Applied Tech Trial Class

May 14, 2020

1 Lesson 0: COVID-19 Outbreak Analysis

WARNING: The reference notebook is meant **ONLY** for a teacher. Please **DON'T** share it with any student.

The contents of the reference notebook are meant only to prepare a teacher for a class. To conduct the class, use the class copy of the reference notebook. The link and the instructions for the same are provided in the **Notes To The Teacher** section.

Particulars	Description
Topic	COVID-19 Outbreak Analysis
Class Description	A student learns to create a live dashboard to monitor the spread of the coronavirus diseases across the globe
Class	Trial
Class Time	45 minutes
Goal	Create a live dashboard to monitor the spread of the coronavirus diseases across the globe Find the countries that have controlled the spread of COVID-19 or flattened the curve Interpret the results and draw conclusions
Teacher Resources	Link to the John Hopkins University coronavirus live dashboard Link to the trial class Google Colab reference notebook Google Account Laptop with internet connectivity Earphones with mic
Student Resources	Link to the John Hopkins University coronavirus live dashboard Google Account Laptop with internet connectivity

Particulars	Description
	Earphones with mic

1.0.1 Class Structure

A class is divided into three parts, as shown in the table below.

Parts	Duration
Warm-Up	9 minutes
Teacher-Student Activities	30 minutes
Wrap-Up	6 minutes

1.0.2 Notes To The Teacher

- For the best class experience, use this Colab notebook in the dark mode.
    1. Click on the settings icon.
    2. A new dialogue box will appear. Click on the tiny downward arrow next to the **light** theme.
    3. Choose the **dark** theme option and then click on the **SAVE** button.
- Before beginning the class:
    1. Open the **AT - Lesson 0 (Trial Class) - Class Copy - v0.9** file by clicking on the link provided in the **Activities** section under the title **COVID-19 Outbreak Analysis (Class Copy)**.
    2. After opening the file mentioned in the first point, create its duplicate copy by following the steps described below:

    – Click on the **File** menu. A new drop-down list will appear.

    – Click on the **Save a copy in Drive** option. A duplicate copy will get created. It will open up in the new tab on your web browser.

    – In the duplicate copy, click on the **Share** button on the top right corner of the notebook. A new dialogue box will appear.

    – Click on the tiny downward arrow next to **Anyone with the link can view** text. A drop-down list will appear.

    – Click on the **More...** option. A new page on the dialogue box will appear.

    – Click on the circle next to **On - Anyone with the link** option. Now, go down and click on the tiny downward box next to **Can edit** text. A drop-down list will appear.

- Click on the **Can edit** option. Then click on the **Save** button. You will be directed back to the first page of the dialogue box.
- Make sure that under the **Link sharing on** section, **Anyone with the link can edit** option is selected.
- Then click on the **Done** button.

3. After creating the duplicate copy of the notebook, please rename it in the **YYYY-MM-DD_StudentName_Lesson0** format. This step is strongly recommended because in future you will conduct the same class for many students on the same day. For each student, you will have to create a duplicate copy of the original class copy. Hence, this step will allow you to organise each duplicate copy for each student in your Google drive.

- Now, finish the **Warm-Up** section. Afterwards, proceed with the class using the duplicate copy of the notebook. You can share the link of the duplicate copy (named as **YYYY-MM-DD_StudentName_Lesson0**) with the student through the chat window.
- When a student executes the code for the first time in the duplicate copy that you shared with them, they may get the following warning:

```
Warning: This notebook was not authored by Google. This notebook was
authored by xyz@gmail.com. It may request access to your data stored
with Google such as files, emails and contacts. Please review the source
code and contact the creator of this notebook at xyz@gmail.com with any
additional questions.

                                                    Cancel. Run Anyway.
```

- Ask the student to click on the `Run Anyway` button.
- If you see the hat ($\hat{}$) sign on the heading of an activity, give a hats-off to the student at the end of the activity.
  - Single hat sign, i.e., $\hat{}$ denotes give hats-off for concentration. You have to assess the concentration level of a student based on their attentiveness and understanding of the concept.
  - Double hat signs, i.e., $\hat{}\hat{}$ denotes give hats-off for creativity. It can be an alternative approach to writing a code for an activity or applying the logic in a totally different problem statement.
  - Triple hat signs, i.e., $\hat{}\hat{}\hat{}$ denotes give hats-off for persistence. It is the ability of a student to not give up on writing code and writing code with minimal teacher support.
- Every time you execute your code in the Google Colab notebook, please don't forget to save it by pressing the **Ctrl + S** keys (or **Command + S** keys if you are using a Mac). While saving the notebook, you or student may encounter the following error:

```
Save failed

The notebook has been changed outside of this session. Would you like to
overwrite existing changes?
```

CANCEL YES

Click on the YES button.

- Occasionally, you may encounter some error such as `NameError, ReferenceError, ImportError` or `ModuleNotFoundError` after executing the code in k^{th} code cell (where $k > 0$). Most likely, due to poor internet speed, the Colab notebook might have lost the information that all the previous codes have been executed already. As a remedy, run codes in the code cells beginning from the first code cell till the $(k-1)^{th}$ code cell. For e.g., if the code in the 5^{th} code cell fails to run because of one of the aforementioned errors, execute the codes in all the first four code cells again.
- For every **Teacher Action**, the teacher is supposed to share their screen with the student. Similarly, for every **Student Action**, the student is supposed to share their screen with the teacher.

1.0.3 Warm-Up

```
TEACHER
  Hi <student_name>! My name is <teacher_name>.
  I am going to be your teacher for this class. Let's get to know each
  other a little bit before we start. Why don't you tell me what are
  your hobbies or what do you like to do in your free time?"

  Notes:
    - Get to know the student - his/her name, what do they like to do in
      their free time, what are their ambitions etc.
    - Get to know the student's grade. If the student is in 11th or
      12th standard, then ask their stream (Maths/Biology/Arts/Commerce).
      Depending on their streams connect the relevance of this course with
      the streams.
    - Get to know the school board (CBSE/ICSE/HSC etc.) of the student.

EXPECTED STUDENT RESPONSE
  Hello teacher! My name is <student_name> I am doing good. I love playing
  football, reading books and watching movies. I want to be a politician
  someday.


TEACHER
  That's great! I love reading too. Do you know that politicians use a lot of
  data to analyse the situation in their constituency? For e.g., using data
  they find out what is their vote bank (who is likely to vote for them).
  Another example is using data they find what is the crime rate in a city,
  how many people are employed or unemployed etc.

  Notes:
    - Connect the relevance of data with the student's stream and with their
```

ambition by quoting a few examples.

Not just politics, data is important in every field whether its science & engineering, humanities, arts, medical science, pharmacy, finance, social (or political) science etc. to solve real-life problems.

EXPECTED STUDENT RESPONSE
Wow! This is interesting. I didn't know that data is that important

TEACHER
Yes. Data is very very important in this new age of technology. So to understand the criticality of data, today in this trial class we are going to observe the trend of spread of coronavirus across the world. I hope you are aware of the coronavirus crisis that has shaken the world.

EXPECTED STUDENT RESPONSE
Yes, teacher. I am aware of the coronavirus situation. It is also spreading in India rapidly.

TEACHER
Absolutely. Today you will learn how using the data you can interpret why the situation is alarming and what can happen in future if we don't take the right measures to tackle this problem and why social isolation is important in cases of spread of diseases like COVID-19 (shorthand for Coronavirus Disease - 2019)

So, are you excited about this class?

EXPECTED STUDENT RESPONSE
Yes. I am very excited.

TEACHER
Great. Let's get started. Now, I am going to share a link with you. Open the link in the new tab of your web browser.

Note:
- Share the duplicate copy of the Colab notebook that you created with the student through the chat window.

EXPECTED STUDENT RESPONSE:
Student opens the duplicate copy of the Colab notebook.

TEACHER
So, this is a Colab notebook developed by Google. We will write Python

```
  codes in a Google Colab notebook which is actually a Jupyter notebook. It
  is just a software to write Python codes. Many data analysts, data
  scientists, artificial intelligence engineers use Jupyter notebooks to
  solve data-related problems.

  Google Colab allows multiple users to work on the same Jupyter notebook
  simultaneously. It is loaded with all the tools required to solve data and
  AI-based problems. Hence, you don't have to install each tool manually.

  So, let's continue the class.

EXPECTED STUDENT RESPONSE:
  The student follows the teacher's lead.
```

1.0.4 Overview

In a tweet on 11 March 2020, WHO declared COVID-19 (or coronavirus) a pandemic. A pandemic is a disease spread over the world.

Here's a link to the tweet: https://twitter.com/WHO/status/1237777021742338049

Coronavirus has claimed the lives of more than **300 thousand** (or 3 lakh) people globally so far and is still counting. You can look at the live dashboard to see the real-time updates.

COVID-19 Live Dashboard

In line with this pandemic, in this class, to understand the virality of the disease, we are going to look at

- How many people get affected by coronavirus every day?
- What is the distribution of the number of people affected across the globe?
- Which countries have flattened the curve? In other words, which countries have controlled the rate of spread of COVID-19?

1.0.5 What Will You Create In This Class?

- Cartograms using Folium Maps
- Line Plots
- Bivariate Bar Charts

1.0.6 Teacher-Student Activities

Now, let's create the required line plots and cartograms. Here's the list of activities that we are going to perform in this class:

1. Download the required data
2. Locate the datasets in the notebook
3. Import the required modules
4. Create DataFrames
5. Create Pandas series
6. Create line plots/charts
7. Create maps
8. Create bar charts

Activity 1: Download The Datasets Here's the link to the data source. It is a GitHub repository. Coders/programmers across the world create GitHub repositories for their codes so that they can download and upload their work anytime from anywhere. Essentially, it's like carrying data in a portable storage device.

COVID-19 Data Source

[1]:
```
# Student Action: Run the code below by either clicking on the play button or␣
↪by hitting the 'shift + enter' keys together.
# Cloning the GitHub repository.
!git clone https://github.com/CSSEGISandData/COVID-19.git
```

```
fatal: destination path 'COVID-19' already exists and is not an empty directory.
```

The repository is saved in the notebook. You can click on the folder icon on the left-hand side of the notebook to access the `COVID-19` repository (or folder)

Activity 2: Dataset Paths To get the path of the datasets:

1. Click on the folder icon on the left-hand side of the notebook.
2. Click on the tiny arrow on the left-hand side of the `COVID-19` folder.
3. Click on the tiny arrow in the left-hand side of the `csse_covid_19_data` folder.
4. Click on the tiny arrow in the left-hand side of the `csse_covid_19_time_series` folder.

   You should be able to see the CSV files which contain data for this class.
5. Right-click on each of the CSV files and then click on the **Copy path** option. The path of the file will get copied.
6. You can now paste the link either in a code cell or in a text cell.

Note:

The files below are no longer available:

1. `time_series_19-covid-Confirmed.csv`
2. `time_series_19-covid-Deaths.csv`
3. `time_series_19-covid-Recovered.csv`

Hence, refer to the following two files for the latest time-series data:

1. `time_series_covid19_confirmed_global.csv`
2. `time_series_covid19_deaths_global.csv`
3. `time_series_covid19_recovered_global.csv`

[0]:
```
# Student Action: Run the code below.
# Storing the path of 'time_series_covid19_confirmed_global.csv'' in 'conf_csv'␣
 ↪variable.
conf_csv = '/content/COVID-19/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_confirmed_global.csv'
# Storing the path of 'time_series_covid19_deaths_global.csv'' in 'deaths_csv'␣
 ↪variable.
deaths_csv = '/content/COVID-19/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_deaths_global.csv'
# Storing the path of 'time_series_covid19_recovered_global.csv' in 'rec_csv'␣
 ↪variable.
rec_csv = '/content/COVID-19/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_recovered_global.csv'
```

This data gets updated every day. Today it has data till yesterday. Tomorrow, it will contain data for this day as well.

Activity 3: Importing Modules To perform different kinds of tasks, we can use different modules in Python.

[3]:
```
# Student Action: Run the code below.
import pandas as pd # Data processing
import matplotlib.pyplot as plt # Data visualisation
import seaborn as sns # Data visualisation
import folium # Cartograms / maps
import datetime # Work with date and time values

%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Activity 4: Create DataFrames & Series Let's create a DataFrames for the total confirmed coronavirus cases across the globe.

A DataFrame is a tabular representation of data containing many rows and columns.

[4]:
```
# Student Action: Run the code below.
# DataFrame for the total confirmed cases.
conf_df = pd.read_csv(conf_csv)
# Display the first five rows of the 'conf_df' DataFrame.
conf_df.head()
```

[4]:

	Province/State	Country/Region	Lat	…	5/11/20	5/12/20	5/13/20
0	NaN	Afghanistan	33.0000	…	4687	4963	5226
1	NaN	Albania	41.1533	…	872	876	880
2	NaN	Algeria	28.0339	…	5891	6067	6253
3	NaN	Andorra	42.5063	…	755	758	760
4	NaN	Angola	-11.2027	…	45	45	45

[5 rows x 117 columns]

As you can see, we have data for the total confirmed cases of coronavirus reported till yesterday across the globe.

Let's find out how many rows and columns are there in the `conf_df` DataFrame.

[5]:
```
# Student Action: Run the code below.
# Number of rows and columns in the 'conf_df' DataFrame.
conf_df.shape
```

[5]: (267, 117)

Now, let's create a series of the total confirmed cases of coronavirus reported:

- Across globe
- In China
- In India
- In the US

[6]:
```
# Student Action: Run the code below.
# Total confirmed cases reported across the globe.
global_cases = conf_df.iloc[:, 4:].apply(sum, axis=0)
# Converting the indices to datetime values.
global_cases.index = pd.to_datetime(global_cases.index) # The datetime value is␣
↪formatted in the yyyy-mm-dd format.
global_cases
```

[6]:
```
2020-01-22      555
2020-01-23      654
```

```
2020-01-24         941
2020-01-25        1434
2020-01-26        2118
                   …
2020-05-09     4024009
2020-05-10     4101699
2020-05-11     4177502
2020-05-12     4261747
2020-05-13     4347018
Length: 113, dtype: int64
```

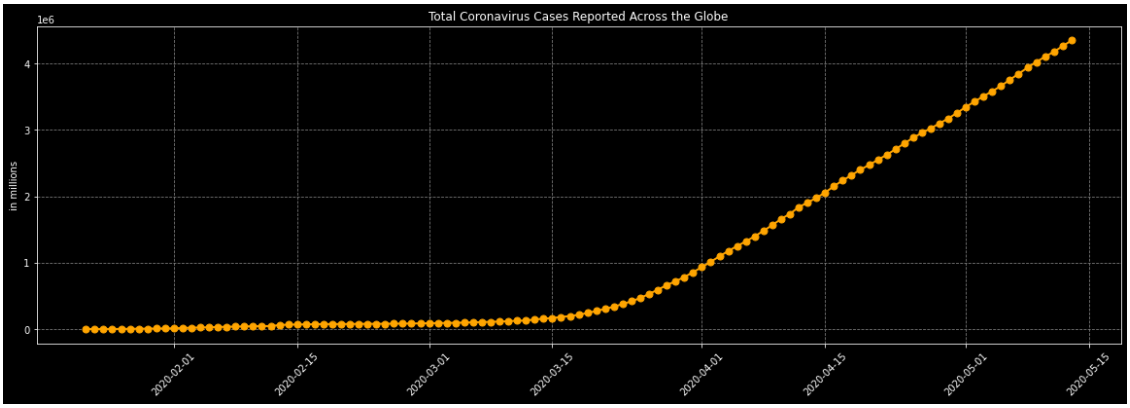
[0]:
```
# Student Action: Run the code below.
# Total confirmed cases in China.
china_cases = conf_df[conf_df['Country/Region'] == 'China'].iloc[:, 4:].
 ↪apply(sum, axis=0)
china_cases.index = pd.to_datetime(china_cases.index) # The datetime value is␣
 ↪formatted in the yyyy-mm-dd format.

# Total confirmed cases in India.
india_cases = conf_df[conf_df['Country/Region'] == 'India'].iloc[:, 4:].
 ↪apply(sum, axis=0)
india_cases.index = pd.to_datetime(india_cases.index)

# Total confirmed cases in the US.
us_cases = conf_df[conf_df['Country/Region'] == 'US'].iloc[:, 4:].apply(sum,␣
 ↪axis=0)
us_cases.index = pd.to_datetime(us_cases.index)
```

Activity 5: Line Plots Now, we will create line plots for the total number of confirmed cases reported

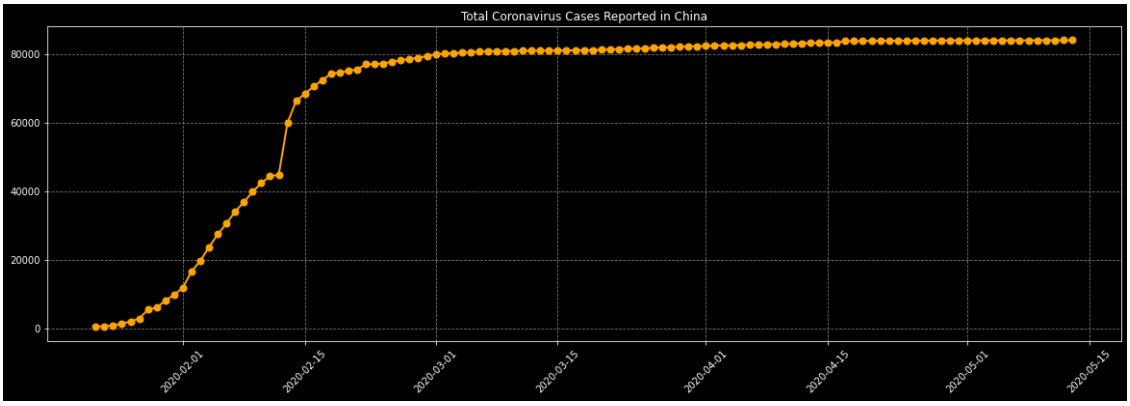
- Across world
- In China
- In India
- In the US

[8]:
```
# Student Action: Run the code below.
# Line plot for the total number of coronavirus confirmed cases reported across␣
 ↪the world.
plt.style.use('dark_background')
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported Across the Globe')
plt.plot(global_cases.index, global_cases, c='orange', linewidth=2, marker='o',␣
 ↪markersize=7)
```

```
plt.xticks(rotation=45)
plt.ylabel('in millions')
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```

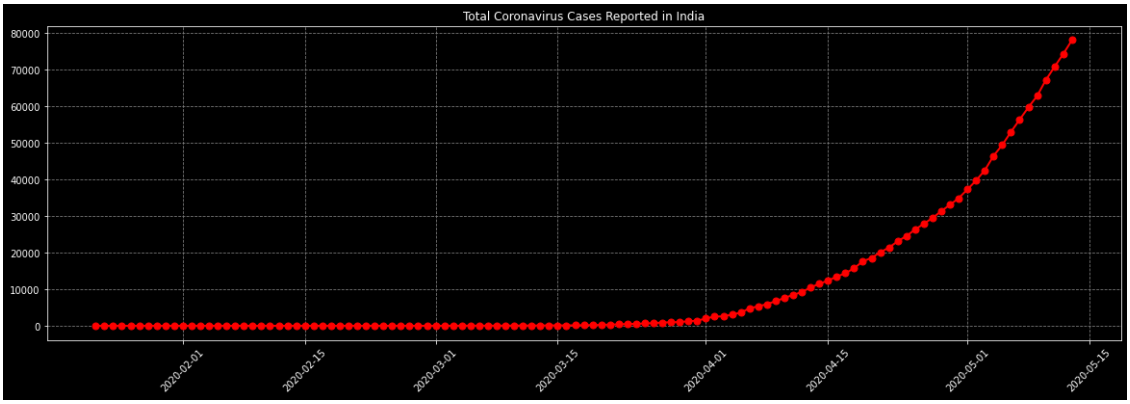


[9]:
```
# Student Action: Run the code below.
# Line plot for the total number of coronavirus confirmed cases reported in␣
 ↪China only.
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported in China')
plt.plot(china_cases.index, china_cases, c='orange', linewidth=2, marker='o',␣
 ↪markersize=7)
plt.xticks(rotation=45)
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```



The curve flattens after 23 February 2020 because Wuhan (the capital of Hubei province in China) was put under total lockdown on 23 January 2020. As a result, the spread of the virus slowed down significantly.

[10]:
```
# Student Action: Run the code below.
# Line plot for the total number of coronavirus confirmed cases reported in␣
 ↪India.
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported in India')
plt.plot(india_cases.index, india_cases, c='r', linewidth=2, marker='o',␣
 ↪markersize=7)
plt.xticks(rotation=45)
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```

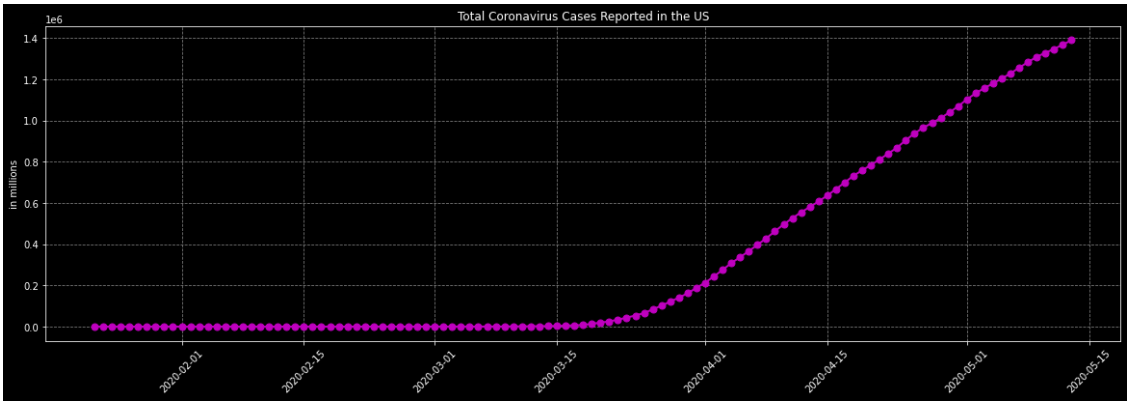


India started testing for coronavirus in bulk in the last week of March 2020.

If you want to look at the trend of the coronavirus outbreak in India across all the states, then click on the link provided below.

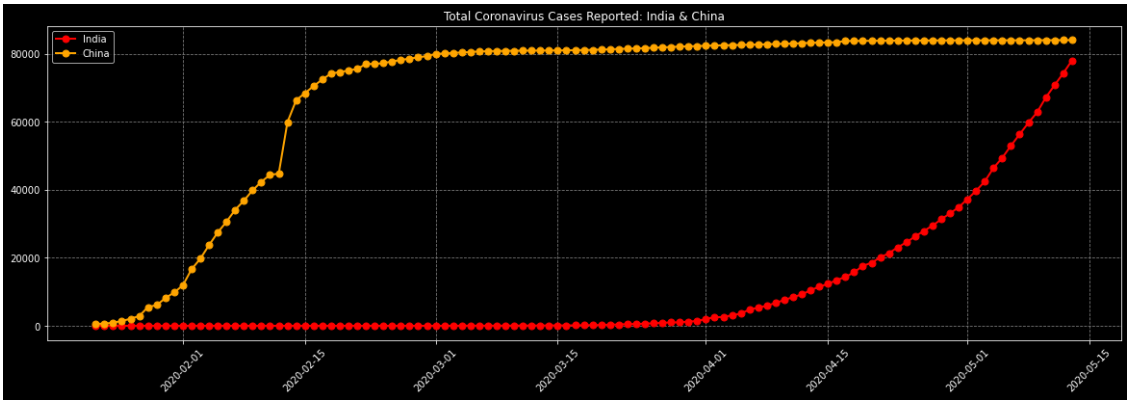
Ministry Of Health & Family Welfare

[30]:
```
# Student Action: Run the code below.
# Line plot for the total number of coronavirus confirmed cases reported in the␣
 ↪US.
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported in the US')
plt.plot(us_cases.index, us_cases, c='m', linewidth=2, marker='o', markersize=7)
plt.xticks(rotation=45)
plt.ylabel("in millions")
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```



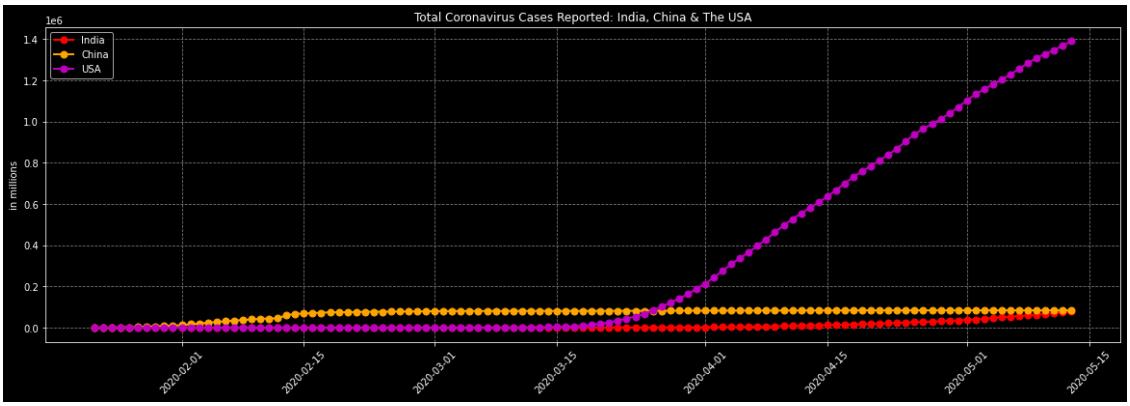
The USA started testing for coronavirus in bulk after March 15, 2020.

[12]:
```
# Student Action: Run the code below.
# Line plot to compare the total number of coronavirus confirmed cases reported␣
 ↪in India and China.
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported: India & China')
plt.plot(india_cases.index, india_cases, c='red', linewidth=2, marker='o',␣
 ↪markersize=7, label='India')
plt.plot(china_cases.index, china_cases, c='orange', linewidth=2, marker='o',␣
 ↪markersize=7, label='China')
plt.xticks(rotation=45)
plt.grid(True, 'major', linestyle='--', c='grey')
plt.legend()
plt.show()
```



[31]:
```
# Student Action: Run the code below.
# Line plot to compare the total number of coronavirus confirmed cases reported␣
 ↪in India, China & the USA.
```

```
plt.figure(figsize=(20, 6))
plt.title('Total Coronavirus Cases Reported: India, China & The USA')
plt.plot(india_cases.index, india_cases, c='red', linewidth=2, marker='o',␣
 ↪markersize=7, label='India')
plt.plot(china_cases.index, china_cases, c='orange', linewidth=2, marker='o',␣
 ↪markersize=7, label='China')
plt.plot(us_cases.index, us_cases, c='m', linewidth=2, marker='o',␣
 ↪markersize=7, label='USA')
plt.xticks(rotation=45)
plt.ylabel('in millions')
plt.grid(True, 'major', linestyle='--', c='grey')
plt.legend()
plt.show()
```



At the beginning of the last week of March, the total confirmed cases of coronavirus in the USA surpassed the same in China.

Activity 6: Grouping Let's identify the top 5 countries having the most number of confirmed coronavirus cases and visualise their trends starting from March 15, 2020 because that's when the USA started testing in bulk. They have recorded the most number of coronavirus confirmed cases till date.

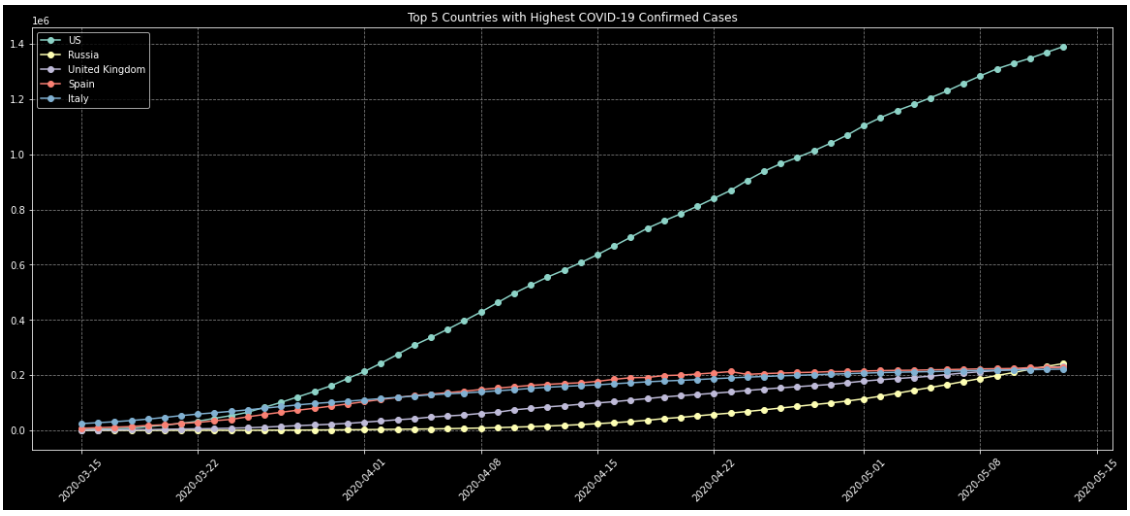
[14]:
```
# Student Action: Run the code below.
# The grouped DataFrame having the total coronavirus confirmed cases across the␣
 ↪globe in descending order.
grouped_conf_df = conf_df.groupby(by='Country/Region', as_index=False).sum()
desc_grp_conf_df = grouped_conf_df.sort_values(by=conf_df.columns[-1],␣
 ↪ascending=False)
desc_grp_conf_df.head()
```

[14]:

	Country/Region	Lat	Long	…	5/11/20	5/12/20	5/13/20
174	US	37.0902	-95.7129	…	1347881	1369376	1390406
140	Russia	60.0000	90.0000	…	221344	232243	242271
178	United Kingdom	270.0299	-482.9247	…	224332	227741	230985
158	Spain	40.0000	-4.0000	…	227436	228030	228691
85	Italy	43.0000	12.0000	…	219814	221216	222104

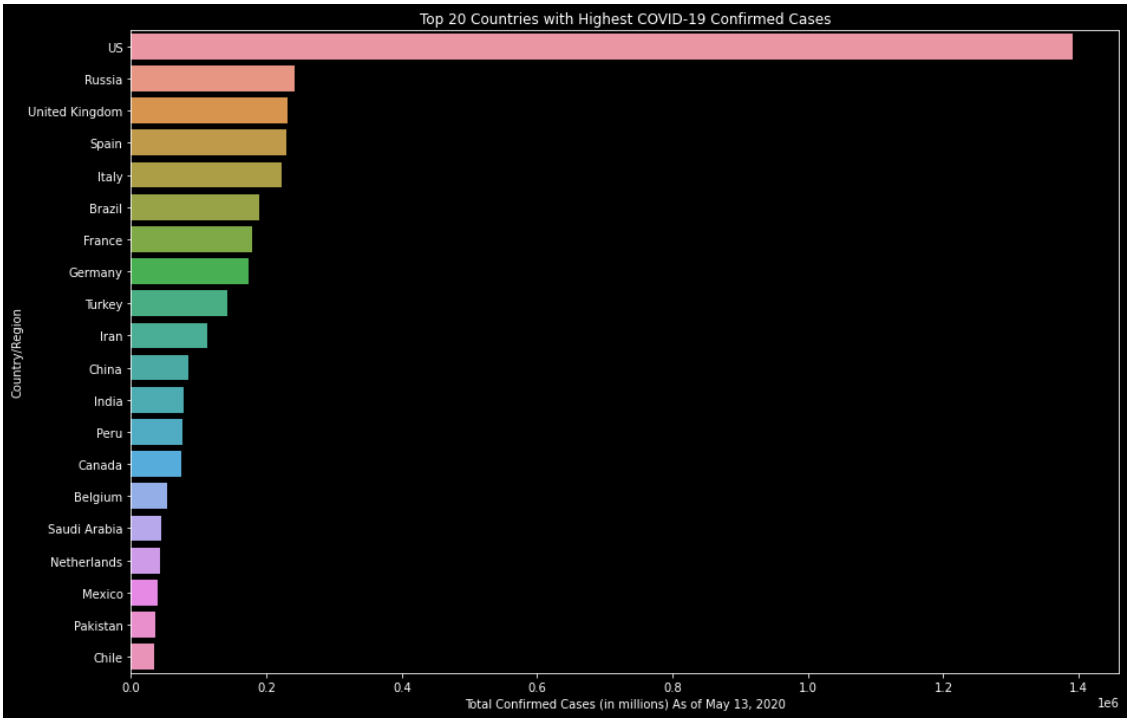
[5 rows x 116 columns]

[15]:
```
# Student Action: Run the code below.
# Top 5 countries with the highest number of coronavirus confirmed cases␣
 ↪starting from March 15, 2020.
plt.figure(figsize=(20, 8))
plt.title('Top 5 Countries with Highest COVID-19 Confirmed Cases')
for region in desc_grp_conf_df.head()['Country/Region']:
  # conf_series = high_to_low_conf_df.loc[high_to_low_conf_df['Country/Region']␣
 ↪== region, high_to_low_conf_df.columns[3:]].values[0]
  plt.plot(global_cases.index[53:], desc_grp_conf_df[desc_grp_conf_df['Country/
 ↪Region'] == region].iloc[0, 56:], '-o', label=region)
plt.xticks(rotation=45)
plt.legend()
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```



Activity 7: Bivariate Bar Plots Let's create a bar chart displaying the top 20 countries having the most number of coronavirus confirmed cases.

[32]:
```
# Student Action: Run the code below.
# Get the latest date.
last_col = conf_df.columns[-1]
# Modify the latest date in the 'Month DD, YYYY' format.
latest_date = datetime.datetime.strptime(last_col, '%m/%d/%y').strftime('%B %d,␣
 ↪%Y')

# Create a bivariate bar chart displaying the top 20 countries having the most␣
 ↪number of coronavirus confirmed cases.
plt.figure(figsize=(15, 10))
plt.title('Top 20 Countries with Highest COVID-19 Confirmed Cases')
sns.barplot(desc_grp_conf_df[last_col].head(20), desc_grp_conf_df['Country/
 ↪Region'].head(20), orient='h')
plt.xlabel(f'Total Confirmed Cases (in millions) As of {latest_date}')
plt.show()
```



Activity 8: Cartograms (or Maps) For China^ Let's create a cartogram to show the distribution of confirmed coronavirus cases in China and mark the affected regions of China with location markers.

The markers will display the name of the region location along with the number of confirmed coronavirus cases in that region.

[17]:
```
# Student Action: Run the code below.
# The DataFrame for the total confirmed cases in China only.
conf_china_df = conf_df[conf_df['Country/Region'] == 'China']

# Map to show the distribution of confirmed coronavirus cases in China (regular␣
 ↪markers).
china_map = folium.Map(location=[30.9756, 112.2707], width='100%',␣
 ↪height='90%', tiles='Stamen Toner', zoom_start=4.25)
for i in conf_china_df.index:
  folium.Marker(location=[conf_china_df.loc[i, 'Lat'], conf_china_df.loc[i,␣
 ↪'Long']],
                popup= conf_china_df.loc[i, 'Province/State'] + "\n" +␣
 ↪str(conf_china_df.loc[i, last_col])).add_to(china_map)
china_map
```

[17]:
```
<folium.folium.Map at 0x7f515c0121d0>
```

Let's create a cartogram to show the distribution of confirmed coronavirus cases in China and mark the affected regions of China with **circular** location markers.

The markers will display the name of the region location along with the number of confirmed coronavirus cases in that region.

[18]:
```
# Student Action: Run the code below.
# Map to show the distribution of confirmed coronavirus cases in China␣
 ↪(circular markers).
china_map = folium.Map(location=[30.9756, 112.2707], width='100%',␣
 ↪height='90%', tiles='Stamen Toner', zoom_start=4.5)
for i in conf_china_df.index:
  folium.Circle(radius=int(conf_china_df.loc[i, last_col]) * 2,
                location=[conf_china_df.loc[i, 'Lat'], conf_china_df.loc[i,␣
 ↪'Long']],
                popup= conf_china_df.loc[i, 'Province/State'] + "\n" +␣
 ↪str(conf_china_df.loc[i, last_col]),
                color='crimson', fill=True, fill_color='crimson').
 ↪add_to(china_map)
china_map
```

[18]:
```
<folium.folium.Map at 0x7f515a6efa20>
```

We know for sure that the Hubei region was affected the most in China; almost 50 times the second most affected region. Hence, the circles for the other regions are very small compared to the circle for the Hubei region.

Let's increase the scale of the radius of the circles to see the variation of people affected in other regions. Also, we will not create a circle for the Hubei region on the map.

[19]:
```
# Student Action: Run the code below.
# Map to show the distribution of confirmed coronavirus cases in China␣
 ↪excluding Wuhan (circular markers).
china_map = folium.Map(location=[30.9756, 112.2707], width='100%',␣
 ↪height='100%', tiles='Stamen Toner', zoom_start=4.5)
for i in conf_china_df.sort_values(by=last_col, ascending=False).index[1:]:
  folium.Circle(radius=int(conf_china_df.loc[i, last_col]) * 100,
                location=[conf_china_df.loc[i, 'Lat'], conf_china_df.loc[i,␣
 ↪'Long']],
                popup= conf_china_df.loc[i, 'Province/State'] + "\n" +␣
 ↪str(conf_china_df.loc[i, last_col]),
                color='crimson', fill=True, fill_color='crimson').
 ↪add_to(china_map)
china_map
```

[19]:
```
<folium.folium.Map at 0x7f515a64df28>
```

As you can see, the number of reported cases of coronavirus are comparable in regions other than Hubei.

Activity 9: Cartogram (or Maps) For World Let's create a cartogram to show the distribution of confirmed coronavirus cases across the world and mark the affected regions in the world with **circular** location markers.

The markers will display the name of the region location along with the number of confirmed coronavirus cases in that region.

[20]:
```
# Student Action: Run the code below.
# Map to show the distribution of confirmed coronavirus cases across the world␣
 ↪(circular markers).
world_map = folium.Map(location=[0, 0], width='100%', height='80%',␣
 ↪tiles='Stamen Toner', zoom_start=2.25)
for i in conf_df.index:
  folium.Circle(location=[conf_df.loc[i, 'Lat'], conf_df.loc[i, 'Long']],
                radius=int(conf_df.loc[i, last_col]),
                popup=conf_df.loc[i, 'Country/Region'] + '\n' + str(conf_df.
 ↪loc[i, last_col]),
                color='crimson', fill=True, fill_color='crimson').
 ↪add_to(world_map)
world_map
```

[20]:
```
<folium.folium.Map at 0x7f515c234198>
```

Activity 10: Cartogram (or Maps) For The USA Let's also create a cartogram specifically for the United States for America.

[21]:
```
# Student Action: Run the code below.
# Map to show the distribution of confirmed coronavirus cases across the USA␣
 ↪(circular markers).
us_conf_csv = '/content/COVID-19/csse_covid_19_data/csse_covid_19_time_series/
 ↪time_series_covid19_confirmed_US.csv'
us_conf_df = pd.read_csv(us_conf_csv)
us_conf_df = us_conf_df.dropna()

us_map = folium.Map(location=[39.381266, -97.922211], width='100%',␣
 ↪height='80%', tiles='Stamen Toner', zoom_start=4)
for i in us_conf_df.index:
  folium.Circle(location=[us_conf_df.loc[i, 'Lat'], us_conf_df.loc[i, 'Long_']],
                radius=int(us_conf_df.loc[i, last_col]),
                popup=us_conf_df.loc[i, 'Province_State'] + '\n' +␣
 ↪str(us_conf_df.loc[i, last_col]),
                color='crimson', fill=True, fill_color='crimson').add_to(us_map)
us_map
```

[21]:
```
<folium.folium.Map at 0x7f515c20c748>
```

Most number of the confirmed cases are recorded in the eastern region in the USA. Also, New York is the most affected city in the USA.

Activity 11: Non-Cumulative Confirmed Cases^^^ Let's identify the countries having the lowest number of daily or non-cumulative coronavirus confirmed cases to further identify the countries that have flattened the curve.

Q: What should be the logic to identify the countries that have flattened the curve?

A: The countries having zero daily (or non-cumulative) confirmed coronavirus cases for a longer period have flattened the curve.

Q: What should be the logic to calculate the non-cumulative confirmed coronavirus cases starting from January 22, 2020?

A: The difference between the current value and the previous value.

Let's create a DataFrame to find out the non-cumulative confirmed coronavirus cases starting from January 22, 2020.

[22]:
```
# Student Action: Run the code below.
# First step to create the DataFrame for the non-cumulative confirmed␣
 ↪coronavirus cases starting from January 22, 2020.
non_cum_conf_df = desc_grp_conf_df.iloc[:, :4]
non_cum_conf_df
```

[22]:
```
       Country/Region         Lat        Long  1/22/20
174                US   37.090200  -95.712900        1
140            Russia   60.000000   90.000000        0
178    United Kingdom  270.029900 -482.924700        0
158             Spain   40.000000   -4.000000        0
85              Italy   43.000000   12.000000        0
..                ...         ...         ...      ...
161          Suriname    3.919300  -56.027800        0
104        MS Zaandam    0.000000    0.000000        0
132  Papua New Guinea   -6.315000  143.955500        0
184    Western Sahara   24.215500  -12.885800        0
98             Lesotho  -29.609988   28.233608        0

[188 rows x 4 columns]
```

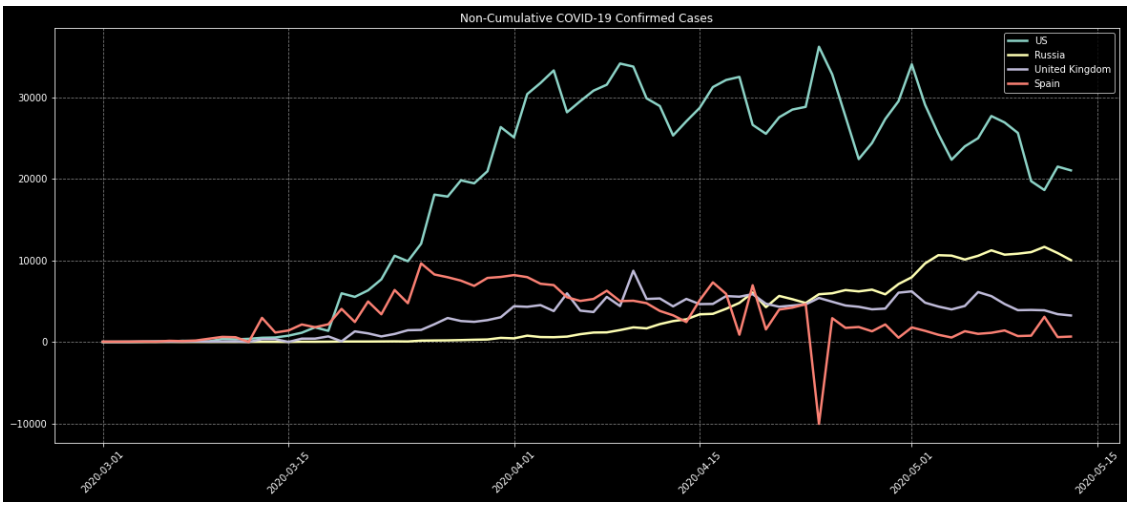
[23]:
```
# Student Action: Run the code below.
# Final step to create the DataFrame for the non-cumulative confirmed␣
 ↪coronavirus cases starting from January 22, 2020.
for i in range(len(desc_grp_conf_df.columns[3:]) - 1):
  series = desc_grp_conf_df[desc_grp_conf_df.columns[3 + (i + 1) ]] -␣
 ↪desc_grp_conf_df[desc_grp_conf_df.columns[3 + i]]
  non_cum_conf_df[desc_grp_conf_df.columns[3 + (i + 1)]] = series

non_cum_conf_df.head()
```

[23]:
```
     Country/Region       Lat      Long  …  5/11/20  5/12/20  5/13/20
174              US   37.0902  -95.7129  …    18621    21495    21030
140          Russia   60.0000   90.0000  …    11656    10899    10028
178  United Kingdom  270.0299 -482.9247  …     3883     3409     3244
158           Spain   40.0000   -4.0000  …     3086      594      661
85            Italy   43.0000   12.0000  …      744     1402      888

[5 rows x 116 columns]
```

Let's create plots for a few countries to visualise the trends for the non-cumulative confirmed cases.

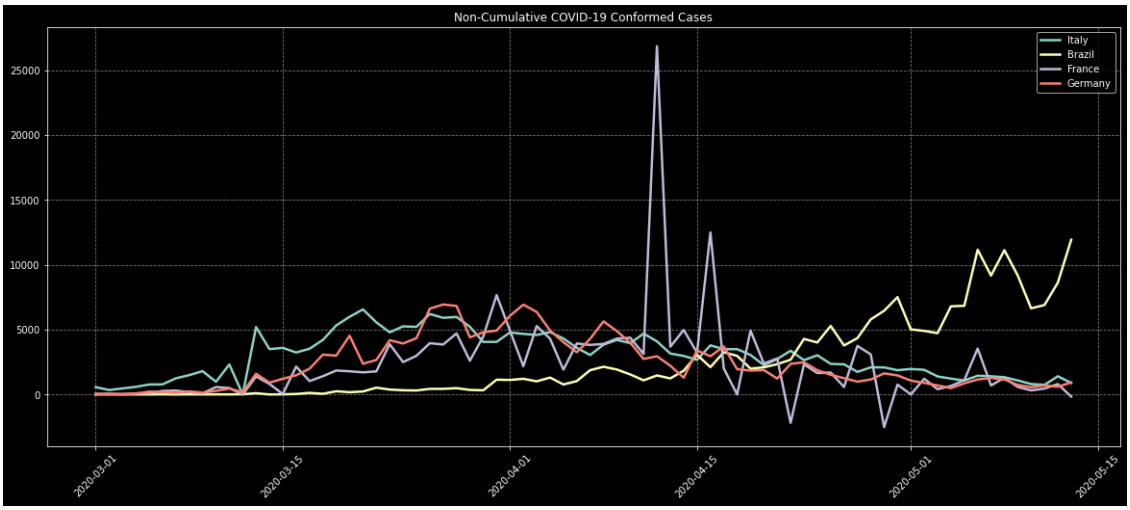
[24]:
```
# Student Action: Run the code below.
# Plot the first four countries having non-cumulative confirmed cases in the␣
 ↪above DataFrame.
plt.figure(figsize=(20, 8))
plt.title('Non-Cumulative COVID-19 Confirmed Cases')
for region in non_cum_conf_df.iloc[:4, :]['Country/Region']:
  plt.plot(global_cases.index[39:], non_cum_conf_df[non_cum_conf_df['Country/
 ↪Region'] == region].iloc[0, 42:], lw=2.5, label=region)
plt.xticks(rotation=45)
plt.legend()
plt.grid(True, 'major', linestyle='--', c='grey')
```

```
plt.show()
```



Q: From the above graph, identify the countries having the lowest daily confirmed cases?

[25]:
```
# Student Action: Run the code below.
# Plot the next four countries having non-cumulative confirmed cases in the␣
 ↪above DataFrame.
plt.figure(figsize=(20, 8))
plt.title('Non-Cumulative COVID-19 Confirmed Cases')
for region in non_cum_conf_df.iloc[4:8, :]['Country/Region']:
  plt.plot(global_cases.index[39:], non_cum_conf_df[non_cum_conf_df['Country/
 ↪Region'] == region].iloc[0, 42:], lw=2.5, label=region)
plt.xticks(rotation=45)
plt.legend()
plt.grid(True, 'major', linestyle='--', c='grey')
plt.show()
```



Q: From the above graph, identify the countries having the lowest daily confirmed cases?

Activity 12: Flattened Curves^^ Let's create a list of countries that have flattened the curve. Ideally, to flatten the curve, the non-cumulative cases for each day should be 0. But in a practical sense, from the point-of-view of healthcare facilities in a country, let's assume that on an average a country can handle 100 COVID-19 cases every day.

So, to find the countries that have flattened the curve, let's calculate the sum of daily coronavirus confirmed cases for each country. If the sum of daily coronavirus confirmed cases for the last 14 days is at most 1400, then we can say that the country has flattened the curve.

[26]:
```
# Student Action: Run the code below.
# List of countries that have flattened the curve.
countries = []
for i in non_cum_conf_df.index:
  last_14_days = []
  for j in range(14):
    last_14_days.append(non_cum_conf_df.loc[i, non_cum_conf_df.columns[-(j +␣
 ↪1)]])
  if sum(last_14_days) <= 1400:
    countries.append(non_cum_conf_df.loc[i, 'Country/Region'])
countries
```

[26]:
```
['China',
 'Switzerland',
 'Israel',
 'Austria',
 'Korea, South',
 'Czechia',
 'Norway',
 'Australia',
 'Malaysia',
 'Finland',
 'Luxembourg',
 'Hungary',
 'Iraq',
 'Thailand',
 'Cameroon',
 'Greece',
 'Azerbaijan',
 'Uzbekistan',
 'Guinea',
 'Croatia',
 'Bosnia and Herzegovina',
```

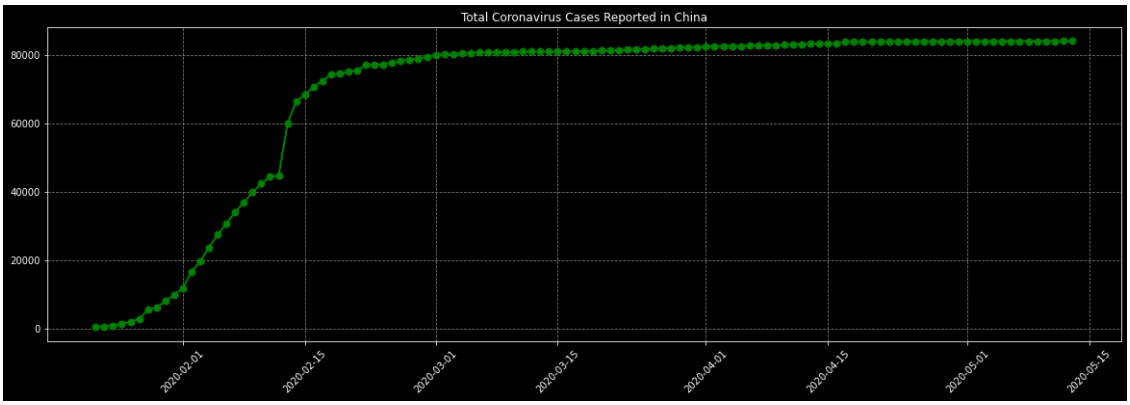
```
'Senegal',
'Bulgaria',
"Cote d'Ivoire",
'Cuba',
'Iceland',
'Estonia',
'North Macedonia',
'Lithuania',
'New Zealand',
'Slovakia',
'Slovenia',
'Guatemala',
'Djibouti',
'Somalia',
'Congo (Kinshasa)',
'Kyrgyzstan',
'El Salvador',
'Tunisia',
'Gabon',
'Maldives',
'Latvia',
'Kosovo',
'Sri Lanka',
'Cyprus',
'Albania',
'Lebanon',
'Niger',
'Guinea-Bissau',
'Costa Rica',
'Tajikistan',
'Burkina Faso',
'Andorra',
'Mali',
'Paraguay',
'Kenya',
'Uruguay',
'Diamond Princess',
'Georgia',
'San Marino',
'Jordan',
'Equatorial Guinea',
'Jamaica',
'Tanzania',
'Malta',
'Zambia',
'Taiwan*',
'Venezuela',
```

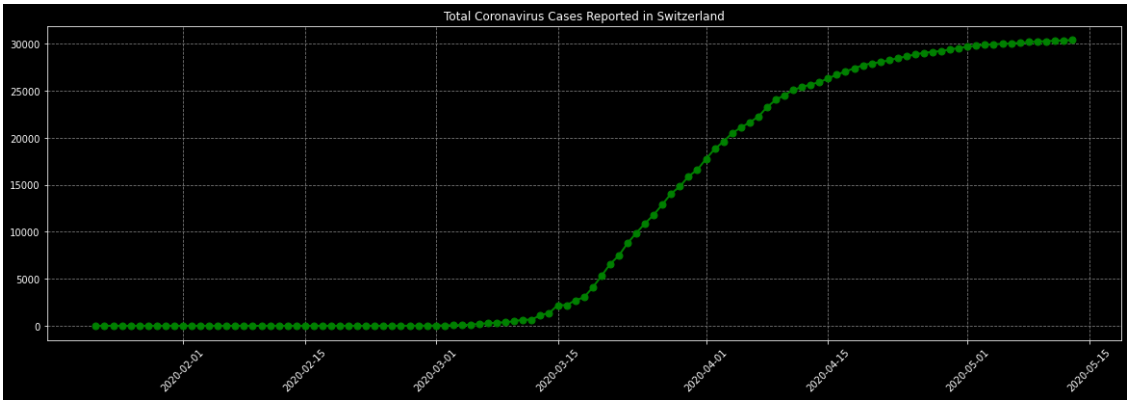
```
'Sierra Leone',
'West Bank and Gaza',
'Chad',
'Congo (Brazzaville)',
'Mauritius',
'Benin',
'Montenegro',
'Cabo Verde',
'Vietnam',
'Rwanda',
'Ethiopia',
'Nepal',
'Haiti',
'Sao Tome and Principe',
'Togo',
'Liberia',
'Madagascar',
'South Sudan',
'Eswatini',
'Burma',
'Central African Republic',
'Brunei',
'Uganda',
'Cambodia',
'Trinidad and Tobago',
'Guyana',
'Mozambique',
'Monaco',
'Bahamas',
'Barbados',
'Liechtenstein',
'Yemen',
'Libya',
'Malawi',
'Syria',
'Angola',
'Mongolia',
'Eritrea',
'Zimbabwe',
'Nicaragua',
'Antigua and Barbuda',
'Botswana',
'Timor-Leste',
'Gambia',
'Grenada',
'Laos',
'Fiji',
```

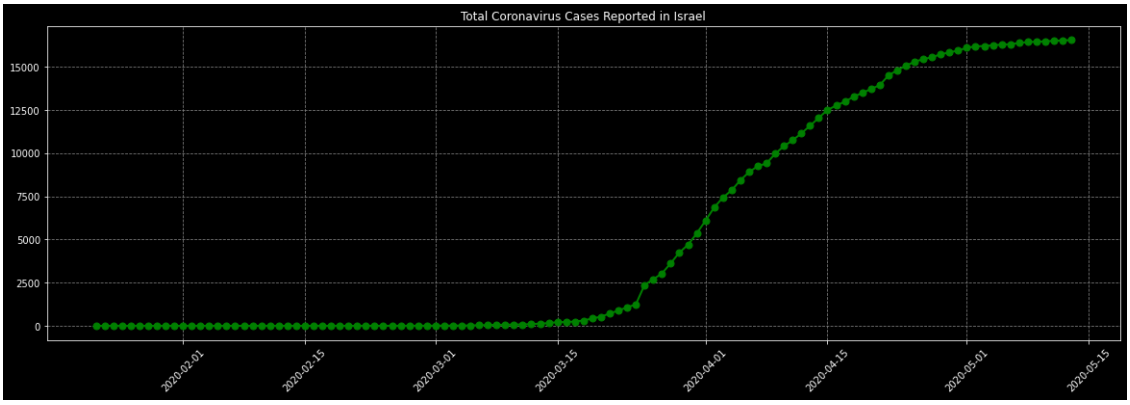
```
 'Belize',
 'Saint Lucia',
 'Saint Vincent and the Grenadines',
 'Dominica',
 'Namibia',
 'Saint Kitts and Nevis',
 'Bhutan',
 'Mauritania',
 'Burundi',
 'Holy See',
 'Comoros',
 'Seychelles',
 'Suriname',
 'MS Zaandam',
 'Papua New Guinea',
 'Western Sahara',
 'Lesotho']
```

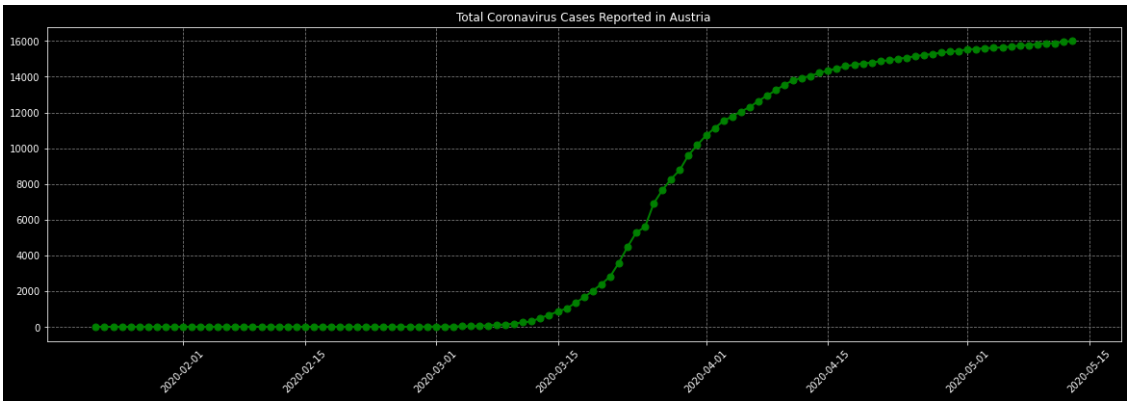
Let's create line plots to visualise the total confirmed cases for the first 10 countries in the above list.

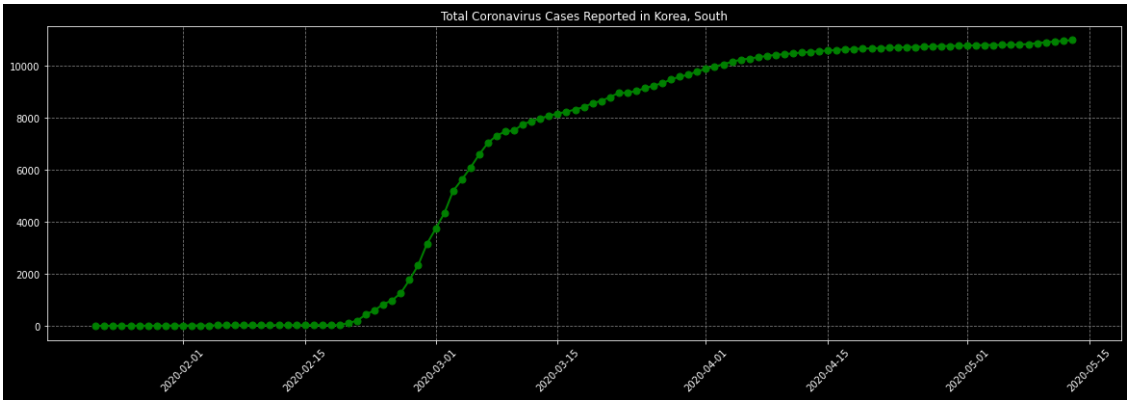
[27]:
```
# Student Action: Run the code below.
# Line plot for the total number of coronavirus confirmed cases reported in the␣
 ↪countries that have flattened the curve.
def flattened_curve(country_list):
  for country in country_list:
    country_cases = conf_df[conf_df['Country/Region'] == country].iloc[:, 4:].
 ↪apply(sum, axis=0)
    country_cases.index = pd.to_datetime(country_cases.index)

    plt.figure(figsize=(20, 6))
    plt.title(f'Total Coronavirus Cases Reported in {country}')
    plt.plot(country_cases.index, country_cases, c='g', linewidth=2,␣
 ↪marker='o', markersize=7)
    plt.xticks(rotation=45)
    plt.grid(True, 'major', linestyle='--', c='grey')
    plt.show()

flattened_curve(countries[:10])
```

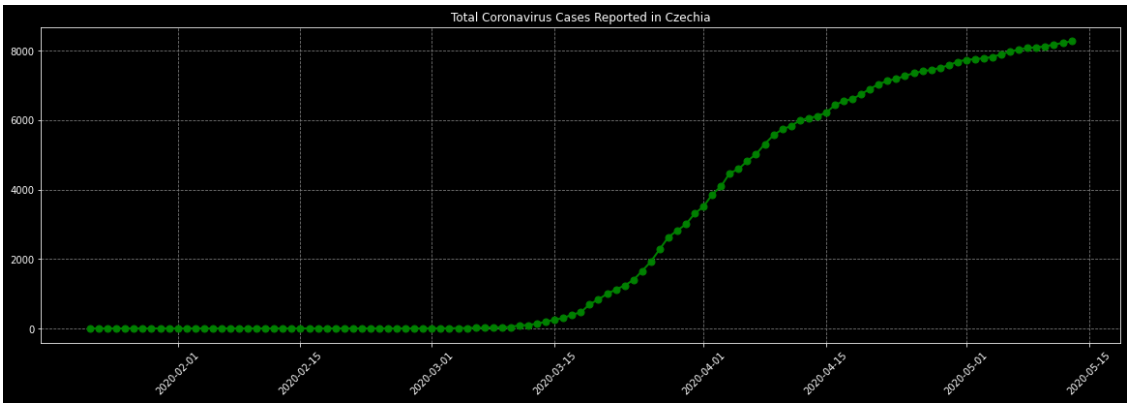

Total Coronavirus Cases Reported in China
80000
60000
40000
20000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15

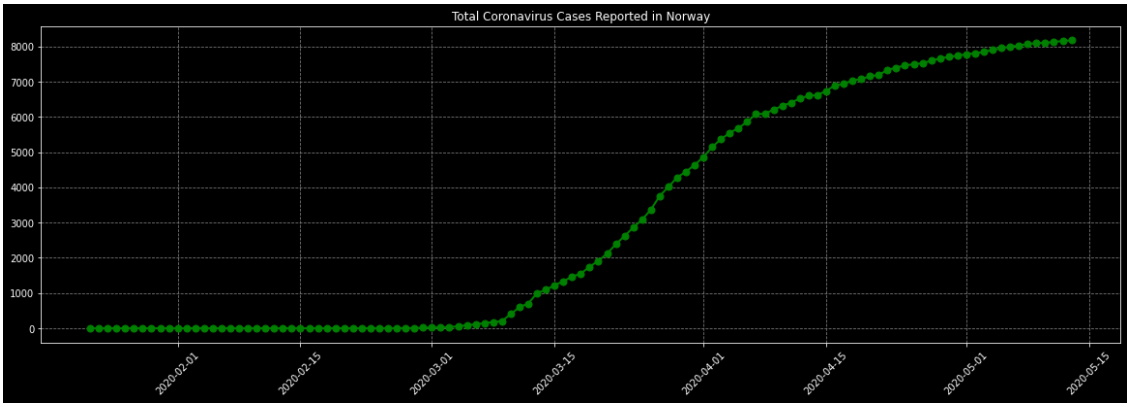

Total Coronavirus Cases Reported in Switzerland
30000
25000
20000
15000
10000
5000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15

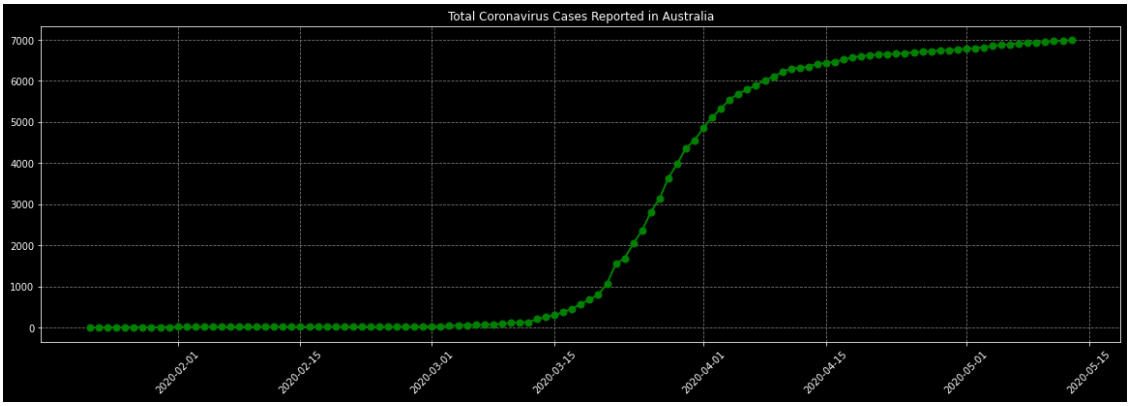

Total Coronavirus Cases Reported in Israel
15000
12500
10000
7500
5000
2500
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15

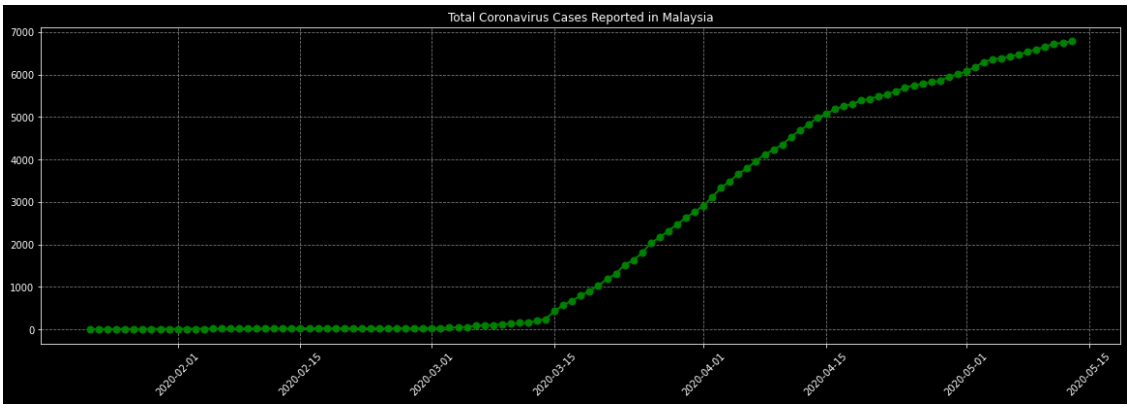

Total Coronavirus Cases Reported in Austria
16000
14000
12000
10000
8000
6000
4000
2000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15

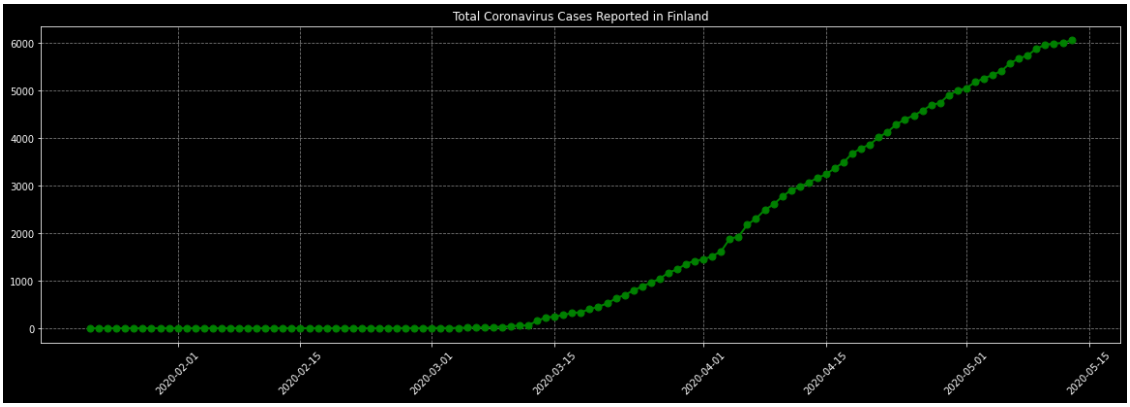

Total Coronavirus Cases Reported in Korea, South
10000
8000
6000
4000
2000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15


Total Coronavirus Cases Reported in Czechia
8000
6000
4000
2000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15


Total Coronavirus Cases Reported in Norway
8000
7000
6000
5000
4000
3000
2000
1000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15


Total Coronavirus Cases Reported in Australia
7000
6000
5000
4000
3000
2000
1000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15


Total Coronavirus Cases Reported in Malaysia
7000
6000
5000
4000
3000
2000
1000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
2020-05-15



We analysed only the confirmed cases. In the same way, you can analyse the deaths due to coronavirus data and number of people recovered from coronavirus data. The dataset structure for these two datasets is the same as that of the confirmed cases dataset.

[28]:
```
# Student Action: Run the code below.
# DataFrame for the total deaths due to coronavirus.
deaths_df = pd.read_csv(deaths_csv)
deaths_df.head()
```

[28]:

	Province/State	Country/Region	Lat	…	5/11/20	5/12/20	5/13/20
0	NaN	Afghanistan	33.0000	…	122	127	132
1	NaN	Albania	41.1533	…	31	31	31
2	NaN	Algeria	28.0339	…	507	515	522
3	NaN	Andorra	42.5063	…	48	48	49
4	NaN	Angola	-11.2027	…	2	2	2

[5 rows x 117 columns]

[29]:
```
# Student Action: Run the code below.
# DataFrame for the total number of patients recovered from coronavirus.
rec_df = pd.read_csv(rec_csv)
rec_df.head()
```

[29]:

	Province/State	Country/Region	Lat	…	5/11/20	5/12/20	5/13/20
0	NaN	Afghanistan	33.0000	…	558	610	648
1	NaN	Albania	41.1533	…	654	682	688
2	NaN	Algeria	28.0339	…	2841	2998	3058
3	NaN	Andorra	42.5063	…	550	568	576
4	NaN	Angola	-11.2027	…	13	13	14

[5 rows x 117 columns]

1.0.7 Wrap-Up

```
TEACHER
  Quickly summarise what we learned in this class.

EXPECTED STUDENT RESPONSE
  We learnt:
    1. How to create a world map using Python.
    2. How to create a line plot using Python.
    3. How to interpret data and draw conclusions.

TEACHER
  That's impressive! You are absolutely correct. You have understood the
  importance of data analysis very well.

  In this course, you will learn to solve real-life problems using data
  analysis and artificial intelligence. The problems will be based on life
  science (like coronavirus), space science, medical science, finance and
  driverless cars.

  I hope you had fun in this class and hope to see you in the upcoming
  classes. Bye!

EXPECTED STUDENT RESPONSE
  Bye teacher!
```

1.0.8 Activities

Teacher Activities

1. COVID-19 Outbreak Analysis (Class Copy)

   https://colab.research.google.com/drive/1L6WEa9lmpKAeAvp4xbFLwJrfI3eJbwrl

2. COVID-19 Outbreak Analysis (Reference)

   https://colab.research.google.com/drive/1xYPZ9v8nFya6iZdrtoS7EbE7g1Pi7i2O

3. WHO Declares COVID-19 a Pandemic

   https://twitter.com/WHO/status/1237777021742338049

4. COVID-19 Live Dashboard by Johns Hopkins University

   https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6

5. COVID-19 Data Source

   https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time

6. Ministry of Health & Family Welfare in India

   https://www.mohfw.gov.in/

7. Prevention Guidelines by World Health Organisation (WHO)

   https://www.youtube.com/watch?v=bPITHEiFWLc&feature=emb_title

Student Activities

1. WHO Declares COVID-19 a Pandemic

   https://twitter.com/WHO/status/1237777021742338049

2. COVID-19 Live Dashboard by Johns Hopkins University

   https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6

3. COVID-19 Data Source

   https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time

4. Ministry of Health & Family Welfare in India

   https://www.mohfw.gov.in/

5. Prevention Guidelines by World Health Organisation (WHO)

   https://www.youtube.com/watch?v=bPITHEiFWLc&feature=emb_title

---

1.0.9 Wrap-Up Pitch

This pitch is to be presented after all the activities are covered.

Kid Motivation: Hats-off Recap of the class followed by asking the student to invite parent and celebrate kids accomplishment in front of the parent:

Congratulations! You are among the exceptional students who get 3 out of 3 hats off in the trial class! You're really bright and I hope you join the full course where you'll solve the real-life problems by applying the machine learning and deep learning algorithms that are used by many data scientists across the globe.

Curriculum Vision Let me give you a quick summary of what we will be learning in the course. Our coursework follows a **project-based learning** philosophy where students apply the concepts they learn to real-life problems (like COVID-19 Growth visualisation) throughout the course. Working on industry-grade projects makes their learning practical instead of theoretical. Our coursework is structured into 3 modules:

1. **Python Developer Certification:** In the first module of 12 classes, your kid will become a certified Python Developer. Kids build AI-based games using core Artificial Intelligence concepts using Python. They learn the basics of Python like sequences, commands, and loops. This dramatically improves their logic and concentration.

2. **Data Analyst Certification:** After the second module of 48 classes, your kid will become among the youngest in the world to be a certified Data Analyst. Kids learn Data Visualisation and Predictive Data Analysis to apply powerful algorithms that can solve real-life problems.

Every kid works on large data sets to generate meaningful insights or outcomes. In one of the projects, they would be able to hunt planets beyond our solar system and even predict house prices.

3. **AI + Space Technology Certification:** In the third module, your kid will become among the 1st in the world to learn advanced data modelling and Machine Learning, using which they'll create Space Tech and Driverless Car simulations. Kids reach the peak of their creative imagination as they apply coding to frontier technologies like Space Technology and Driverless Cars. Top kids in the course who do the best classes and projects get a chance to visit Silicon Valley where they meet top Google scientists and entrepreneurs.

Space-Tech Showcase Let me show you one of the space tech projects built in our course. Here, you will apply machine learning classification-based algorithms such as Random Forest Classifier to detect planets beyond our solar system using the data collected by NASA. You would also be able to work on projects like

- Effects of Greenhouse Gas on Global Warming
- The possible impact of Asteroids on Earth
- Searching all meteor landing sites on Earth

Student Success Now let me share with you what some of our students are doing

Garvit has made an AI-based eye testing application. Using visual recognition, the application recognises the position of the pupil and checks for any impairment. The application generates a score that is sent to eye-clinics for preventive care.

Another student, Shaurya has created an AI-based chatbot that is used by educational institutes for interaction with the parents and students

Both these apps have been featured widely in international media. Isn't it amazing that teenagers are creating apps like these within just 40 hours of learning to code?

During the course, your kid will also be working on industry-grade AI-based projects and get a chance to visit Silicon Valley and meet top Google scientists and entrepreneurs.

Build Urgency To provide you with more details about the course as well as answer any questions that you may have our academic counsellor will get in touch with you. My schedule is almost full but I'd love to have your kid as my student since your kid is exceptionally bright with true entrepreneurship potential!

Congratulations `<kid name>` again on being awarded 3 hats off–you're truly exceptional. Hope you had fun. Thank you for your time today. Kindly stay on the panel and do not close this page when I end the class–our entire curriculum along with the details will be displayed on the panel.
