

Batch:B1 Roll No:1914049 Experiment No.:3

Aim:To use Pandas in built visualization and Matplotlib visualization to perform exploratory data analysis



Resources needed: Python IDE



**Theory:**

**Pandas Built-in Data Visualization**

Pandas have got built-in capabilities for data visualization. It's built-off of matplotlib, but it baked into pandas for easier usage!

import numpy as np import pandas as pd import matplotlib as mp %matplotlib inline

|  |  |
| --- | --- |
| There are some fake data csv files | you can read in as dataframes: |
| df1 **=**​ df1.head().  #Bar plot for df1 can be plot using df2.plot.bar(stacked=True) | pd.read\_csv(​'df1'​,index\_col​**=**​0​) |
| #ploting histogram of only one width less than 1. | column with 50 bins are setting bar |

df1['A'].plot.hist(bins=50, rwidth=0.8)

#line plot in pandas df1.plot.line()

#scatter plot with color and colormaps

df1.plot.scatter()

#boxplot of data frame will helps us to spot the outliers(mild and extream both) df2.plot.box()

#density plots- to explore symmetric or assymetric nature of your dataset. df2.plot.density()

**Mathplotlib for Data Visualization**

Matplotlib is the "grandfather" library of data visualization with Python. It was created by John Hunter. He created it to try to replicate MatLab's (another programming language) plotting capabilities in Python. So if you happen to be familiar with matlab, matplotlib will feel natural to you.

It is an excellent 2D and 3D graphics library for generating scientific figures.

Some of the major Pros of Matplotlib are:

* Generally easy to get started for simple plots
* Support for custom labels and texts
* Great control of every element in a figure
* High-quality output in many formats
* Very customizable in general

Matplotlib allows you to create reproducible figures programmatically

**Installation**

You'll need to install matplotlib first with either:

conda install matplotlib or pip install matplotlib

**Importing**

Import the matplotlib.pyplot​ ​

**Basic Matplotlib Commands**

We can create a very simple line plot using the following

plt.plot(x, y, 'r') # 'r' is the color red #setting x and y axis labels, title of plot plt.xlabel('X Axis Title Here') plt.ylabel('Y Axis Title Here') plt.title('StringTitlehere')

**Using subplot a grid of plots can be created as shown below. Also we can set marker and linestyle along with color of plot.**

plt.subplot(1,2,1) plt.plot(x, y, 'r.--') # plt.subplot(1,2,2) plt.plot(y, x, 'g\*-.'); **Matplotlib’s object oriented api:**

The main idea in using the more formal Object Oriented method is to create figure objects and then just call methods or attributes off of that object. This approach is nicer when dealing with a canvas that has multiple plots on it.

# Create Figure object to represent an empty canvas fig = plt.figure()

# Add set of axes to figure(manually) axes = fig.add\_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (range 0 to 1)

# Plot on that set of axes axes.plot(x, y, 'b') axes.set\_xlabel('Set X Label') # Notice the use of set\_ to begin methods axes.set\_ylabel('Set y Label')

axes.set\_title('Set Title') axes.set\_legend(loc=1)

|  |  |
| --- | --- |
| **Figure size, aspect ratio and** | **DPI** |
| Matplotlib allows the aspect ratio, | DPI and figure size to be specified |
| when the Figure object is created. | You can use |
| the ​figsize​ and ​dpi​ keyword | arguments. |
| ● figsize is a tuple of the​ inches | width and height of the figure in |
| ● dpi is the dots-per-inch​ | (pixel per inch). |

For example:

fig **=**​ plt.figure(figsize​ **=**​ (​ 8​ ,​ 4​ )​ , dpi**=**​ 100​ )​

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Activities:**

**Use Pandas And Matplotlib For Following Activities**

1. Download data set with atleast 1500 rows and 10-20 columns(numeric and non numeric) from valid data sources
2. Visualization to summarize your data set(density, frequency plot)
3. Measures of central tenancy of data set (mean, median etc)
4. Determining presence of outliers in your dataset(boxplot)
5. Correlation of attributes in your dataset( scatter plot and line plot on 2-3 pairs which are correlated)
6. Comparison of data ploted on same scale using barplot( 3 plots for 3 different columns

pairs)

1. Use different, colors, styles, markers,marker with different size, legends, labels, colormaps dpi, figsize etc in the plot
2. Save these plots
3. Write down your comment on each of these plots
4. place legends at appropriate location on the plot
5. Write down observation for your dataset for each of above listed task of analysis.

**Result: (script and output)**

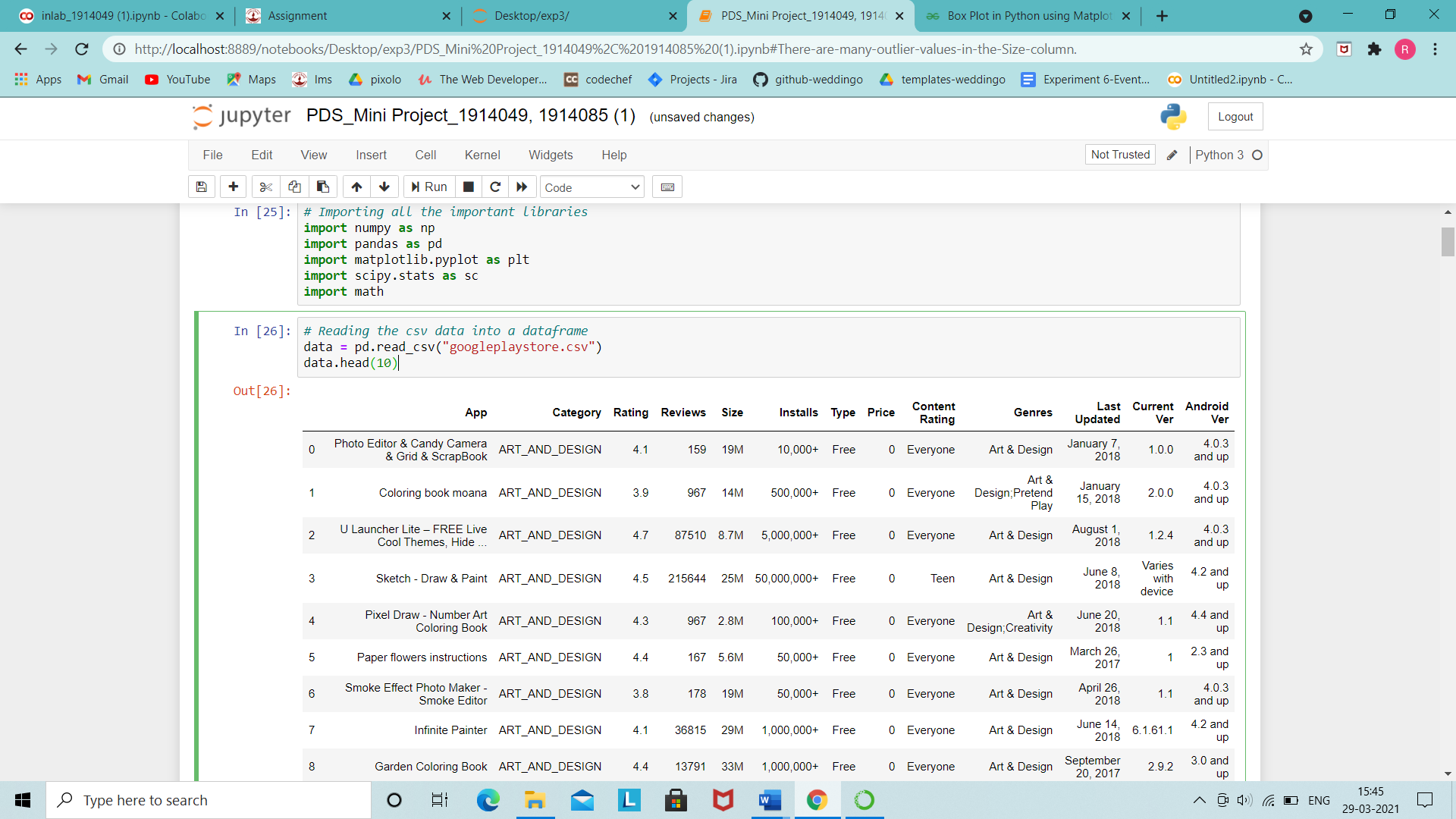


fig = plt.figure(figsize=(8,4), dpi=100)

plt.title("Density graph for Age")

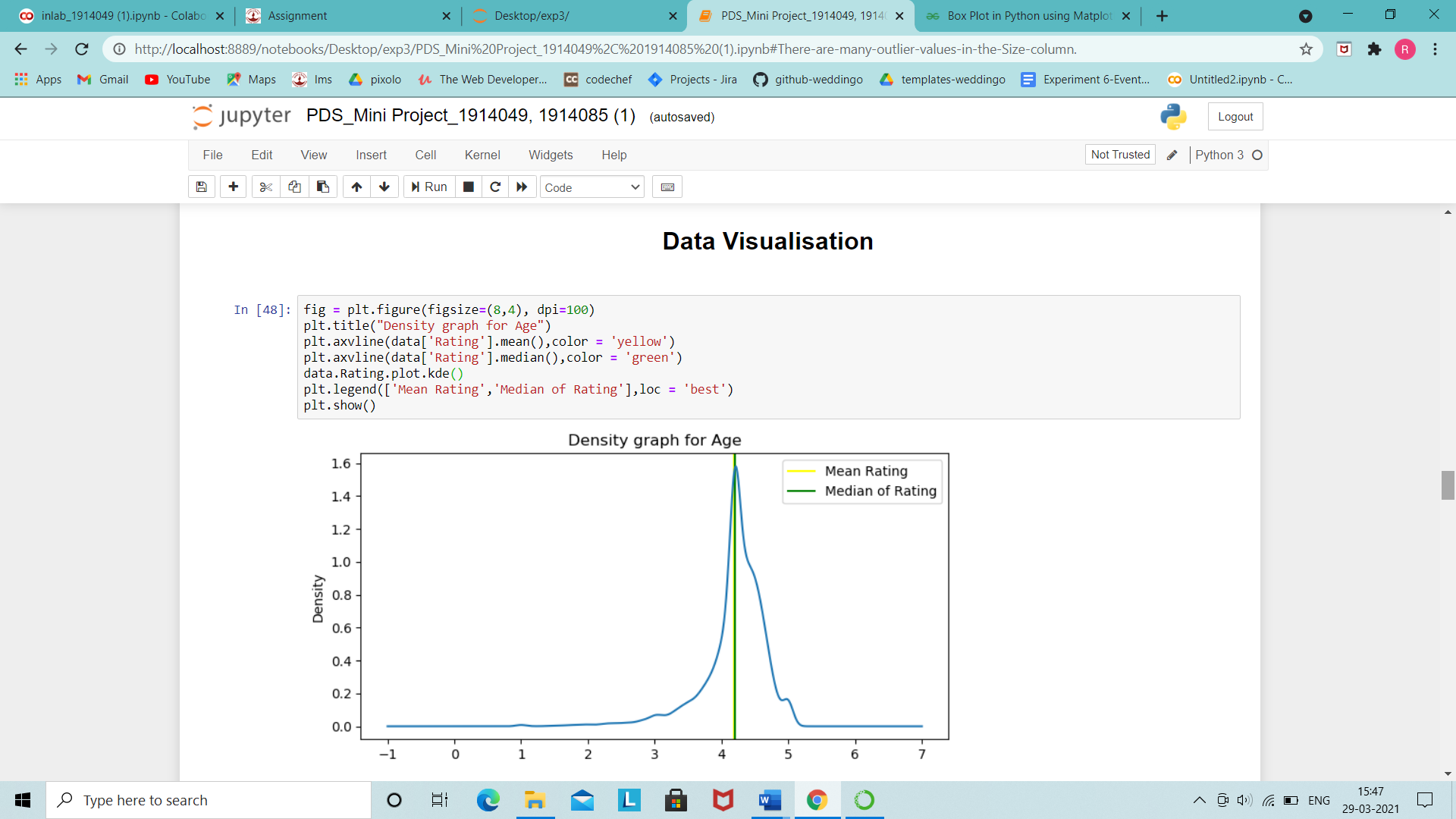
plt.axvline(data['Rating'].mean(),color = 'yellow')

plt.axvline(data['Rating'].median(),color = 'green')

data.Rating.plot.kde()

plt.legend(['Mean Rating','Median of Rating'],loc = 'best')

plt.show()



### From this density graph we can see that an app ratings mostly lie in the range of 4 to 5 ,with around 4.2 being the mean .

fig = plt.figure(figsize=(15,6))

plt.hist(data.Installs)

plt.axvline(data['Installs'].mean(),color = 'red')

plt.axvline(data['Installs'].median(),color = 'green')

plt.legend(['Mean Rating','Median of Rating'])

plt.xlim(0,3500000)

plt.ylim(0,35000)

plt.show()

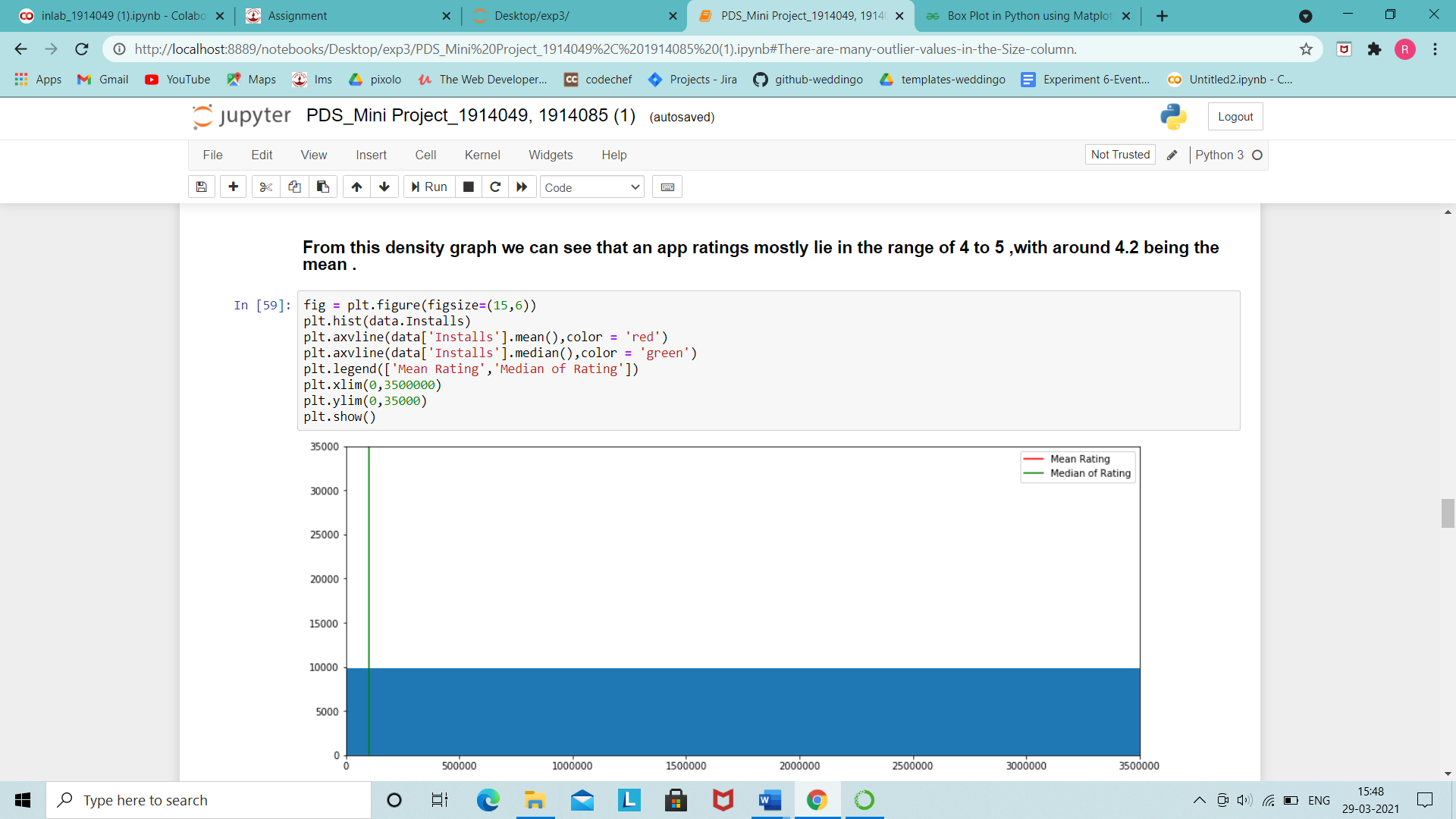


fig = plt.figure(figsize=(10,6))

plt.boxplot(data.Installs)

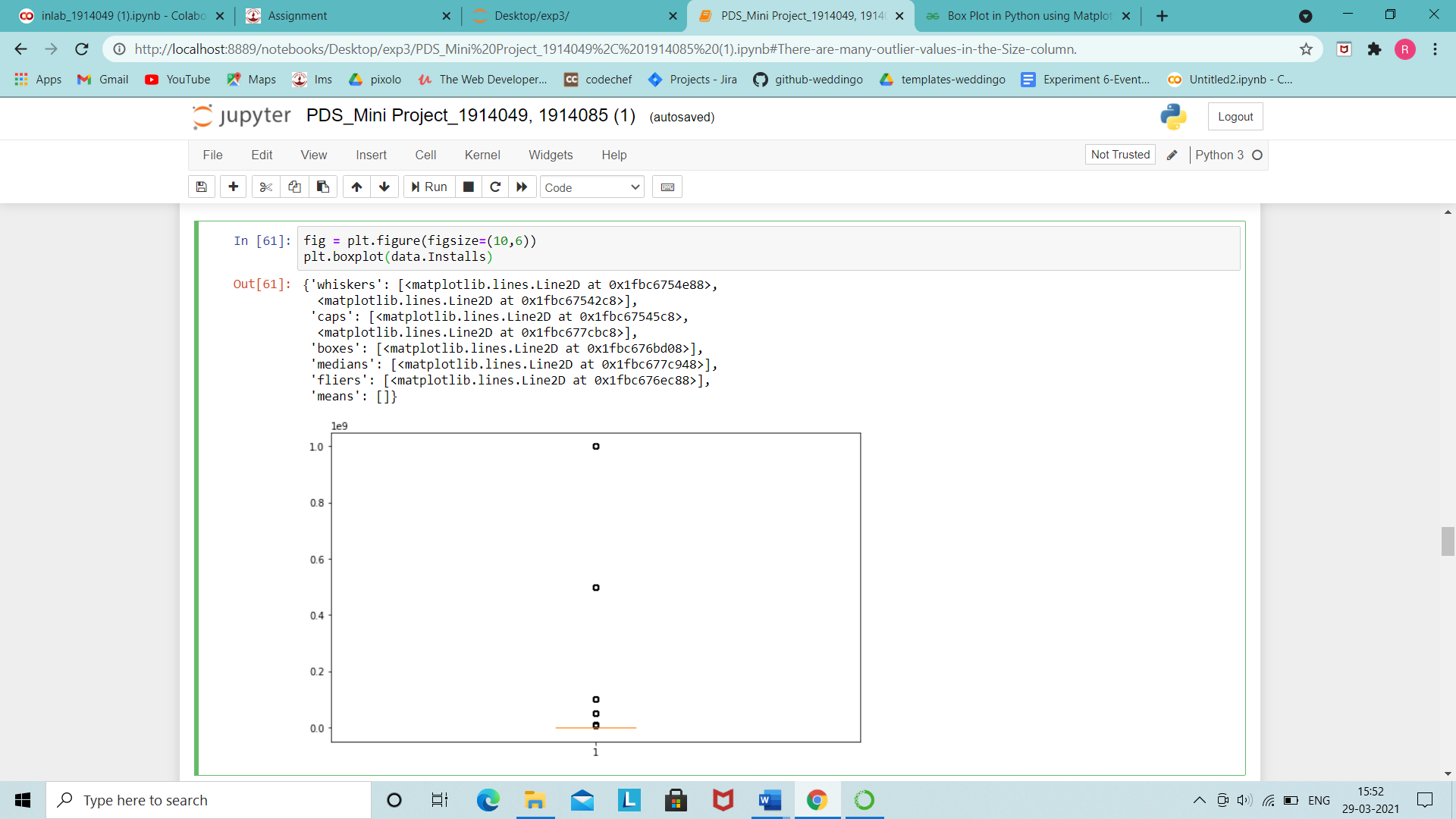
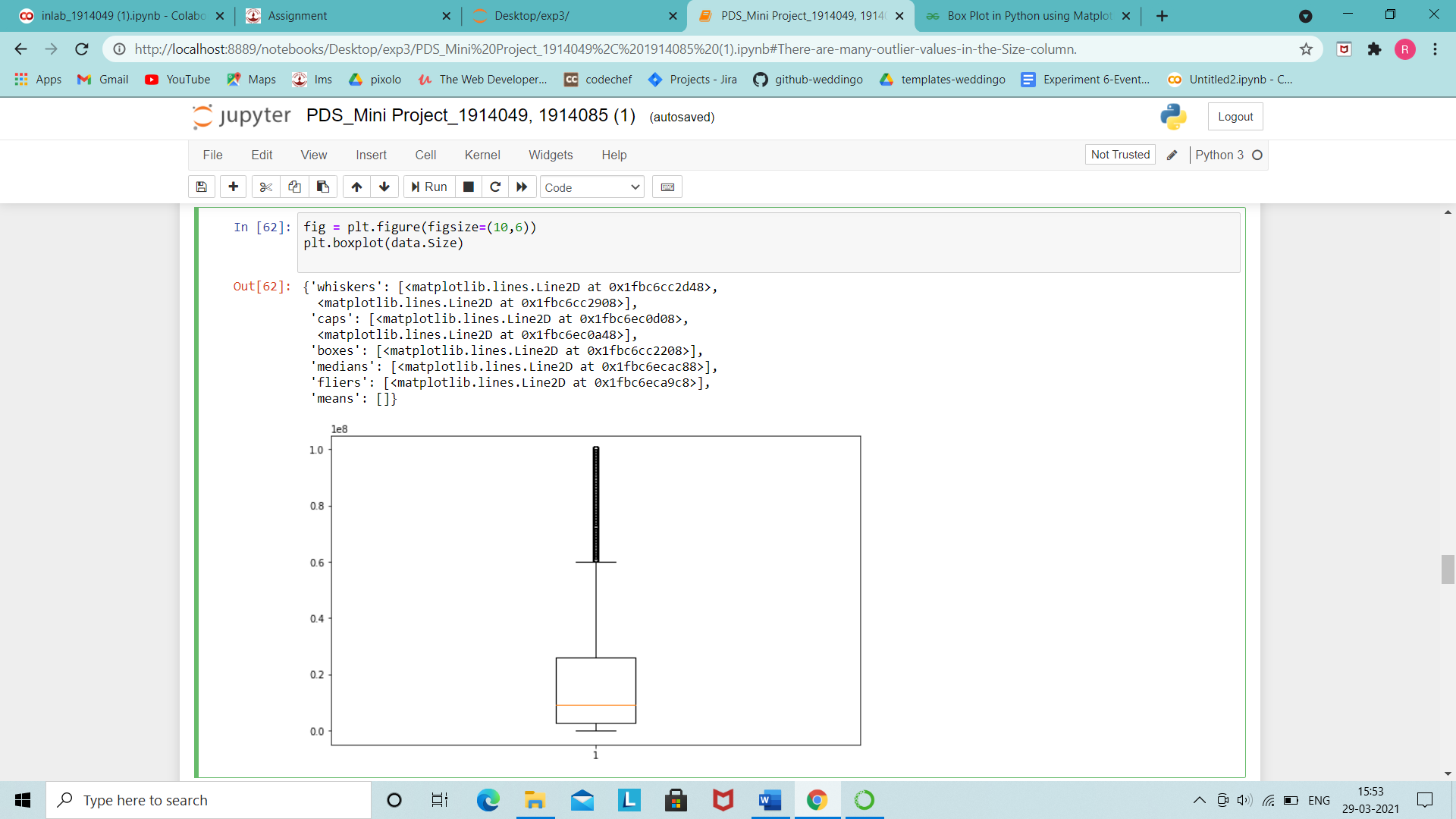


fig = plt.figure(figsize=(10,6))

plt.boxplot(data.Size)



### There are many outlier values in the Size column

# Android version which is most compatible with Apps on Play Store

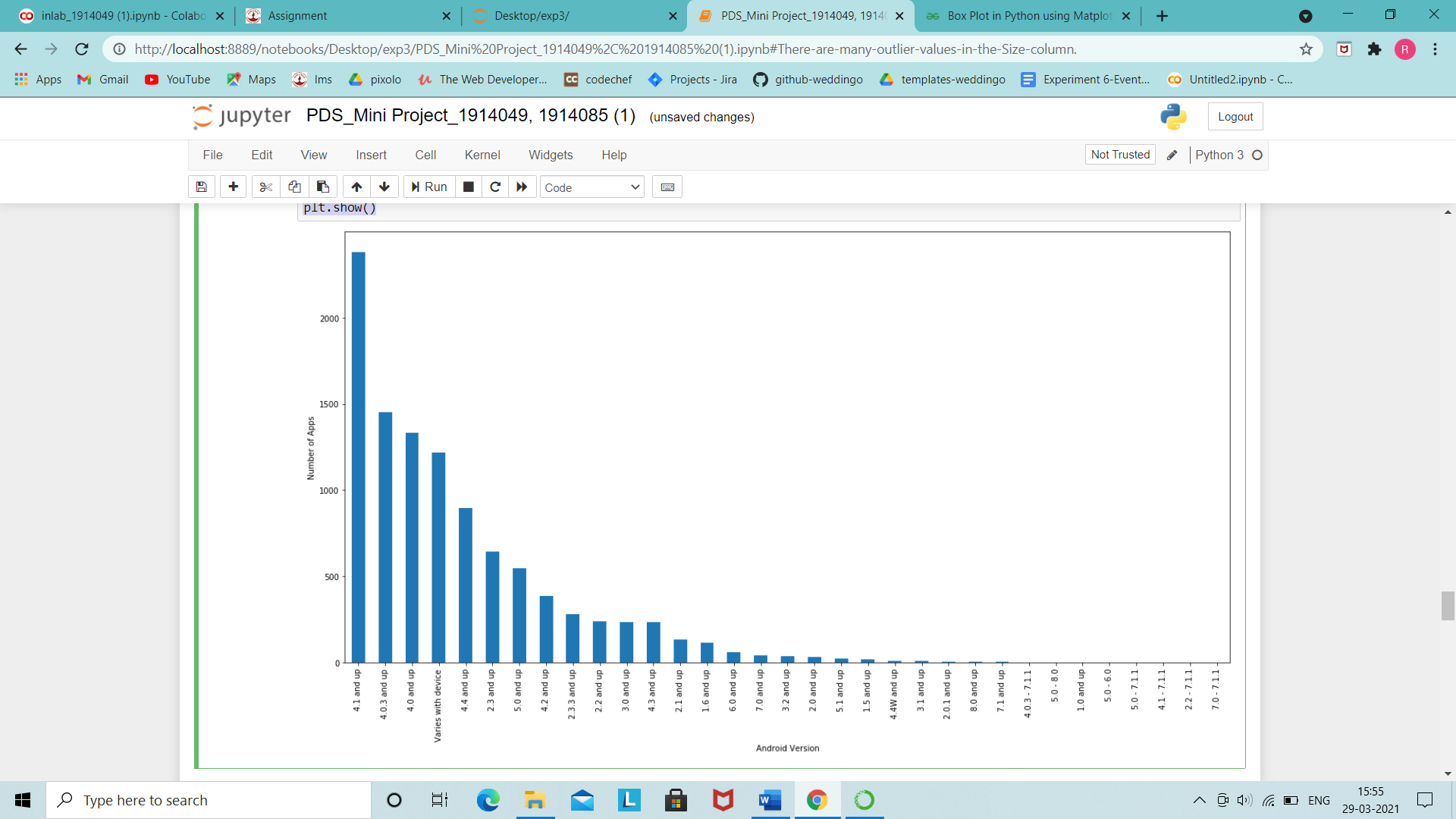
plt.figure(figsize = (20,10))

data['Android Ver'].value\_counts().plot(kind = 'bar')

plt.xlabel('Android Version')

plt.ylabel('Number of Apps')

plt.show()



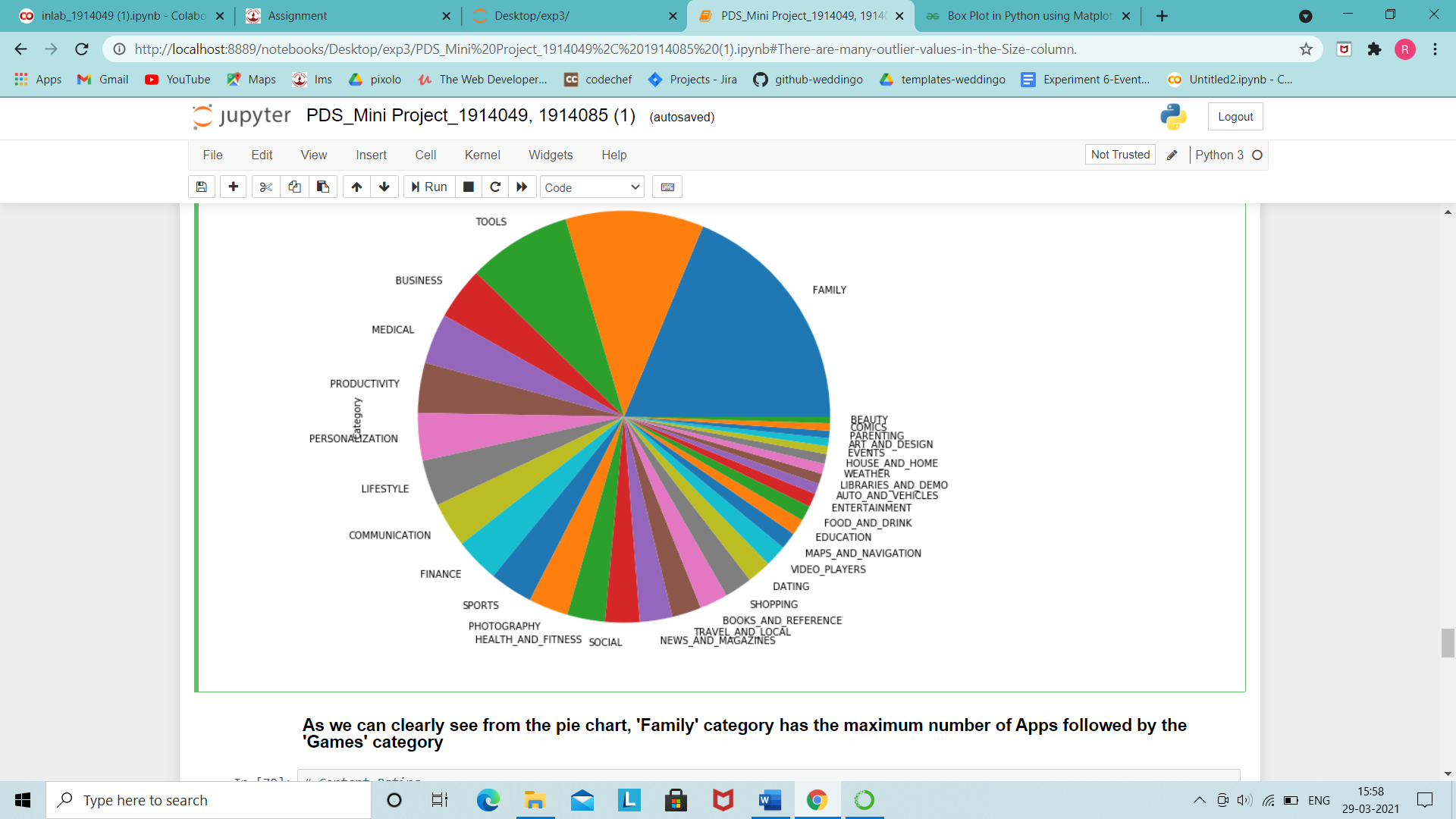
### As we can see most Apps are compatible with devices running Android 4.1 and aboce.

**# Most popular category**

**plt.figure(figsize=(40,10))**

**data['Category'].value\_counts().plot(kind='pie')**

**plt.show()**



### As we can clearly see from the pie chart, 'Family' category has the maximum number of Apps followed by the 'Games' category

**# Content Rating**

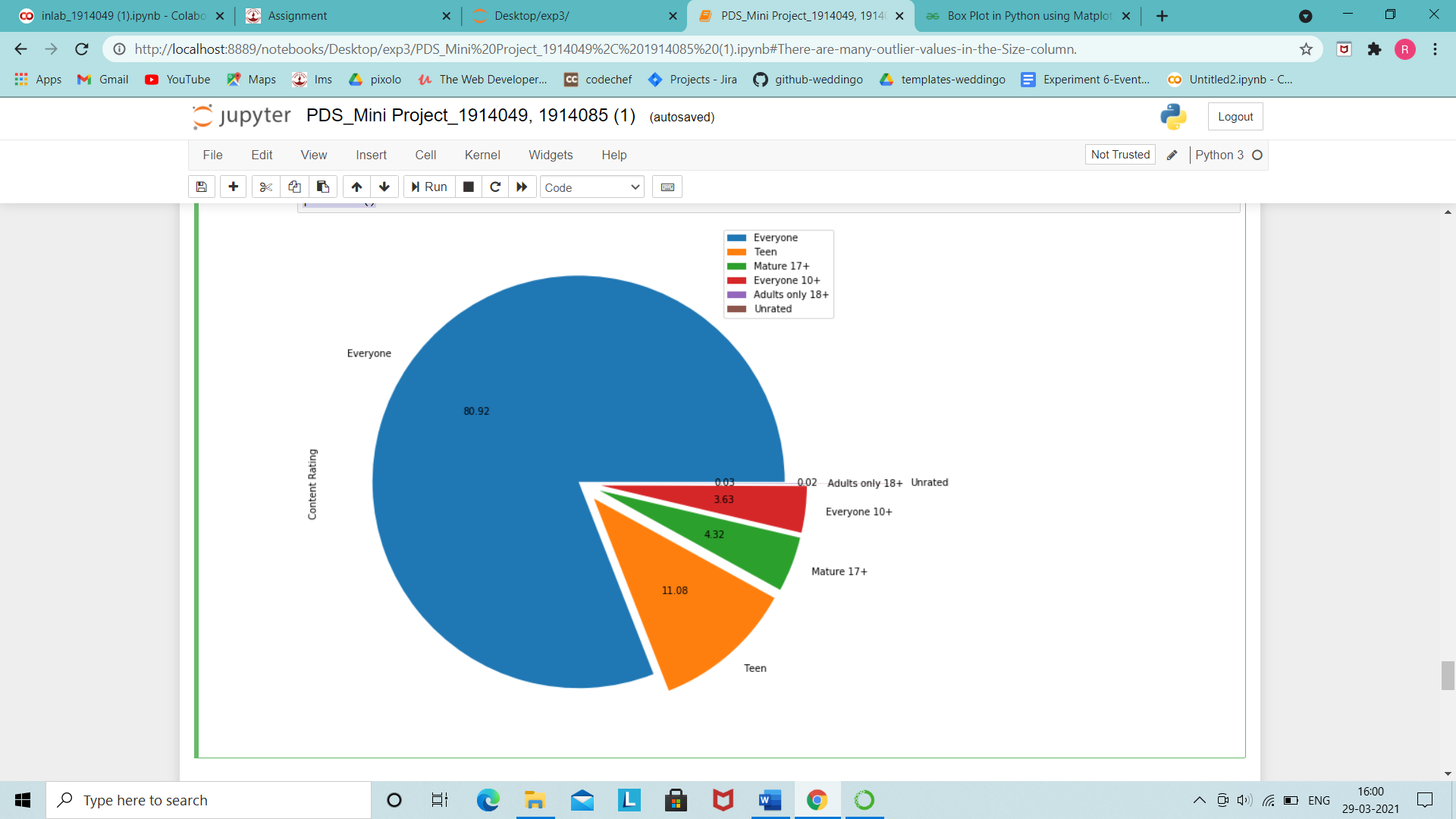
**plt.figure(figsize=(40,10))**

**explode=[0.01,0.1,0.1,0.1,0.1,0.5]**

**data['Content Rating'].value\_counts().plot(kind='pie',autopct="%.2f",explode=explode)**

**plt.legend()**

**plt.show()**



### This pie chart shows that maximum number of Apps on the Play Store are rated to be used by 'Everyone', followed by 'Teen', '17+' etc whereas hardly any apps are rated for 'Adults only' or 'Unrated'.

**# Comparing Type of Apps: 'Free' or 'Paid'**

**plt.figure(figsize=(10,10))**

**explode=[0.1,0]**

**data['Type'].value\_counts().plot(kind='pie',autopct="%.2f",explode=explode)**

**plt.legend()**

**plt.show()**

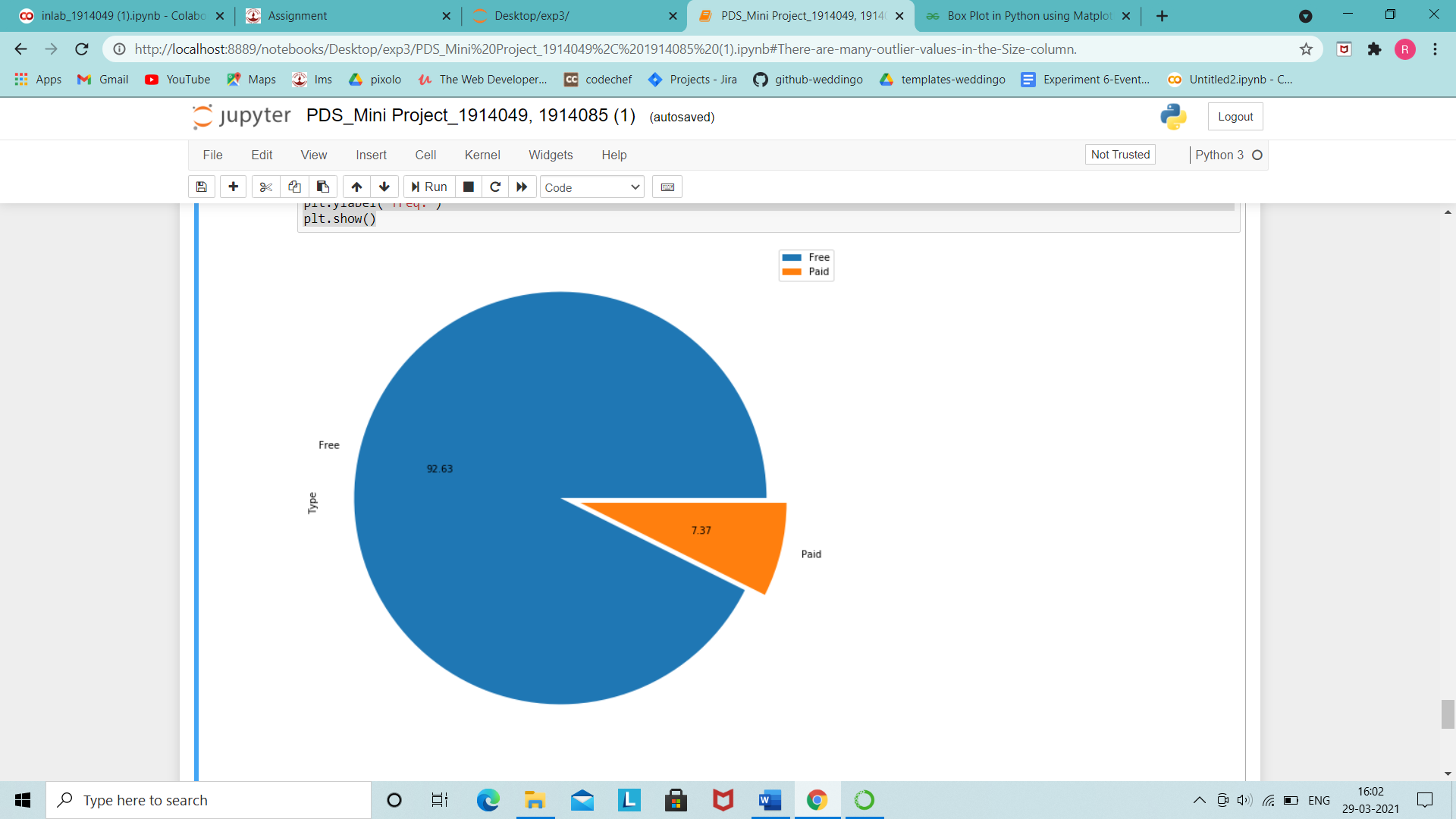
**plt.figure(figsize=(10,10))**

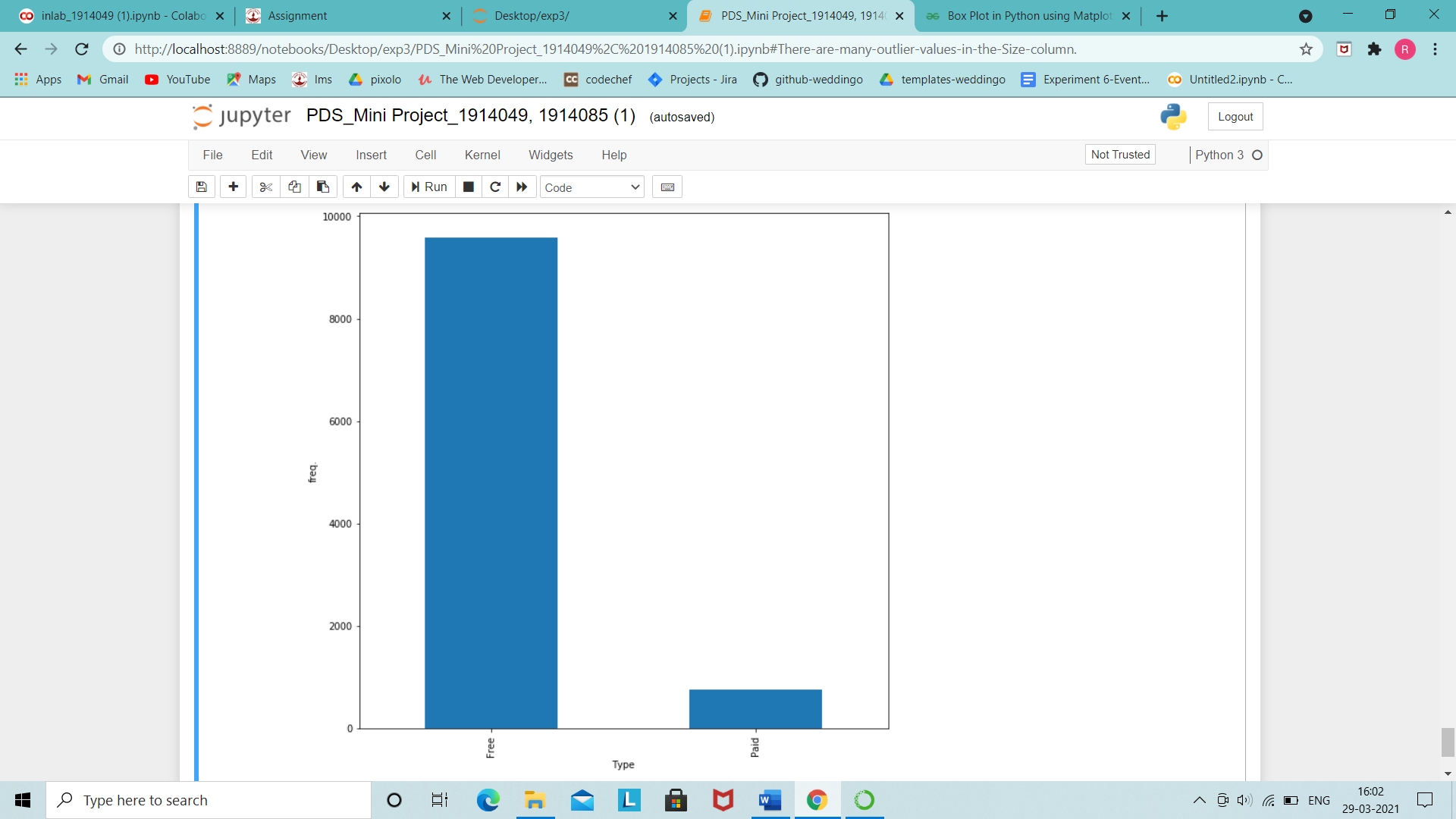
**data['Type'].value\_counts().plot(kind='bar')**

**plt.xlabel('Type')**

**plt.ylabel('freq.')**

**plt.show()**





### The graphs show that almost 93% of Apps are free on the Google Play store.



**Outcomes:**

Use python libraries like matplotlib, numpy, pandas, scipy for data visualization

and scientific-mathematical data computing.



**Conclusion:** (Conclusion to be based on the objectives and outcomes achieved)​

In this experiment we implemented data visualization using matplotlib and analysed our dataset by plotting various types of plots to derive useful conclusions.



References:

1. [https://pandas.pydata.org/pandas-docs/stable/user\_guide/visualization.htm](https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)​ [l](https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)