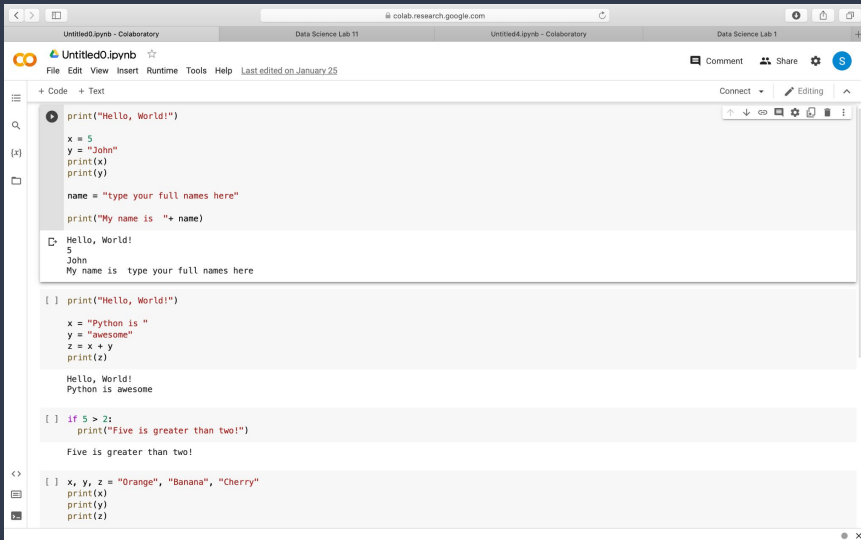


# Data Science Presentation

By:

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.



```
print("Hello, World!")

x = 5
y = "John"
print(x)
print(y)

name = "type your full names here"
print("My name is " + name)

[ ] print("Hello, World!")

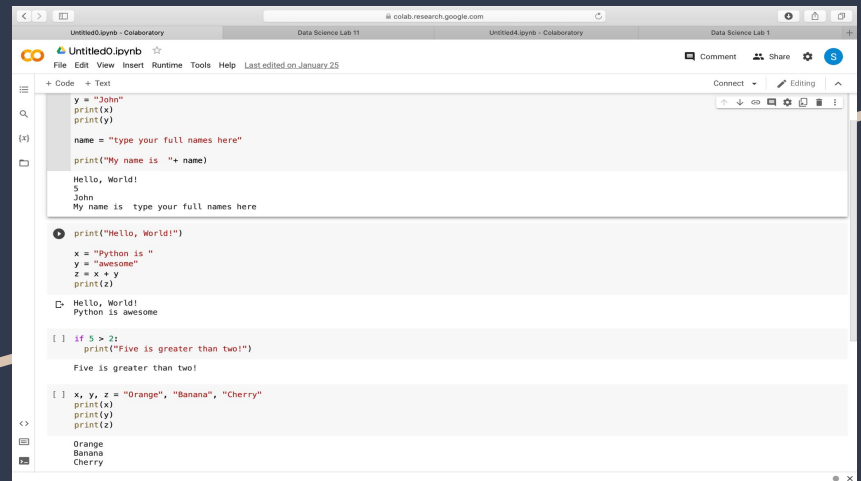
x = "Python is "
y = "awesome"
z = x + y
print(z)

Hello, World!
Python is awesome

[ ] if 5 > 2:
    print("Five is greater than two!")

Five is greater than two!

[ ] x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```



```
y = "John"
print(x)
print(y)

name = "type your full names here"
print("My name is " + name)

Hello, World!
5
John
My name is type your full names here

[ ] print("Hello, World!")

x = "Python is "
y = "awesome"
z = x + y
print(z)

Hello, World!
Python is awesome

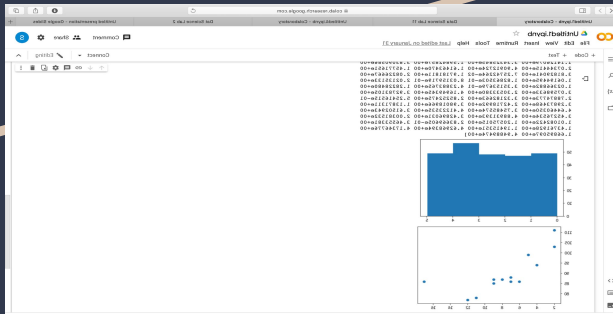
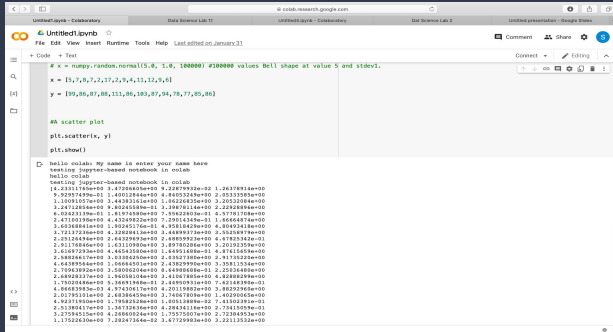
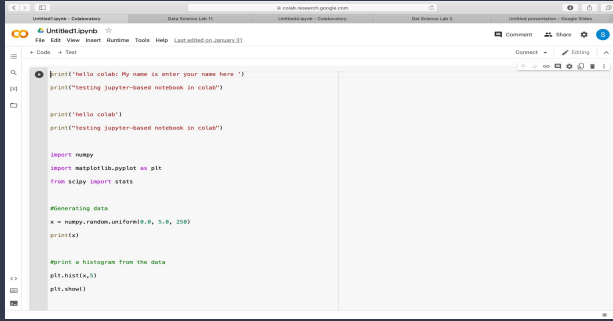
[ ] if 5 > 2:
    print("Five is greater than two!")

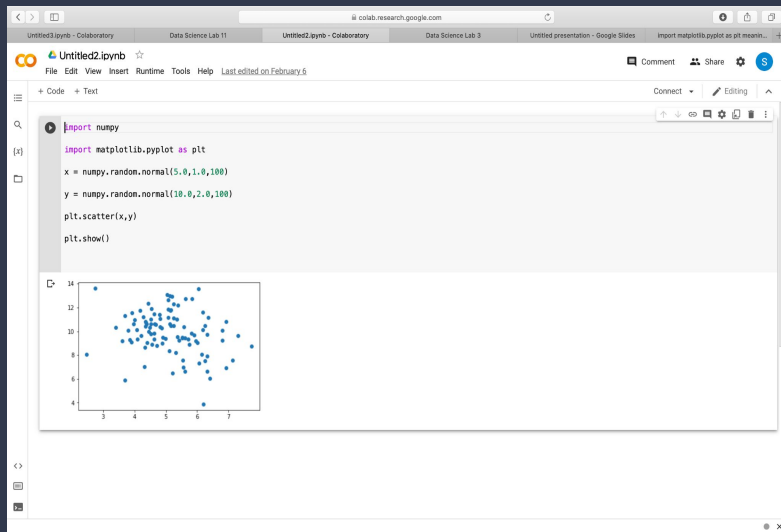
Five is greater than two!

[ ] x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)

Orange
Banana
Cherry
```

1. For lab 1, we used an introductory code to test put the environment of google colab and to make sure that we were comfortable using the platform.
2. I was able to print and copy three codes, one given, and two found on my own to ensure that I was comfortable using google colab as well as comfortable using and understanding python code.



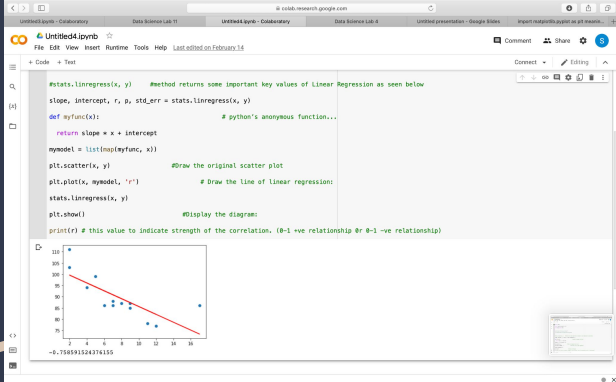


1. For lab 3, we were comparing the data generation to lab 2. The only differences were that the data was only generated through a scatter plot and.
2. The python libraries involved are numpy, matplotlib, and stats.

```
import numpy
import matplotlib.pyplot as plt
from scipy import stats

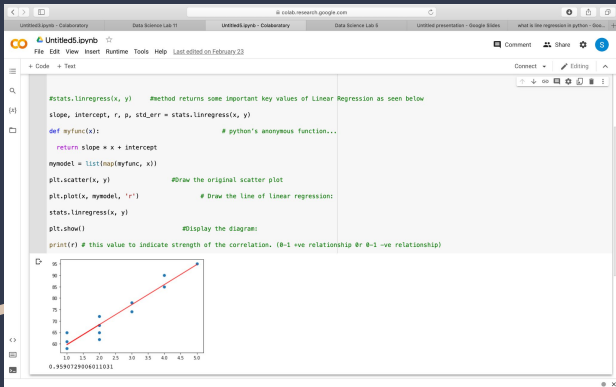
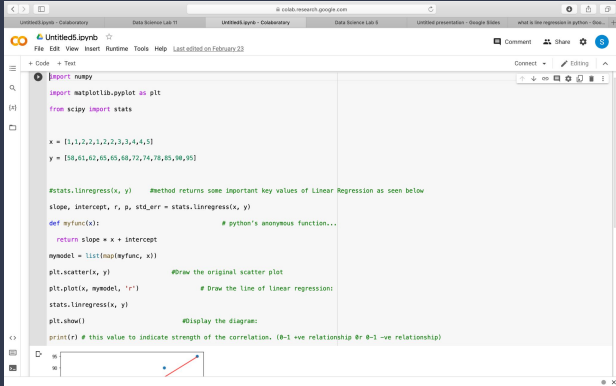
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [19,16,17,16,11,16,19,17,14,7,7,16,16]

#stats.linregress(x, y) #method returns some important key values of Linear Regression as seen below
slope, intercept, r, p, std_err = stats.linregress(x, y)
def myfunc(x):
    return slope * x + intercept
mymodel = list(map(myfunc, x))
plt.scatter(x, y) #Draw the original scatter plot
plt.plot(x, mymodel, "r") # Draw the line of Linear regression
stats.linregress(x, y)
plt.show() #display the diagram
print(r) # this value to indicate strength of the correlation. (0-1 -ve relationship & 0-1 +ve relationship)
```



1. The code used for lab 4 is very similar to the code used for lab 2. The numbers are the same but that data was generated differently.
2. `Stats.linregress(x, y)` :Calculate a linear least-squares regression for two sets of measurements.
3. A line of regression is formed using the slope generated from the data.

1. For lab 5 we used data (numbers) from a table of data to generate data in the form on a scatter graph. This is similar to what we did in lab 4.
2. In this lab we are strictly focusing on line regression which could help us predict the next set of numbers as well as compute strength.
3. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x).



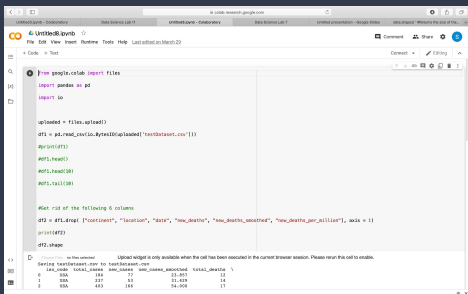
```
1 #Reading data from a local drive
2 from google.colab import files
3 uploaded = files.upload()
4
5 #If you will have to "Run" this to see next instruction to choose file
6 #Follow the instructions to locate your file location
7
8 #After choosing the file:
9 import pandas as pd
10
11 #Run this to see your read data set file
12 df = pd.read_csv(io.BytesIO(uploaded['file.csv'])) #File.csv will be replaced by the name of your file e.g., myFile.csv
13
14 #Prints the data set:
15 df.head() #Prints 1st 5 rows.....Run to see
16 df.shape() #Returns the size of the data set (i.e. Print X #numcolumns)
```

Kernel output is only available when the cell has been executed in the current browser session. Please reset this cell's output.

	date	total_passes	new_passes	%
1	2000-01-01	100	0	0.0
2	2000-01-01	100	0	0.0
3	2000-01-01	100	0	0.0
4	2000-01-01	100	0	0.0
5	2000-01-01	100	0	0.0
6	2000-01-01	100	0	0.0
7	2000-01-01	100	0	0.0
8	2000-01-01	100	0	0.0
9	2000-01-01	100	0	0.0
10	2000-01-01	100	0	0.0
11	2000-01-01	100	0	0.0
12	2000-01-01	100	0	0.0
13	2000-01-01	100	0	0.0
14	2000-01-01	100	0	0.0
15	2000-01-01	100	0	0.0
16	2000-01-01	100	0	0.0
17	2000-01-01	100	0	0.0
18	2000-01-01	100	0	0.0
19	2000-01-01	100	0	0.0
20	2000-01-01	100	0	0.0
21	2000-01-01	100	0	0.0
22	2000-01-01	100	0	0.0
23	2000-01-01	100	0	0.0
24	2000-01-01	100	0	0.0
25	2000-01-01	100	0	0.0
26	2000-01-01	100	0	0.0
27	2000-01-01	100	0	0.0
28	2000-01-01	100	0	0.0
29	2000-01-01	100	0	0.0
30	2000-01-01	100	0	0.0

	total_passes	new_passes	%
1	100	0	0.0
2	100	0	0.0
3	100	0	0.0
4	100	0	0.0
5	100	0	0.0
6	100	0	0.0
7	100	0	0.0
8	100	0	0.0
9	100	0	0.0
10	100	0	0.0
11	100	0	0.0
12	100	0	0.0
13	100	0	0.0
14	100	0	0.0
15	100	0	0.0
16	100	0	0.0
17	100	0	0.0
18	100	0	0.0
19	100	0	0.0
20	100	0	0.0
21	100	0	0.0
22	100	0	0.0
23	100	0	0.0
24	100	0	0.0
25	100	0	0.0
26	100	0	0.0
27	100	0	0.0
28	100	0	0.0
29	100	0	0.0
30	100	0	0.0

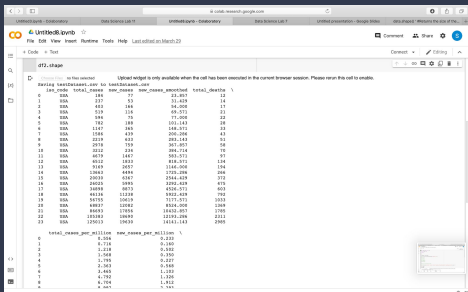
1. Lab 6 was an introductory to importing our own files and using the data from those files.
2. `df = pd.read_csv(io.BytesIO(uploaded['file.csv']))`  
'file.csv' will be : This reads the file that you imported.
3. `data.head()` 'prints 1st 5 rows.....Run to see: This prints only the first 5 rows of the data
4. `data.shape()` '#Returns the size of the data set (i.e. #rows X #numcolumns): This returns the shape of the array.



```
from google.colab import Files
import pandas as pd
import io

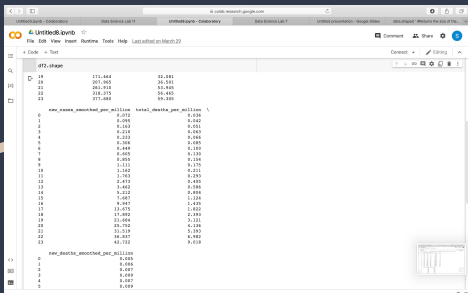
uploaded = Files.upload()
df1 = pd.read_csv(io.BytesIO(uploaded['testdata.csv']))
df1.head()
df1.tail(20)
df1.tail(100)

df1 #id of the following 6 columns
df2 = df1.drop(["continent", "location", "date", "new_deaths", "new_deaths_smoothed", "new_deaths_per_million"], axis = 1)
df2.head()
```



```
df2.head()
```

	new_deaths	total_deaths	new_cases	new_cases_smoothed	total_deaths_1
0	USA	200	22	21,454	15
1	USA	207	22	21,454	15
2	USA	209	22	21,454	15
3	USA	219	22	21,454	15
4	USA	221	22	21,454	15
5	USA	224	22	21,454	15
6	USA	226	22	21,454	15
7	USA	227	22	21,454	15
8	USA	229	22	21,454	15
9	USA	230	22	21,454	15
10	USA	231	22	21,454	15
11	USA	232	22	21,454	15
12	USA	233	22	21,454	15
13	USA	234	22	21,454	15
14	USA	235	22	21,454	15
15	USA	236	22	21,454	15
16	USA	237	22	21,454	15
17	USA	238	22	21,454	15
18	USA	239	22	21,454	15
19	USA	240	22	21,454	15
20	USA	241	22	21,454	15
21	USA	242	22	21,454	15
22	USA	243	22	21,454	15
23	USA	244	22	21,454	15



```
df2.head()
```

	new_deaths	total_deaths	new_cases	new_cases_smoothed	total_deaths_1
0	USA	200	22	21,454	15
1	USA	207	22	21,454	15
2	USA	209	22	21,454	15
3	USA	219	22	21,454	15
4	USA	221	22	21,454	15
5	USA	224	22	21,454	15
6	USA	226	22	21,454	15
7	USA	227	22	21,454	15
8	USA	229	22	21,454	15
9	USA	230	22	21,454	15
10	USA	231	22	21,454	15
11	USA	232	22	21,454	15
12	USA	233	22	21,454	15
13	USA	234	22	21,454	15
14	USA	235	22	21,454	15
15	USA	236	22	21,454	15
16	USA	237	22	21,454	15
17	USA	238	22	21,454	15
18	USA	239	22	21,454	15
19	USA	240	22	21,454	15
20	USA	241	22	21,454	15
21	USA	242	22	21,454	15
22	USA	243	22	21,454	15
23	USA	244	22	21,454	15

1. Lab 7 is similar to lab 6. Instead of reading the first five rows we are dropping 6 columns from the data that we want to read.
2. #Get rid of the following 6 columns

```
df2 = df1.drop(["continent", "location", "date", "new_deaths", "new_deaths_smoothed", "new_deaths_per_million"], axis = 1)
```

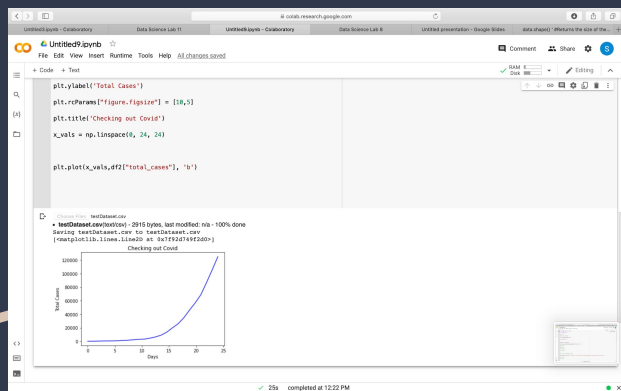


```
from google.colab import files
import pandas as pd
import io

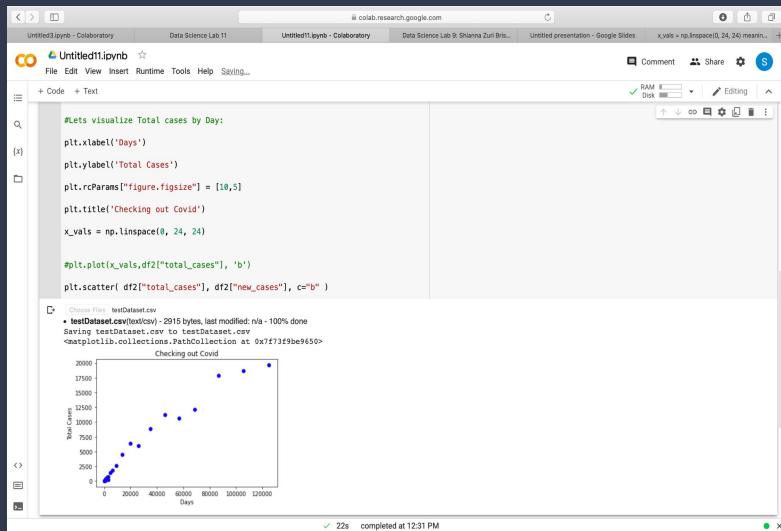
#Add these 2 libraries
import matplotlib.pyplot as plt
import numpy as np

uploaded = files.upload()
df1 = pd.read_csv(io.BytesIO(uploaded['testDataset.csv']))
#print(df1)
df1.head()
df1.head(10)
df1.tail(10)

#Get rid of the following 6 columns
df2 = df1.drop(['continent', 'location', 'date', 'new_deaths', 'new_deaths_smoothed', 'new_deaths_per_million'], axis = 1)
#print(df2)
df2.shape
```



1. For lab 8, we also imported our own data sheet to read from. 6 columns from the data were also dropped. In addition to that we generated data using two of the columns which allowed us to see the amount of total cases per day.
2. `plt.xlabel('Days')`: Days would be our x values and `plt.ylabel('Total Cases')` would be our y values in the graph.
3. `plt.rcParams["figure.figsize"] = [10,5]`: This makes the figures width and height.
4. The NumPy `linspace` function (sometimes called `np.linspace`) is a tool in Python for creating numeric sequences.



1. For lab 9, we are doing the exact same thing we did for lab 8. The difference is that we are generating the data using a scatter plot.

1. For lab 10 we did new deaths by day instead of total cases by day. We did the exact same thing that we did in lab 8, we just used two different columns.
2. We imported the data ourselves, displayed the size of the data, and generated a graph.

```
File Edit View Insert Runtime Tools Help Last loaded at Aug 23
+ Code - Test
+ Comment + / Library

10 From google.colab import Files
11 registered = Files.upload()
12 import pandas as pd
13
14 import matplotlib.pyplot as plt
15 from sklearn.linear_model import LinearRegression, stats
16 df1 = pd.read_csv('new_deaths.csv')
17
18 df1.head()
19 df1.tail()
20 df1.info()
21 df1.describe()
22 df1.shape
23
24 # Create visualization Total cases by Day
25
```

```
10 # Create visualization Total cases by Day
11 plt.xlabel('Day')
12 plt.ylabel('New Deaths')
13 plt.figure(figsize=(10,10))
14 plt.title('Checking out Covid')
15 x_val = np.linspace(0, 400, 400)
16 plt.plot(x_val, df1['new_deaths'], 'b')
17 plt.plot(x_val, df1['new_deaths'], 'r')
18 x = np.array(df1['new_deaths'])
19 y = np.array(df1['new_deaths'])
20 # Create Linear Regression
21 from sklearn.linear_model import LinearRegression
22 plt.plot(x, y, label='Actual Value')
23 plt.legend(['Linear Regression', 'Actual Value'])
24 plt.show()
25 print('RMSE value: ', rms.vals22.df1)
```

