Hi everyone my name is 詹景淳you can call me Johnny, too. I am a TA (teaching assistant) for the Big Data Analytics course.

In today’s course, I’m going to introduce you how to use some simple methods to visualize data in Python.

If you have any questions during the course, please raise your hand immediately, I will explain in English again, if the explanation in English is not clear. I will use Chinese to explain for you, is there anyone who can not accept? Please raise your hand to let me know!

Thank you all for your cooperation, let's start the course.

----------------------------------------**P1**-------------------------------------------------------------

First, we will give you a brief introduction to what data visualization is, and then we will introduce the tools we need to use to visualize data.

Next is the implementation part. We will explain three simple visualization programs for you. They are word cloud, line chart, and heat map.

----------------------------------------**P2**-------------------------------------------------------------

The definition of visualization: the conversion of invisible data into concrete graphics.

So, when we visualizing data such as values, articles, or positioning information is called data visualization.

Data visualization usually uses statistical graphs, charts, or other tools.

We can use dots, lines, or bars to encode data to visually display quantitative information.

----------------------------------------**P3**-------------------------------------------------------------

You can see the following picture. This is a famous example of data visualization. In Chinese we call it 南丁葛爾玫瑰圖.

If you are interested, you can link the following URL to learn more.

----------------------------------------**P4**-------------------------------------------------------------

There are three main purposes of data visualization:

First is to overview, second is to Discover, and third is to Convey.

**Overview**: Confirm the distribution of data and conduct simple summaries to grasp the profile of the data.

**Discover**: Organizing the data can reveal parts that were previously unnoticed. Visualization of data makes it easier to detect features and identify key points. (After sorting out the data, you can find the parts that you haven't noticed before. Data visualization makes it easier for us to perceive the characteristics of the data and find the key points)

**Convey**: Presenting data analysis results and data characteristics in an appropriate manner allows people without relevant background to better understand the data.

----------------------------------------**P5**-------------------------------------------------------------

Now we are going to introduce you Jupyter Notebook.

Jupyter Notebook is a common data visualization tool.

Jupyter Notebook is an open-source integrated development environment built on web applications.

Allowing data scientists to write programs, display programs, and visualize output as if they were computing in a notebook file.

Jupyter Notebook will display the execution results below the code, so it is very suitable for beginners to practice data visualization.

In this course, we will introduce you how to use Jupyter Notebook to visualize data.

----------------------------------------**P6**-------------------------------------------------------------

First, we need to teach you how to open jupyter notebook.

You can search jupyter on windows, and the search result of Jupyter Notebook will pop up and click it to enter your local host in browser.

----------------------------------------**P7**-------------------------------------------------------------

After clicking Jupyter Notebook, the default browser will be launched.

If you want to change the browser, you can copy the URL in the Prompt of Jupyter Notebook (in the red box of the picture), and then paste the URL into the browser you want to use.

----------------------------------------**P8**-------------------------------------------------------------

After open your Jupyter Notebook, we need to create a new notebook file.

Select "Python 3" from the "New" drop-down menu in the upper right corner of the picture.

And you will successfully create a new notebook file.

----------------------------------------**P9**-------------------------------------------------------------

You can change file name by clicking "Untitled" on the right side of the Jupyter logo and enter your new file name, then click "Rename" button to confirm that the file name update is complete.

----------------------------------------**P10**------------------------------------------------------------

Now we are going to execute the program.

Please enter print("Hello World") in the cell.

You can Click "Run" to execute the program, or directly press "Shift" + "Enter" to execute.

----------------------------------------**P11**------------------------------------------------------------

Now please Upload the file we give you to Jupyter Notebook.

Click the red box in this picture to select all the files and upload them.

----------------------------------------**P12**---------------------------------------------------------

Assuming an error occurs when running the sample file, please confirm whether the necessary library is installed.

If it has not been installed, please start Anaconda Prompt to install it.

----------------------------------------**P13**------------------------------------------------------------

There are two ways to install packages.

First is enter **conda install + (library name)** to install the necessary packages.

If you can't install it with the conda command, change it to **pip install + (library name)**.

----------------------------------------**P14**------------------------------------------------------------

After finishing the basic introduction in Jupyter, we are going to enter the data visualization part.

We will teach everyone how to use crawler tools to crawl PTT website and draw the contents into a word cloud

So, what is word cloud? A word cloud is a collection, or cluster, of words depicted in different sizes.

The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is.

----------------------------------------**P15**------------------------------------------------------------

First, let's introduce the two libraries needed by web crawlers.

**1. requests** is used to send HTTP requests and get responses.

2. **BeautifulSoup** is used to analyze and crawl elements in HTML.

----------------------------------------**P16**------------------------------------------------------------

Next, we need to get Website Source Code.

We will set a variable url to store the website we want retrieved,

And then use the **get()** function to obtain the source code of the website (get website return information) and print out the crawled data in Jupyter Notebook.

You can use the URL we provide:

<https://www.ptt.cc/bbs/NBA/M.1635814687.A.AE4.html>

**(# text attributes are html files)**

----------------------------------------**P17**------------------------------------------------------------

The printed content will probably look like this:

----------------------------------------**P18**------------------------------------------------------------

Here is a hint. If the URL you enter has R-18 restrictions, you can use the following code to crawl pushes.

----------------------------------------**P19**---------------------------------------------------------

Our example needs to crawl out all the pushes in an article on PTT.

Since we need to crawl the PTT webpage later, we need to know the structure of the PTT website first.

We can press (F12, Developer Tools) to observe where ptt’s pushes will be placed.

We find that the pushes will be placed in the **<div class = “push“>**. (Red box in the picture on the righthand side)

And store the data in <div> tag into the variable response. (將<div>內資料存入變數response中)。

----------------------------------------**P20**------------------------------------------------------------

Here we will Use the **find\_all()** function to operate BeautifulSoup to retrieve all the contents in **<div class = “push“>** and print them out.

# lxml套件是用來作為BeautifulSoup的解析器，根據官方文件的推薦，我們使用解析速度最快的lxml

# The lxml suite is used as a parser for BeautifulSoup. According to the official documentation recommendations, we use the fastest parser lxml.

----------------------------------------**P21**------------------------------------------------------------

Print out all the contents in **<div class = “push“>** as shown below.

You can find that the <div> and <span> HTML tags are still mixed in with the printed data, so we need to remove them.

----------------------------------------**P22**------------------------------------------------------------

**We can u**se the **find()** function to retrieve the pushes, and we find that the pushes exist in the tag which name is <span>, <class ="f3 push-content">.

**find()** function helps to find the index of occurrence of the substring in the given string. It will return -1 if the substring is not present.

**getText()** function provides us to get text and catalog management.

----------------------------------------**P23**------------------------------------------------------------

After executing code on previous page the printed results are as shown below.

Although <div> and <span> are removed, But it still have colons in pushes, so we need to remove them either.

----------------------------------------**P24**------------------------------------------------------------

If we want to write a push without a colon in a file named Olympic2020.txt, we can use the **replace()** function to remove the colon, then use the **strip()** function to remove the spaces, and finally use **write()** towrite the sorted data into the file.

Find\_all returns all matched results, which is different from find (find only returns the first result found)

#Find\_all 返回所有匹配到的结果， find只返回查找到的第一个结果

#Find\_all returns all matched results, find returns only the first result found

----------------------------------------**P25**------------------------------------------------------------

The pushes in the PTT website were successfully saved without colons to Olympic2020.txt.

----------------------------------------**P26**------------------------------------------------------------

After crawling the data. Now, we are going to Start Drawing Word Cloud.

To draw a word cloud, we will use the **wordcloud** library.

**wordcloud** library can convert the strings of words separated by blank characters into single words, and then create a word cloud.

The picture below is the library needed to draw a word cloud.

# %matplotlib是Python內建的魔法函式，inline讓程式執行後印出的折線圖結果顯示在cell下方，不會另外跳出視窗顯示結果

# %matplotlib is a magic function built into Python. inline allows the result of the printed line chart to be displayed below the cell after the program is run, without popping up a separate window to display the result.

----------------------------------------**P27**------------------------------------------------------------

If library has not been installed, please type **(Exclamation Mark) !pip install (package name)** in Jupyter to install the library.

----------------------------------------**P28**---------------------------------------------------------

The parameter **width** in WordCloud can set the width, **height** can set the height, **background\_color** can set the background color of the word cloud, **font\_path** can specify the path of the font file.

**generate()** function will help us generate a word cloud.

#Use utf-8 to ensure that files are not read in garbled code.

----------------------------------------**P29**------------------------------------------------------------

The result after the above program is executed is as shown in the figure on right hand side.

The picture on the left is an article on PTT discussing the reasons for Taiwan's medal improvement in the Tokyo Olympics 2020.

The word clouds on the right are comments(thoughts) from netizens.

Most people believe that the record was broken because of the time difference, the government and the economic class.

----------------------------------------**P30**------------------------------------------------------------

Next, we will introduce how to change the mask image.

First, we need to prepare a silhouette picture.

Upload the mask image we provided to Jupyter Notebook.

Please note that the image format must be png and jpg.

In the following example, we use the heart image as a word cloud mask.

----------------------------------------**P31**------------------------------------------------------------

The parameter **mask** in **WordCloud** can specify a mask image.

In order to see the shape of the mask clearly, we can add a border to the mask.

At this time, we use the parameter **contour\_width** to set the thickness of the border, and the parameter **countou\_color** to determine the border color.

**colormap** can sets the color of text in the word cloud.

----------------------------------------**P32**------------------------------------------------------------

Use **imshow()** to display the word cloud.

Use **axis()** and set its parameter to "**off**" so that the x-axis and y-axis will not be displayed in the word cloud.

----------------------------------------**P33**------------------------------------------------------------

This is the Output after changing the mask image:

----------------------------------------**P34**------------------------------------------------------------

Now we have our first exercise.

1. Please find an article you are interested in on PTT and the number of pushes are greater than 50 to draw a word cloud.
2. Change the mask of word cloud above to a circle.

----------------------------------------**P35**------------------------------------------------------------

And this is example of exercise 1.

----------------------------------------**P36**------------------------------------------------------------

In the next section, we will introduce how to draw a temperature line chart by using weather data on the internet.

What is line chart? Define by wiki, A line chart (you can also call line plot or line graph or curve chart) is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments.

----------------------------------------**P37**------------------------------------------------------------

First, let's introduce the library needed to draw the line chart.

**Numpy** is a basic function library required for scientific operations using Python, including array operations that we will use later.

Generally speaking, we will invoke the Numpy function with np.

We can load Numpy with **import numpy as np**.

----------------------------------------**P38**------------------------------------------------------------

**matplot** is a classic library for visualizing data with Python, available for Python development environments such as Jupyter Notebook.

**seaborn** is a Python library for visualizing data based on matplotlib, which makes visualizing statistical data more beautiful.

----------------------------------------**P39**------------------------------------------------------------

**plotly** is a library that can draw dynamic charts and can also use preset values to beautify the chart.

----------------------------------------**P40**------------------------------------------------------------

The picture below is the library needed to draw a line chart.

If you want to import full library please check this page.

----------------------------------------**P41**------------------------------------------------------------

In order to draw a line chart, we first need to obtain weather data.

We provide you with the following two methods to obtain weather data from the website:

1. Click the download button on the website, download the file to your computer, and then upload the file to Jupyter notebook.
2. Use web crawlers to obtain website information and data.

You can use the URL we provide for implementation.

----------------------------------------**P42**------------------------------------------------------------

You can use the URL we provided on the previous page to link to the website and download the data.

Click the red box in this picture to download csv file.

----------------------------------------**P43**------------------------------------------------------------

Upload the downloaded file to Jupyter Notebook.

Click the red box in this picture to upload csv file.

----------------------------------------**P44**------------------------------------------------------------

Second method is web crawlers.

First of all, we need to print out the website information in jupyter to determine what data we need.

----------------------------------------**P45**------------------------------------------------------------

After executing the program described on the previous page, we can show the details of the webpage in Jupyter Notebook.

The weather data (temperature) we look for is in <table> and the table id is “**MyTable**”.

The picture below shows the details of the webpage.

----------------------------------------**P46**------------------------------------------------------------

<table> is used to wrap the structure and content of the entire table.

<tr> (table row) is used to define how many rows the table has, and <tr> contains <td> (table data) and <th> (table header).

<td> defines how many columns the table has. <th> defines a header cell in an HTML table.

----------------------------------------**P47**------------------------------------------------------------

Find the weather data we need, and we can recognize that they are all wrapped in <tr>, <td> and <th> label.

----------------------------------------**P48**------------------------------------------------------------

First, we create a new CSV file to write data in.

Search table ID=MyTable with BeautifulSoup.

Find all <tr> in <table ID=MyTable > to be written in CSV file we create.

----------------------------------------**P49**------------------------------------------------------------

We use the for loop to write all the weather data about <td><th> which are in <tr> tag into the csv file.

**append()** is used to add new objects at the end of the list.

**writerows()** write all the elements in rows into the CSV file and format it according to the current variables.

#The newline parameter will help us to automatically wrap lines as we write

----------------------------------------**P50**------------------------------------------------------------

After performing the above steps, we need to delete the data that may cause errors.

Please go to Jupyter's home page to find weather1.csv and open it.

----------------------------------------**P51**------------------------------------------------------------

Please delete data in the red box in the figure below.

----------------------------------------**P52**------------------------------------------------------------

After deleting, please press “**Ctrl**” + “**S**” to confirm all the data has been written to the CSV file successfully.

----------------------------------------**P53**------------------------------------------------------------

After completing the data crawling, we will next introduce how to draw the data into a line chart.

First, Read the “觀測時間(day)” column from weather1.csv.

“觀測時間(day)" is the key of column in weather1.csv.

Specify the **parse\_dates** parameter as “觀測時間(day)” to ensure that the line chart can be drawn with the timeline data.

If you do not perform this step, you will not be able to convert this data into timeline data when drawing line charts.

----------------------------------------**P54**------------------------------------------------------------

This picture show the result after performing the above steps.

----------------------------------------**P55**------------------------------------------------------------

The data type crawled by the web crawler may be object, which may cause the problem that the line chart cannot be drawn.

At this time, we need to manually force it to **int** or **float** data type.

----------------------------------------**P56**------------------------------------------------------------

From the figure above, we can see that the variables have been successfully transformed from object to int and float data types.

----------------------------------------**P57**------------------------------------------------------------

Now we can finally draw the line chart.

Line charts can be drawn using the **sns.lineplot()** function.

We can Specify the column of the horizontal axis in the parameter x, and specify the vertical axis field in the parameter y.

Because the minimum value of the horizontal axis cannot be 0 (our horizontal axis is “觀測時間(day)“), so we specify the value of the y axis.

----------------------------------------**P58**------------------------------------------------------------

The output after performing the previous steps.

We will get a temperature line chart.

----------------------------------------**P59**------------------------------------------------------------

If the string of the horizontal axis is too long, it may cause the scale to be covered, so we use the setting of **plt.xticks()** to make the scale of the horizontal axis become straight.

----------------------------------------**P60**------------------------------------------------------------

Then we introduce how to draw multiple line charts.

The first step is to create a data frame with “觀測時間(day) ” as the first column. So when we use the **read\_csv()** function to load the CSV file, we can specify **index\_col** parameter as 0 to create a data frame with the “觀測時間(day) ” column of the date data type as the index.

Load “氣溫", "最高氣溫", "最低氣溫" in weather\_index and print them out.

----------------------------------------**P61**------------------------------------------------------------

You can see that “觀測時間(day)” become the first column(that is, become an index) and is followed by “氣溫”、”最高氣溫”、”最低氣溫” columns.

----------------------------------------**P62**------------------------------------------------------------

If you want to display the legend of the chart somewhere, you can use the **loc (location)** parameter of the **legend()** function to specify the legend position.

----------------------------------------**P63**------------------------------------------------------------

As the figure you can see that there are three different types of line charts, so the next step is setting line chart to the same type.

----------------------------------------**P64**------------------------------------------------------------

We Put the attributes to be adjusted in the same column and adjust the format of the data (in this program is to be “氣溫種類”).

rename\_axis() function is used to rename the name of an index.

reset\_index() takes the current index and places it in the column.

We can use the rename() function to change the column name to "溫度(℃)".

----------------------------------------**P65**------------------------------------------------------------

This is the output after combining “氣溫”, “最高氣溫”, “最低氣溫” into a single column named “氣溫種類”.

----------------------------------------**P66**------------------------------------------------------------

We can use the parameter **hue** to specify the column to be drawn in different colors.

In this example, we specify “氣溫種類” to draw a line chart in different colors.

In this way the types of line chart will be the same, but the color is different.

----------------------------------------**P67**------------------------------------------------------------

Compared to the previous line chart, the type of line chart will be the same.

----------------------------------------**P68**------------------------------------------------------------

Next, we will introduce how to highlight one of the line charts. That is, how to change the color of the line chart.

First, we use the **len** function and the **unique** function to calculate the number of categories of "氣溫種類“. Set the highlight color. Set the number of the line chart to be changed and create the original line chart palette.

Use the parameter **palette** of the **sns.lineplot** function to change the color of the palette.

----------------------------------------**P69**------------------------------------------------------------

This picture is the whole steps of highlighting one of the line charts.

----------------------------------------**P70**------------------------------------------------------------

It’s the output after performing the previous page.

You can see that “氣溫(℃)" is highlighted in red.

----------------------------------------**P71**------------------------------------------------------------

Next, I will introduce another method of drawing a line chart, by using the plotly function.

If you use plotly to draw a line chart, the code can be very simple and the data can be displayed dynamically.

For example, when the mouse is moved on the line chart, the value at each point will be displayed.

We Use **px.line()** function to draw a line chart.

We Use **show()** function to display line chart.

----------------------------------------**P72**------------------------------------------------------------

When our mouse is moved on the line chart, the value at each point will be displayed.

In the figure “觀測時間" and "氣溫" is display on this line chart.

----------------------------------------**P73**------------------------------------------------------------

To use plotly to draw multiple line charts, you must define each line chart then gather the line charts in a single chart.

We Specify “**lines**” in the parameter **mode** of the **Scatter()** function to create line chart data of different temperatures.

Our X-axis is “觀測時間(day)”, and Y-axis are different types of temperature.

----------------------------------------**P74**------------------------------------------------------------

To combine multiple line charts into a single chart, we specify the parameter **data** of the **Figure()** function to combine data in different “氣溫種類” (tmp\_ave, tmp\_high, tmp\_low).

----------------------------------------**P75**------------------------------------------------------------

Figure below is the data output:

----------------------------------------**P76**------------------------------------------------------------

Draw the output data to multiple line chart by plotly.

----------------------------------------**P77**------------------------------------------------------------

Now we will teach you how to smooth the line chart.

Use the **scipy** library to smooth the line of the line chart.

The **linspace()** function in Numpy can spread points evenly within a certain range.

For example, if we want to generate 10 points between 1 and 5, our program can be written as follows.

----------------------------------------**P78**------------------------------------------------------------

Use **make\_interp\_spline()** to determine the coefficients of the spline to draw a smooth line chart.

We plot the curve by taking 300 equally spaced samples between the minimum and maximum values along the x-axis.

----------------------------------------**P79**------------------------------------------------------------

The scatter chart can be drawn with the **scatter()** function, the parameter x can specify the column of the horizontal axis, and the parameter y can specify the column of the vertical axis.

The parameter **color** can specify the color of the dots, and the parameter **alpha** can specify a value between 0 and 1. Make the color of the dots transparent.

The larger the value, the opaquer it is, and the smaller the value, the more transparent it is.

---------------------------------------**P80**------------------------------------------------------------

This picture is the whole steps of smoothing the line chart.

----------------------------------------**P81**------------------------------------------------------------

And it’s The output after smoothing the line chart.

----------------------------------------**P82**------------------------------------------------------------

Now we establish x-axis and y-axis matrix as "觀測時間(day) and "氣溫(℃)“ than smooth the data.

----------------------------------------**P83**------------------------------------------------------------

We can use the **title()** function to add a title to the line chart.

Use the **x.label()**, **y.label()** functions to add labels to the x and y axes.

----------------------------------------**P84**------------------------------------------------------------

This picture is the whole steps of smoothing the line chart which x-axis and y-axis matrix are "觀測時間(day) and "氣溫(℃)“.

----------------------------------------**P85**------------------------------------------------------------

This is the output after smoothing the line chart of weather data.

----------------------------------------**P86**------------------------------------------------------------

If you have encountered an abnormal display of Chinese, please go to the URL provided by us, and follow the instructions of the website to solve the abnormal display.

URL link: <https://www.getit01.com/p20190120250957403/>

----------------------------------------**P87**------------------------------------------------------------

Now we have second exercise for you.

1. Please get the weather data for any month and any region from the website we provide. Draw the "氣溫(℃)", "最高氣溫(℃)", and "最低氣溫(℃)" as a line chart and highlight the "最高氣溫(℃)".
2. Draw "氣溫(℃)", "最高氣溫(℃)", and "最低氣溫(℃)" as a line chart and smooth it.

----------------------------------------**P88**------------------------------------------------------------

This is an example of exercise 2.

----------------------------------------**P89**------------------------------------------------------------

In this section, we will introduce how to use Covid-19 related data to draw heat maps.

What is heat map? A heat map is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions.

----------------------------------------**P90**------------------------------------------------------------

First, we need to load the library we will use later.

----------------------------------------**P91**------------------------------------------------------------

First of all, we have to define the base map.

We use the parameter **location** of the **folium.map** function to specify the latitude and longitude of the map,

and the parameter **zoom\_start** is the zoom level of the map. The smaller the value, the larger the coverage of the map.

On the contrary, the larger the value, the smaller the coverage of the map.

----------------------------------------**P92**------------------------------------------------------------

Then use the **folium.Choropleth** function to draw a colored map of the administrative regions of the US states.

The **add\_map** function can help us add map color, map transparency, border color and border thickness to our **base\_map**.

**json** function is used to import data (because usually map files are json files).

----------------------------------------**P93**------------------------------------------------------------

This is the map of the United States after coloring.

----------------------------------------**P94**------------------------------------------------------------

Of course, just coloring the map is not enough. Next, we will introduce how to

draw a heat map based on number of cases of COVID-19 in U.S. states.

First, we need to create a csv file (named **covid\_usastate**) of the number of people diagnosed with covid-19 in each state in the United States. And use this file to draw a heat map.

Since this csv file was made a month ago, there may be little differences with the current figures, but the difference is not obvious if we drawn it as a heat map.

But you can also get current data from our link.

----------------------------------------**P95**------------------------------------------------------------

(When **folium** manipulates map information, you must first call the base map with the **folium.map** function, and then assign the data frame to the parameter **data** of the **folium.Choropleth** function.)

The **read\_csv** function can help us read CSV files stored in Jupyter.

Import **covid\_usastate.csv** with the **data** parameter to read the covid-19 confirmation cases for each state.

The **fill\_color** parameter is used to set the heat map palette.

----------------------------------------**P96**------------------------------------------------------------

The **columns** parameter is used to specify the name of the key (e.g. USA State, Total Cases) in the column where the heat map is drawn.

----------------------------------------**P97**------------------------------------------------------------

Figure below drawing a heat map based on the number of diagnoses in each state.

----------------------------------------**P98**------------------------------------------------------------

However, due to the uneven distribution of the population, the previous picture failed to show the actual severity of each state truthfully.

Therefore, we need to add the population of each state and convert the number of diagnosed cases into a ratio to increase the readability of the data.

So, we create another csv file (named **new\_covid\_usastate**), and draw the heat map with the ratio of diagnoses cases in each state as the data.

----------------------------------------**P99**------------------------------------------------------------

We reload the new data (**new\_covid\_usastate**), and then we modify the key in the **columns** parameter from **“Total Cases”** to **“Rate”**.

----------------------------------------**P100**----------------------------------------------------------

Figure below drawing a heat map based on the ratio of diagnoses in each state.

----------------------------------------**P101**----------------------------------------------------------

Now we have the third exercise for you.

1. Please go to the GitHub folder we provided, download, and upload all files to Jupyter Notebook at first.
2. Draw the number of death cases of COVID-19 in U.S. states as a heat map.
3. Draw the heat map of death rate with COVID-19 in U.S. states.

* GitHub link: <https://github.com/emotionmax/Python-Data-Visualization/tree/main/Data>
* Please draw a heat map based on the Death\_cases and Death\_rate in covid\_us\_states.csv according to the file we provided in GitHub.

----------------------------------------**P102**----------------------------------------------------------

We have an example about drawing the number of death cases of COVID-19 in U.S. states as heat map.

----------------------------------------**P103**----------------------------------------------------------

This figure is an example about drawing the heat map of death rate with COVID-19 in U.S. states.

----------------------------------------**P104**----------------------------------------------------------

There goes the final exercise.

* Search the information on the internet you are interested in and draw it to a heat map.
* For example, GDP per capita in various countries, global vaccine coverage, number of confirmed cases of COVID-19 in Japan, etc.
* There are no restrictions, you can do whatever you want.

----------------------------------------**P105**----------------------------------------------------------

This is the end of today’s course. Thank you for listening. Wish you have a nice day! Bye

----------------------------------------**P106**----------------------------------------------------------