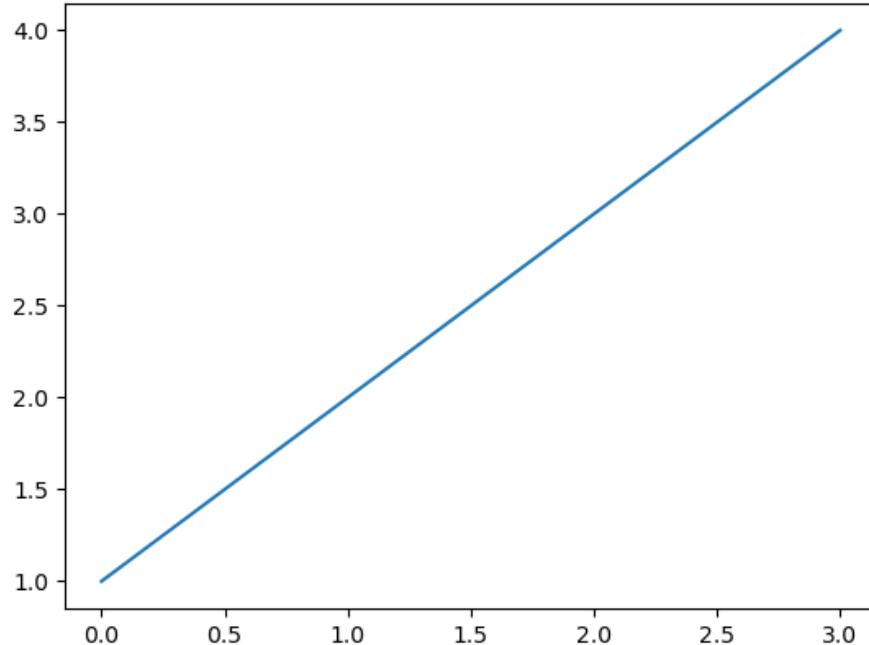
```
1 import matplotlib.pyplot as plt  
2 plt.plot([1, 2, 3, 4])  
3  
4 plt.show()
```

Line 2

- If not assigning separate x values, this value is assigned to the sequence of y-axis.

11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot



- | From the chart above, x-axis is 0-3 and y-axis is 1-4. This is because when providing a single list or array for floating, matplotlib assumes that it is the sequence of y value and automatically generates x value.
- | Python range starts from 0, so the basic x vector has the same length with y, but it starts from 0. Thus, the x data become [0, 1, 2, 3].

11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot

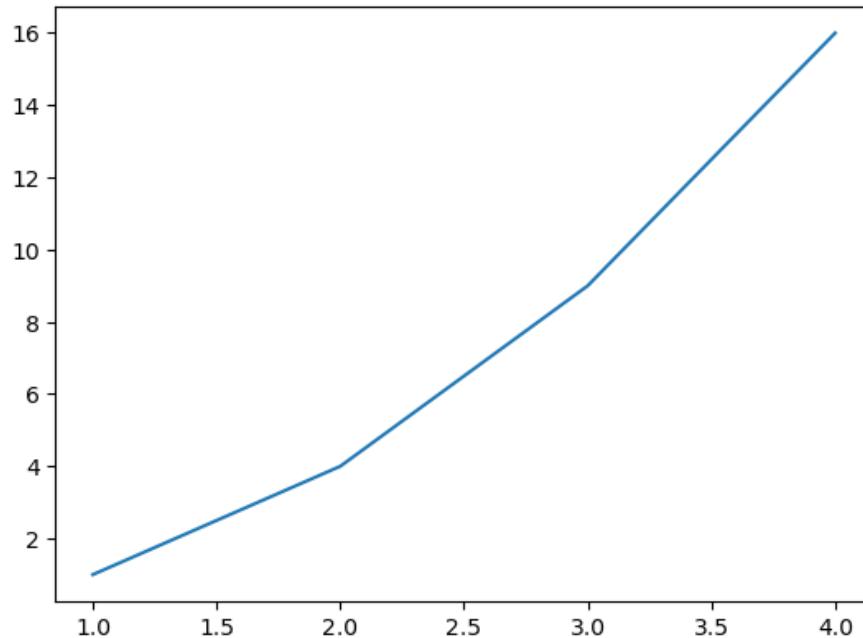
```
1 import matplotlib.pyplot as plt  
2 plt.plot([1, 2, 3, 4], [1, 4, 9, 16])  
3  
4 plt.show()
```

Line 2

- Assigns both x-axis and y-axis

11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot



11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot

```
1 import matplotlib.pyplot as plt
2
3 x = range(100)
4 y = [value **2 for value in x]
5
6 plt.plot(x,y, linewidth=5.0, color='red')
7
8 plt.title('hello world of chart')
9 plt.ylabel('y-some numbers')
10 plt.xlabel('x-some numbers')
11
12
13 plt.show()
```

Line 1, 4, 6, 8

- 1: matplotlib is an external module that needs to be installed.
- 4: uses list comprehension.
- 6: linewidth is a property that adjusts the thickness of a line graph, while color designates specific colors.
- 8: adds the title of the chart.

11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot

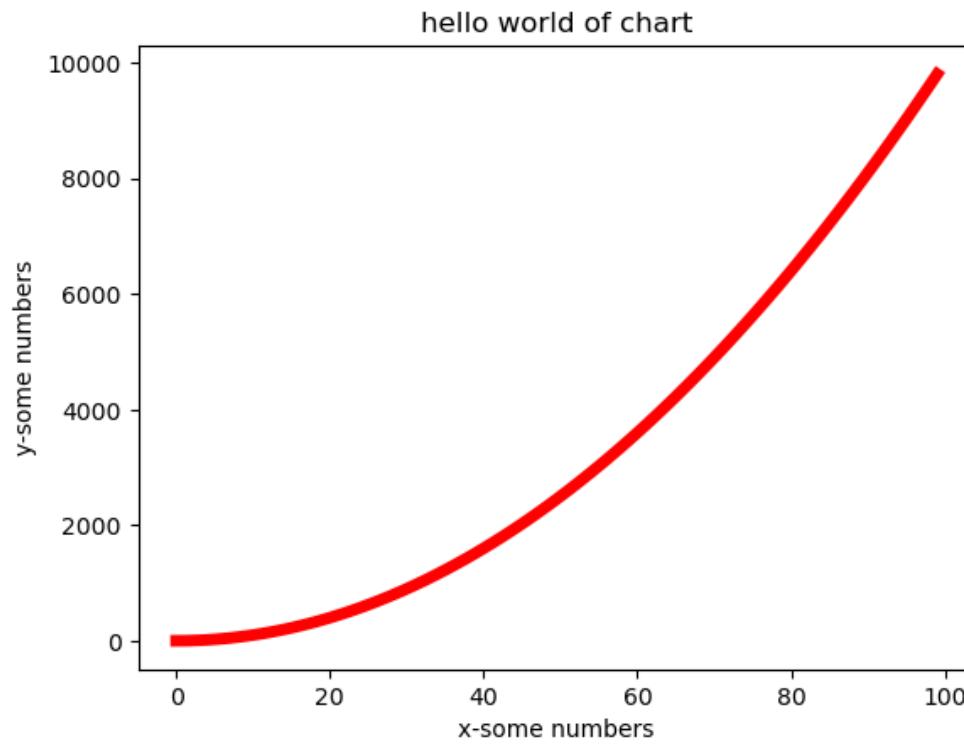
```
1 import matplotlib.pyplot as plt
2
3 x = range(100)
4 y = [value **2 for value in x]
5
6 plt.plot(x,y, linewidth=5.0, color='red')
7
8 plt.title('hello world of chart')
9 plt.ylabel('y-some numbers')
10 plt.xlabel('x-some numbers')
11
12
13 plt.show()
```

Line 9, 10, 13

- 9: adds the name of x-axis of the chart.
- 10: adds the name of y-axis of the chart.
- 13: command function to print a graph.

11. Basic of Data Visualization Tool Matploylib

11.1. Drawing a line plot



- ▶ It is result of drawing $y = x^{**2}$ curve where x is between $[0,99]$.

11. Basic of Data Visualization Tool Matploylib

11.2. Drawing a bar plot

- A bar plot represents the height proportional to data value size as a rectangular bar. It is possible to compare the data size through the relative difference between bar height, and there are two different bar types including vertical and horizontal bar.

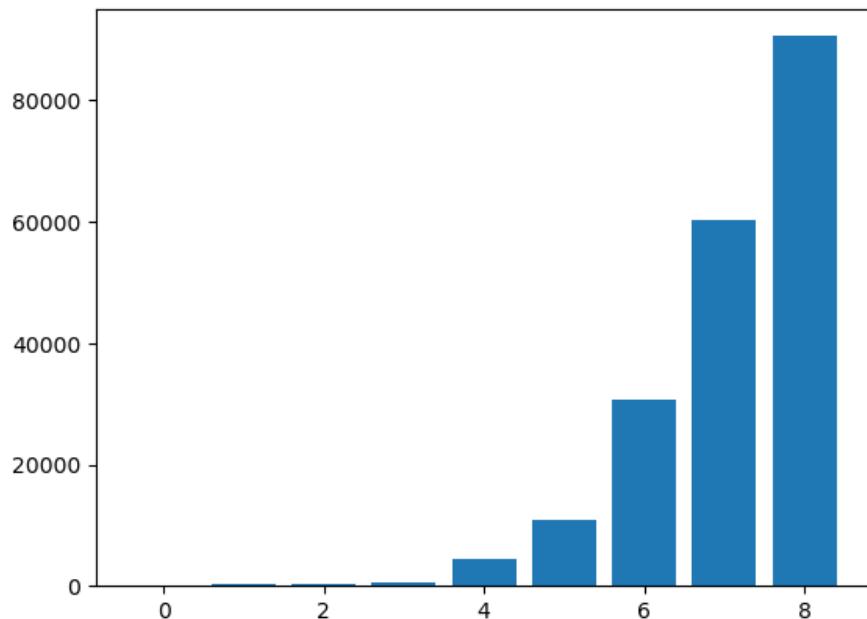
Vertical bar plt.bar()	Horizontal bar plt.bart()
<ul style="list-style-type: none">- Suitable to show differences between data values of two points that have time difference.- In other words, it is suitable for time series data expression.	<ul style="list-style-type: none">- Suitable so show differences between variable value sizes.

11. Basic of Data Visualization Tool Matploylib

11.2. Drawing a bar plot

- If the elements of each index are not matched, the operation result returns NaN.

```
1 from matplotlib import pyplot as plt
2 days_in_year = [88, 225, 365, 687, 4333, 10756, 30687, 60190, 90553]
3 plt.bar(range(len(days_in_year)), days_in_year)
4 plt.show()
```



| Let's code

Step 1

1) Consider which libraries would be required to solve the mission. Find necessary libraries and install ones that you don't have. Import the module to the code.

- ▶ Numpy for making an array while data processing
- ▶ Pandas for creating series objects
- ▶ Matplotlib for data visualization

```
1 import numpy as np  
2 import pandas as pd  
3 import matplotlib.pyplot as plt
```

Step 1

2) Convert each of three dictionaries into a series object.

- ▶ Make sure to check the index label of each series object for operation because if index label or index number (number of data) is different, it may return NaN.

```
1 s_2020 = pd.Series(population_2020)
2 s_2021 = pd.Series(population_2021)
3 s_area = pd.Series(city_area)
```

Line 1~3

- 1: population_2020 is dictionary data object that stores total population of each city in 2020.
- 2: population_2021 is dictionary data object that stores total population of each city in 2021.
- 3: city_area is dictionary data object that stores area of each city.

Step 2

- | Use two series arithmetic operations to calculate population fluctuation of each year.

```
1 growth = s_2020 - s_2021  
2 print(growth)
```

```
Tokyo          -53324  
Delhi         890440  
Shanghai       737222  
Sao Paulo     194444  
Mexico City    136558  
Dhaka          735230  
Cairo          422146  
Karachi        365686  
Istanbul       224861  
Buenos Aires   103944  
Kinshasa       628021  
Lagos          493779  
Manila          235121  
Rio de Janeiro  86387  
Moscow          55298  
Bogota          189032  
Paris           61316  
Jakarta         144877  
Lima            163569  
dtype: int64
```

Step 2

- Use two series arithmetic operations to calculate population fluctuation of each year.

```
1 growth = s_2020 - s_2021  
2 print(growth)
```

Line 1~2

- 1: Creates a new series object by calculating population fluctuation of each city (index label) through two series arithmetic operations (-).
- 2: Checks data of new series object that stores arithmetic operation results.

Step 3

- | As previously explained, when visualizing data through a bar graph, vertical graph is suitable for time series data that have consecutive values, but horizontal bar graph is suitable here because it would show data difference between each variable (city name).
- | The x-axis of the graph is index label of time series object, and the y-axis consists of data elements.
- | To draw a chart of time series object of 'growth,' save index information to x-axis and data values to y-axis.

```
1 x=growth.index  
2 y=growth.values
```

Line 1~2

- 1: Extracts index value only.
- 2: Extracts data elements only.

Step 3

- | When creating a bar graph, overlapping sometime occurs if there are many label names on x-axis. To solve this problem, adjust the graph size to clearly see label names on x-axis.
- | Use the plt.figure() function, figsize =(horizontal, vertical size), and parameters in inch to adjust the chart size. If not designating the size, the default chart size becomes 6.4 and 4.8 inches.
- | Use the plt.xticks() function to adjust the angle of label name printing direction.
 - rotation = enters an angle that rotates the number counterclockwise.
 - size = font size

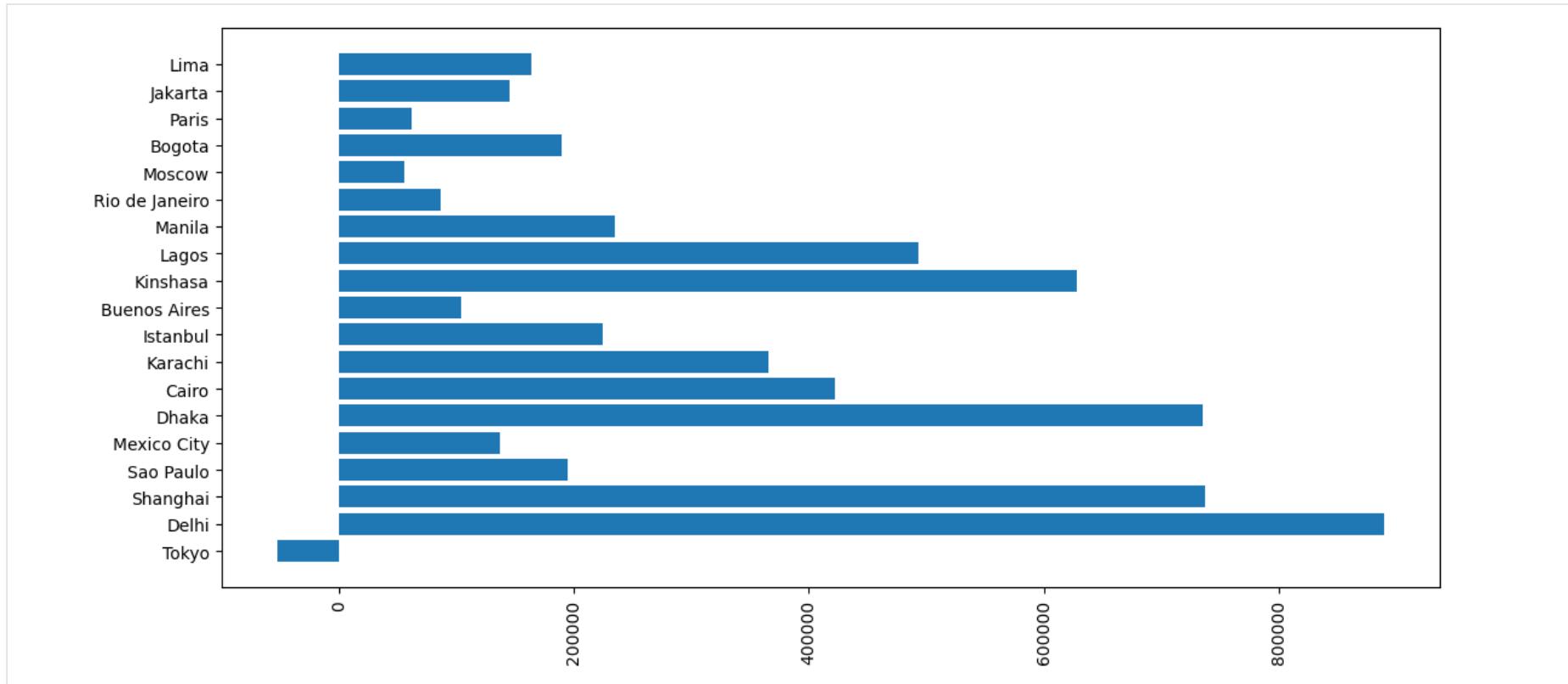
Step 3

```
1 plt.figure(figsize = (13, 6))
2
3 plt.xticks(rotation='vertical', size = 10)
4
5 plt.barh(x, y)
6
7 plt.show()
```

Line 1, 3, 4, 5

- 1: Figure dimension (width, height) in inches.
- 3: It is possible to enter a number that signifies an angle instead of vertical. If rotation=90, it means that it rotates 90 degrees counterclockwise.
- 4: size=10 refers to font size.
- 5: Use plt.barh() to plot a horizontal bar plot.

Step 3



- When calculating data of cities with the largest population, it shows that population growth rate is decreased in Tokyo.

Step 4

- | Let's calculate population density.
- | The series objects for calculation include `s_2020` that has population data in 2020 and `s_area` that has area data of each city.

```
1 population_density = s_2020 / s_area  
2 population_density
```

```
Tokyo          17019.053783  
Delhi         21011.708895  
Shanghai       4384.180126  
Sao Paulo     14620.297173  
Mexico City    14760.226263  
Dhaka          70956.560052  
Cairo          6911.750405  
Karachi        4354.357672  
Istanbul       2885.120157  
Buenos Aires   75160.950739  
Kinshasa       1502.304064  
Lagos          12691.811272  
Manila          330190.601679  
Rio de Janeiro 10792.400000  
Moscow          5015.233771  
Bogota          6291.488451  
Paris           105109.544592  
Jakarta         16500.928193  
Lima            4072.888099  
dtype: float64
```

Step 4

```
1 x=population_density.index  
2 y=population_density.values
```

 Line 1~2

- 1: Extracts index values only.
- 2: Extracts data elements only.

Step 4

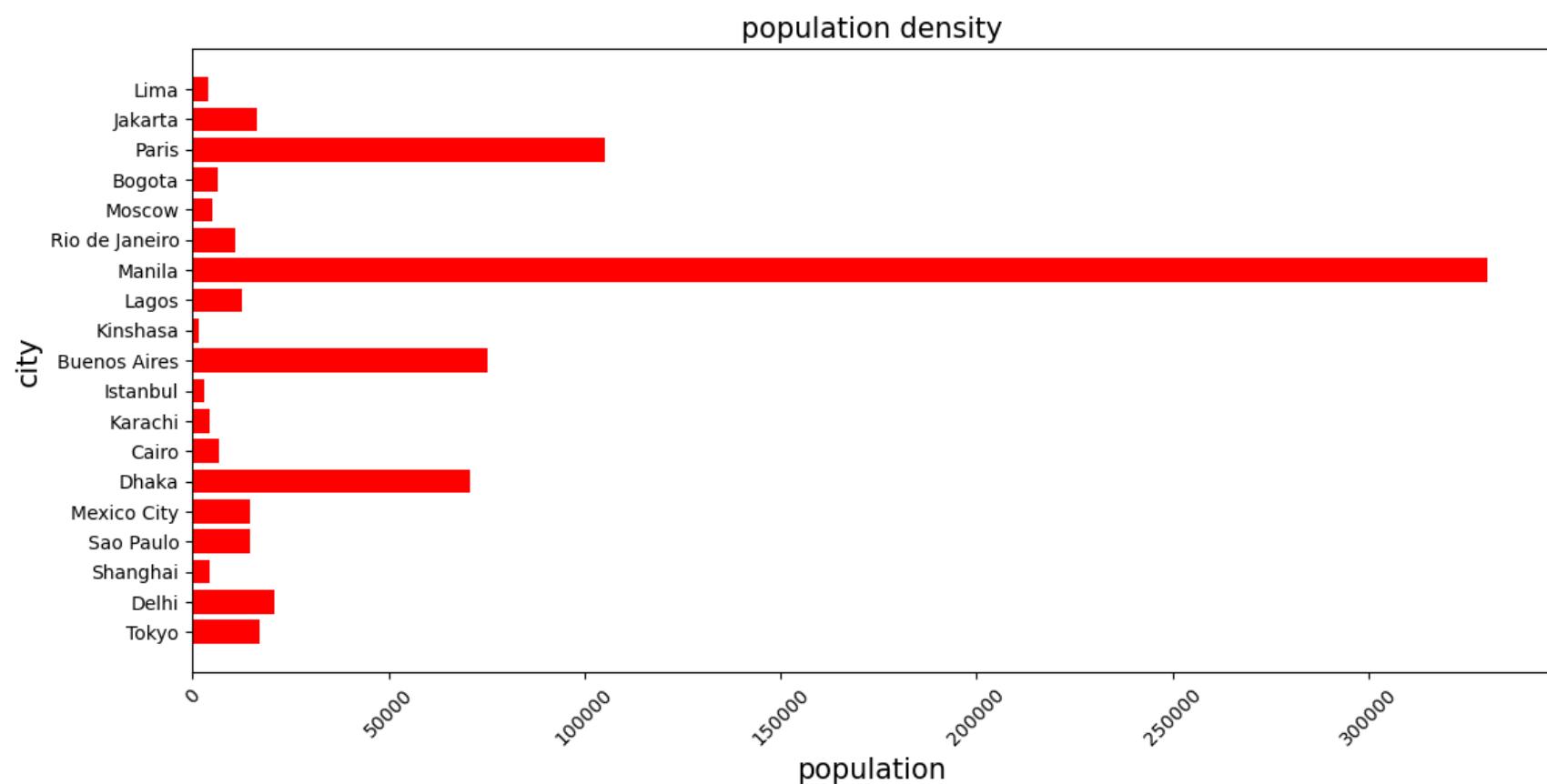
```
1 plt.figure(figsize = (13,6))
2
3 plt.xticks(rotation=45, size = 10)
4
5 plt.barh(x, y, color='red')
6
7 plt.title('population density', size = 15)
8 plt.ylabel('city', size = 15)
9 plt.xlabel('population', size = 15)
10
11 plt.show()
```



Line 1, 5, 7, 8, 9

- 1: Figure dimension (width, height) in inches.
- 5: Use color parameter to designate color of the graph.
- 7: Adds the title of the chart.
- 8: Adds the name of x-axis of the chart.
- 9: Adds the name of y-axis of the chart.

Step 4



Step 5

- | Calculate the cities with the highest and lowest population density by using the descriptive statistics functions of the pandas series object.

<code>count()</code>	Number of non-null observations
<code>sum()</code>	Sum of values
<code>mean()</code>	Mean of Values
<code>median()</code>	Median of Values
<code>mode()</code>	Mode of values
<code>std()</code>	Standard Deviation of the Values
<code>min()</code>	Minimum Value
<code>max()</code>	Maximum Value
<code>abs()</code>	Absolute Value
<code>prod()</code>	Product of Values
<code>cumsum()</code>	Cumulative Sum
<code>cumprod()</code>	Cumulative Product

Step 5

- | Calculate the cities with the highest and lowest population density by using the descriptive statistics functions of the pandas series object.

```
1 population_density.mean()
```

38117.44238880675



Line 1

- Finds the minimum value among the entire data element values of the series object.

```
1 population_density.max()
```

330190.60167910444



Line 1

- Finds the maximum value among the entire data element values of the series object.

| 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. The Spotify dataset has various column values other than the artist name, ranking, and popularity that we used in the mission. Discuss with your classmate to create a special playlist made with other column values.

Unit 36.

Pandas DataFrame for Data Processing

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.

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.”

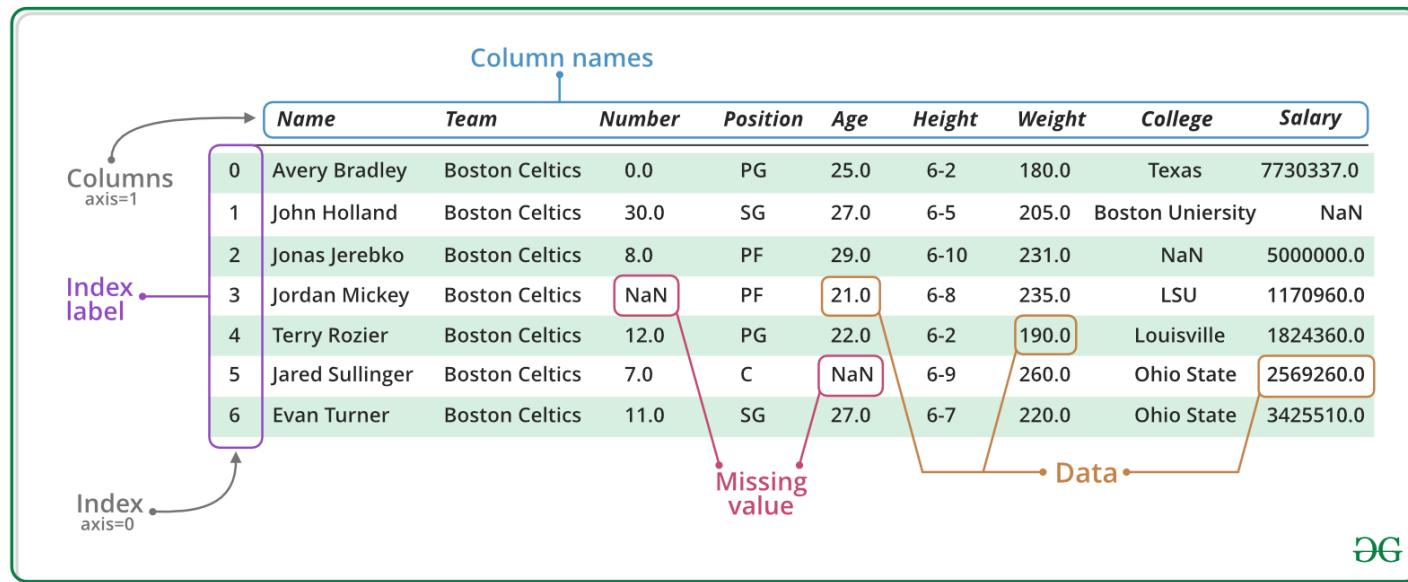
- | Search and download the 2019 Most Streamed Tracks data set from the global streaming service Spotify from Kaggle.
Find out which artist has the most popular songs based on this data set.
- | Also, check whether the tempo of music is correlated with popularity through scatter plot data visualization.
- | 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.



<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.

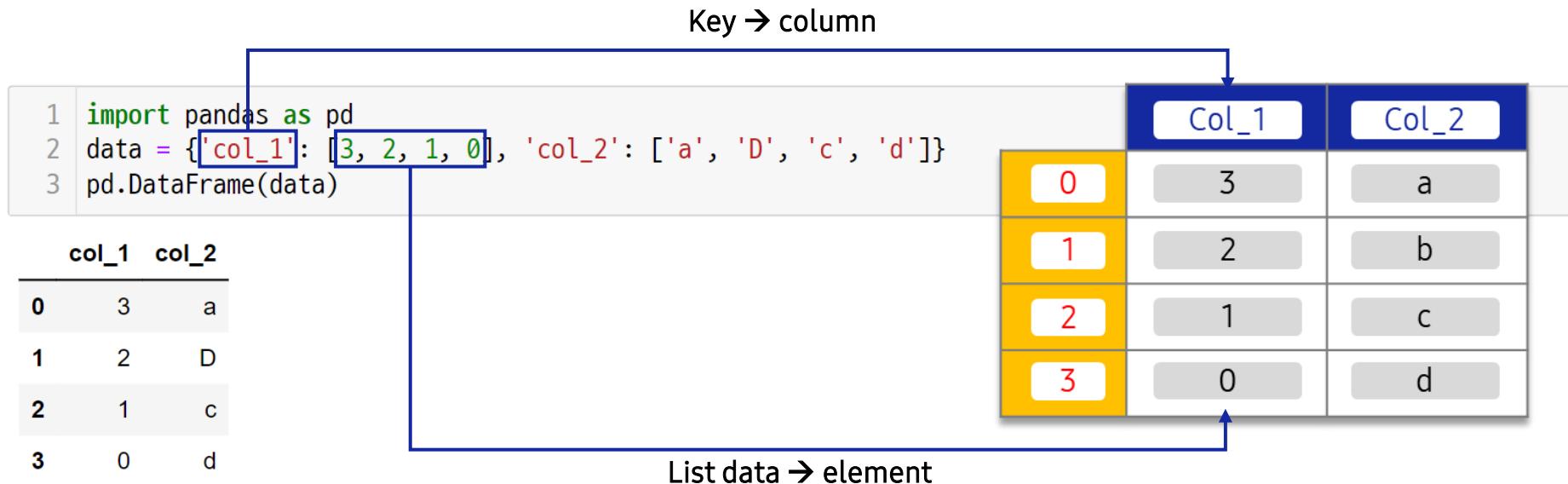
2. Creating a DataFrame

- | 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', 'D', 'c', 'd']}
```

	col_1	col_2
0	3	a
1	2	D
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)
```

	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)
```

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

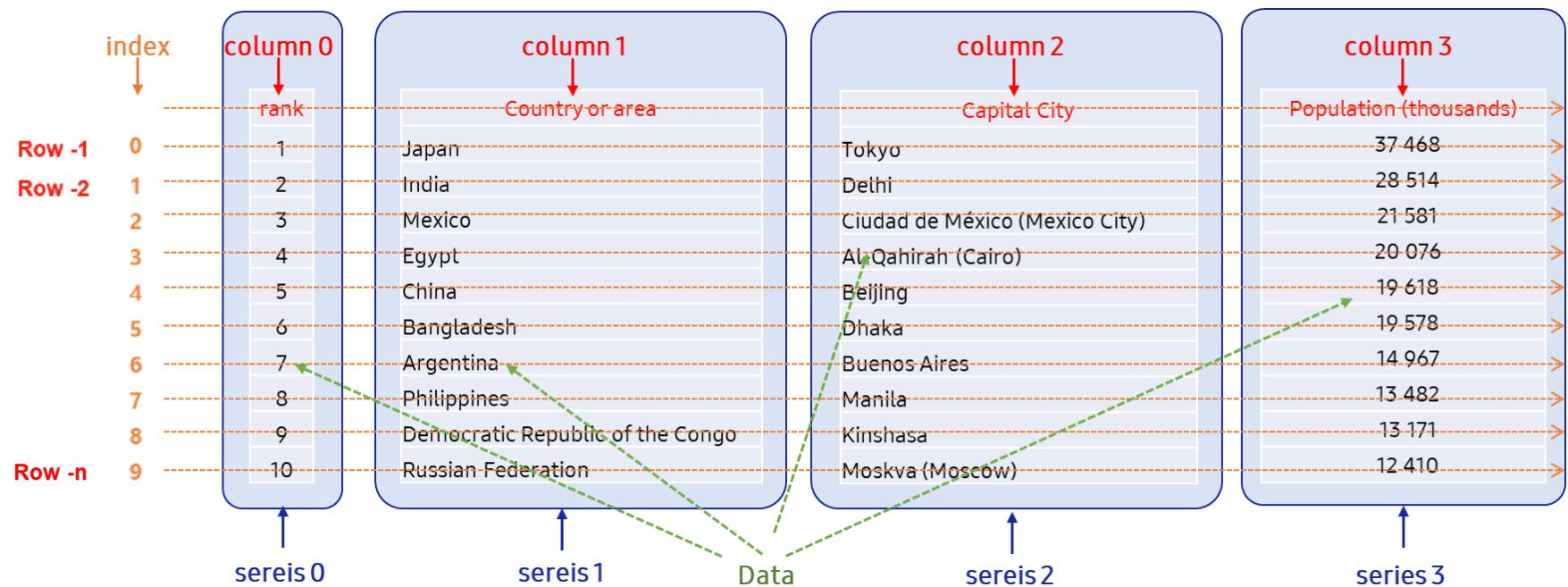
Line 5, 7, 8

- 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"})
```

```
coll  col2
0      1    3
1      2    4
```

	coll	col2
new index0	1	3
	1	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
1	2



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_dict()`.
 - 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

```
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 dic_data = {"one":s, "two":s2}
6
7 type(dic_data)
8
9 pd.DataFrame(dic_data)
10
11 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

- I 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.
- I 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
	The "picture.jpg" file is located in the same folder as the current page
	The "picture.jpg" file is located in the images folder in the current page
	The "picture.jpg" file is located in the images folder at the root of the current web
	The "picture.jpg" file is located in the folder one level up from the current folder

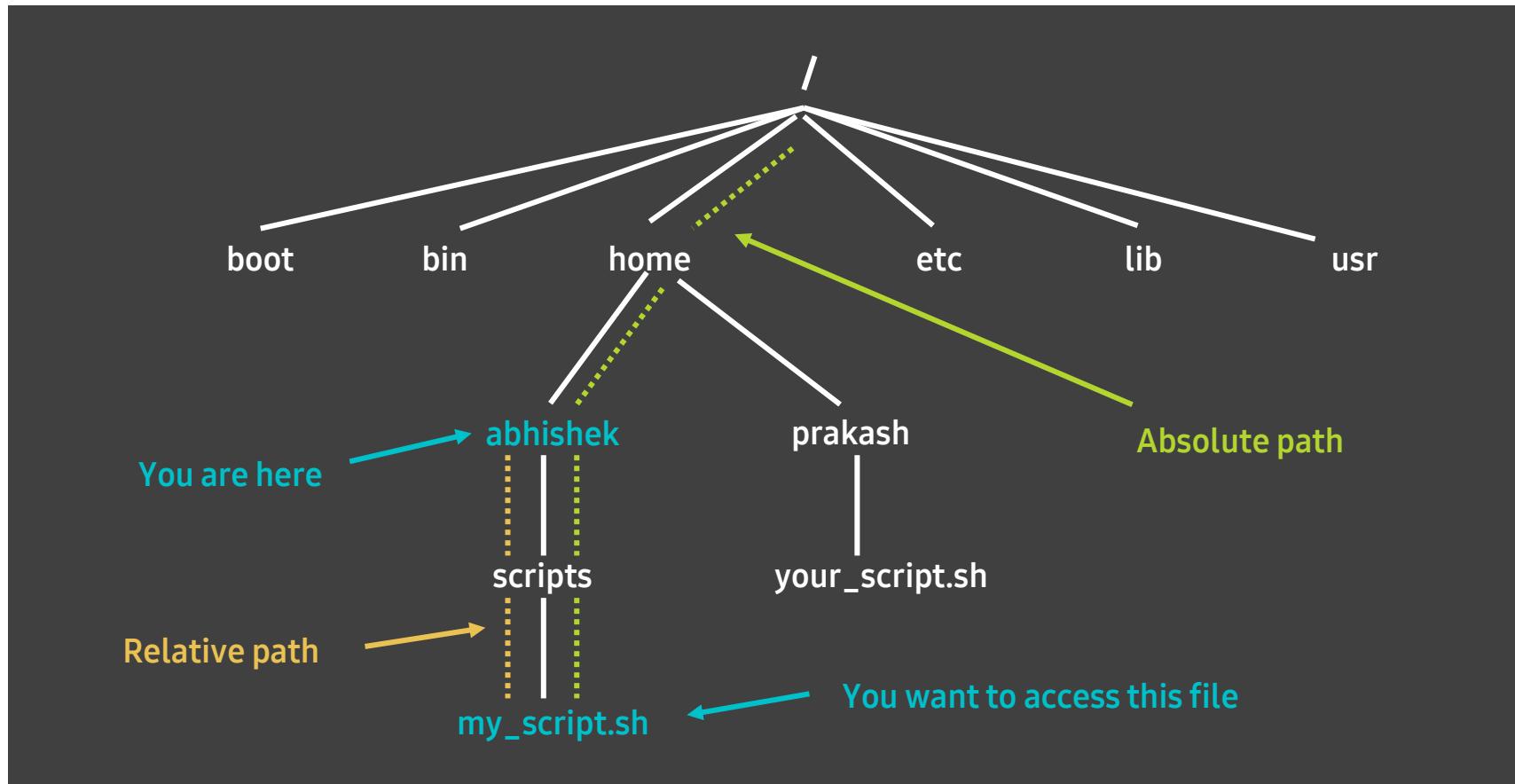
I Why do we need relative paths?

- 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 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

- I 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... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Allen, Mr. William Henry	male	35.0	1	0	113803	53.1000	C123	S
4	5	0	3				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

```
<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
```

RangeIndex: You can see the total number of rows (890) and number of columns (12) 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 percentile value
50%	50 percentile value
75%	75 percentile 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



- 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)	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
header = 2	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

header = 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'])  
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, 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)
```

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Name										
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)
```

- ▶ 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



- 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

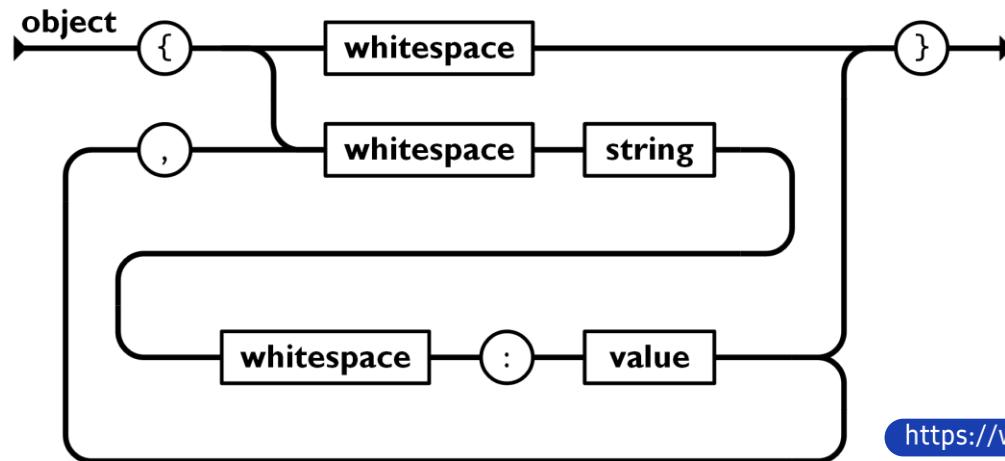
6.1. JSON

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 : "Datename": 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}]
```

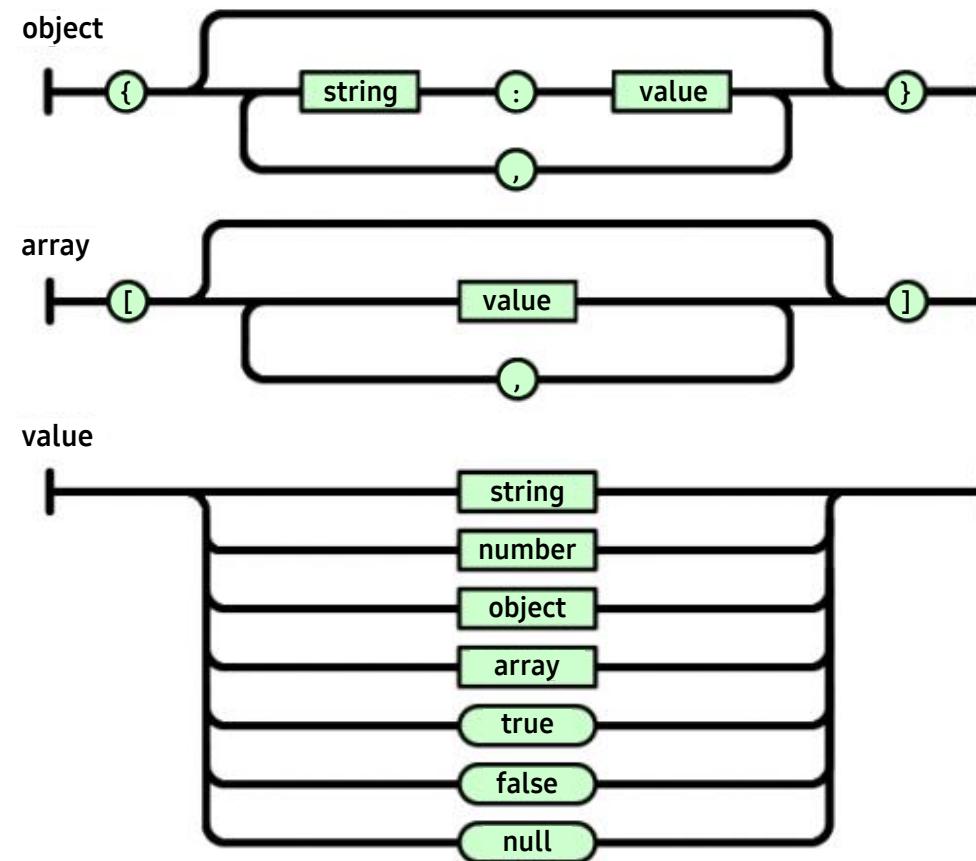


Line 1

- 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.
- Download the data from <https://www.kaggle.com/datasets/matheusgratz/world-university-rankings-2021>

6. Converting JSON File to DataFrame

6.4. Reading a file and creating a DataFrame object

I 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

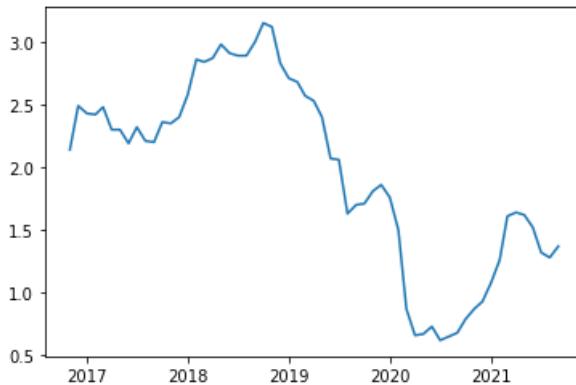
- ▶ You should remember that what we are learning here is loading the json object into a DataFrame using pandas. It is different from encoding to load json into Python and convert it to a Python string. Hope you don't get confused.

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()
```

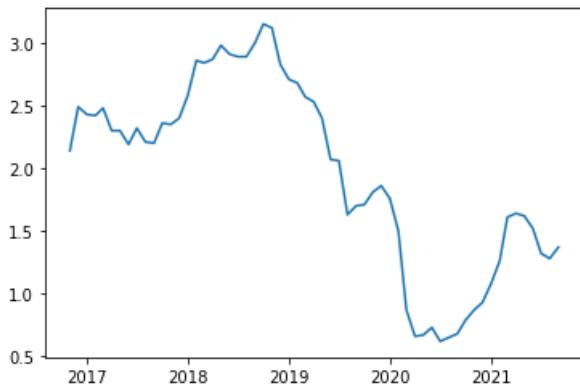


Line 2, 4

- 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.

- I World Bank supports thousands of datasets through DataReader. You can see the World Bank's data catalog at <https://datacatalog.worldbank.org/>
- I 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.

The screenshot shows the World Bank Data Catalog interface. On the left, there is a sidebar titled "REFINED BY" with options like COUNTRY, LICENSE, DATA TYPE, RESOURCE TYPE, LANGUAGES SUPPORTED, PERIODICITY, RATING, and COLLECTIONS, each with a plus sign to expand. There is also a "RESET" button. The main search area has a "Search Criteria" section with radio buttons for "All Words" (selected) and "Any Word". A search input field contains "Poverty Headcount (\$1.90 a day)". To the right of the input is a magnifying glass icon. Below the search bar are navigation tabs: "All" (selected), "Datasets", "Indicators", and "Visualizations". Underneath these tabs are sorting options: "Sort By: Most Relevant" (selected), "Alphabetical", and "Last Updated". To the right, it says "Showing 1 - 10 of 52 results". The first result listed is "Poverty Headcount Ratio At \$1.90 A Day (2011 PPP) (% Of Population)". This title is highlighted with a yellow box. Below the title is a brief description: "Poverty headcount ratio at \$1.90 a day is the percentage of the population living on less than \$1.90 a day at 2011 international prices. As a result of revisions in PPP exchange rates, poverty rates for individual countries...". There is a "See More" link. At the bottom of the result card, there is a detailed description of the code, data type, periodicity, dataset, source, and methodology, along with download links for "Query Tool, API" and "Download". At the very bottom of the catalog page, there are links for "Preview", "Data Availability", and "Also Found In".

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
12208		SE.SCH.LIFE		School life expectancy, primary to tertiary, b...
12209		SE.SCH.LIFE.FE		School life expectancy, primary to tertiary, f...
12210		SE.SCH.LIFE.MA		School life expectancy, primary to tertiary, m...
13802	SP.DYN.LE00.FE.IN			Life expectancy at birth, female (years)
13803	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
12208		SE.SCH.LIFE		School life expectancy, primary to tertiary, b...
12209		SE.SCH.LIFE.FE		School life expectancy, primary to tertiary, f...
12210		SE.SCH.LIFE.MA		School life expectancy, primary to tertiary, m...
13802	SP.DYN.LE00.FE.IN			Life expectancy at birth, female (years)
13803	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	life_expectancy
Canada	1999	78.839024
	1998	78.623659
	1997	78.412439
	1996	78.220244
	1995	77.977317
	1994	77.871707
	1993	77.704878
	1992	77.838049
	1991	77.593415
	1990	77.436585
	1989	77.124878
	1988	76.858780
	1987	76.778780
	1986	76.463902

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')
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.

year	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990
country											
Afghanistan	39.618000	40.164000	37.766000	38.187000	33.329000	33.550000	39.396000	39.844000	43.958000	45.158000	45.967000
Africa Eastern and Southern	49.636538	50.057073	50.296849	48.703331	48.652665	49.011631	49.639719	50.075888	49.359727	50.684100	50.607728
Africa Western and Central	47.015239	47.297190	47.529378	47.785262	47.931920	48.021676	48.066764	48.237852	48.512913	48.689847	48.650003
Albania	70.478000	70.730000	71.023000	71.296000	71.502000	71.656000	71.950000	72.352000	72.641000	72.880000	73.144000
Algeria	53.261000	55.276000	57.428000	59.510000	61.658000	63.588000	65.159000	65.830000	66.971000	67.085000	67.416000
...
West Bank and Gaza	NaN	67.934000									
World	62.233556	62.611283	62.972214	63.215641	63.520124	63.812289	64.208232	64.512757	64.684863	65.022815	65.188407
Yemen, Rep.	50.654000	51.709000	52.405000	53.923000	54.934000	55.693000	53.819000	57.096000	57.759000	58.218000	58.699000
											59.04



One More Step

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

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

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



Line 3

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



One More Step

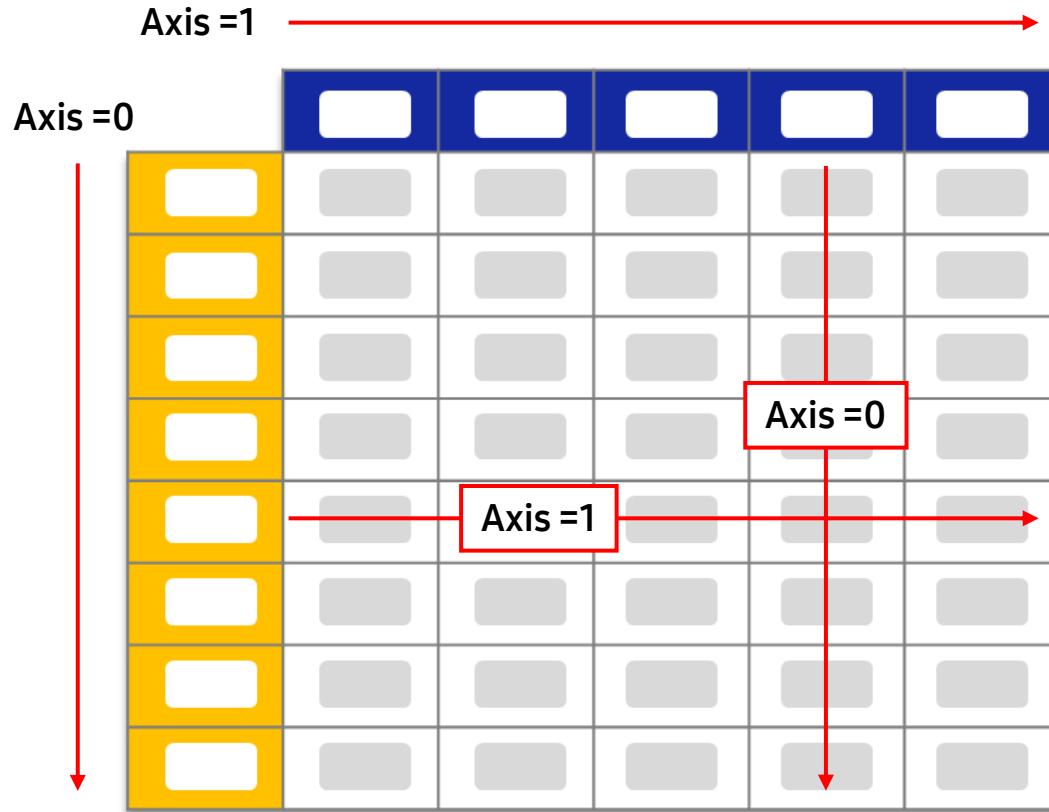
```
DataFrame.idxmin(axis=0, skipna=True)
```

```
year
SP.DYN.LE00.IN 1980    Timor-Leste
                1981    Timor-Leste
                1982    Timor-Leste
                1983    Timor-Leste
                1984  South Sudan
                1985  South Sudan
                1986  South Sudan
                1987  South Sudan
                1988  South Sudan
                1989  South Sudan
                1990  South Sudan
                1991      Somalia
                1992  South Sudan
                1993  South Sudan
                1994      Rwanda
                1995  South Sudan
                1996  South Sudan
                1997  South Sudan
                1998  South Sudan
                1999  Sierra Leone
dtype: object
```

- | 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=1, 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 10.

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)
```

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

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.iloc[1:10]



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
```

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



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
```

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



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

```
1 import pandas as pa
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 b = index_position[["title"]]
9
10 print (a)
11 print('\n')
12 print (b)
13
14 c = index_position[["title", "location", "students staff ratio"]]
15 print('\n')
16 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      Princeton University
9      The University of Chicago
```

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

8.3. Select column

```
1 import pandas as pa
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 b = index_position[["title"]]
9
```



Line 7, 9

- 7: Select only one column named title and save it as a new Series.
- 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 print (a)
11 print('\n')
12 print (b)
13
14 c = index_position[["title", "location", "students staff ratio"]]
15 print('\n')
16 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 and 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.

Unit 36. Pandas DataFrame for Data Processing

| 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

I 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	U
0	global	1.0	25sgk305KZfyuqVBQIahim	1166185736	Sweet but Psycho	Ava Max https://open.spotify.com/track/25sgk305KZfyuqVBQIahim
1	global	2.0	2Fxmhks0bxGSBdJ92vM42m	1052358787	bad guy	Billie Eilish https://open.spotify.com/track/2Fxmhks0bxGSBdJ92vM42m
2	global	3.0	6ocbgoVGwYJhOv1Ggl9NsF	789094044	7 rings	Ariana Grande https://open.spotify.com/track/6ocbgoVGwYJhOv1Ggl9NsF
3	global	4.0	1rgnBhdG2JDFTbYkYRZAku	764208309	Dance Monkey	Tones and I https://open.spotify.com/track/1rgnBhdG2JDFTbYkYRZAku
4	global	5.0	6v3KW9xbzN5yKLt9YKDYA2	763064359	Señorita	Shawn Mendes https://open.spotify.com/track/6v3KW9xbzN5yKLt9YKDYA2
5	global	6.0	5w9c2J52mkdntKOmRLeM2m	715027898	Con Calma	Daddy Yankee https://open.spotify.com/track/5w9c2J52mkdntKOmRLeM2m
6	global	7.0	2VxeLyX666F8uXCJ0dZF8B	714097238	Shallow	Lady Gaga https://open.spotify.com/track/2VxeLyX666F8uXCJ0dZF8B



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 
 ..   ...             ...           ...
```



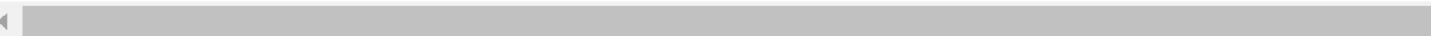
Line 1

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



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
Artist	
Post Malone	45
Juice WRLD	31
Ariana Grande	28
Billie Eilish	27
Taylor Swift	27
	..
Nicki Minaj	1
J. Cole	1
Rihanna	1
Bass Santana	1
Bruno Mars	1
Name: count, 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

	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) - Pt. 1	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) - Pt. 2	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.

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

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

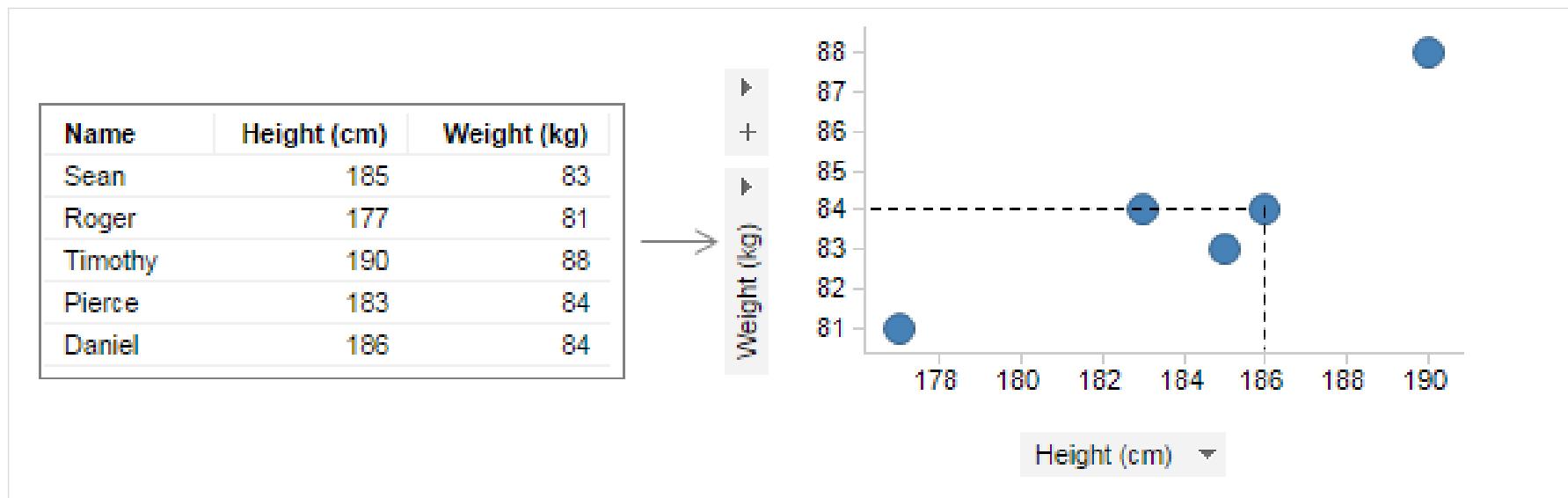
- Initialize the index with reset_index.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reset_index.html

Step 6

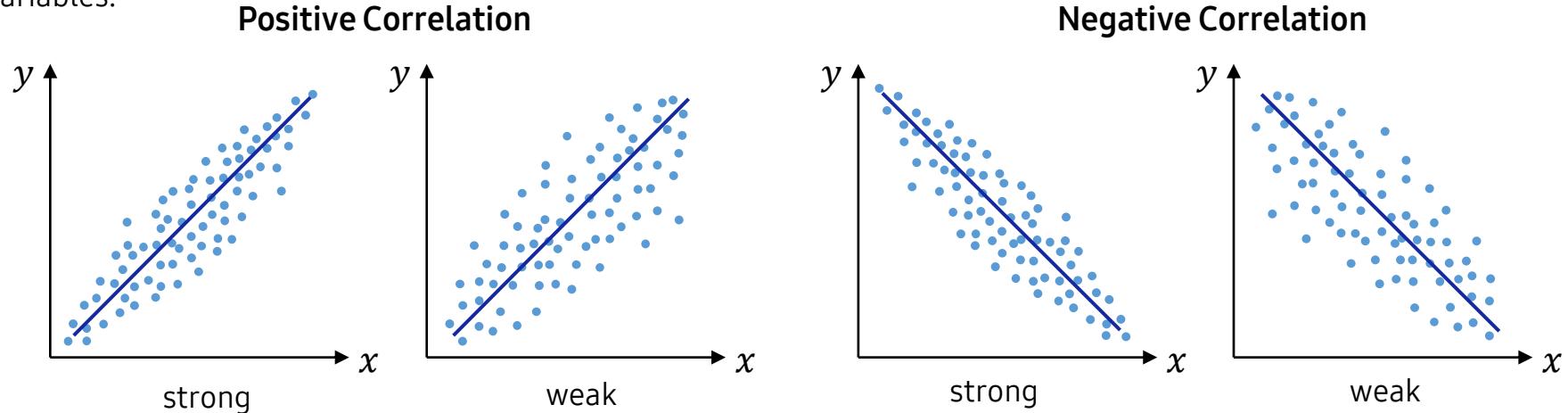
I Expressing the tempo by rank as a scatter plot graph

- ▶ 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.



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.

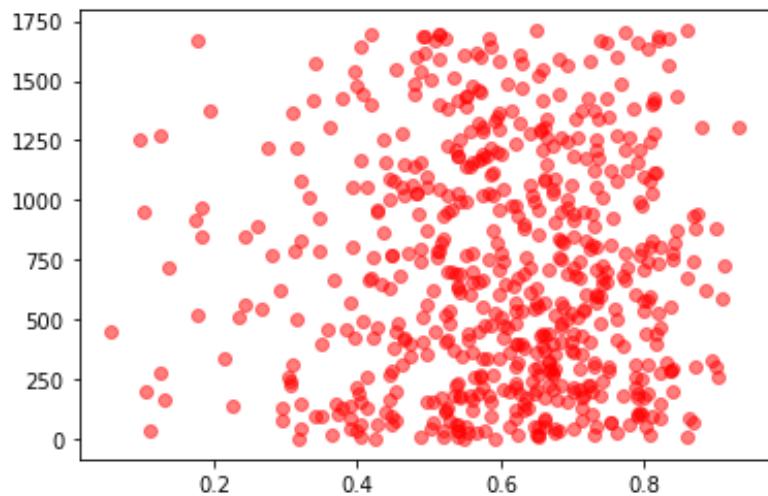


Step 6

- | 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', numeric_only=True)  
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

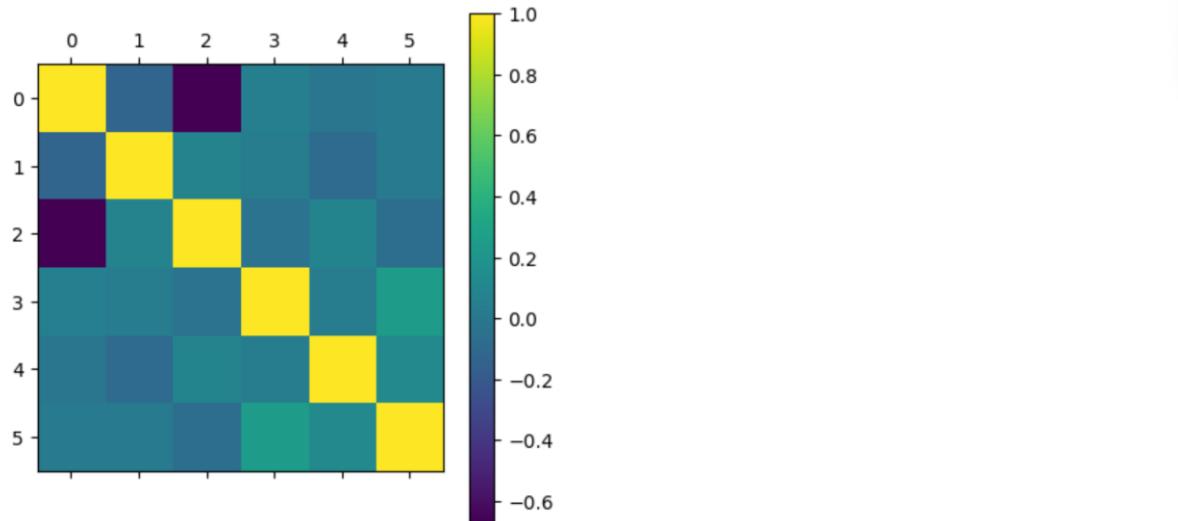


Line 2

- 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

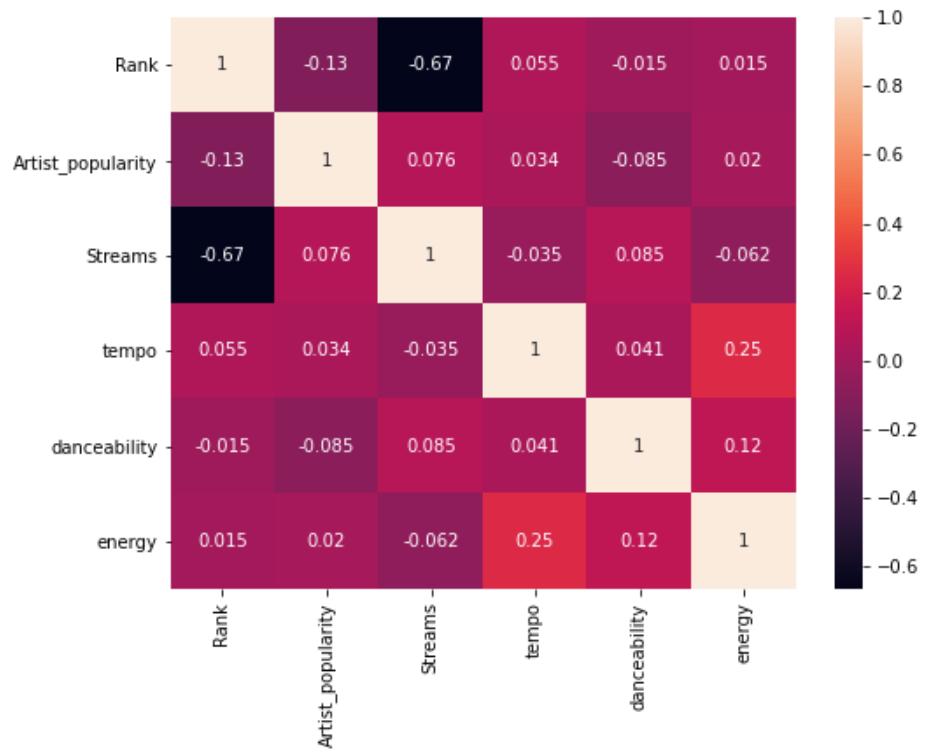
```
1 plt.matshow(cor)
2 plt.colorbar()
3 plt.show()
```

**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

Unit 37.

Data Tidying

Learning objectives

- ✓ Learners will be able to check for missing data within a given data frame.
- ✓ Learners will be able to delete or replace missing data.
- ✓ Learners will be able to perform descriptive statistics such as mean, median, mode, variance, and standard deviation for data frames.

Learning overview

- ✓ Differentiate situations that require data tidying.
- ✓ Search and check missing data.
- ✓ Delete missing data or replace it with another value.
- ✓ Learn the basics of descriptive statistics such as mean, median, mode, variance, standard deviation, and correlation coefficient.
- ✓ Data visualization: learning box about box plots, histograms, and scatter plots.

Concepts you will need to know from previous units

- ✓ Know how to select and slice data elements in a data frame.
- ✓ Know the basics of Matplotlib visualization.

Keywords

NaN

Tidy data

seaborn

Descriptive
Statistics

Boxplot

Mission

Descriptive Statistics and Visualization of Student Grades

- | The University of California, Irvine provides a sample dataset for data learning.
- | Among these datasets, download and unzip **student.zip** from
<https://archive.ics.uci.edu/ml/datasets/student%2Bperformance>

- ▶ The student-mat.csv file is used as the target data for data analysis.
- ▶ It is converted into a data frame using Pandas.
- ▶ Identify the characteristics of the data.
- ▶ If checking for missing data, run data pre-processing.
- ▶ Calculate the mean, median, and mode.
- ▶ Visualize in a box plot graph.
- ▶ Visualize all the variables in a histogram and scatterplot.

UCI 

Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact
 Search
 Repository Web Google

[View ALL Data Sets](#)

Browse Through: **588 Data Sets** [Table View](#) [List View](#)

Default Task	Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Classification (442) Regression (137) Clustering (117) Other (56)	 Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Categorical (38) Numerical (396) Mixed (55)	 Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Multivariate (456) Univariate (27) Sequential (57) Time-Series (121) Text (66) Domain-Theory (23) Other (21)	 Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Life Sciences (138) Physical Sciences (57) CS / Engineering (215) Social Sciences (38) Business (44) Game (11) Other (80)	 Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
	 Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
	 Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
	 Audiology (Original)	Multivariate	Classification	Categorical	226		1987

| Key concept

1. Why is Data Tidying Necessary?

- | So far, we have learned to create data frames and series, or to bring in other files and turn them into data frames. In fact, the most frequently encountered situation when doing such similar tasks in the field is when the data form of the analysis target is abnormal.
- | Abnormal data refers to cases where data is missing, units do not match, or are overlapped without rules. The process of organizing the abnormal data is called data tidying.
- | This term was coined by Hadley Wickham(<http://hadley.nz/>) in his paper "Tidy Data"(Paper Link: <https://vita.had.co.nz/papers/tidy-data.html>). The paper is available for download, so please check it out.



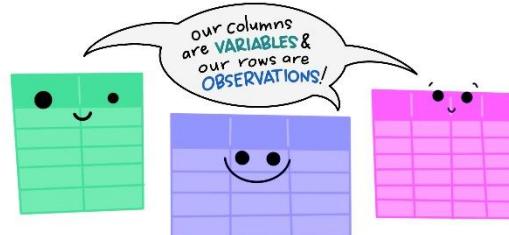
1. Why is Data Tidying Necessary?

| There is something that data scientists in the field always say.

"80% of data analysis is spent on tidying and preparing data."

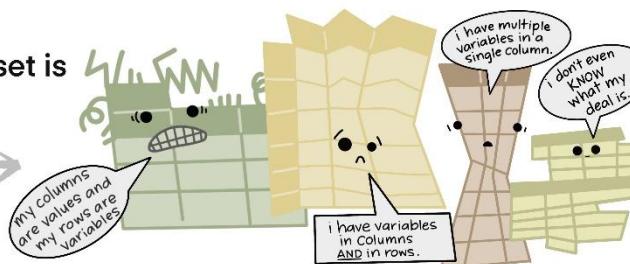
- | In addition, the reason why this work is so tedious and repetitive is because data preparation and tidying do not end in a single process, but rather, requires repeatedly tidying new problems that are constantly found in the process.
- | Therefore, Tidy Data provides a standard method of configuring data values within a standardized data set. The standard makes it easy to clean up initial data because you don't have to start from scratch and perform new tasks every time.

The standard structure of
tidy data means that
"tidy datasets are all alike..."



...but every messy dataset is
messy in its own way.

—HADLEY WICKHAM



<https://cfss.uchicago.edu/notes/tidy-data/>

2. Definition of Tidy Data

"Tidy data is a standard way of mapping the meaning of a dataset to its structure."

- Hadley Wickham

I The 3 rules of Tidy Data

- ▶ Each variable must have its own column.
- ▶ Each observation must have its own row.
- ▶ Each value must have its own cell.

country	year	cases	population
Afghanistan	1990	745	1998071
Afghanistan	2000	2666	2059360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	128042583

variables

country	year	cases	population
Afghanistan	1990	745	1998071
Afghanistan	2000	2666	2059360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	128042583

observations

country	year	cases	population
Afghanistan	99	745	1998071
Afghanistan	00	2666	2059360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	212258	1272915272
China	00	213766	128042583

values

<https://cfss.uchicago.edu/notes/tidy-data/>

3. Messy Data

| The following is a summary of realistic situations in which data tidying is necessary.

Messy Data

- ▶ If the column is not the name of the variable, but the value itself
- ▶ If there is not just one variable in the column, but multiple variables
- ▶ If the variables are stored in both columns and rows (should be stored in columns)
- ▶ If missing data exists
- ▶ If it's not a sample for the desired period of time
- ▶ If quantitative data is needed but the variable is qualitative
- ▶ If the data types of the values are wrong
- ▶ If there is duplication of data

4. Setting Up Data for Practice

`seaborn.load_dataset`

- | You can call up and use the needed amount of dataset at seaborn's online repository (<https://github.com/mwaskom/seaborn-data>). By accessing this link, you can also see what data sets there are. The small drawback is that Internet connection is required in order to access the repository.
- | Since the result value is returned in the Pandas data frame, it is useful for learning purposes.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('titanic', cache=True)
```



Line 5

- Brings in the data set of titanic survivors. At this time, check the local cache first and set it to cache=True in order to set it up for use.

4. Setting Up Data for Practice

```
seaborn.load_dataset
```

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

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
1 df.shape
```

(891, 15)

5. Checking for Missing Data

- | In many cases, the value of the element data is omitted in the data frame. In Pandas, which we are learning now, missing data is displayed as NaN(Not a number).
- | The following are the reasons why missing data occurs. There are many other reasons, but only the common cases are summarized here.

- ▶ If there are no values corresponding to each other in the two data sets joining E
- ▶ If the data from an external source is incomplete
- ▶ If the data is missing at the time of collection because it will be filled up later
- ▶ If the event continues to occur and accumulate despite an error in the value
- ▶ If the shape of the data is changed due to adding a new column or row (that was not checked during data reshaping)

5. Checking for Missing Data

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
  0   survived    891 non-null    int64  
  1   pclass      891 non-null    int64  
  2   sex         891 non-null    object  
  3   age         714 non-null    float64 
  4   sibsp       891 non-null    int64  
  5   parch       891 non-null    int64  
  6   fare         891 non-null    float64 
  7   embarked    889 non-null    object  
  8   class        891 non-null    category 
  9   who          891 non-null    object  
  10  adult_male  891 non-null    bool    
  11  deck         203 non-null    category 
  12  embark_town 889 non-null    object  
  13  alive        891 non-null    object  
  14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```



Line 1

- You can see that the number of valid values among the data held for each column.

5. Checking for Missing Data

- The RangeIndex shows that each column has 891 data elements. However, it can be seen that the number of non-null data is less than the total amount of data in columns such as age and deck. Simply calculating, it can be confirmed that there are NaN data elements of $891 - 714 = 177$ for the case of age.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column      Non-Null Count   Dtype  
 ---  --          --          --      
 0   survived    891 non-null     int64  
 1   pclass      891 non-null     int64  
 2   sex         891 non-null     object 
 3   age         714 non-null     float64
 4   sibsp       891 non-null     int64  
 5   parch       891 non-null     int64  
 6   fare         891 non-null     float64
 7   embarked    889 non-null     object 
 8   class        891 non-null     category
 9   who          891 non-null     object 
 10  adult_male  891 non-null     bool    
 11  deck         203 non-null     category
 12  embark_town 889 non-null     object 
 13  alive        891 non-null     object 
 14  alone        891 non-null     bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
df['age'].value_counts(dropna=False)
```

Age	Count
NaN	177
24.00	30
22.00	27
18.00	26
28.00	25
36.50	1
55.50	1
0.92	1
23.50	1
74.00	1

Name: age, Length: 89, dtype: int64

```
df['deck'].value_counts(dropna=False)
```

Deck	Count
NaN	688
C	59
B	47
D	33
E	32
A	15
F	13
G	4

Name: deck, dtype: int64

5. Checking for Missing Data

- If the dropna=False parameter is used in the df.value_counts() method, the number of missing data can be returned in the form of a Series.
 - ▶ https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html



TIP

- If the parameter is set from df.value_counts() to dropna=True, or if the dropna parameter is not used at all, the number of data excluding missing data is calculated.

```
1 df['deck'].value_counts(dropna=True)
```

```
deck
C    59
B    47
D    33
E    32
A    15
F    13
G     4
Name: count, dtype: int64
```

5. Checking for Missing Data



TIP

- If the parameter is set from df.value_counts() to dropna=True, or if the dropna parameter is not used at all, the number of data excluding missing data is calculated.

```
1 df['deck'].value_counts(dropna=False)
```

```
deck
NaN    688
C      59
B      47
D      33
E      32
A      15
F      13
G       4
Name: count, dtype: int64
```

6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
<code>df.isnull()</code>	0	False	False	False	False	False	False	False	False	False	False	True
<code>Nan → True</code>	1	False	False	False	False	False	False	False	False	False	False	False
	2	False	False	False	False	False	False	False	False	False	False	True
	3	False	False	False	False	False	False	False	False	False	False	False
	4	False	False	False	False	False	False	False	False	False	False	True

6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
<code>df.notnull()</code>	0	True	True	True	True	True	True	True	True	True	True	False
<code>Nan → False</code>	1	True	True	True	True	True	True	True	True	True	True	True
	2	True	True	True	True	True	True	True	True	True	True	False
	3	True	True	True	True	True	True	True	True	True	True	True
	4	True	True	True	True	True	True	True	True	True	True	False

6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

```
1 df.isnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	False	False	False	False	False	False	False	False	False	False	False	True		False	False
1	False	False	False	False	False	False	False	False	False	False	False	False		False	False
2	False	False	False	False	False	False	False	False	False	False	False	True		False	False
3	False	False	False	False	False	False	False	False	False	False	False	False		False	False
4	False	False	False	False	False	False	False	False	False	False	False	True		False	False
...
886	False	False	False	False	False	False	False	False	False	False	False	True		False	False
887	False	False	False	False	False	False	False	False	False	False	False	False		False	False
888	False	False	False	True	False	False	False	False	False	False	False	True		False	False
889	False	False	False	False	False	False	False	False	False	False	False	False		False	False
890	False	False	False	False	False	False	False	False	False	False	False	True		False	False

6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

```
1 df.notnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone	
0	True	True	True	True	True	True	True	True	True	True	True	False		True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True		True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	False		True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True		True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	False		True	True	True
...
886	True	True	True	True	True	True	True	True	True	True	True	False		True	True	True
887	True	True	True	True	True	True	True	True	True	True	True	True		True	True	True
888	True	True	True	False	True	True	True	True	True	True	True	False		True	True	True
889	True	True	True	True	True	True	True	True	True	True	True	True		True	True	True
890	True	True	True	True	True	True	True	True	True	True	True	False		True	True	True

6. Finding Missing Data

The `.sum()` method treats True and False as 1 and 0. If applied, the total NaN can be obtained.

```
1 df.isnull().sum()
```

```
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare          0
embarked     2
```

- ▶ If you check the results through this code, there are 177 in the page column.
- ▶ There are two missing data in the `embark_town` column and two in the `deck` column.



Line 1

- Check how much missing data are for each column in the entire data with numbers.

6. Finding Missing Data

```
1 df.isnull().sum().sum()
```

869



Line 1

- In the previous result series, if .sum() is applied once more, we can know the total number of NaN values in the data frame.

```
1 (len(df)-df.count()).sum()
```

869



Line 1

- df.count() returns the number of values other than missing data for each column. The number of missing data can be obtained by subtracting this from the total amount of data.

7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

I Delete the column.

- ▶ The total number of passengers is 891, and 688 do not have their deck information.
- ▶ Since the proportion of missing data is very high, it can be said that the column is meaningless from the standpoint of processing and analyzing data. In this case, the most common way to deal with missing data is to delete columns with missing data.

7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

I Delete the column.

```
1 new_df = df.dropna(axis = 1, thresh = 500)
2
3 new_df. head ()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True

 Line 1, 3

- 1: thresh = 500 is a command to delete all columns with more than 500 NaN values.
- 3: Since the deck column shows 688 NaNs and there are more than 500 NaNs, we can confirm that all the results are deleted.

7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

Delete the row.

- ▶ 177 out of 891 have no data on age. If the passenger's age is considered an important variable in data analysis, it is recommended to delete the passenger's data(row) without age data.
- ▶ If subset='age,' delete all rows with NaN values (axis = 0) from the rows in the age column.
- ▶ how = 'all' is deleted only if all data is NaN.

```
1 age_df = df.dropna(axis = 0, how= 'any', subset =[ 'age'])
2 len(age_df)
```

891

Line 1, 2

- 1: There were 177 rows without age data above, We deleted this.
- 2: Therefore, it is only normal that the number of the result data is 714.

7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

- Delete the column.

```
1 no = df.dropna()  
2 no.isnull().sum()
```

```
survived      0  
pclass        0  
sex           0  
age           0  
sibsp         0  
parch         0  
fare           0  
embarked      0
```



Line 1

- If there is any NaN in the column, delete it.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

- | Just because there is missing data in the preprocessing stage of the data set, deleting all the columns or rows is unfavorable because it reduces the number of data sets to be analyzed.
- | However, replacing NaN's data with values such as 0 or 1 will affect data analysis.
- | Therefore, it is usually a value that replaces missing data, and the mean value and mode value, which represent the distribution and characteristics of the data set, are obtained and filled.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

- ▶ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html?highlight=mean#pandas.DataFrame.mean>

```
1 import pandas as pd
2 import seaborn as sns
3
4 df = sns.load_dataset('titanic', cache=True)
5
6 df['age'].head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5    NaN
6    54.0
7    2.0
8    27.0
9    14.0
Name: age, dtype: float64
```

8. Replacing Missing Data with Other Data

```
1 import pandas as pd  
2 import seaborn as sns  
3  
4 df = sns.load_dataset('titanic', cache=True)  
5  
6 df['age'].head(10)
```

```
0    22.0  
1    38.0  
2    26.0  
3    35.0  
4    35.0  
5    NaN  
6    54.0  
7     2.0  
8    27.0  
9    14.0  
Name: age, dtype: float64
```

Line 4, 6

- 4: Bring up the Titanic Dataset.
- 6: Check the data in the age column, the data is found as NaN.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

```
1 avg_age = df['age'].mean(axis=0)
2 avg_age
```

```
29.69911764705882
```

 Line 1

- The mean of the data age is stored in avg_age. It is the mean of the data in the age column.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

```
1 median_age = df['age'].median(axis=0)
2
3
4 median_age
```

28.0

Line 1, 2

- 1: The median, using the median() method, can also be used as replacement instead of the mean.
- 2: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.median.html?highlight=median#pandas.DataFrame.median>

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

```
1 df['age'].fillna(avg_age, inplace=True)
```

 Line 1

- NaN data elements are substituted with the mean using `fillna()`. Let's replace it using the median value `median_age`, calculated earlier.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

```
1 df['age'].head(10)
```

```
0    22.000000
1    38.000000
2    26.000000
3    35.000000
4    35.000000
5    29.699118
6    54.000000
7     2.000000
8    27.000000
9    14.000000
Name: age, dtype: float64
```

 Line 1

- We can see that the missing data is replaced with the mean.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

1) Replacing with the Mean

```
1 df.isnull().sum()
```

```
survived      0      class        0
pclass        0      who          0
sex           0      adult_male    0
age           0      deck         688
sibsp         0      embark_town  2
parch         0      alive         0
fare           0      alone        0
embarked      2      dtype: int64
```

Line 1

- It appears that there is all NaN replaced in the age column.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value

- ▶ https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.value_counts.html?highlight=value_counts
- ▶ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.idxmax.html?highlight=idxmax#pandas.DataFrame.idxmax>
- ▶ Let's search for the missing data of embark_town by searching for the name of the city with the most passengers and replace it with the data.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value

```
1 import pandas as pd
2 import seaborn as sns
3
4 df = sns.load_dataset('titanic', cache=True)
5
```

 Line 4

- Bring up the Titanic data set.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value

```
1 town_count = df['embark_town'].value_counts(dropna=True)
2
3 town_count
```

```
embark_town
Southampton    644
Cherbourg      168
Queenstown     77
Name: count, dtype: int64
```

 Line 1

- Return the time series including unique rows in the corresponding column. Exclude missing data.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value

```
1 most=town_count.idxmax()  
2  
3 most
```

'Southampton'

 Line 1

- df.idxmax() returns the index in which the maximum value first occurs on the requested axis.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value

```
1 df['embark_town'].fillna(most, inplace=True)
2
3 df.isnull().sum()
```

```
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare          0
embarked      2
```

Line 1, 3

- 1: Using the `fillna()` method, NaN data elements are substituted with the names of the most embarking towns stored in the variable `most`.
- 3: Missing data in the `embark_town` column has been replaced by Southampton, resulting in no missing data in the column.

8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

3) Replacing with Nearest Neighbor Values

- ▶ This is not true for all data, but usually due to its nature, neighboring data often have similarities. After checking the characteristics of the dataset, the missing data is replaced with previous data or the one immediately after.

```
1 df_01=df['embark_town'].fillna(method = 'ffill', inplace=True)
2
3 df_01
```

```
1 df_02=df['embark_town'].fillna(method = 'bfill', inplace=True)
2
3 df_02
```

| Let's code

Step 1

I Preparing Data and Creating Data Frame

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 student_data = pd.read_csv("./data/student/student-mat.csv")
7 student_data.head()
```

```
school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activities;nursery;higher;internet;romantic;famrel;freetime;goout;Dalc;Walc;health;absences;G1;G2;G3
0 GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher...
1 GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";...
2 GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";...
3 GP;"F";15;"U";"GT3";"T";4;2;"health";"services...
4 GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"h...
```

- ▶ Although the data input has been confirmed, the data is difficult to handle in this state. It can be confirmed that the downloaded data is divided into ' ; '.
- ▶ Usually, csv files are divided by ' , ', but this file is divided into semicolons, make it difficult to check visually.
- ▶ In order to change the character symbol that separates the data, the parameter sep= 'separating character symbol' is used to designate it

Step 1

| Preparing Data and Creating Data Frame

```
1 student_data = pd.read_csv("./data/student/student-mat.csv", sep=';')
2 student_data.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5

5 rows × 33 columns

Step 1

Checking Data Characteristics

```
1 student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   school      395 non-null    object  
 1   sex          395 non-null    object  
 2   age          395 non-null    int64  
 3   address     395 non-null    object  
 4   famsize     395 non-null    object  
 5   Pstatus      395 non-null    object  
 6   Medu         395 non-null    int64  
 7   Fedu         395 non-null    int64  
 8   Mjob          395 non-null    object  
 9   Fjob          395 non-null    object  
 10  reason        395 non-null    object  
 11  guardian     395 non-null    object
```



Line 1

- non-null means that there is no null data.

Step 1

| Searching for any NaN in the Data

```
1 student_data.isnull().sum()
```

```
school      0
sex         0
age         0
address     0
famsize     0
Pstatus     0
Medu        0
Fedu        0
Mjob        0
Fjob        0
reason      0
guardian    0
```

 Line 1

- Looking at the results, it can be confirmed that the number of NaN data for each column is 0.

Step 1

| Searching for any NaN in the Data

```
1 student_data.isnull().sum().sum()
```

```
0
```

**Line 1**

- Obtain the number of columns with #NaN data.

Step 1

I Understanding the Meaning of the Columns for Data Comprehension

- ▶ For data analysis, it is fundamental to understand the meaning of each column name of the data frame.
- ▶ In the case of this data, it is helpful because it is written in detail in the study.txt file included in the compressed file. However, on a daily basis, there are often no documents explaining these column names. In this case, it is necessary to contact the person who received the data to check the information on the column of the data set.

Step 1

I Understanding the Meaning of the Columns for Data Comprehension

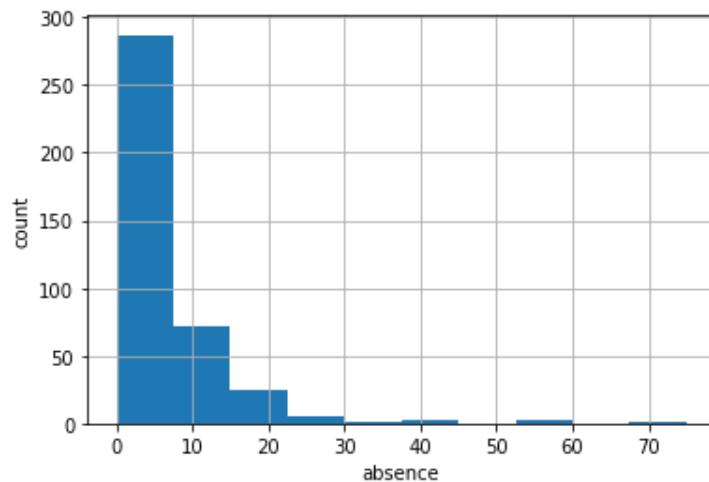
1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
2 sex - student's sex (binary: "F" - female or "M" - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: "U" - urban or "R" - rural)
5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
11 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
12 guardian - student's guardian (nominal: "mother", "father" or "other")
13 traveltimes - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)

Step 2

- | Let's visualize the number of absent days of students in a histogram. The number of days of absence is the column 30 absences.

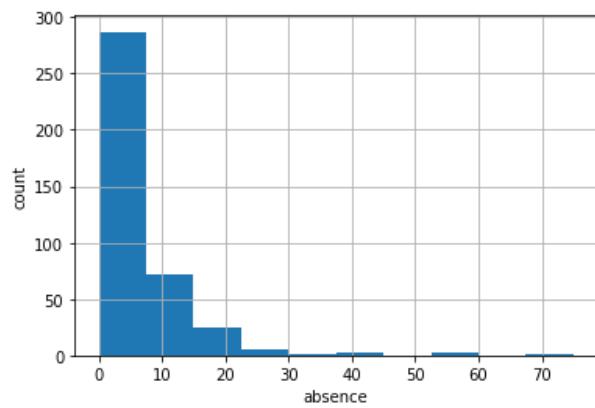
► https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib-pyplot-hist

```
1 plt.hist(student_data['absences'])
2
3 plt.xlabel('absence')
4 plt.ylabel('count')
5
6 plt.grid(True)
7
```



Step 2

```
1 plt.hist(student_data['absences'])  
2  
3 plt.xlabel('absence')  
4 plt.ylabel('count')  
5  
6 plt.grid(True)  
7
```



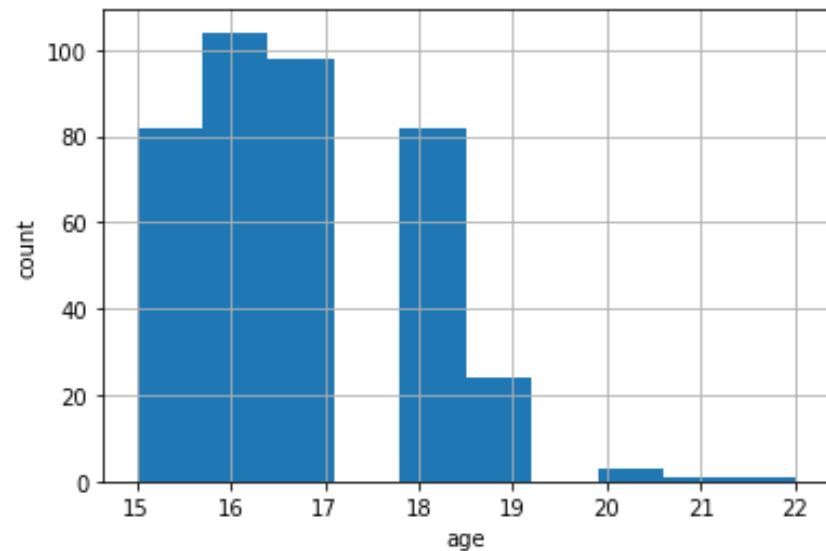
Line 1, 6

- 1: Specify the variable to be graphed on the histogram.
- 6: Add a grid to the graph.

Step 2

- | Let's make a histogram graph of other variables that have numeric data perform an exploratory data analysis.

```
1 plt.hist(student_data['age'])  
2  
3 plt.xlabel('age')  
4 plt.ylabel('count')  
5  
6 plt.grid(True)
```



Step 3

- | Find the basic descriptive statistics such as mean, median, mode, variance, and standard deviation etc.
- ▶ First of all, we learned in the previous lessons that we can check the results of various and basic descriptive statistics of the corresponding data frame using Pandas' `describe()` method.

```
1 student_data.describe()
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	3.235443	3.108861	1.481013
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	0.998862	1.113278	0.890741
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000

Step 3

| Median

- ▶ The median refers to the middle value of a data that is rearranged in the order of size.
- ▶ Assuming that the median value of studytime is obtained,

```
1 student_data['studytime'].median()
```

2.0

Step 3

I Mode

- ▶ The mode is the most frequent value in the data.
- ▶ Assuming that we obtain the mode of studytime,

```
1 student_data['studytime'].mode()
```

```
0    2  
Name: studytime, dtype: int64
```

Step 3

I Variance

- ▶ It is possible to check whether the data is scattered or concentrated around the mean by calculating the variance. After designating the reference variable, the var() method is used.
- ▶ Square the observed value minus the average, add it all, and divide it by the total number. That is, it's the sum of all squared differences. If you add all the deviations minus the mean from the observed values, you get zero, so you add them in squares.

```
1 student_data['studytme'].var()
```

0.7043243590567373



Line 1

- The smaller the result value, the smaller the degree of scattering the data.

Step 3

I Standard Deviation

- Where there is a lot of data, the average is often used as the value that represents the data. The standard deviation, one of the scatter plots, is a representative figure indicating how spread out the data is around the average. The unit of the standard deviation is identical to the unit of data, unlike the variance that uses the square root. If the standard deviation is close to the center, it means that the data values are concentrated near the average. The larger the standard deviation, the more widespread the data values are.

```
1 student_data['studytme'].std()
```

0.8392403464185556



Line 1

- The smaller the result value, the smaller the degree of scatter in the data.

Step 3

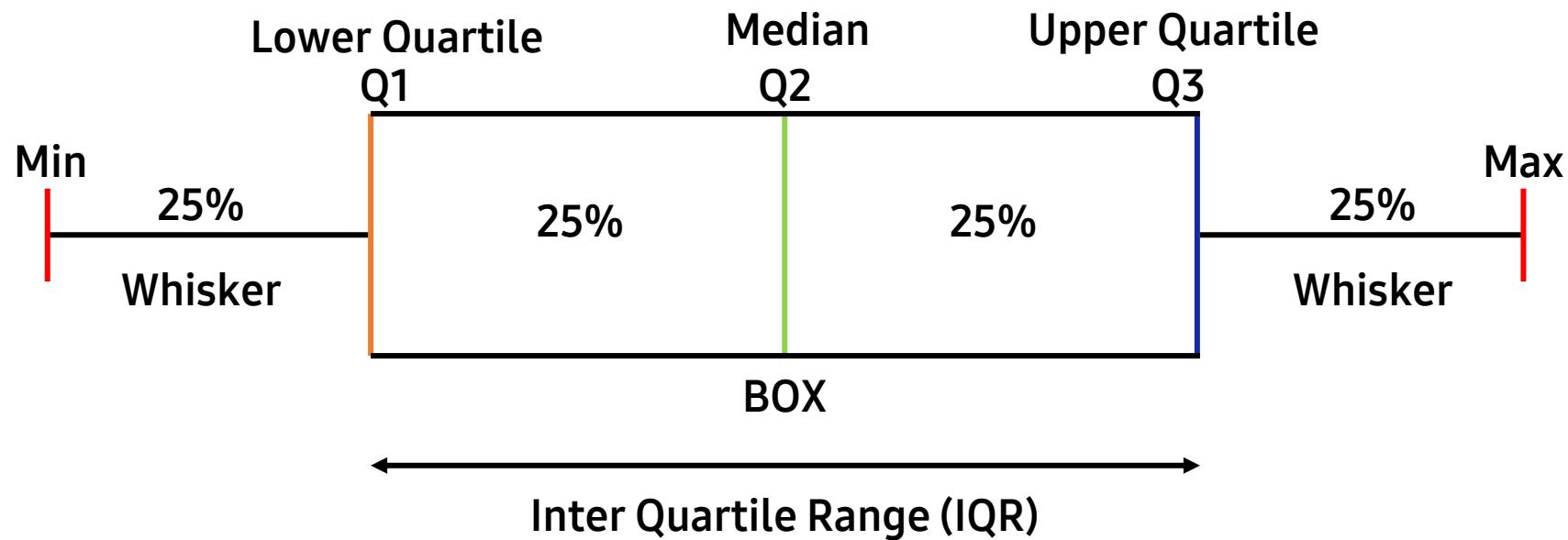
I Visualization through Box Plot

- ▶ Necessary Concepts for Understanding Box Plots

Percentile	Data divided into hundred equal parts
Quartile	Data divided into four parts
Median Value(Q2)	The value at the middle of the data (Half of the observations are greater than or equal, and the other half are smaller or equal)
The 3rd (Upper) Quartile (Q3)	The median of the top 50% based on the median value. The value corresponding to the top 25% of the entire data.
The 1st (Lower) Quartile (Q1)	The median of the bottom 50% based on the median value. The value corresponding to the bottom 25%, that is 75% of the total data.

Step 3

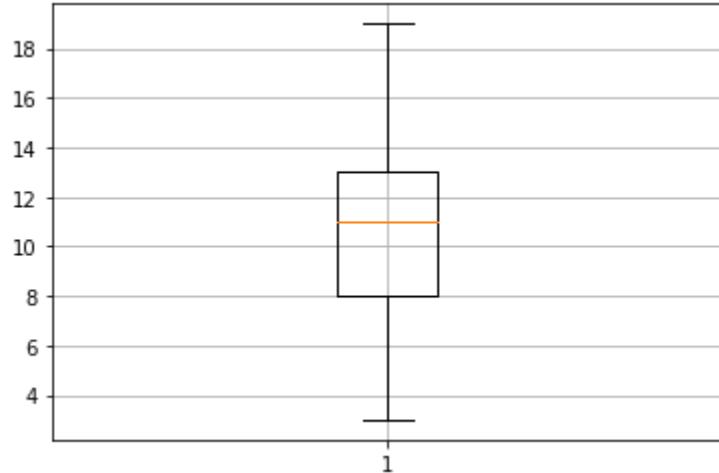
- Visualization through Box Plot



Step 3

I Visualization through Box Plot

```
1 plt.boxplot(student_data['G1'])  
2 plt.grid(True)
```



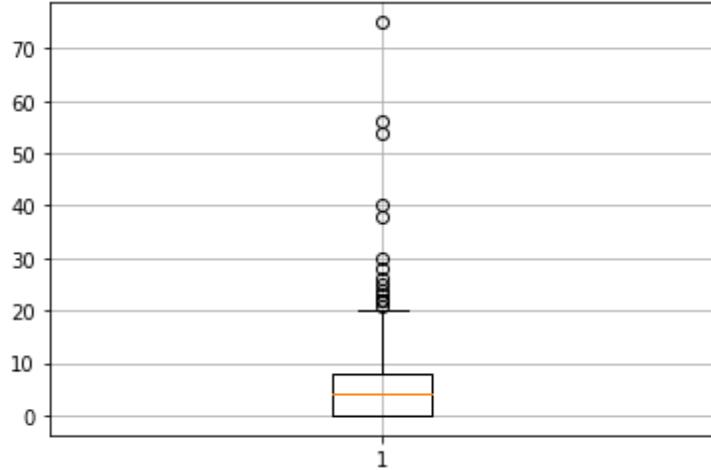
Line 1

- 1st Semester Grades

Step 3

I Visualization through Box Plot

```
1 plt.boxplot(student_data['absences'])  
2 plt.grid(True)
```



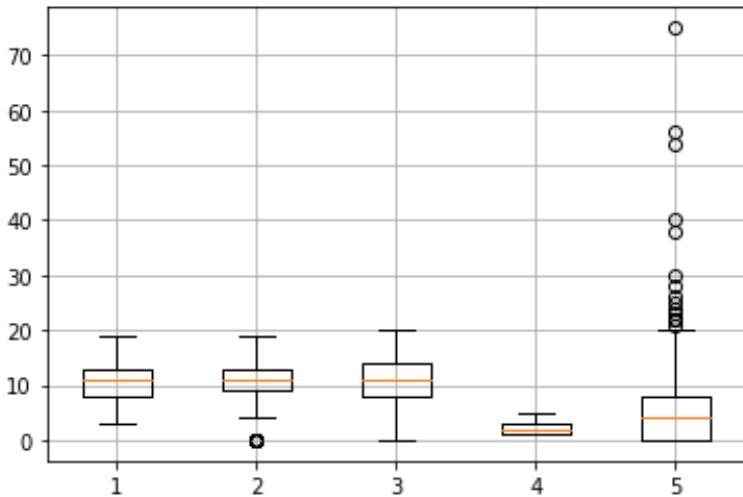
Line 1

- Number of absent days
- ▶ Looking at the graph, it can be seen that there are many abnormal data in the case of the box plot of the number of absences.

Step 3

Visualization through Box Plot

```
1 plt.boxplot([student_data['G1'],student_data['G2'],student_data['G3'],student_data['Walc'],student_data['absences']])
2 plt.grid(True)
```



- If you look at the box plot of the first semester grades, the second semester grades, the third semester grades, the weekend night outs statistics, and the number of days of absence, you can get insights on student performance.

Step 4

I Coefficient of Variation (CV)

- ▶ Data on variables with different units of measurement cannot be compared simply. This is because if the size of the data is different, the deviation tends to increase when the measurement unit is large.

Ex The standard deviation between stock prices and gas prices cannot be compared simply.

- ▶ What we can use in these situations is the coefficient of the variation.
- ▶ It is the value of the standard deviation divided by the mean. This can be used to compare data of different specifications regardless of size.

$$\text{Coefficient of Variation (CV)} = \frac{\text{Standard Deviation}}{\text{Mean}}$$

Step 4

| Coefficient of Variation (CV)

```
1 study_time_cv= student_data['studytime'].std() / student_data['studytime'].mean ()  
2 print(study_time_cv)
```

0.4123133542727978

```
1 absences_cv= student_data['absences'].std() / student_data['absences'].mean()  
2 print(absences_cv)
```

1.4018726369879073

Step 4

I Coefficient of Variation (CV)

- ▶ The `describe()` method does not show the result of the coefficient of variation for the whole, It can be applied as follows.
- ▶ In order to find the coefficient of variation for the whole, it can be applied as follows.
- ▶ However, it should be noted that if the mean to be compared is 0 or close to 0, the coefficient of variation may be infinitely large.

Step 4

| Coefficient of Variation (CV)

```
1 cv = student_data.std(numeric_only=True) / student_data.mean(numeric_only=True)
2 cv
```

```
age          0.076427
Medu        0.398177
Fedu        0.431565
traveltime   0.481668
studytime    0.412313
failures     2.225319
famrel       0.227330
freetime      0.308725
goout         0.358098
Dalc          0.601441
Walc          0.562121
health        0.391147
absences      1.401873
G1            0.304266
G2            0.351086
G3            0.439881
dtype: float64
```



Line 1

- Return the CV for the entire column as Series.

Step 4

| Coefficient of Variation (CV)

```
1 cv = student_data.std(numeric_only=True) / student_data.mean(numeric_only=True)
2 cv
```

```
age          0.076427
Medu        0.398177
Fedu        0.431565
traveltime   0.481668
studytime    0.412313
failures     2.225319
famrel       0.227330
freetime      0.308725
goout         0.358098
Dalc          0.601441
Walc          0.562121
health        0.391147
absences      1.401873
G1            0.304266
G2            0.351086
G3            0.439881
dtype: float64
```



Line 2

- However, since the entire data may not be numeric, it is recommended to use numeric_only=True option, or error will occur.

Step 4

I Covariance

- ▶ The covariance represents the relationship between the two variables.
 - If the values of the covariance is positive, the two variables are positive.
 - If the value of the covariance is negative, the two variables are negative.
- ▶ Multiply the deviation between the two variables and calculate it by averaging it. It is used to calculate the variance of two or more variables.
- ▶ It can be calculated using the cov() method.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.cov.html>

Step 4

Covariance

```
1 student_data.cov(min_periods=None, ddof=1, numeric_only=True)
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc
age	1.628285	-0.228619	-0.226949	0.062873	-0.004434	0.231221	0.061717	0.020947	0.180364	0.149039	0.192733
Medu	-0.228619	1.198445	0.742717	-0.131061	0.059667	-0.192681	-0.003842	0.033779	0.078115	0.019341	-0.066440
Fedu	-0.226949	0.742717	1.184180	-0.120073	-0.008379	-0.202641	-0.001337	-0.013963	0.052220	0.002313	-0.017702
traveltime	0.062873	-0.131061	-0.120073	0.486513	-0.059070	0.047844	-0.010512	-0.011861	0.022162	0.085941	0.120478
studytime	-0.004434	0.059667	-0.008379	-0.059070	0.704324	-0.108321	0.029898	-0.120041	-0.059706	-0.146533	-0.274304
failures	0.231221	-0.192681	-0.202641	0.047844	-0.108321	0.553017	-0.029564	0.068329	0.103123	0.090118	0.135964
famrel	0.061717	-0.003842	-0.001337	-0.010512	0.029898	-0.029564	0.803997	0.134974	0.064454	-0.061974	-0.130952
freetime	0.020947	0.033779	-0.013963	-0.011861	-0.120041	0.068329	0.134974	0.997725	0.316944	0.185954	0.190163
goout	0.180364	0.078115	0.052220	0.022162	-0.059706	0.103123	0.064454	0.316944	1.239388	0.264763	0.602744
Dalc	0.149039	0.019341	0.002313	0.085941	-0.146533	0.090118	-0.061974	0.185954	0.264763	0.793420	0.742852



Line 1

- Returns the covariance value of each column in the data frame.

Step 4

Covariance

```
1 np.cov(student_data['G1'],student_data['G3'])
```

```
array([[11.01705327, 12.18768232],  
       [12.18768232, 20.9896164 ]])
```

Line 1

- The covariance of NumPy can also be calculated through the cov() method. Two series columns, that is, the result of covariance for two series data.
- ▶ Analysis of the matrix above is as follows.
- The covariance values of G1 and G3: matrix elements (1,2) and (2,1) 12.18768232
 - Variance of G1: matrix element (1,1) 11.01705327
 - Variance of G3: matrix element (2,2) 20.9896164

Step 4

Covariance

```
1 print(student_data['G1'].var())
2 print(student_data['G3'].var())
```

11.017053267364904
20.989616397866733

Line 1, 2

- 1: Verification through var() to obtain the variance is the same as the result of the matrix element above.
- 2: Verification through var() to obtain the variance is the same as the result of the matrix element above.

Step 4

| Correlation Coefficient

- ▶ The covariance equation itself depends on the scale and unit of each variable. The correlation coefficient eliminates the dependence of each variable on the scale and finds out the relationship between data.
- ▶ The covariance can tell you what the relationship between the two variables is, but there are cases where a correlation coefficient is needed because the size of the relationship cannot be explained.
- ▶ Simply put, the correlation coefficient measures the degree to which two variables move together.

1.0	Complete positive correlation. This means that when one variable moves to a specific size, the other moves in the same direction at the same rate.
0.0	It means that the two variables have no relationship.
-1.0	Complete negative correlation or inverse correlation. This means that when one variable moves to a specific size, the other moves in the opposite direction.

- ▶ Can be calculated using the corr() method.
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html>

Step 4

Correlation Coefficient

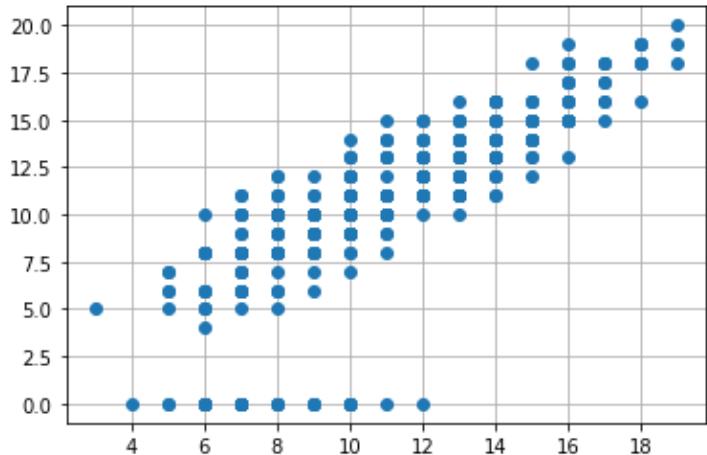
```
1 student_data.corr(numeric_only=True)
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc
age	1.000000	-0.163658	-0.163438	0.070641	-0.004140	0.243665	0.053940	0.016434	0.126964	0.131125	0.117276
Medu	-0.163658	1.000000	0.623455	-0.171639	0.064944	-0.236680	-0.003914	0.030891	0.064094	0.019834	-0.047123
Fedu	-0.163438	0.623455	1.000000	-0.158194	-0.009175	-0.250408	-0.001370	-0.012846	0.043105	0.002386	-0.012631
traveltime	0.070641	-0.171639	-0.158194	1.000000	-0.100909	0.092239	-0.016808	-0.017025	0.028540	0.138325	0.134116
studytime	-0.004140	0.064944	-0.009175	-0.100909	1.000000	-0.173563	0.039731	-0.143198	-0.063904	-0.196019	-0.253785
failures	0.243665	-0.236680	-0.250408	0.092239	-0.173563	1.000000	-0.044337	0.091987	0.124561	0.136047	0.141962
famrel	0.053940	-0.003914	-0.001370	-0.016808	0.039731	-0.044337	1.000000	0.150701	0.064568	-0.077594	-0.113397
freetime	0.016434	0.030891	-0.012846	-0.017025	-0.143198	0.091987	0.150701	1.000000	0.285019	0.209001	0.147822
goout	0.126964	0.064094	0.043105	0.028540	-0.063904	0.124561	0.064568	0.285019	1.000000	0.266994	0.420386
Dalc	0.131125	0.019834	0.002386	0.138325	-0.196019	0.136047	-0.077594	0.209001	0.266994	1.000000	0.647544
Walc	0.117276	-0.047123	-0.012631	0.134116	-0.253785	0.141962	-0.113397	0.147822	0.420386	0.647544	1.000000

Step 4

Scatter Plot

```
1 plt.plot(student_data['G1'],student_data['G3'], 'o')
2 plt.grid(True)
```



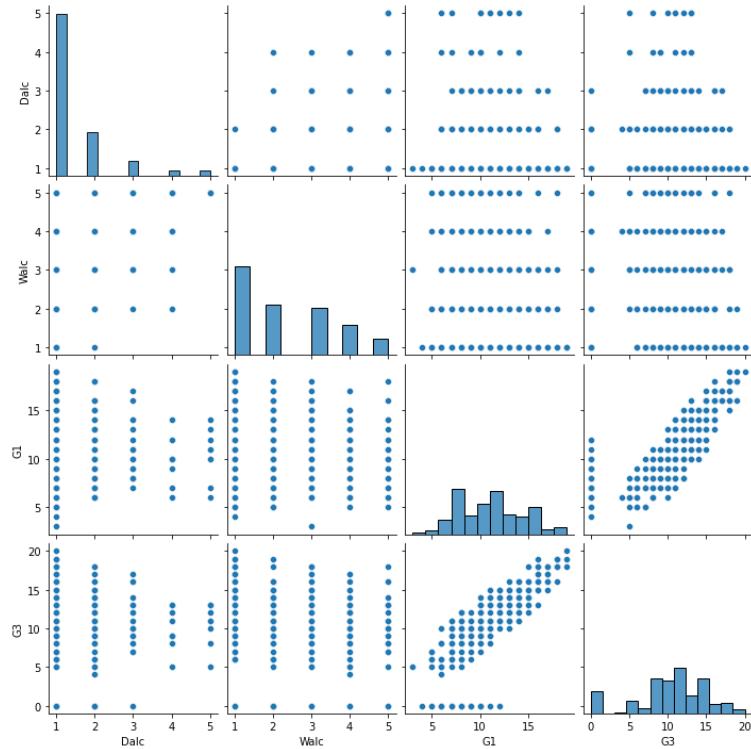
Line 1

- The graph displays the comparison results of the first and final tests. It shows that people with good grades from the beginning often perform well until the end.

Step 5

| Draw a histogram and scatter plot of the variables you want to compare.

```
1 sns.pairplot(student_data[['Dalc', 'Walc', 'G1', 'G3']])
2 plt.grid(True)
```



| 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. After converting the universities_ranking.csv file in the practice folder 'data folder' into a data frame, check if there is missing data, and decide how to replace it or delete it altogether with your colleague.

| The results of the discussion should be organized through actual codes.

```
1 import pandas as pd  
2 import seaborn as sns
```

```
1 a = pd.read_csv("./data/World University Rankings 2021/universities_ranking.csv")
```

Unit 38.

Time Series Data

Learning objectives

- ✓ Be able to determine whether it is an analysis target using time series data when a data frame is presented.
- ✓ Be able to explain the two data types of time series data.
- ✓ Be able to create date, time, interval, and array data as a time series data type.
- ✓ Be able to index and slice a data frame using DatetimeIndex.
- ✓ Be able to obtain a moving average, a representative descriptive statistic of time series data.
- ✓ Be able to visualize data as an area graph for comparison of two or more series data.

Learning overview

- ✓ Learn real-world applications of time series data analysis.
- ✓ Learn types and elements of time series data types.
- ✓ Learn how to generate time series data (date, time, interval, array of time series data).
- ✓ Learn how to index and slice time series data.
- ✓ Learn basic descriptive statistics using time series data functions.

Concepts you will need to know from previous units

- ✓ How to index and slice of data frames and series
- ✓ How to create an external file as a dataframe
- ✓ How to draw graphs using Matplotlib
- ✓ How to view the descriptive statistics summary of the datafram using Dataframe.info() function
- ✓ How to retrieve missing data and replacing data

Keywords

Time Series Data

Timestamp

Period

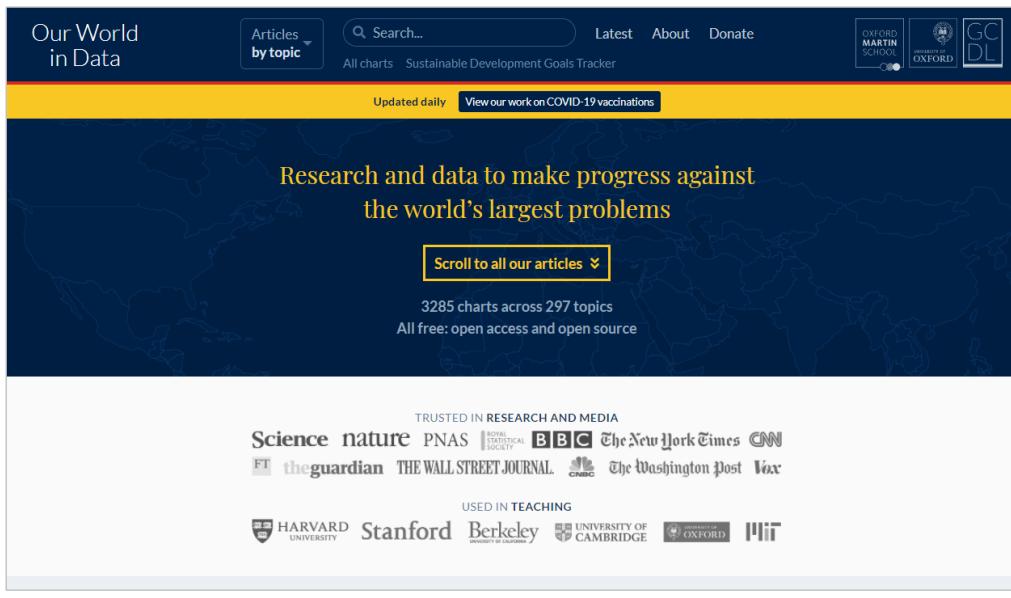
Moving Average

Area Graph

Categorical Data

Mission

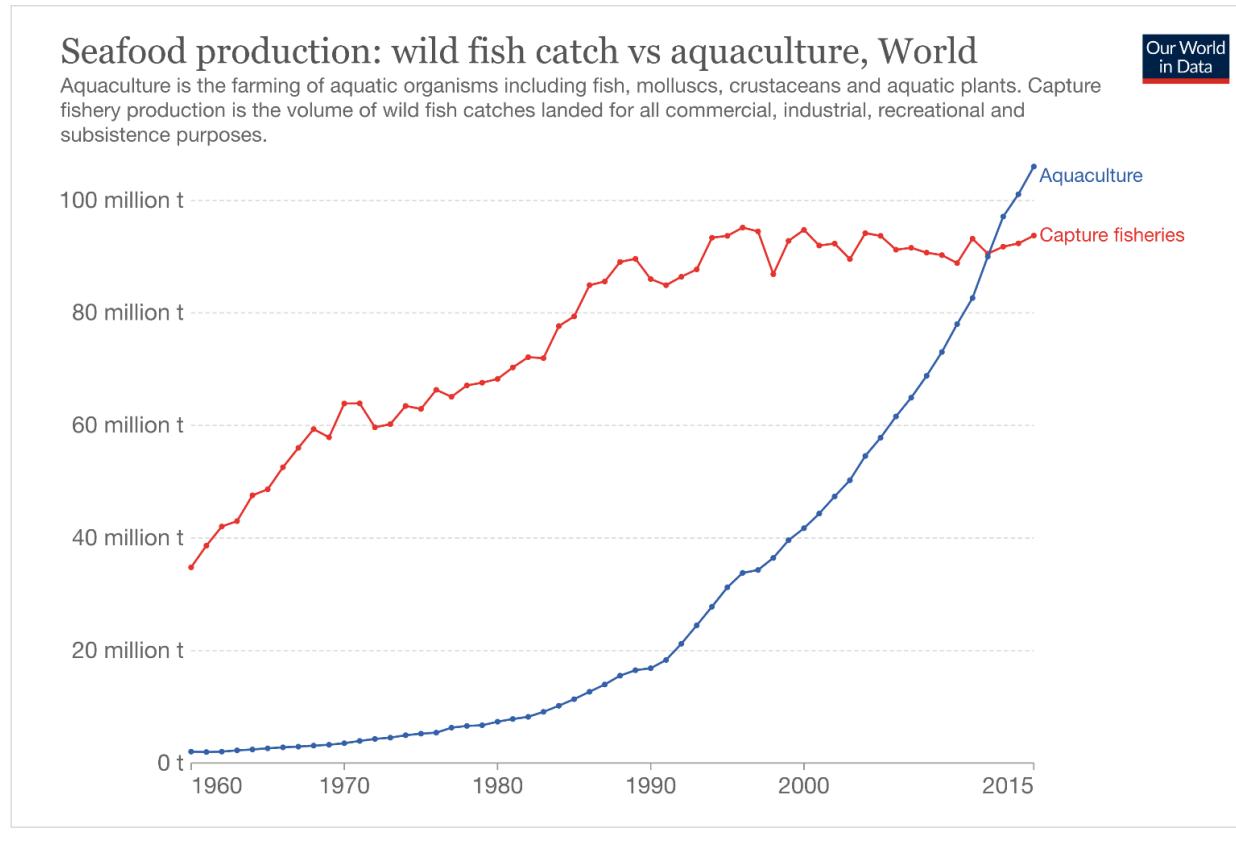
- | In the current global Seafood Production business, are most of the fish caught and processed directly by fishermen, or are they processed through aquaculture?
- | This may be different depending on the geographical location of each country or the difference in food culture. An organization called Our World in Data provides related data. Based on this, let's compare what type of raw material supply occupies the most in the current global Seafood Production business.
- | In addition, since this data provides data for each country, let's search for the country you want and create related data statistics separately.



| This mission must comply with the tasks below.

- ▶ Data resource location: <https://ourworldindata.org/fish-and-overfishing>
- ▶ Data as of November 6, 2021 are downloaded and provided for practice.
`df = pd.read_csv("./data//fish/capture-fisheries-vs-aquaculture.csv")`
- ▶ Convert the csv file to a data frame.
- ▶ Delete unnecessary columns in the data frame.
- ▶ Convert the year data to a time series data type and convert it to an index.
- ▶ Transform country data into categorical data.
- ▶ Check if NaN data exists and replace it.
- ▶ Use visualization tools to see trends in aquaculture and direct catch data around the world (All countries) from the 1960s to the present.
- ▶ Search for 3 or more individual countries and compare them through visualization.

- | The goal of this mission is to produce the same type of results as the graph below.
- | The graph below was created by data analysis experts.



| Key concept

1. What is time series data?

- | Time series data is a set of time-organized values collected over a certain time or period.
- | It can be explained a little more technically:

“A time series is data about one or more variables measured over a period of time at specific intervals.”

- | Machine learning technology is widely used for business predictive analysis. It analyzes business problems such as stock price, budget, sales and asset flow, forecast maintenance and sales forecasting, and predicts future indicators by composing related data such as trend, periodic theory, and seasonality together.
- | Pandas was created to handle financial data, and time series data analysis can be the core of data analysis using Pandas.
- | In addition to the financial field, time series data analysis is used in various fields.

- ▶ Medicine
- ▶ Meteorology
- ▶ Astronomy
- ▶ Economics
- ▶ IOT

2. Time Series Data and Analysis used in the Real World

- | The image below is the Manufacturing Purchasing Management Index (PMI) of the United States. Values above 50 indicate economic expansion, and values below 50 indicate contraction. It is a meaningful graph that can be a leading indicator of overall economic performance. This is an example where time series data is used in the real world. Even the predicted and actual figures are compared, and this can also be generated by analyzing time series data accumulated in the past.
- | <https://www.markiteconomics.com/Public/Release/PressReleases>

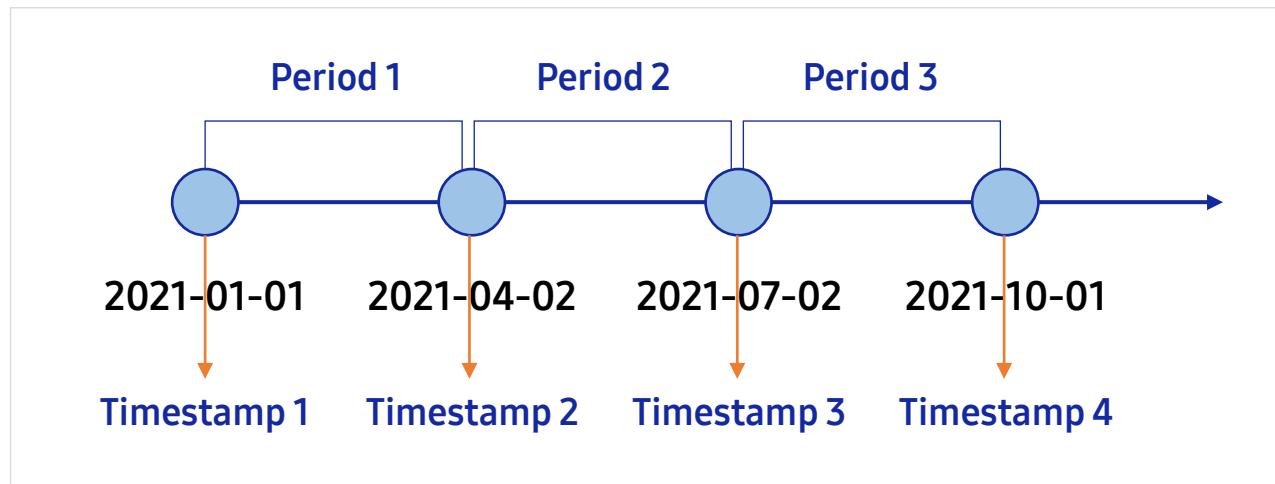


2. Time Series Data and Analysis used in the Real World

- | Time series data are sorted by time order, so there are continuous observation values. There is a high probability that each data value has a correlation with each other.
- | To understand time series data, you should first understand how pandas represents dates, times, and intervals. Pandas has many functions that can be used when transforming data at different frequencies or using a calendar that reflects business days and holidays for financial calculations.
- | Pandas was created to handle financial data. When analyzing financial data, changing time series data into an index within a data frame and using it has many advantages.
- | As it will be explained in the example code, when most of the external data is called and the columns of date are checked, there are many cases of string or object type. It starts with converting to Pandas time series data type for effective analysis.

3. Two Time Series Data Types in Pandas

- timestamp: one point in time
- period: two time points. i.e., a constant period between two timestamps
 - ▶ Multiple timestamps are gathered to create an array and become a DatetimeIndex.

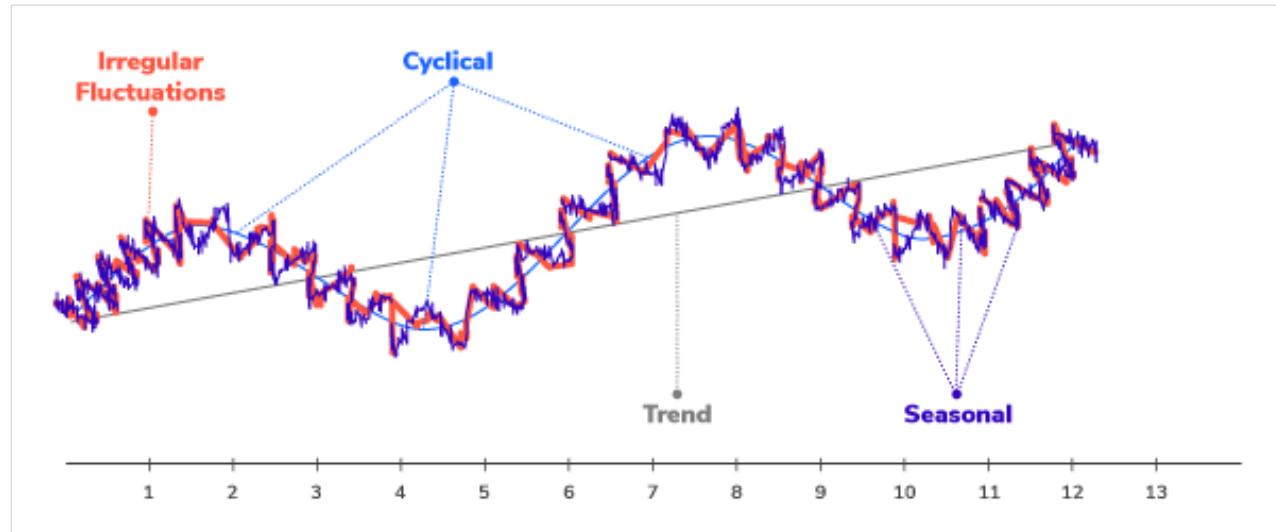


4. Elements of Time Series Data

Trend	A fluctuating pattern in data that appear in the long term (The values of the data are increased or decreased in a reasonably predictable pattern)
Seasonal	A fluctuating pattern in data that appears in a period of unit time such as a week, month, quarter, or half year, etc. (The patterns of the data are repeated over a specific period.)
Cycle	A long-term fluctuation rather than fixed periods that appear in a period of at least two years (The values of the data exhibit rises and falls that are not of a fixed frequency often due to economic conditions.)
Random	A completely irregular pattern that does not belong to the above three categories

4. Elements of Time Series Data

- | You need to decompose the components of each of the four representative time series data to discover insights.



- | This analysis of time series data is called a time series additive model (Time series additive model).
- | Trend factor + Cycle factor + Seasonal factor + Irregular/Random factor

$$Y_t = T_t + C_t + S_t + I_t$$

5. Creating an Interval of Date, Time, Frequency and Time

5.1. Create date and time with python datetime class

- The datetime object is part of Python's datetime library. This class is used to express general and various patterns, such as a specific point in time using both date and time, only the date minus the time, or only the time.
 - ▶ <https://docs.python.org/3/library/datetime.html#module-datetime>
- Most of the functions supported by the date and time class of the datetime library are supported.
- The disadvantage is that it reduces the precision needed for massive computations on time series data. The datetime class receives year, month, day, hour, minute, second, microsecond, and time zone as arguments. The time argument is not a required value. If empty, 0 is returned as a default value.
 - ▶ You can use Pandas by converting datetime to timestamp object.

5. Creating an Interval of Date, Time, Frequency and Time

5.1. Create date and time with python datetime class

```
1 import pandas as pd
2 import numpy as np
3 import datetime
4
5 from datetime import date, datetime, time, timezone
6
7 datetime (2023, 7, 7)
```

Line 7

- We need at least 3 parameters corresponding to year, month and day. Hour and minute are returned as 0 by default.

5. Creating an Interval of Date, Time, Frequency and Time

5.1. Create date and time with python datetime class

```
1 datetime(2021, 7, 7, 15, 50)
```

```
datetime.datetime(2021, 7, 7, 15, 50)
```

Line 1

- In case of setting the parameters of 15:50 in the hour and minute.

```
1 d = date(2021, 7, 7)
2 t = time(13, 26, 10)
3 datetime.combine(d, t)
4
```

```
datetime.datetime(2021, 7, 7, 13, 26, 10)
```

Line 4

- If you use the combine() method of the datetime class, you can create a datetime object using an existing date or time object.

5. Creating an Interval of Date, Time, Frequency and Time

5.1. Create date and time with python datetime class

```
1 datetime.now()
```

```
datetime.datetime(2023, 8, 20, 18, 52, 44, 800163)
```

 Line 1

- Current date and time, local time.

```
1 datetime.now(timezone.utc)
```

```
datetime.datetime(2023, 8, 20, 9, 52, 57, 981152, tzinfo=datetime.timezone.utc)
```

 Line 1

- If a time zone is used as a parameter in the now method, a datetime object applied with the time zone is created.

5. Creating an Interval of Date, Time, Frequency and Time

5.1. Create date and time with python datetime class

```
1 datetime.now().date()
```

```
datetime.date(2023, 8, 20)
```

 Line 1

- Get the current time and date and returns only the date.

```
1 datetime.now().time()
```

```
datetime.time(18, 53, 44, 386969)
```

 Line 1

- Get the current time and date and returns only the time.

5. Creating an Interval of Date, Time, Frequency and Time

5.2. Create a point in time with Pandas timestamp

`pandas.Timestamp()`

- | The date and time created with datetime can also be created with `pandas.Timestamp()`. The difference is that the datatype is `datetime64`, which has higher precision than Python `datetime`.
- | <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Timestamp.html>

5. Creating an Interval of Date, Time, Frequency and Time

5.2. Create a point in time with Pandas timestamp

```
pandas.Timestamp()
```

```
1 pd.Timestamp('2017, 5, 3, 12:00')
```

```
Timestamp('2017-05-03 12:00:00')
```

 Line 1

- Both date and time can be set when creating.

```
1 pd.Timestamp('12:00')
```

```
Timestamp('2023-08-20 12:00:00')
```

 Line 1

- Created by specifying the time.

5. Creating an Interval of Date, Time, Frequency and Time

5.2. Create a point in time with Pandas timestamp

```
pandas.Timestamp()
```

```
1 pd.Timestamp('12:00')
```

```
Timestamp('2023-08-20 12:00:00')
```

   Line 1

- If only time is specified, today's date is applied as default.

5. Creating an Interval of Date, Time, Frequency and Time

5.3. Create time intervals with Pandas Timedelta

- We have learned to create a moment of date and time so far. Now, we will learn how to express time intervals. timedelta is a subclass of Python's datetime.timedelta and is also supported by Pandas.
- You can use Pandas' Timedelta class for time intervals. The time interval is important for time series data analysis because it is necessary when determining the number of days or analyzing by a specific time interval.

```
pandas.Timedelta(value, unit=None, **kwargs)
```

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Timedelta.html>

5. Creating an Interval of Date, Time, Frequency and Time

5.3. Create time intervals with Pandas Timedelta

```
pandas.Timedelta(value, unit=None, **kwargs)
```

```
1 import pandas as pd
2 import datetime
3 from datetime import datetime
4
5 my_birth = datetime(1973, 7, 7)
6 today = datetime.today()
7 tomorrow = today + pd.Timedelta(days = 1)
8
9 print(tomorrow - my_birth)
```

```
18307 days, 18:57:47.726682
```



Line 5, 6, 7, 9

- 5: Create a specific date with datetime.
- 6: Create today's date.
- 7: Calculate the day plus one day using timedelta.
- 9: How long is tomorrow from my birthday?

5. Creating an Interval of Date, Time, Frequency and Time

5.3. Create time intervals with Pandas Timedelta

```
pandas.Timedelta(value, unit=None, **kwargs)
```

```
1 my_birth-today
```

```
datetime.timedelta(days=-18307, seconds=18132, microseconds=273318)
```

Line 1

- When calculating the number of days between two dates, data is returned in timedelta format.

5. Creating an Interval of Date, Time, Frequency and Time

5.3. Create time intervals with Pandas Timedelta

```
pandas.Timedelta(value, unit=None, **kwargs)
```

```
1 today= datetime.today()
2 data = today + pd.Timedelta(hours = 10)
3
4 print(today)
5 print (data)
```

```
2023-08-20 18:59:18.551611
```

```
2023-08-21 04:59:18.551611
```

 Line 2, 5

- 2: The difference in time can also be calculated arithmetically.
- 5: You can see that 10 hours are added to today.

5. Creating an Interval of Date, Time, Frequency and Time

5.4. Represent a period with Period

pandas.Period()

| <https://pandas.pydata.org/docs/reference/api/pandas.Period.html?highlight=pd%20period>

- | Most of time series data analysis is event analysis for a specific time interval. An example is an analysis of a company's sales over a specific period of time. However, when analyzing events by grouping multiple periods, it is difficult to use only timestamps. Pandas provides a standardized time interval through a class called Period to facilitate this kind of data organization and calculation.
- | Period creates a period based on a specified frequency such as daily, weekly, monthly, yearly, quarterly, etc. and provides Timestamp that indicates the start time and end time.
- | A period can be created using a timestamp corresponding to a reference point and a frequency that indicates the period.
- | If you create a period corresponding to a month based on July 1973, you can do it as follows.

```
1 import pandas as pd  
2  
3 special_day = pd.Period('1973-7', freq='M')  
4 special_day
```

Period('1973-07', 'M')

5. Creating an Interval of Date, Time, Frequency and Time

5.4. Represent a period with Period

```
pandas.Period()
```

- | pd.Period has attributes showing start time and end time.
- | https://pandas.pydata.org/docs/reference/api/pandas.Period.start_time.html#pandas.Period.start_time

```
1 special_day.start_time
```

```
Timestamp('1973-07-01 00:00:00')
```



- Line 1
- Return the start time of that point in time.

```
1 special_day.end_time
```

```
Timestamp('1973-07-31 23:59:59.999999999')
```



- Line 1
- Return the end time of that point in time.

5. Creating an Interval of Date, Time, Frequency and Time

5.4. Represent a period with Period

`pandas.Period()`

Period can be shifted through simple arithmetic operation. You can create a new period object by shifting the frequency. The example below is an example of +2 (shifting two months) because the frequency of special_day is one month.

```
1 n= special_day + 2
2
3
4 print(special_day)
5 print(n)
```

1973-07

1973-09



Line 1

- You cannot understand that adding 2 makes two months shifted just because 2 means two months.
You should understand the way that the period is shifted by the unit that created the period.

5. Creating an Interval of Date, Time, Frequency and Time

5.4. Represent a period with Period

```
pandas.Period()
```

```
1 n
```

```
Period('1973-09', 'M')
```

```
1 n.start_time, n.end_time
```

```
(Timestamp('1973-09-01 00:00:00'), Timestamp('1973-09-30 23:59:59.999999999'))
```

   Line 1

- See the results. You can see that Pandas is properly judging the full date of September 1973 (there are 30 days).

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data Timestamp and indexing with DatetimeIndex

- | You can convert data type to datetime64 type with timestamp with pandas.to_datetime().

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

- | https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html?highlight=to_datetime#pandas.to_datetime
- | The key to working with time series data in Pandas is indexing using DatetimeIndex objects.
- | The indexing is very useful, it automatically sorts data based on date and time.

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data Timestamp and indexing with DatetimeIndex

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

```
1 import pandas as pd  
2 wti = pd.read_csv("./data/wti/DCOILWTICO.csv")  
3 wti.head()
```

	DATE	DCOILWTICO
0	2016-11-01	46.66
1	2016-11-02	45.32
2	2016-11-03	44.66
3	2016-11-04	44.07
4	2016-11-07	44.88



Line 2

- As example data, we use the international crude oil price data provided by the FRED.
- Download the data from <https://fred.stlouisfed.org/series/DCOILWTICO>

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data **Timestamp** and indexing with **DatetimeIndex**

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

| You can see that the data type of the date column is object.

```
1 wti.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1770 entries, 0 to 1769  
Data columns (total 2 columns):  
 #   Column      Non-Null Count  Dtype     
 ---    
 0   DATE        1770 non-null    object    
 1   DCOILWTICO  1770 non-null    object    
 dtypes: object(2)  
 memory usage: 27.8+ KB
```

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data Timestamp and indexing with DatetimeIndex

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

```
1 wti['N_DATE'] = pd.to_datetime(wti['DATE'])  
2 wti['N_DATE']
```

```
0    2016-11-01  
1    2016-11-02  
2    2016-11-03  
3    2016-11-04  
4    2016-11-07  
     ...  
1765  2023-08-08  
1766  2023-08-09  
1767  2023-08-10  
1768  2023-08-11  
1769  2023-08-14  
Name: N_DATE, Length: 1770, dtype: datetime64[ns]
```

Line 2

- The data type is changed to datetime64 type.

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data **Timestamp** and indexing with **DatetimeIndex**

```
1 wti.head()
```

	DATE	DCOILWTICO	N_DATE
0	2016-11-01	46.66	2016-11-01
1	2016-11-02	45.32	2016-11-02
2	2016-11-03	44.66	2016-11-03
3	2016-11-04	44.07	2016-11-04
4	2016-11-07	44.88	2016-11-07

Line 1

- A column named N_Date is created.



TIP

- Sometimes, in the process of converting to timestamp, an error occurs if the data cannot be converted.
In this case, there is a way to force the conversion.
- If you use errors = 'coerce' parameter, NaT is forcibly assigned to data that cannot be converted.

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data Timestamp and indexing with DatetimeIndex

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

```
1 df = wti.drop(['DATE'], axis=1)  
2 df.head()
```

	DCOILWTICO	N_DATE
0	46.66	2016-11-01
1	45.32	2016-11-02
2	44.66	2016-11-03
3	44.07	2016-11-04
4	44.88	2016-11-07

Line 1

- Delete the DATE column that was the existing object data type.

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data Timestamp and indexing with DatetimeIndex

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

```
1 df.set_index('N_DATE', inplace = True)  
2 df.head()
```

DCOILWTICO	
N_DATE	
2016-11-01	46.66
2016-11-02	45.32
2016-11-03	44.66
2016-11-04	44.07
2016-11-07	44.88



Line 1

- Designate the newly created datetime64 column as the index of the data frame.

6. Time Series Data Indexing and Basic Use

6.1. Convert other data types to time series data **Timestamp** and indexing with **DatetimeIndex**

```
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True,  
unit=None, infer_datetime_format=False, origin='unix', cache=True)
```

| Now, the dataframe is very easy to index or slice in chronological order because it supports the time series index class.

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
DatetimeIndex: 1770 entries, 2016-11-01 to 2023-08-14  
Data columns (total 1 columns):  
 #   Column      Non-Null Count  Dtype    
 ---  --          --          --  
 0   DCOILWTICO  1770 non-null    object  
dtypes: object(1)  
memory usage: 27.7+ KB
```

 Line 1

- This completes the data frame indexed by DatetimeIndex.

6. Time Series Data Indexing and Basic Use

6.2. Slicing using DatetimeIndex

- In the previous lesson, we learned how to slice specific rows and columns of a dataframe. We check how easy and convenient it is to slice from time series data.

```
1 df['2016-11-04': '2016-11-10']
```

DCOILWTICO

N_DATE

2016-11-04	44.07
2016-11-07	44.88
2016-11-08	44.96
2016-11-09	45.20
2016-11-10	44.62



Line 1

- Select a specific date, and you can slice only the data you want very conveniently. Isn't it very intuitive?

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

- | Time series data can be created not only in units of one unit but also in a specific time interval, that is, in a form with a specific frequency.
- | You can use the freq parameter for pd.date_range() to create a time series with a frequency you want. The default is one unit.
- | The principle is similar to creating an array of numbers with Python range().

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

```
pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
```

start: the start of date range, **end**: the end of date range, **periods**: the number of timestamps to be created

| For more details on freq, refer to https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases

If we describe only representative examples of freq,

- ▶ D: calendar day frequency
- ▶ B: business day (Business day frequency)
- ▶ W: weekly frequency
- ▶ M: month and frequency
- ▶ MS: month start frequency
- ▶ Q: quarter and frequency

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

```
pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
```

- Since there are 5 periods, it means to create 5 Timestamps.

```
1 import pandas as pd  
2 pd.date_range(start='2021-01-01', end=None, periods=5, freq='D', tz=None)
```

```
DatetimeIndex(['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',  
                 '2021-01-05'],  
                dtype='datetime64[ns]', freq='D')
```

Line 2

- In case of a native timezone state where the timezone is not set, count the number of data created.

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

```
pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
```

```
1 import pandas as pd  
2 pd.date_range(start='2021-01-01', end='2021-12-31', periods=5, freq=None, tz='Asia/Seoul')
```

```
DatetimeIndex(['2021-01-01 00:00:00+09:00', '2021-04-02 00:00:00+09:00',  
                '2021-07-02 00:00:00+09:00', '2021-10-01 00:00:00+09:00',  
                '2021-12-31 00:00:00+09:00'],  
               dtype='datetime64[ns, Asia/Seoul]', freq=None)
```



- In case the time zone is set to Seoul.

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

```
pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
```

```
1 import pandas as pd  
2 pd.date_range(start='2021-01-01', end=None, periods=5, freq='M', tz=None)
```

```
DatetimeIndex(['2021-01-31', '2021-02-28', '2021-03-31', '2021-04-30',  
                 '2021-05-31'],  
                dtype='datetime64[ns]', freq='M')
```



Line 2

- In case of using the parameter as the frequency based on the end of the month.

7. Creating Time Series Data

7.1. Create time series of specific frequency from Timestamp array

```
pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)
```

```
1 import pandas as pd  
2 pd.date_range(start='2021-01-01', end=None, periods=5, freq='MS', tz=None)
```

```
DatetimeIndex(['2021-01-01', '2021-02-01', '2021-03-01', '2021-04-01',  
                 '2021-05-01'],  
                dtype='datetime64[ns]', freq='MS')
```



Line 2

- In case of using the frequency as a parameter based on the month start date.

7. Creating Time Series Data

7.2. Create time series of specific frequency from Period array

| You can create time series data containing multiple periods and frequencies with period_range(). It returns PeriodIndex as a result.

```
pandas.period_range(start=None, end=None, periods=None, freq=None, name=None)
```

| https://pandas.pydata.org/docs/reference/api/pandas.period_range.html

```
1 import pandas as pd
2
3 p_data= pd.period_range(start='2020-1-1',
4                         end= '2020-12-31',
5                         periods = None,
6                         freq='M')
7
8 p_data
```

```
PeriodIndex(['2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06',
       '2020-07', '2020-08', '2020-09', '2020-10', '2020-11', '2020-12'],
      dtype='period[M']')
```



Line 3, 4

- 3: Start of date range
- 4: End of date range

7. Creating Time Series Data

7.2. Create time series of specific frequency from Period array

| You can create time series data containing multiple periods and frequencies with period_range(). It returns PeriodIndex as a result.

```
pandas.period_range(start=None, end=None, periods=None, freq=None, name=None)
```

| https://pandas.pydata.org/docs/reference/api/pandas.period_range.html

```
1 import pandas as pd
2
3 p_data= pd.period_range(start='2020-1-1',
4                         end= '2020-12-31',
5                         periods = None,
6                         freq='M')
7
8 p_data
```

```
PeriodIndex(['2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06',
       '2020-07', '2020-08', '2020-09', '2020-10', '2020-11', '2020-12'],
      dtype='period[M]')
```

 Line 5, 6

- 5: Number of frequencies to generate
- 6: Length of period

7. Creating Time Series Data

7.2. Create time series of specific frequency from Period array

| You can create time series data containing multiple periods and frequencies with period_range(). It returns PeriodIndex as a result.

```
pandas.period_range(start=None, end=None, periods=None, freq=None, name=None)
```

| https://pandas.pydata.org/docs/reference/api/pandas.period_range.html

```
1 import pandas as pd
2
3 p_data= pd.period_range(start='2020-1-1',
4                         end= '2020-12-31',
5                         periods = None,
6                         freq='M')
7
8 p_data
```

```
PeriodIndex(['2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06',
       '2020-07', '2020-08', '2020-09', '2020-10', '2020-11', '2020-12'],
       dtype='period[M']')
```



Line 8

- The label of the index is period.

7. Creating Time Series Data

7.2. Create time series of specific frequency from Period array

```
pandas.period_range(start=None, end=None, periods=None, freq=None, name=None)
```

```
1 for i in p_data:  
2     print("{0} {1}".format(i.start_time, i.end_time))
```

```
2020-01-01 00:00:00 2020-01-31 23:59:59.999999999  
2020-02-01 00:00:00 2020-02-29 23:59:59.999999999  
2020-03-01 00:00:00 2020-03-31 23:59:59.999999999  
2020-04-01 00:00:00 2020-04-30 23:59:59.999999999  
2020-05-01 00:00:00 2020-05-31 23:59:59.999999999  
2020-06-01 00:00:00 2020-06-30 23:59:59.999999999  
2020-07-01 00:00:00 2020-07-31 23:59:59.999999999  
2020-08-01 00:00:00 2020-08-31 23:59:59.999999999  
2020-09-01 00:00:00 2020-09-30 23:59:59.999999999  
2020-10-01 00:00:00 2020-10-31 23:59:59.999999999  
2020-11-01 00:00:00 2020-11-30 23:59:59.999999999  
2020-12-01 00:00:00 2020-12-31 23:59:59.999999999
```



Line 2

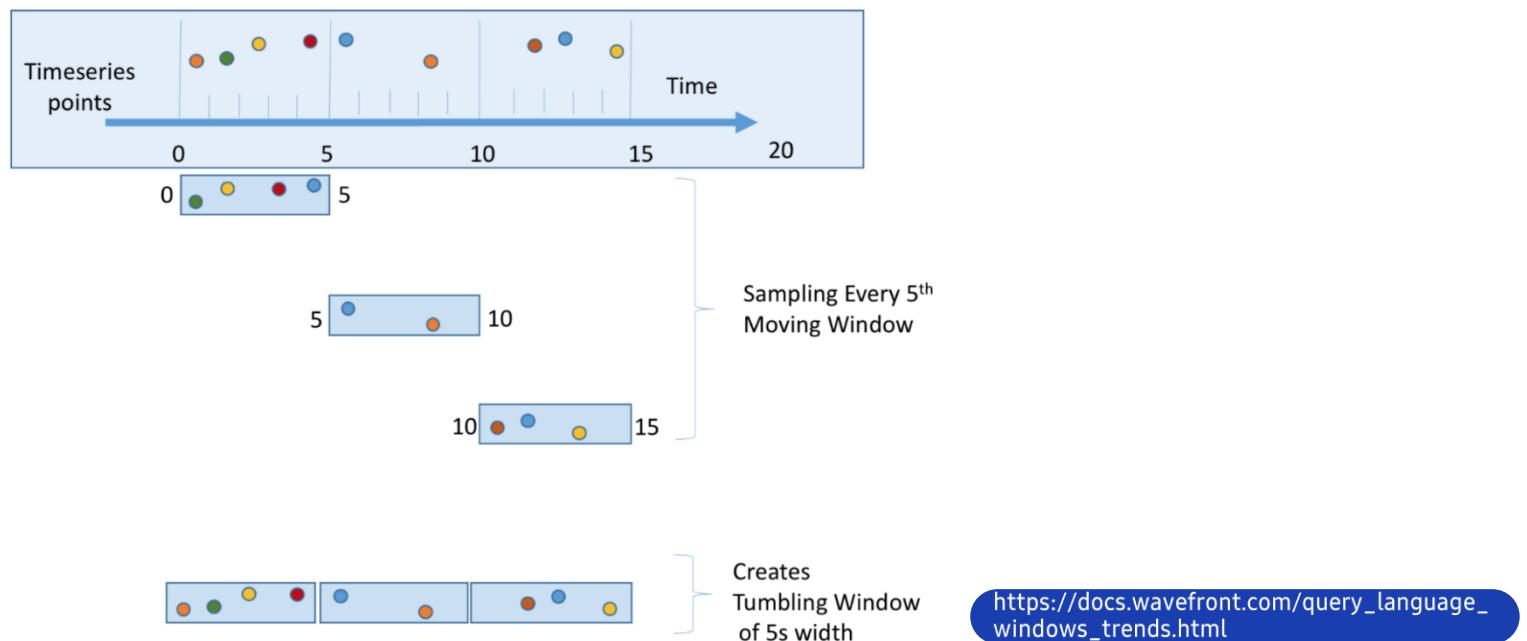
- See the data carefully. It automatically creates when each month actually starts and ends.

8. Moving Statistics Function (Rolling Window Calculations)

- | Pandas provides a function that can easily calculate the moving statistics (Rolling window) for DataFrame and Series.
- | The expression “window” refers to a period or section in which specific data is expressed.
- | It is a format in which the window is automatically moved according to the specified interval, the statistics are calculated accordingly, and the entire time series data is applied.

8. Moving Statistics Function (Rolling Window Calculations)

- To explain by referring to the image below, the moving average is the most used method for time series financial data analysis in order to smooth the short-term volatility of the target data and analyze long-term trends.



- TIP
- Smoothing means smooth processing by removing small fluctuations or discontinuities that are not good in the data due to noise when sampling from large sample data.

8. Moving Statistics Function (Rolling Window Calculations)

DataFrame rolling function: `pandas.DataFrame.rolling()`

- ▶ <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html>

Series rolling function: `pandas.Series.rolling()`

- ▶ <https://pandas.pydata.org/docs/reference/api/pandas.Series.rolling.html?highlight=series%20rolling#pandas.Series.rolling>

I Representative method

<code>.rolling().mean()</code>	Average in window
<code>.rolling().std()</code>	Standard deviation in window
<code>.rolling().var()</code>	Dispersion in window
<code>.rolling().sum()</code>	Total in window
<code>.rolling().min()</code>	Minimum value in window
<code>.rolling().max()</code>	Maximum value in window

8. Moving Statistics Function (Rolling Window Calculations)

| Let's practice the stock price data using the moving average concept as an example.

```
1 import pandas as pd
2 import datetime
3 from datetime import date, datetime, time
4 import matplotlib.pyplot as plt
5
6 import yfinance as yf
7 start = datetime(2020, 1, 1)
8 end = datetime(2020, 12, 31)
9
10 sec = yf.download(tickers=['005930.KS'],
11                     start=start,
12                     end=end)
13 sec.head()
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-01-02	55500.0	56000.0	55000.0	55200.0	50037.406250	12993228
2020-01-03	56000.0	56600.0	54900.0	55500.0	50309.347656	15422255
2020-01-06	54900.0	55600.0	54600.0	55500.0	50309.347656	10278951
2020-01-07	55700.0	56400.0	55600.0	55800.0	50581.289062	10009778
2020-01-08	56200.0	57400.0	55900.0	56800.0	51487.765625	23501171

8. Moving Statistics Function (Rolling Window Calculations)

| Let's practice the stock price data using the moving average concept as an example.

```
1 import pandas as pd
2 import datetime
3 from datetime import date, datetime, time
4 import matplotlib.pyplot as plt
5
6 import yfinance as yf
7 start = datetime(2020, 1, 1)
8 end = datetime(2020, 12, 31)
9
10 sec = yf.download(tickers=['005930.KS'],
11                   start=start,
12                   end=end)
13 sec.head()
```



Line 6, 7, 10

- 6: It is a Korean stock data set provided by Yahoo Finance in datareader.
- 7: If you want to practice with a wide window, it is better to specify a longer period.
- 10: Ticker code

8. Moving Statistics Function (Rolling Window Calculations)

| Let's practice the stock price data using the moving average concept as an example.

```
1 import pandas as pd
2 import datetime
3 from datetime import date, datetime, time
4 import matplotlib.pyplot as plt
5
6 import yfinance as yf
7 start = datetime(2020, 1, 1)
8 end = datetime(2020, 12, 31)
9
10 sec = yf.download(tickers=['005930.KS'],
11                   start=start,
12                   end=end)
13 sec.head()
```



Line 11, 12

- 11: Start date of search, use timestamp stored in start variable
- 12: Search end date, using the timestamp stored in the end variable

8. Moving Statistics Function (Rolling Window Calculations)

| Let's practice the stock price data using the moving average concept as an example.

1 sec.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 248 entries, 2020-01-02 to 2020-12-30
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  --  
 0   Open         248 non-null    float64 
 1   High         248 non-null    float64 
 2   Low          248 non-null    float64 
 3   Close        248 non-null    float64 
 4   Adj Close    248 non-null    float64 
 5   Volume       248 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 13.6 KB
```

 Line 1

- You can see that this data set is already in DatetimeIndex.

8. Moving Statistics Function (Rolling Window Calculations)

| Let's practice the stock price data using the moving average concept as an example.

```
1 ma5 = sec['Adj Close'].rolling(window=5).mean()  
2 ma5.head(10)
```

```
Date  
2020-01-02      NaN  
2020-01-03      NaN  
2020-01-06      NaN  
2020-01-07      NaN  
2020-01-08    50545.031250  
2020-01-09    51161.435156  
2020-01-10    51886.614844  
2020-01-13    52702.443750  
2020-01-14    53463.884375  
2020-01-15    53862.733594  
Name: Adj Close, dtype: float64
```



- Line 1
- It is easy to calculate the moving average data over 5 days.

8. Moving Statistics Function (Rolling Window Calculations)

| Let's visualize each moving average data.

```
1 close = sec['Adj Close']
2
3 ma5 = sec['Adj Close'].rolling(window=5).mean()
4 ma10 = sec['Adj Close'].rolling(window=10).mean()
5 ma60 = sec['Adj Close'].rolling(window=60).mean()
6 ma120 = sec['Adj Close'].rolling(window=120).mean()
7
8 plt.figure(figsize = (15,10))
9
10 plt.plot(close, label='close', linewidth=4)
11 plt.plot(ma5, label='5 window')
12 plt.plot(ma10, label='10 window')
13 plt.plot(ma60, label='60 window')
14 plt.plot(ma120, label='120 window')
15
16 plt.legend ()
17 plt.title('Rolling window calculations')
18 plt.xlabel('Date')
19 plt.ylabel('close price')

Text(0, 0.5, 'close price')
```

8. Moving Statistics Function (Rolling Window Calculations)

| Let's visualize each moving average data.

 Line 1, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14

- 1: To compare with the moving average in the graph, only the closing price is stored separately.
- 3: Average over Windows 5 days
- 4: Average over Windows 10 days
- 5: Average over Windows 60 days
- 6: Average over Windows 120 days
- 8: Set the size of the chart.
- 10: Express the closing price data as a line graph and name the label close. Since it is the reference data, it is expressed a little thicker.
- 11: Express the daily moving average data as a line graph and name the label 5 window.
- 12: Express the 10-day moving average data as a line graph and name the label 10 window.
- 13: Express the 60-day moving average data as a line graph and name the label 60 window.
- 14: Express the 120-day moving average data as a line graph and name the label 120 window.

8. Moving Statistics Function (Rolling Window Calculations)

The figure below is the result graph.



| Let's code

Step 1

| Data acquisition and data frame transformation

```
1 import pandas as pd
2 import numpy as np
3 import datetime
4 import matplotlib.pyplot as plt
5
6 from datetime import date, datetime, time, timezone
7
8 df = pd.read_csv("./data/fish/capture-fisheries-vs-aquaculture.csv")
9 df.head()
```

	Entity	Code	Year	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
0	Afghanistan	AFG	1969	60.0	400.0
1	Afghanistan	AFG	1970	60.0	400.0
2	Afghanistan	AFG	1971	60.0	500.0
3	Afghanistan	AFG	1972	60.0	500.0
4	Afghanistan	AFG	1973	60.0	500.0



Line 8

- <https://ourworldindata.org/fish-and-overfishing>

Step 1

| Data acquisition and data frame transformation

1 df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12727 entries, 0 to 12726
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Entity          12727 non-null   object 
 1   Code             11842 non-null   object 
 2   Year            12727 non-null   int64  
 3   Aquaculture production (metric tons) 9710 non-null   float64
 4   Capture fisheries production (metric tons) 12569 non-null   float64
dtypes: float64(2), int64(1), object(2)
memory usage: 497.3+ KB
```

Line 1

- Let's check the data type of each column of the data frame.

Step 2

Preprocessing including data cleaning

```
1 df.drop(['Code'], axis=1, inplace=True)  
2 df.head()
```

	Entity	Year	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
0	Afghanistan	1969	60.0	400.0
1	Afghanistan	1970	60.0	400.0
2	Afghanistan	1971	60.0	500.0
3	Afghanistan	1972	60.0	500.0
4	Afghanistan	1973	60.0	500.0



Line 1

- Delete unnecessary columns. Let's delete the code column in the statistics we want to do because we don't need it.

Step 2

I Preprocessing including data cleaning

```
1 # Search NaN data. You can check the number of missing data for each column.  
2 df.isnull().sum()
```

```
Entity          0  
Year            0  
Aquaculture production (metric tons)    3017  
Capture fisheries production (metric tons) 158  
dtype: int64
```

Step 2

| Preprocessing including data cleaning

```
1 change_value=0  
2 df.fillna(change_value, inplace=True)  
3 df.isnull().sum()
```

```
Entity          0  
Year           0  
Aquaculture production (metric tons) 0  
Capture fisheries production (metric tons) 0  
dtype: int64
```

Line 1~3

- 1: Create a variable with the data you want to replace.
- 2: Replace NaN with 0 in order not to affect the sum statistic.
- 3: When the substitution result was confirmed, all NaN data were substituted with 0, and there is currently no number of NaNs.

Step 2

- Change to time series data type and replace index

```
1 df['new_Year'] = pd.to_datetime(df['Year'].astype(str), format='%Y')
2 df.set_index('new_Year', inplace=True)
3 df.drop(['Year'], axis=1, inplace=True)
4
5 df.head()
```

	Entity	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
--	--------	--------------------------------------	--

`new_Year`

1969-01-01	Afghanistan	60.0	400.0
1970-01-01	Afghanistan	60.0	400.0
1971-01-01	Afghanistan	60.0	500.0
1972-01-01	Afghanistan	60.0	500.0
1973-01-01	Afghanistan	60.0	500.0



Line 1, 2

- 1: Year is an int64 type. An error occurs if you change it to datetime. It is necessary to change the data type.
- 2: Specify the new_Year column changed in the time series format as an index.

Step 2

- Change to time series data type and replace index

```
1 df['new_Year'] = pd.to_datetime(df['Year'].astype(str), format='%Y')
2 df.set_index('new_Year', inplace=True)
3 df.drop(['Year'], axis=1, inplace=True)
4
5 df.head()
```

	Entity	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
--	--------	--------------------------------------	--

new_Year	Entity	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
1969-01-01	Afghanistan	60.0	400.0
1970-01-01	Afghanistan	60.0	400.0
1971-01-01	Afghanistan	60.0	500.0
1972-01-01	Afghanistan	60.0	500.0
1973-01-01	Afghanistan	60.0	500.0



Line 3, 5

- 3: Delete the existing Year column because it is no longer needed.
- 5: Confirm the above processing result.

Step 2

| Change to time series data type and replace index

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 12727 entries, 1969-01-01 to 1969-01-01
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Entity          12727 non-null   object 
 1   Aquaculture production (metric tons) 12727 non-null   float64 
 2   Capture fisheries production (metric tons) 12727 non-null   float64 
dtypes: float64(2), object(1)
memory usage: 397.7+ KB
```



Line 1

- See the result, and you can confirm that the DatetimeIndex has been changed.

Step 2

- | Change to time series data type and replace index

```
1 new_df=df.sort_index()  
2 new_df.head()
```

	Entity	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
new_Year			
1960-01-01	Cayman Islands	0.0	0.0
1960-01-01	Bahrain	0.0	1500.0
1960-01-01	North Korea	5236.0	299190.0
1960-01-01	Northern Mariana Islands	0.0	100.0
1960-01-01	Norway	1900.0	1609362.0



- Line 1
- Sort dataframes based on set index.

Step 2

- Change to time series data type and replace index

```
1 new_df['Entity']=new_df['Entity'].astype('category')
2 new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 12727 entries, 1960-01-01 to 2018-01-01
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Entity            12727 non-null   category
 1   Aquaculture production (metric tons) 12727 non-null   float64 
 2   Capture fisheries production (metric tons) 12727 non-null   float64 
dtypes: category(1), float64(2)
memory usage: 333.1 KB
```

Line 2

- You can see that the data type for the country name has been changed from object to categorical.

Step 3

```
1 new_df['Entity'].value_counts()
```

```
Entity
Afghanistan      59
Portugal          59
Mozambique        59
Myanmar           59
Namibia           59
...
Montenegro       13
Saint Martin (French part) 12
Curacao          8
Sint Maarten (Dutch part)  8
South Sudan       7
Name: count, Length: 231, dtype: int64
```

 Line 1

- Let's check how many unique data there are in the column with country name.
There are data for a total of 231 countries.

Step 3

- | By adding up each country's catch and aquaculture production by year, we can visualize this to see trends in global catch and aquaculture.
- | If you see the data, there are separate data for each country and year. This can be solved by calculating the sum of the world (The sum of data from each country.) for each year through the group operation of pandas based on the year and visualizing this.

```
1 g = new_df.groupby(['new_Year'])
```

 Line 1

- Groups are grouped based on the new_Year column and stored them in a new data frame called g.

Step 3

```
1 g.head()
```

		Entity	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
		new_Year		
1960-01-01		Cayman Islands	0.00	0.0
1960-01-01		Bahrain	0.00	1500.0
1960-01-01		North Korea	5236.00	299190.0
1960-01-01	Northern Mariana Islands		0.00	100.0
1960-01-01		Norway	1900.00	1609362.0
...	
2018-01-01		Slovakia	2224.41	1937.0
2018-01-01		Cayman Islands	0.00	125.0
2018-01-01		Qatar	10.00	14669.0
2018-01-01		Romania	12298.45	11179.0
2018-01-01		Liberia	240.00	14115.0



Line 1

- If you check the saved result, you can see that the data frame was created based on the year.

Step 3

```
1 #  
2 for key, group in g:  
3     print('+key:', key)  
4     print('+number:', len(group))  
5     print(group.head())  
6     print('\n')  
  
+key: (Timestamp('1960-01-01 00:00:00'),)  
+number: 199  
          Entity Aquaculture production (metric tons) \\\n  
new_Year  
1960-01-01      Cayman Islands           0.0  
1960-01-01        Bahrain                0.0  
1960-01-01       North Korea             5236.0  
1960-01-01 Northern Mariana Islands      0.0  
1960-01-01         Norway                1900.0  
  
          Capture fisheries production (metric tons)\\n  
new_Year  
1960-01-01            0.0  
1960-01-01          1500.0  
1960-01-01        299190.0  
1960-01-01            0.0
```

 Line 1

- Print the contents of the g object using a loop.

Step 4

```
1 world_total = g.sum(numeric_only=True)  
2 world_total.head()
```

Aquaculture production (metric tons) Capture fisheries production (metric tons)

new_Year	Aquaculture production (metric tons)	Capture fisheries production (metric tons)
1960-01-01	10977127.0	161077154.0
1961-01-01	10322252.0	180573888.0
1962-01-01	10628086.0	198022581.0
1963-01-01	11882700.0	201874864.1
1964-01-01	12981280.0	226019748.1

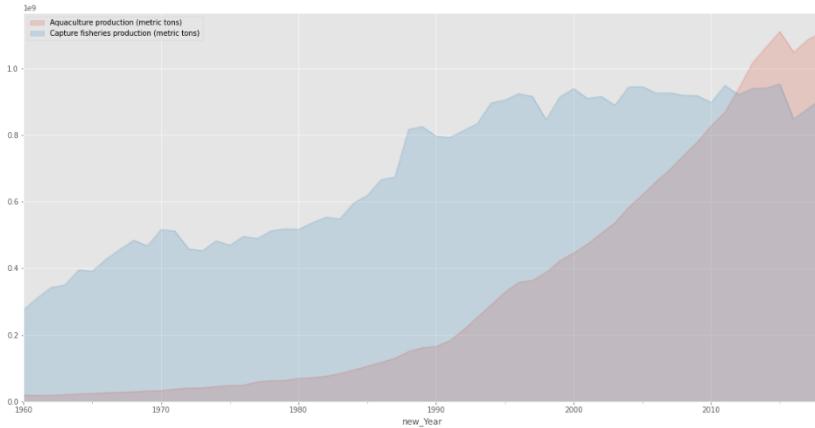
 Line 1

- Through group operations for each created group, the sum of each year is obtained and a new data frame is created.

Step 4

I Visualize global data

```
1 plt.style.use('ggplot')
2 world_total.plot(kind='area',
3                   alpha=0.2,
4                   stacked=False,
5                   figsize = (20, 10) )
6
7 plt.legend()
8 plt.show()
```



- ▶ Compare the graph we created with the graph we created from Our World data. You can make a result to the level that experts process and visualize.

Step 4

I Visualize global data

```
1 plt.style.use('ggplot')
2 world_total.plot(kind='area',
3                   alpha=0.2,
4                   stacked=False,
5                   figsize = (20, 10) )
6
7 plt.legend()
8 plt.show()
```



Line 1, 2, 3, 4, 5, 7, 8

- 1: Specify the style of the graph.
- 2: Draw an area graph in which the rest of the line graph is colored.
- 3: Adjust the opacity of the color to increase the visibility of overlapping graphs.
- 4: Select the option with False.
- 5: Specify the size of the graph.
- 7: Show legend.
- 8: If you check the results, you can see that the world has been increasing the amount of artificial aquaculture rather than the amount caught little by little starting in 2010.

Step 5

- | Let's search for a specific country name and visualize it. You can do this easily if you remember the practice of searching for artist names in the DataFrame lecture.

```
1 country = new_df['Entity'].value_counts()  
2 print(country)  
3 print("Data type =>", type(country))
```

Entity	count
Afghanistan	59
Portugal	59
Mozambique	59
Myanmar	59
Namibia	59
	..
Montenegro	13
Saint Martin (French part)	12
Curacao	8
Sint Maarten (Dutch part)	8
South Sudan	7
Name: count, Length: 231, dtype: int64	
Data type => <class 'pandas.core.series.Series'>	



Line 1~3

- 1: Create only non-duplicate names as series data in the country column to search for a country.
- 2: Print only non-duplicate country names.
- 3: If you check the processed data type, you can see that it has been successfully converted into a series.

Step 5

```
1 'South Korea' in country
```

True

 Line 1

- If it is True when the country name you want is searched, it means that there is data for the country.

Step 5

- | Search for 3 or more countries with different cultural and geographic requirements, visualize and get data insights.

```
1 s= new_df.loc[new_df['Entity']=='South Korea']
2 c= new_df.loc[new_df['Entity']=='China']
3 a= new_df.loc[new_df['Entity']=='Afghanistan']
```

Line 1~3

- 1: Create a separate data frame after searching for the country name you want in the series data created earlier.
- 2: Create a separate data frame after searching for the country name you want in the series data created earlier.
- 3: Create a separate data frame after searching for the country name you want in the series data created earlier.

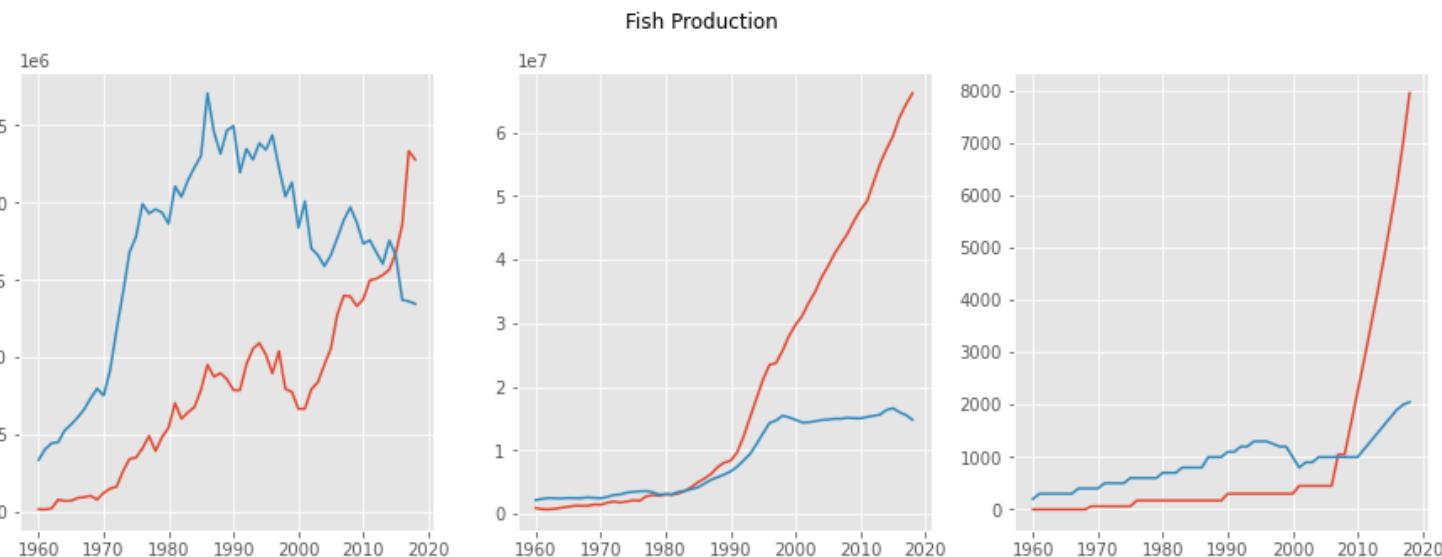
Step 5

```
1 s_y = s[['Aquaculture production (metric tons)', 'Capture fisheries production (metric tons)']]  
2 s_x = s.index  
3  
4 c_y = c[['Aquaculture production (metric tons)', 'Capture fisheries production (metric tons)']]  
5 c_x = c.index  
6  
7 a_y = a[['Aquaculture production (metric tons)', 'Capture fisheries production (metric tons)']]  
8 a_x = a.index
```

Step 5

```
1 import matplotlib.pyplot as plt
2 fig, axs = plt.subplots(1, 3, figsize= (15, 5))
3 axs[0].plot(s_x, s_y)
4 axs[1].plot(c_x, c_y)
5 axs[2].plot(a_x, a_y)
6
7 fig.suptitle('Fish Production')
```

Text(0.5, 0.98, 'Fish Production')



| 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. Discuss and practice the data you practiced in the key concept section of this lecture with your learning colleagues as shown below.

- | Change it to another company's data.
- | Try slicing based on the learning date.

Q2.

Access the University of California, Irvine, one of the open datasets used in the previous lecture, Data Organization Learning, and explore together with your learning colleagues which data is good data to utilize the advantages of time series data analysis.

A photograph of a person working at a desk. They are wearing an orange long-sleeved shirt and are holding a brown paper coffee cup with a black lid in their left hand. Their right hand is on a black computer keyboard. In the background, there are two computer monitors on stands. One monitor shows a dark screen with some white text or code. The other monitor is partially visible on the left. On the desk in front of the keyboard, there is an open book or manual with text and diagrams. A pair of glasses is resting on the desk to the left of the book.

End of
Document



Together for Tomorrow! Enabling People

Education for Future Generations

©2023 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this book other than the curriculum of Samsung Innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.