**Web dashboard for Data Visualization using Python**

**(Python Dash)**

**The goal of the Activities:**

In this activity, you will learn the fundamentals of how to create a visualization dashboard using Python Dash. Dash is an open source Python Library to build web applications which are optimized for data visualization. A dashboard is an information management tool that visually tracks, analyzes and displays key performance indicators (KPI), metrics and key data points to monitor the situation of a business, department or specific process. The function of a dashboard is to present information visually by aggregating the data and extracting interesting knowledge. It simplifies the data into more manageable chunks of visual information as it relates to the nature of the data. Dashboards can be used to help business managers make informed decisions that dramatically impact business performance – which in turn affects their bottom line.

**Resources:**

Dash by Plotly is a framework, build on top of Plotly.js, React, and Flask, that can be used to build web applications and modern UI elements like charts, tables, and interactive input and output functionalities where the user would engage with this platform with Python.

You can learn Plotly and Dash from the below URLs.

# Official Plotly documentation for Python [link](https://plot.ly/python/)

# Official Dash User Guide [link](https://dash.plot.ly/)

**Preparation:**

# Learn Plotly: 8 videos series (by SuperDataScience) [link](https://www.youtube.com/watch?v=j0wvKWb337A&list=PLE50-dh6JzC4onX-qkv9H3HtPbBVA8M94)

# Dash By Plotly: 6 videos series (by codebliss) [link](https://www.youtube.com/watch?v=Ldp3RmUxtOQ&list=PLCDERj-IUIFCaELQ2i7AwgD2M6Xvc4Slf)

**Activities:**

This activity includes the following parts:

Part 1: Set up the project and datasets on your computer from Git Repository provided and run the first example. (Individual Submission)

Part 2: Creating Bar chart, Line chart, Box plot and Heatmap with given datasets with Plotly.(Group Submission)

Part 3: Designing and creating Dashboard with plots drawn with plotly in the part2. (Group Submission)

You need to work on the above parts in the following steps and submit a document:

Step 1: Create a repository for this project under your own individual repository.

Step 2: Do part 1 on each of your computers individually and add it to your repository. Prepare your Part 1 individual report and submit it (make sure to include necessary screenshots).

Step 3: Present to each other your works after you submitted that and start creating a team version of part 1 by merging ideas and codes from individual works.

Step 4: Start working on Part 2 as a team. You need to identify tasks and create your sprint schedule using a template or you can use Trello application/mobile app (or similar tools) as well. It is a part of grading to show how tasks are distributed. You need to include screenshots in the team report. Also, include log of Git Activities in the team report.

**Python Dash Activity, PART 1:**

**Setting up the project and datasets on your computer and run first example**

**The following two steps are similar to TDD assignment.**

1. **Python 3.8:** If you haven't already installed Python on your computer, download and install Python from this [link](https://www.python.org/downloads/) according to your platform.
2. **IDE:** Download and install PyCharm through the links below:

[Download the Windows version](https://download.jetbrains.com/python/pycharm-community-2019.3.2.exe).

[Download the Mac version](https://download.jetbrains.com/python/pycharm-community-2019.3.2.dmg).

1. **GitHub:** Retrieve project into your personal and team repository.

* *Perform the below steps like you did in previous activities*
* *Create a GitHub Repository.*
* *git clone https://github.com/ProgrammingClassrooms/Software-Eng.git*
* *git config --global user.name "John Doe"*
* *git config --global user.email* [*johndoe@example.com*](mailto:johndoe@example.com)
* *git init*
* *git add -A*
* *git commit -m "first commit"*
* *git remote add myorigin <repository link> (provide your own repository link)*
* *git push -u myorigin master*

1. **Source Code:**

* Open Pycharm and select ‘Open’

****

* Set the directory path to the cloned project.
* The application is imported into your IDE.

1. **Project Structure:**

Project contains three folders:

* **Datasets:** CSV files which are used for creating plots.
* **Plots:** Data visualization plots will be created here for part 2. It also contains some examples initially.
* **Dashboards:** Final Dashboards will be created here for part 3.

1. **Project Overview:**

We want to implement a Dashboard for Coronavirus Disease 2019 (COVID-19) data visualization. In this program, we have two phases. First, preparing plots that we want to show in the dashboard with Plotly individually. Second, designing a dashboard to show the plots in the one page.

1. **Datasets:**

In the Dataset folder you have two datasets at first. ([From Novel Coronavirus (COVID-19) Cases, provided by JHU CSSE](https://github.com/CSSEGISandData/COVID-19))

This is the data repository for the 2019 Coronavirus Visual Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). Also, Supported by ESRI Living Atlas Team and the Johns Hopkins University Applied Physics Lab (JHU APL).

* **CoronavirusTotal.csv**: Contains statistics for number of confirmed cases, deaths and recovered cases in the world between 1/22/2020 to 3/17/2020.
* **CoronaTimeSeries.csv**: Contains number of cases per each day between 1/22/2020 to 3/17/2020

**For assignments two more dataset will be added:**

* **Olympic2016Rio.csv**: Contains number of medals per each country in Olympic 2016 Rio.
* **Weather2014-15.csv**: Contains weather statistics of different temperature in the years 2014 and 2015.

1. **Packages:**

All required packages are imported at the top of the file. If libraries are missing in py files, it will show as RED under-line, you can right-click on it and click on “Show context action” / “Install package” to import those libraries.

1. **First example:**

In the Plots folder open the *barchart.py*, look at the code and read comments carefully, then right click on it and select “Run barchart.py” after running code, you will see the result in the browser. It creates a *barchart.html* file in the same py file directory and if the browser didn’t open automatically, you can open this file manually.

This chart shows the number of confirmed cases in the first 20 states of the US. Figure 1 illustrates the result.

Submit the screenshot similar to figure 1.

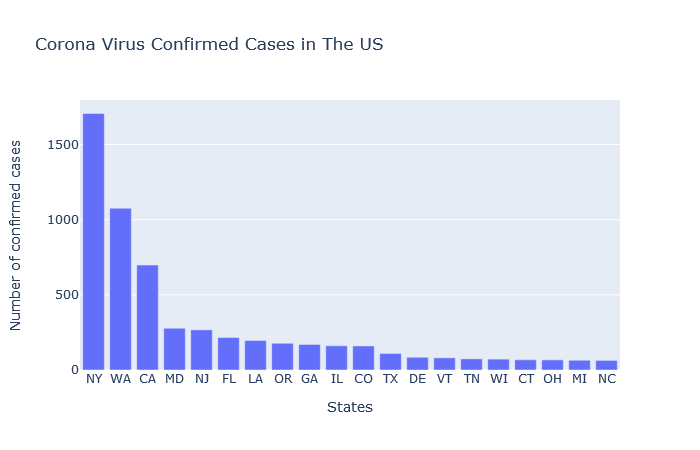


Figure 1. Number of confirmed cases for different states as of 3/17/20.

**Python Dash Activity, PART 2:**

**Creating Bar chart, Line chart, Box plot and Heatmap with given datasets with Plotly**

Once you set up and test the application follow the below instruction. Each task should be completed in a separate file under the Plots folder.

1. **Stack bar chart:** Draw a stacked bar chart to represent the CoronaVirus deaths, recovered and under-treatment (cases which are confirmed but are still under the recovering process) of all reported first 20 countries except China as shown in figure 2. (use CoronavirusTotal.csv file)

**Source code:**

import pandas as pd  
import plotly.offline as pyo  
import plotly.graph\_objs as go  
  
# Load CSV file from Datasets folder  
df = pd.read\_csv('../Datasets/CoronavirusTotal.csv')  
  
# Removing empty spaces from State column to avoid errors  
df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
  
# Creating unrecovered column  
df['Unrecovered'] = df['Confirmed'] - df['Deaths'] - df['Recovered']  
  
# Removing China and Others from data frame  
df = df[(df['Country'] != 'China')]  
  
# Creating sum of number of cases group by Country Column  
new\_df = df.groupby(['Country']).agg(  
 {'Confirmed': 'sum', 'Deaths': 'sum', 'Recovered': 'sum', 'Unrecovered': 'sum'}).reset\_index()  
  
# Sorting values and select 20 first value  
new\_df = new\_df.sort\_values(by=['Confirmed'], ascending=[False]).head(20).reset\_index()  
  
# Preparing data  
trace1 = go.Bar(x=new\_df['Country'], y=new\_df['Unrecovered'], name='Unrecovered', marker={'color': '#CD7F32'})  
trace2 = go.Bar(x=new\_df['Country'], y=new\_df['Recovered'], name='Recovered', marker={'color': '#9EA0A1'})  
trace3 = go.Bar(x=new\_df['Country'], y=new\_df['Deaths'], name='Deaths', marker={'color': '#FFD700'})  
data = [trace1, trace2, trace3]  
  
# Preparing layout  
layout = go.Layout(title='Corona Virus Cases in the first 20 country expect China', xaxis\_title="Country",  
 yaxis\_title="Number of cases", barmode='stack')  
  
# Plot the figure and saving in a html file  
fig = go.Figure(data=data, layout=layout)  
pyo.plot(fig, filename='stackbarchart.html')

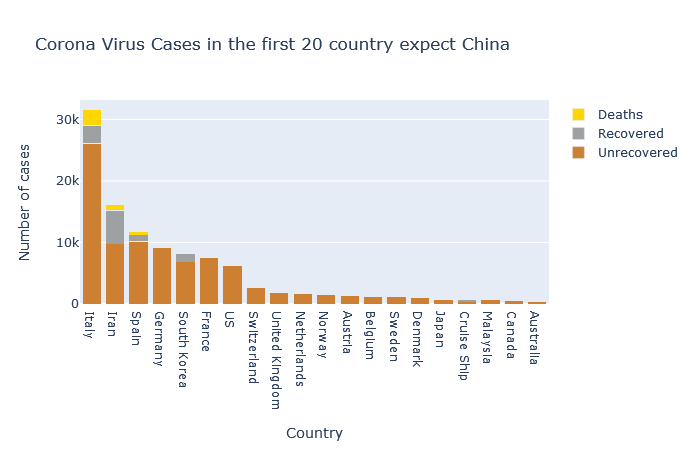


Figure 2. Number of cases vs. country as of 3/17/20.

1. **Line chart:** Draw a line chart to represent the Corona Virus confirmed cases of all reported cases in the period as shown in figure 3. (use CoronaTimeSeries.csv file)

**Source code:**

import pandas as pd  
import plotly.offline as pyo  
import plotly.graph\_objs as go  
  
# Load CSV file from Datasets folder  
df = pd.read\_csv('../Datasets/CoronaTimeSeries.csv')  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Preparing data  
data = [go.Scatter(x=df['Date'], y=df['Confirmed'], mode='lines', name='Death')]  
  
# Preparing layout  
layout = go.Layout(title='Corona Virus Confirmed Cases From 2020-01-22 to 2020-03-17', xaxis\_title="Date",  
 yaxis\_title="Number of cases")  
  
# Plot the figure and saving in a html file  
fig = go.Figure(data=data, layout=layout)  
pyo.plot(fig, filename='linechart.html')

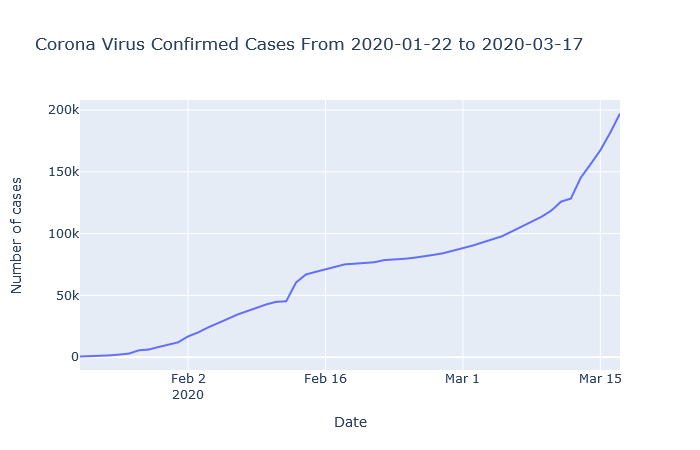
**Result:**

Figure 3. Number of cases over different times as of 3/17/20.

1. **Multi line chart:** Draw a multi line chart to represent the CoronaVirus death, recovered and under-treatment cases of all reported cases in the period as shown in figure 4. (use ConfirmedTimeSeries.csv file)

**Source code:**

import pandas as pd  
import plotly.offline as pyo  
import plotly.graph\_objs as go  
  
# Load CSV file from Datasets folder  
df = pd.read\_csv('../Datasets/CoronaTimeSeries.csv')  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Preparing data  
trace1 = go.Scatter(x=df['Date'], y=df['Death'], mode='lines', name='Death')  
trace2 = go.Scatter(x=df['Date'], y=df['Recovered'], mode='lines', name='Recovered')  
trace3 = go.Scatter(x=df['Date'], y=df['Unrecovered'], mode='lines', name='Unrecovered')

data = [trace1,trace2,trace3]

# Preparing layout  
layout = go.Layout(title='Corona Virus Death and Recovered Cases From 2020-01-22 to 2020-03-17', xaxis\_title="Date",  
 yaxis\_title="Number of cases")  
  
# Plot the figure and saving in a html file  
fig = go.Figure(data=data, layout=layout)  
pyo.plot(fig, filename='multilinechart.html')

**Result:**

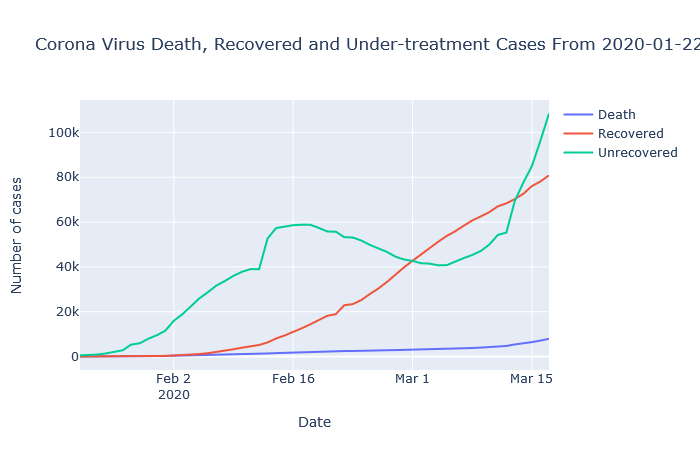


Figure 4. Number of deaths, recovered, still under-treatment cases over times as of 3/17/20.

1. **Bubble chart:** Draw a bubble chart to represent the Corona Virus recovered and under-treatment (cases which are confirmed but are still under the recovering process) of all reported countries except China. (use CoronavirusTotal.csv file)

import pandas as pd  
import plotly.offline as pyo  
import plotly.graph\_objs as go  
  
df = pd.read\_csv('../Datasets/CoronavirusTotal.csv')  
# Removing empty spaces from State column to avoid errors  
df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  
  
# Creating unrecovered column  
df['Unrecovered'] = df['Confirmed'] - df['Deaths'] - df['Recovered']  
  
# Removing China and Others from data frame  
df = df[(df['Country'] != 'China') & (df['Country'] != 'Others')]  
  
# Creating sum of number of cases group by Country Column  
new\_df = df.groupby(['Country']).agg(  
 {'Confirmed': 'sum', 'Recovered': 'sum', 'Unrecovered': 'sum'}).reset\_index()  
  
# Preparing data  
data = [  
 go.Scatter(x=new\_df['Recovered'],  
 y=new\_df['Unrecovered'],  
 text=new\_df['Country'],  
 mode='markers',  
 marker=dict(size=new\_df['Confirmed'] / 100,color=new\_df['Confirmed'] / 100, showscale=True))  
]  
  
# Preparing layout  
layout = go.Layout(title='Corona Virus Confirmed Cases', xaxis\_title="Recovered Cases",  
 yaxis\_title="Unrecovered Cases", hovermode='closest')  
  
# Plot the figure and saving in a html file  
fig = go.Figure(data=data, layout=layout)  
pyo.plot(fig, filename='bubblechart.html')



Figure 5. Number of under-treatment cases vs recovered ones as of 3/17/20.

1. **Heatmap:** Draw a heatmap to represent the Corona Virus recovered cases of all reported cases per day of week and week of month. (use CoronaTimeSeries.csv file)

import plotly.offline as pyo  
import plotly.graph\_objs as go  
import pandas as pd  
  
# Load CSV file from Datasets folder  
df = pd.read\_csv('../Datasets/CoronaTimeSeries.csv')  
  
# Preparing data  
data = [go.Heatmap(x=df['Day'],

y=df['WeekofMonth'],

z=df['Recovered'].values.tolist(),

colorscale='Jet')]

# Preparing layout

layout = go.Layout(title='Corona Virus Recovered Cases', xaxis\_title="Day of Week",

yaxis\_title="Week of Month")

# Plot the figure and saving in a html file

fig = go.Figure(data=data, layout=layout)

pyo.plot(fig, filename='heatmap.html')

**Result:**

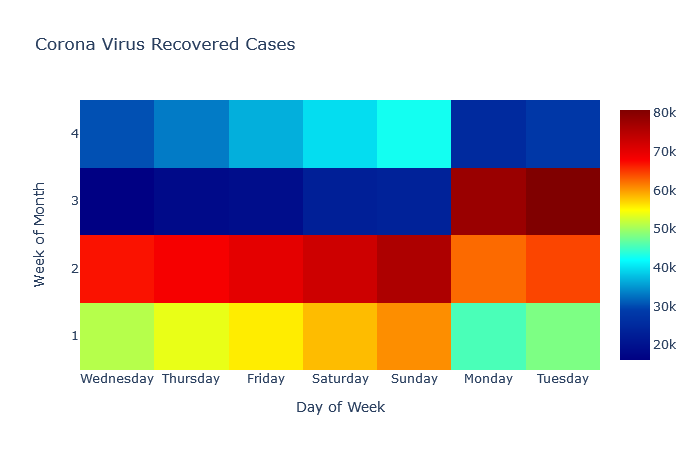
****

Figure 6. Heatmap as of 3/17/20.

**Assignment Submission:**

Take screenshots of the above 6 figures in a word document. Then at the end answer the following questions and submit the word file.

Using different data sets as shown below (or a new CoronaVirus Dataset that you can download to find updated graphs).

* 1. In each of the source codes, there is a comment and corresponding code about “Preparing data”. Find each one of them and explain what each one does as far as data preprocessing before drawing the figures.
  2. Draw a bar chart to represent the total medals of Olympic 2016 of 20 first top countries. (use Olympic2016Rio.csv file)
  3. Draw a stack bar chart to represent the Gold, Silver, Bronze medals of Olympic 2016 of 20 first top countries. (use Olympic2016Rio.csv file)
  4. Draw a line chart to represent the actual max temperature of each month in weather statistics. (use Weather2014-15.csv file)
  5. Draw a multi line chart to represent the actual max, min and mean temperature of each month in weather statistics. (use Weather2014-15.csv file)
  6. Draw a bubble chart to represent the average min and max temperature of each month in weather statistics. (use Weather2014-15.csv file)
  7. Draw a heatmap to represent the recorded max temperature on day of week and month of year. (use Weather2014-15.csv file)
  8. Reflect on the creation of these graphs in python as far as complexity of the creation process, clarity of the figures, and usefulness as far as providing information.