Data Visualisation in Python

# **Introduction to Matplotlib**

**Data visualisation** is an important skill to possess for anyone trying to extract and communicate insights from data. Great business narratives and presentations often stem from brilliant visualisations that convey the key ideas in a concise and aesthetic manner. In the field of machine learning, visualisation plays a key role throughout the entire process of analysis - to obtain relationships, observe trends and portray the final results as well. Therefore, it is imperative that you learn and master this tool which will aid you throughout this program.

This session will help you learn how to visualise data in Python using the **Matplotlib** library.

In the previous module, you learnt how to handle numerical data using the NumPy library. You could load data in the form of arrays and then perform operations on the same. In this session, we will try to visualise the arrays using another library in Python, namely, **Matplotlib**.

Data visualisation is a crucial step in the process of data analysis; in the upcoming video, let’s hear it out from Behzad how visualisation could play the most crucial role for your data toolkit.

Play Video

1569002

This session will cover the following topics:

* Creating and plotting graphs
* Different chart types
* Modification of charts for better understanding and presentation

## Guidelines for in-session questions

The in-video and in-content questions for this session are **not graded**. The graded questions will be provided later in a separate session.

## Lecture Notes for this Module

Please find the lecture notes pdf document for this module available below:

Lecture Notes - Matplotlib

Download

## People you will hear from in this session

**Subject Matter Expert**

[Behzad Ahmadi](https://www.linkedin.com/in/behzad-ahmadi)

Data Scientist at Walmart Labs

Behzad is a Doctor of Philosophy (PhD) in Electrical and Computer Engineering; Communication and Signal Processing from the New Jersey Institute of Technology. He has been working in the software engineering and data science field for more than 12 years. Behzad currently employs his machine learning skillset to create retail graphs for Walmart Labs.

[Mirza Rahim Baig](https://www.linkedin.com/in/rahim-baig/)Analytics Lead, Flipkart

Rahim, a BITS Pilani graduate, has around 9 years of experience in advanced analytics and machine learning. He is currently the Analytics Lead at Flipkart Pvt Ltd., an Indian electronic commerce company based in Bengaluru, India.

[S. Anand](https://www.linkedin.com/in/sanand0/)CEO, Gramener

Gramener is one of the most prominent data analytics and visualization companies in India. Anand, currently the CEO, was previously the Chief Data Scientist at Gramener and has extensive experience in management consulting and equity research.

# **The Necessity of Data Visualisation**

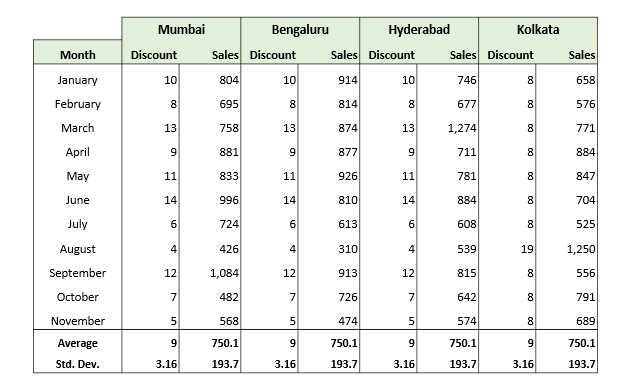
**“There are three kinds of lies: lies, damned lies, and statistics.” - Mark Twain**

Before we get to learn about the various nuances involved in data visualisation, it is essential to appreciate why it is so important to ‘look’ at the data from the perspective of plots and graphs. To begin with, it is difficult for the human eye to decipher patterns from raw numbers only. Sometimes, even the statistical information summarised from the data may mislead you to wrong conclusions. Therefore, you should visualise the data often to understand how different features are behaving. Let’s listen to Rahim as he demonstrates this idea using a brilliant example.

Play Video

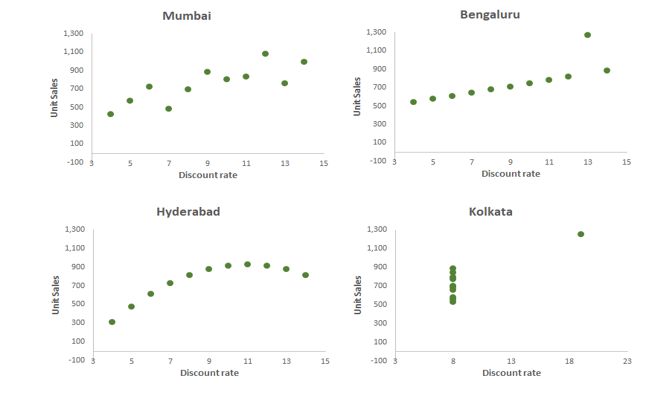
1569002

As explained in the video above, it is very easy to be deceived by the numbers and summary statistics. In the example that you saw, each of the branches had similar average sales and discount rates, and the corresponding standard deviations were similar as well, as shown in the table below.



**Table 1**

However, the patterns in the underlying data and the difference became apparent when visualised through appropriate plots.



Each of the branches had actually employed a different strategy to calculate its discount rate, and the sales numbers were also quite different across all of them. It is difficult to draw this type of insight and understand the difference between each of the branches using raw numbers alone; therefore, you should utilise an appropriate visualisation technique to ‘look’ at the data.

**References:**

The discount and sales example that you saw above is actually a modified version of a popular dataset called the [Anscombe’s Quartet](https://en.wikipedia.org/wiki/Anscombe%27s_quartet). As explained in the linked article (Anscombe's Quartet), the statistician Frances Anscombe constructed this example to counter the notion that **“numerical calculations are exact, but graphs are rough.”**

# **Visualisations - Some Examples**

In this segment, you'll be looking at some real-life examples of how data visualisation can help you derive insights out of data very quickly.

Play Video

1569002

You saw how data visualisation can improve data density and improve the amount of information being conveyed. Just imagine how difficult it would be to interpret a spreadsheet with the score of each inning of each batsman being recorded along with his strike rate. In this case, data visualisation can help you figure out many insights just by looking at the plot. You can see the visualisation below:

You can also play around with the visualisation [here](https://gramener.com/cricket/).

Now, let's see another example of where data visualisation can help you derive industry-relevant insights.

Play Video

1569002

You saw how a tree map diagram showed how multiple companies and sectors react to the budget. You can find the interactive graph [here](https://gramener.com/budget/?Year=2007).

An important point to remember is that visualisations should be accompanied by a voice or text narrative if possible - it improves the experience of the user drastically.

Next, let's see how visualisation can help in visual exploratory analytics. Here, you will see how data visualisation helped in understanding the connections between different software and clustering them together based on common features. You can find the visual [here](https://gramener.com/software/).

Play Video

1569002

**Report an error**

PREVIOUS

The Necessity of Data Visualisation

NEXT

Facts and Dimensions

# **Facts and Dimensions**

Graphics and visuals, when used intelligently and innovatively, can convey a lot more than what raw data alone can. Matplotlib serves the purpose of providing multiple functions to build graphs from the data stored in your lists, arrays, etc. So, let’s start with the first lecture on Matplotlib.

Before we start discussing different types of plots, you need to learn about the elements that help us create charts and plots effectively. There are two types data, which are as follows:

* Facts
* Dimensions

Facts and dimensions are different types of variables that help you interpret data better. Facts are numerical data, and dimensions are metadata. Metadata explains the additional information associated with the factual variable. Both facts and dimensions are equally important for generating actionable insights from a given data set. For example, in a data set about the height of students in a class, the height of the students would be a fact variable, whereas the gender of the students would be a dimensional variable. You can use dimensions to slice data for easier analysis. In this case, the distribution of height based on the gender of a student can be studied.

Identifying facts and dimensions among variables effectively will help you start the analysis of a given data set.

Question 2/2

Mandatory

Question 1

Correct

Question 2

Correct

In the next segment, you will start building graphical plots using Python. The first visualisation that you will try to create is a Bar Graph.

Question 1/2

Mandatory

#### **Facts and Dimensions**

Consider a bank having thousands of ATMs across India. In every transaction, the following variables are recorded:

* Withdrawal amount
* Account balance after withdrawal
* Transaction charge amount
* Customer ID
* ATM ID
* Date of withdrawal

Which among the following are fact variables (mark all that apply)?

Withdrawal amount

✓ Correct

Feedback:

Fact variable: It is numerical data

Account balance after a withdrawal

✓ Correct

Feedback:

Fact variable: it is numerical data

Transaction fee

✓ Correct

Feedback:

Fact variable: It is numerical data

Customer ID

ATM ID

Date of withdrawal

Your answer is Correct.

Question 2/2

Mandatory

#### **Dimensional Modelling**

What are the benefits of having dimension variables apart from facts?

More than one option may be correct.

Performing various types of analyses, such as sector-wise, country-wise or funding type-wise analyses.

✓ Correct

Feedback:

This activity needs metadata along with the fact data.

Summarising fact variables by calculating their sum, average, range, etc.

Extracting specific, useful information such as the total investment made in the automobile sector in India between 2014 and 2015.

✓ Correct

Feedback:

This activity needs metadata along with the fact data.

Your answer is Correct.

Attempt 2 of 2

Continue

In the next segment, you will start building graphical plots using Python. The first visualisation that you will try to create is a Bar Graph.

# **Bar Graph**

Plots are used to convey different ideas. For example, you can use certain plots to visualise the spread of data across two variables and other plots to gauge the frequency of a label. Depending on the objective of your visualisation task, you can choose an appropriate plot. As part of this session, you will learn how to select an appropriate plot. You can download the Jupyter Notebook attached below. The same notebook will be used in the demonstrations throughout the session.

In this segment, you will learn how to create the first plot: Bar Graph. Let’s watch the next video to understand the process of creating a bar graph.

Play Video

1569002

As you saw in the video above, the subpackage pyplot is used to build plots and graphs throughout this session. To load the subpackage, you need to run the following command:

**import** **matplotlib.pyplot** **as** **plt**

To recap, Matplotlib allows you to use a simple and intuitive workflow to create plots. The important Matplotlib commands used in the video above are as follows:

* plt.bar(x\_component, y\_component): Used to draw a bar graph
* plt.show(): Explicit command required to display the plot object

A bar graph is helpful when you need to visualise a numeric feature (fact) across multiple categories. In the example covered in the video, you plotted the sales amount (numeric feature) under three different product categories. Using the bar graph, you could easily distinguish between the performance of these categories.

Let’s watch the next video to learn how to add elements to our graph in order to make it more easily understandable.

Play Video

1569002

Matplotlib Code Along

Download

In the video above, you learnt the different ways in which you can modify your chart to make it more understandable. You can use the following code to add a title and labels to your graph:

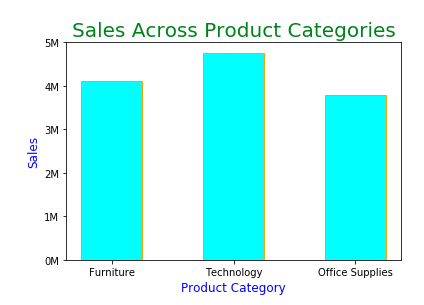
* plt.xlabel(), plt.ylabel(): Specify labels for the x and y axes
* plt.title(): Add a title to the plot object.

You can also try to make the charts more appealing by using different attributes such as font size and colour. Adding labels and a title to your plot helps the audience interpret the graphs easily and also relays the required information to the viewer.

You can use the attributes of plt.bar() to make the desired changes to the bars of a graph. For example, you can use the following code to change the values and ticks on the x and y axes of a graph:

plt.yticks(tick\_values, tick\_labels)

After making all the changes demonstrated in the video, your graph will look like the one given below.



Bar Graph

Now that you have understood the basics of a bar plot, answer the following questions.

Question 1/2

Mandatory

#### **Bar Graph**

How will you change the width of the bars in a bar graph?  
Check the Maptlotlib [documentation](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar) for bar graphs.

matplotlib.pyplot.bar(x, y, breadth = **0.8**)

matplotlib.pyplot.bar(x, y)

bar\_width = **0.8**

matplotlib.pyplot.bar(x, y, width = **0.8**)

✓ Correct

Feedback:

'width' is the correct attribute for changing the width of the bars in a bar graph.

matplotlib.pyplot.bar(x, y)

change\_bar\_width = **0.8**

Your answer is Correct.

Question 2/2

Mandatory

#### **Bar Graph**

Is it possible to have a separate colour for each category in a bar graph?

Yes

✓ Correct

Feedback:

Colours can be provided as a list to the matplotlib.pyplot.bar function under the attribute ‘color’.: matplotlib.pyplot.bar(x, y, color = [‘red’, ‘blue’, ‘green’]) [Note: if there are more than three bars, the colours will start repeating themselves.]

No

✕ Incorrect

Feedback:

Check the Maptlotlib [documentation](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar) for bar graphs.

Your answer is Wrong.

In this segment, you learnt how to build a bar graph and add or modify the required elements within it. In the next segment, you will learn about another visualisation: Scatter Plot.

# **Scatterplot**

**Scatter plot**, as the name suggests, displays how the variables are spread across the range considered. It can be used to identify a relationship or pattern between two quantitative variables and the presence of outliers within them.

Let’s watch the next video to understand how a scatter plot can be useful when you are dealing with two quantitative variables

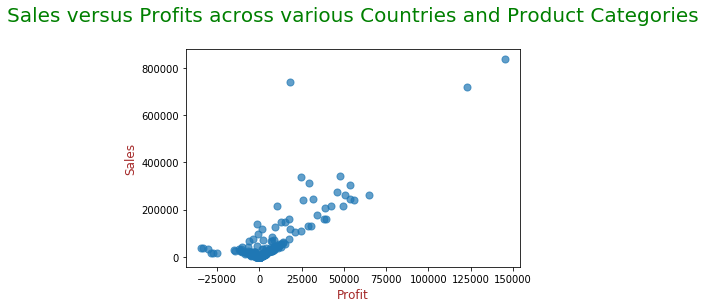
Play Video

1569002

You can use the following command to build a scatter plot:

plt.scatter(x\_axis, y\_axis)

Using this command, the data points will spread across the graph corresponding to the values on the y and x axes. The plot that was developed in the demonstration is given below.



Take a look at the rightmost point in the plot. It represents a product that made a profit of 1,50,000 units when the sales generated were 8,00,000 units. Similarly, all the points in the plot represent a product and its profit and sales values. Just by looking at the chart, can you find the products that are more lucrative to trade than others? Yes, a lucrative product has high profit value with preferably low sales value, that is the points to the right side in the plot preferably also towards the bottom.

Now that you have learnt how to read a scatter plot, let’s proceed to more complex features of a scatter plot. Matplotlib offers a feature that allows you to incorporate a categorical distinction between the points plotted on a scatter plot. Using this feature, you can colour-code the points based on the category and distinguish them accordingly. Let’s watch the next video to learn how to colour-code points on a scatter plot.

Play Video

1569002

You can run the scatter function with the following attributes to specify the colours and labels of the categories in a data set:

plt.scatter(x\_axis, y\_axis, c = color, label = labels)

Here, all the information (x\_axis, y\_axis, colour, labels) need to be provided in the form of a list or an array. You can use this command to assign colours to categories and distinguish them accordingly.   
   
Another feature of a scatter plot allows you to use labels to further distinguish points over another dimension variable. Suppose you have the array ‘country’, which indicates the country where the sales were generated. Now you want to highlight the points belonging to a particular country in the previous scatter plot.

You can use the following command to add a note (annotate) with a point in the scatter plot:

plt.scatter(profit[product\_category == "Technology"], sales[product\_category == "Technology"],

c= 'Green', alpha= **0.7**, s = **150**, label="Technology" )

plt.scatter(profit[product\_category == "Office Supplies"], sales[product\_category == "Office Supplies"],

c= 'Yellow', alpha= **0.7**, s = **100**, label="Office Supplies" )

plt.scatter(profit[product\_category == "Furniture"], sales[product\_category == "Furniture"],

c= 'Cyan', alpha= **0.7**, s = **50**, label="Furniture" )

**for** xy **in** zip (profit[country == "India"], sales[country == "India"]):

plt.annotate(s = "India", xy = xy)

# Adding and formatting title

plt.title("Sales versus Profits across various Countries and Product Categories**\n**", fontdict={'fontsize': **20**, 'fontweight' : **5**, 'color' : 'Green'})

# Labeling Axes

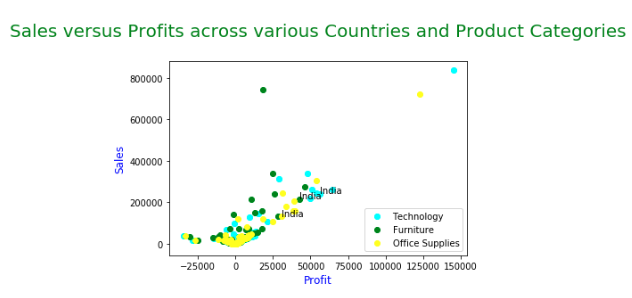
plt.xlabel("Profit", fontdict={'fontsize': **12**, 'fontweight' : **5**, 'color' : 'Brown'})

plt.ylabel("Sales", fontdict={'fontsize': **12**, 'fontweight' : **5**, 'color' : 'Brown'})

plt.legend()

plt.show()

After using the command to add a note, your scatter plot will look like the one given below.



Scatter Plot

As you can see in the figure above, the products that were traded in India are marked. This is how the annotate statement that was added to a point in the scatter plot helps you distinguish the data points.

In this segment, you learnt how a scatter plot helps you visualise two numeric variables. Attempt the following questions to cement the concepts that were covered in this segment.

Question 1/1

Mandatory

#### **Scatterplot**

Select the cases where a scatterplot would be helpful in generating insights.  
(More than option one can be correct.)

To check whether a relationship exists between the age of a person and their income.

✓ CorrectYou missed this!

Feedback:

A scatterplot shows the relationship between two sets of data. Hence, this is the correct option.

To check whether there are any irregular entries in the data range.

To check whether stock prices are positively related to the profit of a company.

✓ CorrectYou missed this!

Feedback:

Since a scatterplot helps to visualise the relationship between two sets of data, it also reveals whether they are related positively or negatively.

To understand the distribution of the salaries of the employees in a company.

✕ Incorrect

Feedback:

The distribution of data can be better visualized using distribution plot which requires one input, whereas a scatterplot needs two variables as input.

Your answer is Wrong.

Attempt 2 of 2

Continue

Question 1/1

Mandatory

#### **Scatterplot**

Which attribute helps set the transparency of points in a scatterplot?  
Check the Maptlotlib [documentation](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html?highlight=scatterplot) for scatterplots.

weight

marker

color

alpha

✓ Correct

Feedback:

This is the correct attribute. You can assign a value between 0 (transparent) and 1 (opaque).

Your answer is Correct.

Matplotlib also offers multiple features to make these plots as descriptive as possible using the different dimension variables associated with the plot.bar() method.

In the next segment, you will learn about another set of graphs: Line Graph and Histogram.

# **Line Graph and Histogram**

In this segment, you will learn about two new visualisation charts, which are as follows:

* Line graph
* Histogram

A **line graph** is used to present continuous time-dependent data. It accurately depicts the trend of a variable over a specified time period. Let’s watch the next video to learn how to plot a line chart using the Matplotlib library.

Play Video

1569002

You can use the following command to plot a line graph:

plt.plot(x\_axis, y\_axis)

Remember to be careful while using the plt.plot function. This function also helps you create a scatter plot when you tweak the syntax and specify the markers. Try to run the following code to understand the difference between the outputs of the function:

y = np.random.randint(**1**,**100**, **50**)

plt.plot(y, 'ro') # ‘ro’ represents color (r) and marker (o)

(if you are getting an error, check the quotation marks.)

If you specify the colour and marker separately, then you will get a line plot with the points marked. Try to run the following the code for this:

plt.plot(y, 'red', marker = 'o')

A line graph can be helpful when you want to identify the trend of a particular variable. Some key industries and services that rely on line graphs include financial markets and weather forecast. Although you successfully created a line chart in the previous video, you can make certain visual improvements to create a chart that is more easily understandable. Let’s watch the next video to learn about these visual modifications.

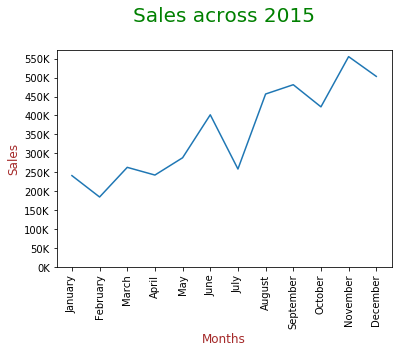
Play Video

1569002

In the video above, you learnt how to rotate the tick labels on the axes using the following command:

plt.yticks(rotation = number) #could do for xticks as well

After running the command, the chart will look like the one given below.



Line Graph

As you can see in the diagram, the x-ticks and y-ticks are much more readable.

To further improve the readability of the chart, you can add markers to the data points. Let’s watch the next video to learn how to make modifications to add data labels.

Play Video

1569002

In the video above, you learnt how to use the annotate method to add data labels to the plot. The code given below was used in the previous video.

plt.plot(months, sales)

# Adding and formatting title

plt.title("Sales across 2015**\n**", fontdict={'fontsize': **20**, 'fontweight' : **5**, 'color' : 'Green'})

# Labeling Axes

plt.xlabel("Months", fontdict={'fontsize': **12**, 'fontweight' : **5**, 'color' : 'Brown'})

plt.ylabel("Sales", fontdict={'fontsize': **12**, 'fontweight' : **5**, 'color' : 'Brown'} )

ticks = np.arange(**0**, **600000**, **50000**)

labels = ["{}K".format(i//**1000**) **for** i **in** ticks]

plt.yticks(ticks, labels)

plt.xticks(rotation=**90**)

**for** xy **in** zip(months, sales):

plt.annotate(s = "{}K".format(xy[**1**]//**1000**), xy = xy, textcoords='data')

plt.show()

After running this code, your plot will look like the one given below.



In the earlier segment on scatter plot, you used the annotate method to add data labels to a scatter plot. Similarly, the annotate method can be used to add data labels to graphs as well. Now that you know the basics of line graphs, attempt the following questions.

Question 1/1

Mandatory

#### **Specifying Line Style**

Let's say 'x' and 'y' are two lists that you want to plot. You have initialised matplotlib.pyplot as plt. Now, you want to plot a blue line with cross (x) as markers for your points. What is the function call for this?  
For your reference, here is the [official documentation](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) of the Pyplot API.

plt.plot(x, y, 'bo') plt.show()

plt.plot(x, y, 'bO') plt.show()

plt.plot(x, y, 'b', marker = 'x') plt.show()

✓ Correct

Feedback:

The syntax for plotting a line graph is plt.plot(x\_component, y-component, 'graph\_features'). You have to specify the marker as a separate attribute to plot a line graph.

plt.plot(x, y, 'b', marker = 'cross') plt.show()

Your answer is Correct.

## Histograms

A **histogram** is a frequency chart that records the number of occurrences of an entry or an element in a data set. It can be useful when you want to understand the distribution of a given series. Let’s watch the next video to learn how to plot a histogram.

Play Video

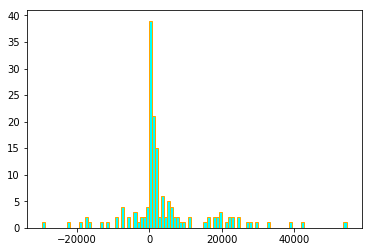
1569002

As shown in the video above, you can use the following command to plot a histogram:

plt.hist(profit, bins = **100**,edgecolor='Orange',color='cyan')

plt.show()

After running this code, your histogram will look like the one given below.



The x-ticks in the histogram above are not very informative. Let's try to add more detailed x-ticks so that the data is more readable. In the next video, you will learn how to add more information to classes and x-ticks.

Play Video

1569002

In the video above, you learnt how to use the hist() function to add more information to classes and x-ticks. Now that we have covered the basics of histograms, attempt the following questions.

Question 1/1

Mandatory

#### **Histogram**

You are provided with a list of weights of individuals stored as float:  
list\_1 = [48.49, 67.54, 57.47, 68.17, 51.18, 68.31, 50.33, 66.7, 45.62, 43.59, 53.64, 70.08, 47.69, 61.27, 44.14, 51.62, 48.72, 65.11]

You want to check which bucket has maximum entries when divided into the following four groups:

* [40-50)
* [50-60)
* [60-70)
* [70-80)

Which code will help you find the correct answer? For your reference, here is the [official documentation](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist) of the Pyplot API.

plt.hist(list\_1, edgecolor = 'white')

plt.hist(list\_1, bins = **4**, edgecolor = 'white')

plt.hist(list\_1, bins = **4**, range =[**40**, **80**], edgecolor = 'white')

✓ Correct

Feedback:

This is the correct answer. You are provided with four bars representing the desired ranges.

plt.hist( list\_1, range =[**40**, **80**])

Your answer is Correct.

Attempt 1 of 2

Continue

In the next segment, you will learn about another plot, namely, Box Plot.

## Additional Resources

You can also use line graphs to plot and visualize time series data. You can learn more about plotting time series data [here](https://www.youtube.com/watch?v=_LWjaAiKaf8).

# **Box Plot**

**Box plots** are quite effective in summarising the spread of a large data set into a visual representation. They use percentiles to divide the data range.

The percentile value gives the proportion of the data range that falls below a chosen data point when all the data points are arranged in the descending order. For example, if a data point with a value of 700 has a percentile value of 99% in a data set, then it means that 99% of the values in the data set are less than 700.

Let’s watch the next video to learn more about box plots.

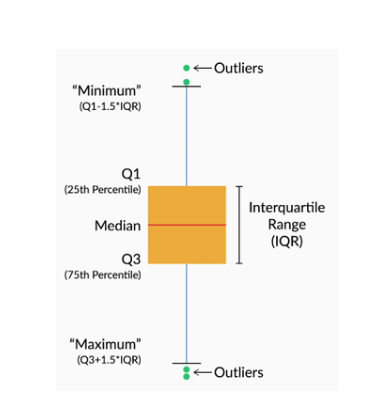
Play Video

1569002

You can use the following command to create a box plot in Python using Matplotlib:

plt.boxplot([ list\_1, list\_2])

The figure below shows a typical box plot with explanations for each element in its construction.

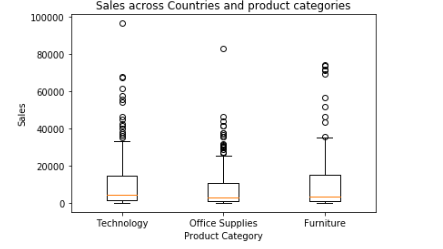


Box Plot

Box plots divide the data range into three important categories, which are as follows:

* **Median value:** This is the value that divides the data range into two equal halves, i.e., the 50th percentile.
* **Interquartile range (IQR):** These data points range between the 25th and 75th percentile values.
* **Outliers:** These are data points that differ significantly from other observations and lie beyond the whiskers.

The box plot given below was constructed in the video.



Box Plot

As you can see, there are quite few outliers in the given data set.

Attempt the quiz given below to test your understanding of the box plots.

Question 1/3

Mandatory

#### **Box Plot**

Find the interquartile range for the box plot created over the following list:  
list\_1 = [48.49, 67.54, 57.47, 68.17, 51.18, 68.31, 50.33, 66.7, 45.62, 43.59, 53.64, 70.08, 47.69, 61.27, 44.14, 51.62, 48.72, 65.11]  
[Hint: use np.percentile to find the percentile values]

48.55-66.30

✓ Correct

Feedback:

This is the correct answer. The IQR can be calculated by finding the 25th and 75th percentile values.

48.55-52.63

✕ Incorrect

Feedback:

Check the percentile values for the IQR.

37.68-66.30

37.68-52.63

Your answer is Wrong.

Question 2/3

Mandatory

#### **Types of Plots**

Which of the following plots can be used to show the relationship between two quantitative variables?

1. Box plot  
2. Line plot  
3. Scatter plot  
4. Histogram

1 and 2

✕ Incorrect

Feedback:

A box plot shows the distribution of points with respect to different categories.

A line plot shows how one variable varies with a variation in another (could be time or any other quantitative variable).

2 and 3

✓ Correct

Feedback:

Correct Answer! A line plot and a scatterplot essentially plot the x-y relationship between two quantitative variables. They show how one variable changes with respect to another quantitative variable. To find a relationship using a line plot, one additional step that you have to complete is to first sort the elements of x-axis.

2 and 4

1 and 4

Your answer is Wrong.

Question 3/3

Mandatory

#### **Deciding the Type of Plot to Use**

You have some data of credit card transactions. In this data, you're trying to find transaction values that are way outside the usual data range, in order to investigate some fraud.  
Which type of plot will you use in this case?

Line Chart

Scatter Plot

Histogram

Box Plot

✓ Correct

Feedback:

A box plot will be helpful here as it easily segregates the values as outliers.

Your answer is Correct.

Attempt 1 of 2

Continue

In the next segment, you will learn about some additional Matplotlib features that can help make your visualisations more effective.

# **Subplots**

In the previous segments, you learnt the basic ways to create plots.

Sometimes, it is beneficial to draw different plots on a single grid next to each other to get a better overview of the data set. For example, suppose you have some data on e-commerce purchases. If you want to analyse the number of purchases across different categories, you can create multiple bar charts for each category, for example, one for male buyers and another for female buyers. These two charts, when placed next to each other, make it easy for you to compare the buying patterns of the male and female consumers.

Different plots presented in a single plot object are commonly referred to as **subplots**. Let's watch the next video to learn how to create subplots inside a single plot in Matplotlib.

Play Video

1569002

In the video above, you learnt how to create multiple plots in a single graph. Behzad added plots for Asia, USCA and the Asia-Pacific region. Let's watch the next video and continue filling up the same plot, and learn about some new features.

Play Video

1569002

To recap, you can use the following Matplotlib command to create subplots in Python:

* fig, ax = plt.subplots(): It initiates a figure that will be used to comprise multiple graphs in a single chart.

Subplots are a good way to show multiple plots together for comparison. In the video above, you learnt how to plot different categories together in the same chart. Subplots offer the ability to create an array of plots, and you can create multiple charts next to each other to make it look like the elements of an array.

Let’s watch the next video to understand this with the help of an example.

Play Video

1569002

You can use the following command to create an array of plots.

* plt.subplot(nrow, ncol, x): It creates a subplot. 'nrow' and 'ncol' are the dimensions of the array inside the figure, and 'x' is the position in the array.  
  (You can visit this [web page](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.subplots.html) to understand the attributes associated with the subplot method in detail.)
* In plt.subplot()method, the numbering of subplots starts from the top-left element of the grid and moves rightward along each row. The numbering then continues to the next row from left to right. For example, suppose you have created a grid of nine subplots. The numbering would look like this:

| Subplot 1 | Subplot 2 | Subplot 3 |
| --- | --- | --- |
| Subplot 4 | Subplot 5 | Subplot 6 |
| Subplot 7 | Subplot 8 | Subplot 9 |

* Suppose you need to access the subplot present in the second row and the second column (i.e., 'Subplot 5' in the table above). You can do so by using the following command:

plt.subplot(**3**, **3**, **5**)

In this segment, you learnt how to add multiple plots to the same graph and create an array of plots in the same picture. However, as you saw in an earlier video, the picture was quite small in size and the information was not easily understandable. Let’s watch the next video to learn how to modify the size of the overall plot.

Play Video

1569002

In the video above, you learnt how to use the set\_size\_inches() function to modify the size of the plot. You also learnt how to combine both the two techniques, plotting on the same graph and plotting an array of subplots. With this, we have covered all the basics of subplots. Let’s attempt the following question.

Question 1/1

Mandatory

#### **Arrays of Plots**

As you learnt, in plt.subplot(), the numbering starts from the top-left element of the grid and moves rightward along each row. Out of the following subplots, which one does NOT represent a plot that is part of the second column from the left in the array of subplots?

plt.subplot(**4**,**4**,**2**)

plt.subplot(**4**,**4**,**14**)

plt.subplot(**3**,**3**,**7**)

✓ Correct

Feedback:

This is the correct answer as the code represents row 3, column 1.

plt.subplot(**3**,**3**,**8**)

Your answer is Correct.

## Additional Resources:

Here is a [video tutorial by Corey Shafer](https://www.youtube.com/watch?v=XFZRVnP-MTU) that can provide you additional example on subplot input parameters

# **Choosing Plot Types**

Before this segment, you have seen a lot of different types of plots. Each of the plot types is good at communicating a specific type of information. Which means, in certain situations, certain plot types are preferred over the others. So, how do you select the best possible plot type in a given situation? To answer this question, you need to first define the objective of creating a plot. A good visualisation, along with the right type of graph, presents the relationship between different variables effectively and allows you to analyse them at a quick glance. Let’s take a look at some principles that can help you select the right type of chart. Let's hear about selecting a plot type from Behzad.

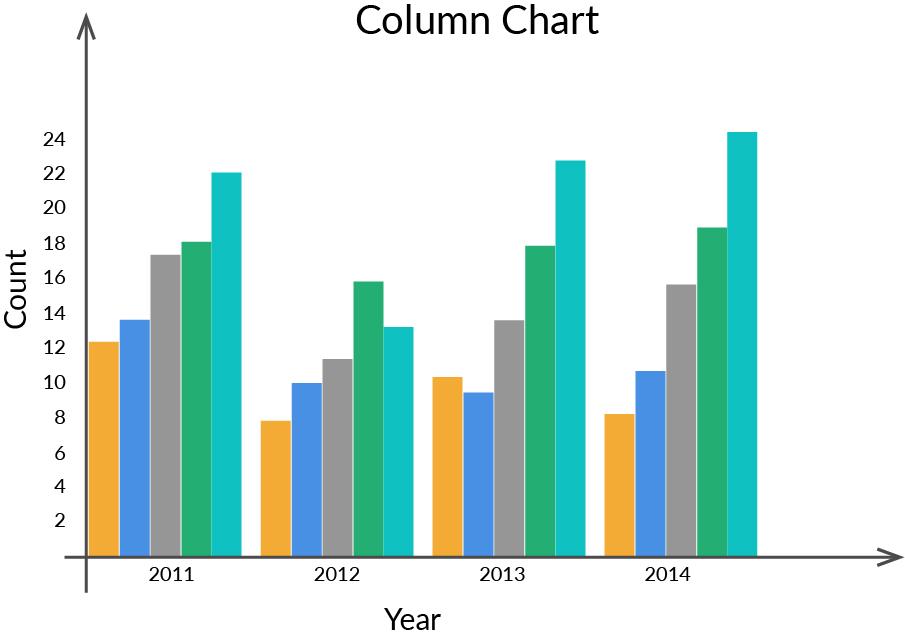
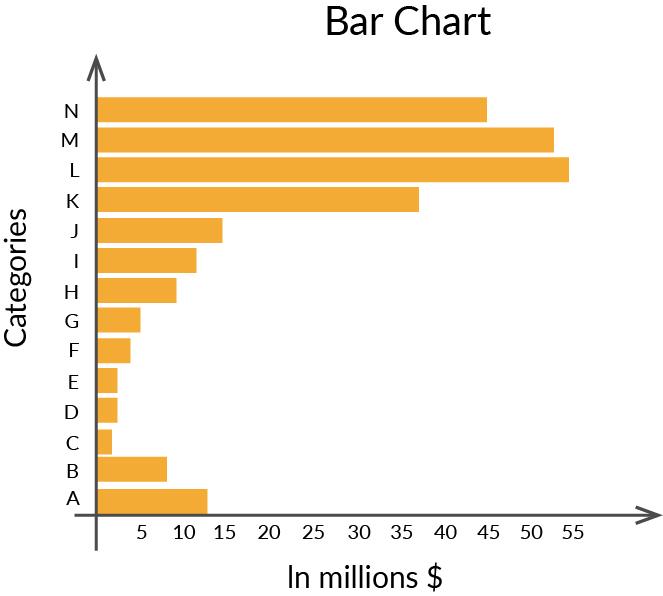
Play Video

1569002

Let's summarise the points discussed in the video.

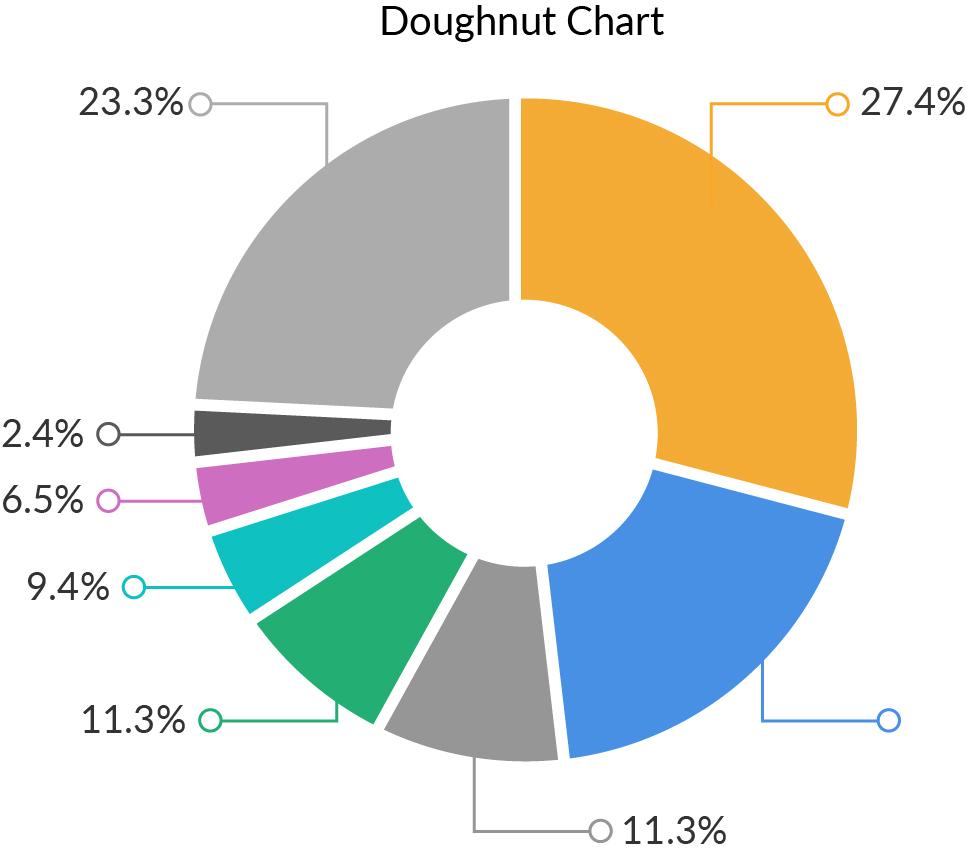
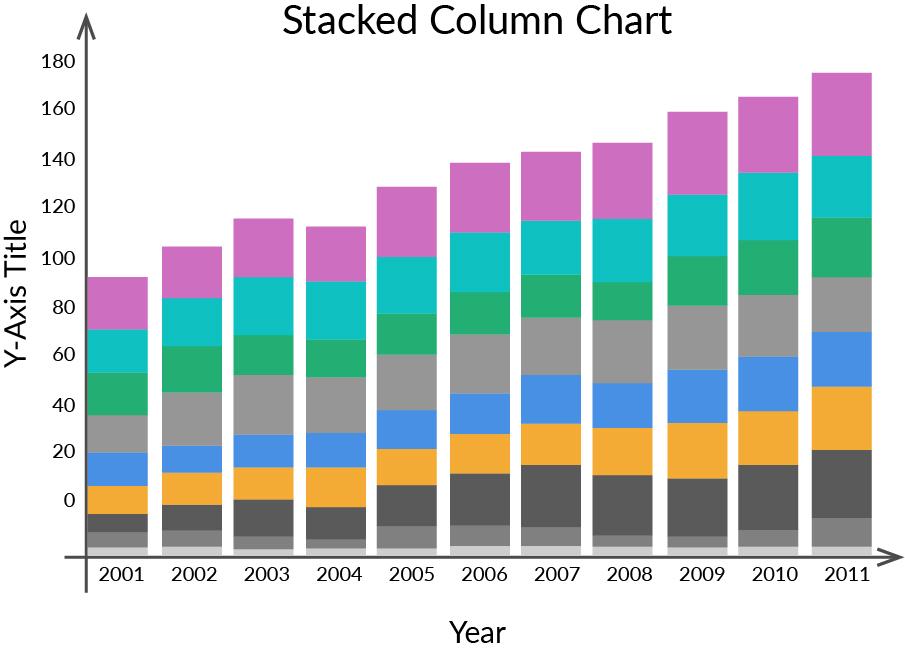
## **Comparison**

These charts can be used when you want to compare one set of values with other sets of values. The objective is to differentiate one particular set of values from the other sets, for example, quarterly sales of competing phones in the market. The following two types of charts are used to show a comparison:

* Column chart  
     
  
* Bar chart  
  

## **Composition**

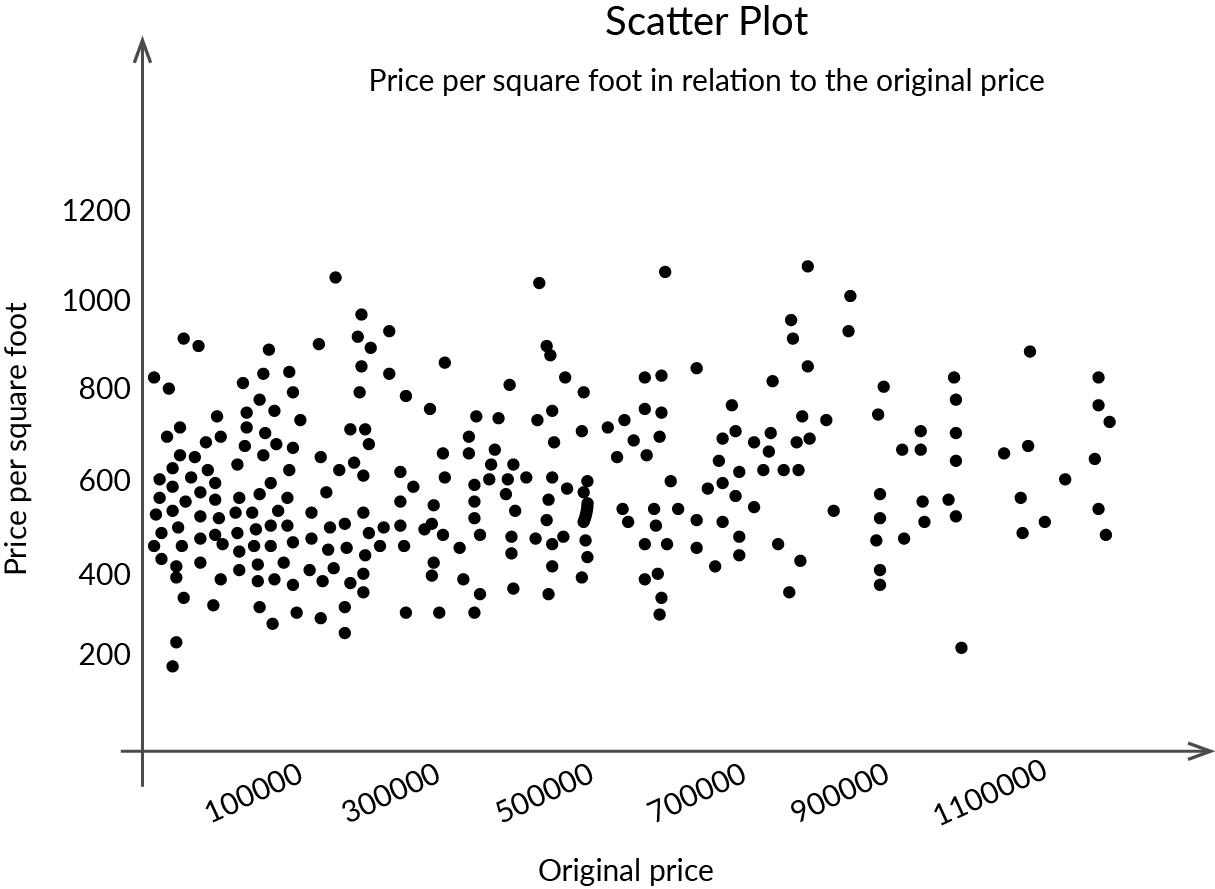
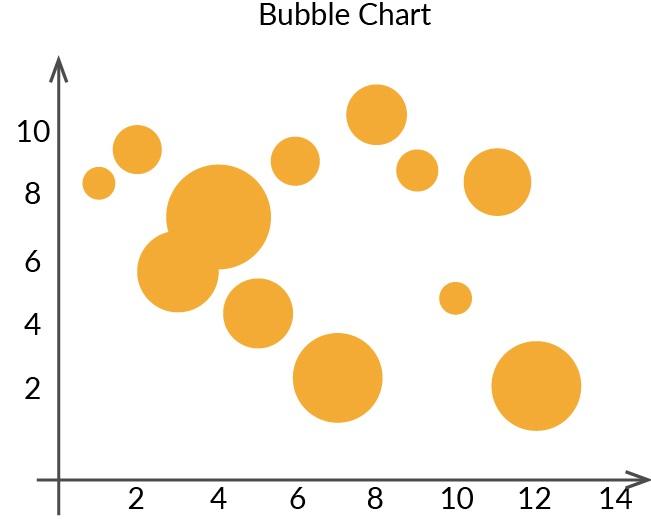
You would need to use a composition chart to display how the various elements make up the complete data. Composition charts can be static, which shows the composition at a particular instance of time, or dynamic, which shows the changes in the composition over a period of time. Two of the popular composition charts are as follows:

* Pie/ Doughnut Chart  
  
* Stacked Column chart  
     
  

The pie chart is by far the most common way to represent static composition, while the stacked column chart can be used to show the variation of composition over a period of time.

## **Relationship**

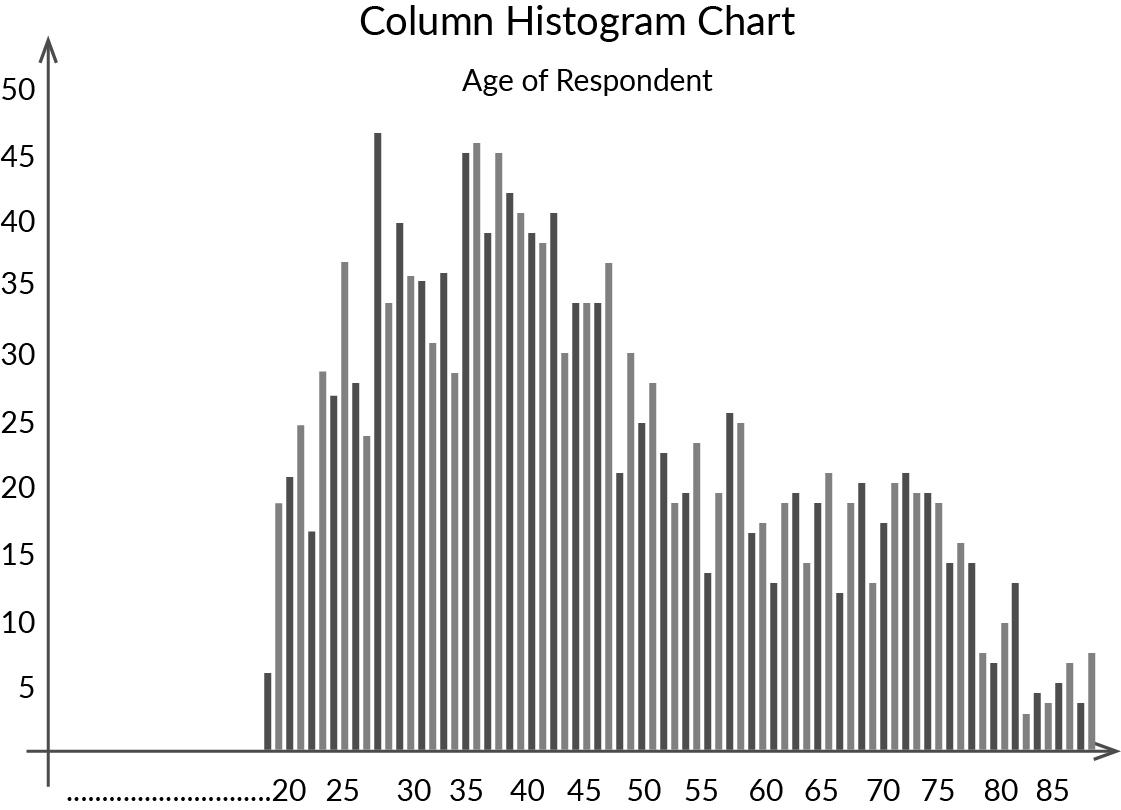
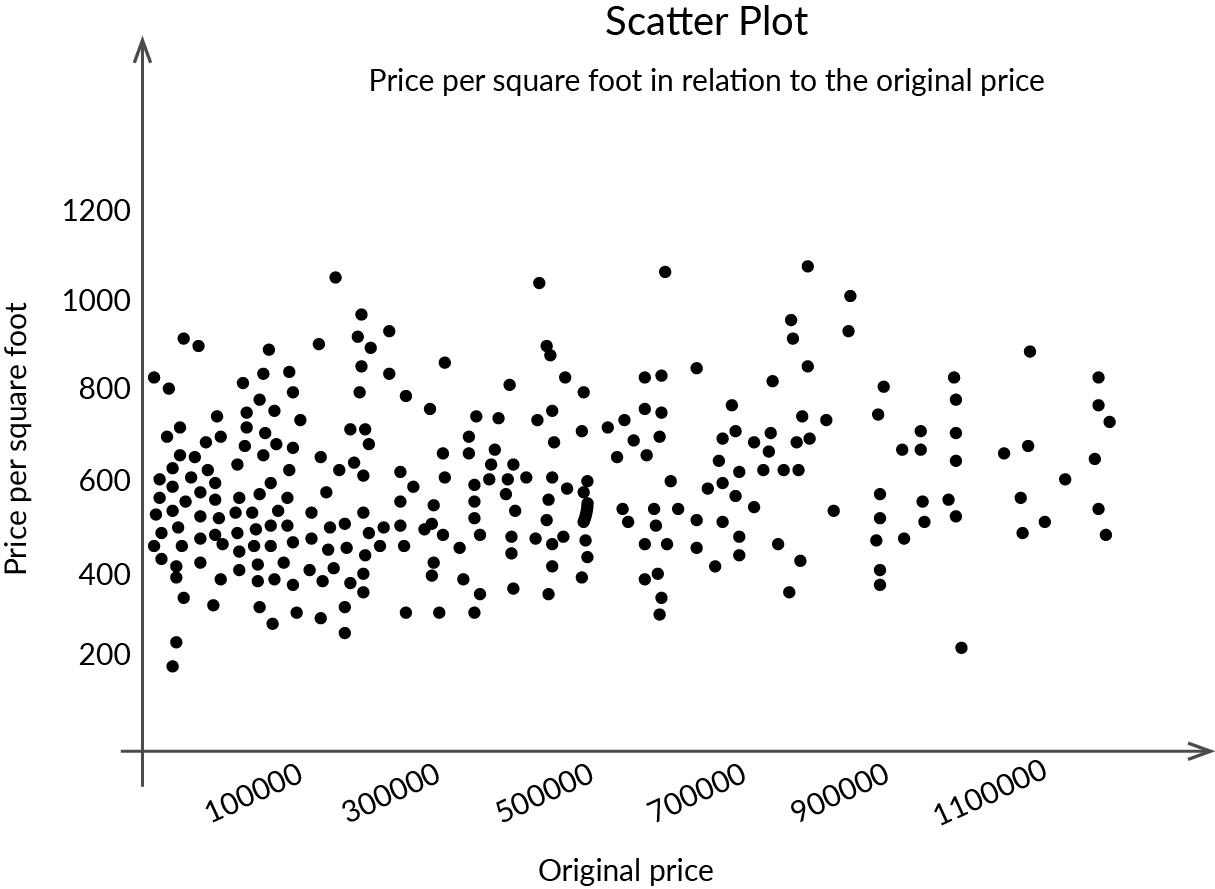
A relationship chart helps in visualising the correlation between variables. It can help in answering questions such as ‘Is there a correlation between the amount spent on marketing and the sales revenue?’ and ‘How does the gross profit vary with the change in offers?’. Two of the most common types of charts used to visualise relationships between variables are as follows:

* Scatter plot  
     
  
* Bubble Plot  
     
     
     
  

A scatter plot can help correlate two variables, whereas a bubble chart adds one more dimension, i.e., the size of the bubble (usually indicative of the frequency of occurrence of that particular data point)

## Distribution

A distribution chart tries to answer the question ‘How is the data distributed?’. For example, suppose you asked everyone their age in a survey. Using a distribution chart will help you visualise the distribution of ages in the data set. The distribution can be over a variable, or it can also be over a period of time. Two of the most commonly used charts for visualising distribution are as follows:

* Histogram  
     
    
     
  2. Scatter plots   
     
     
  

Histograms are quite good at displaying the distribution of data over intervals, whereas scatter plots are good at visualising the distribution of data over two different variables.

In the discussion above, you must have realized that it is beneficial to use different charts in different situations. Even though we did discuss the applications of some of the charts above, you can visit this [link](https://support.google.com/docs/answer/190718?visit_id=637388534929139547-822461942&rd=1) to gain more exposure to plot selection.

Also here is a list of links of youtube videos which will help you in picking the appropriate chart type for your data.

* [Youtube link 1](https://www.youtube.com/watch?v=C07k0euBpr8)
* [Youtube link 2](https://www.youtube.com/watch?v=aUk4npRmjL8)

After choosing the right plot, you also need to make sure that the plot conveys the intended information quickly. You can find a comprehensive paper on presentation data visualizations [here](https://pubs.aeaweb.org/doi/pdfplus/10.1257/jep.28.1.209). It is quite a long read but has valuable information about communicating stories in a clutter-free way to your audience. You can also listen to [this](https://youtu.be/5Zg-C8AAIGg) ted talk on applications of data visualization. Think about what charts you would have used to visualize the data shown by the narrator.

One of the key takeaways from this segment is to observe the chart that you have prepared and check whether it serves the purpose for which it was created. If it does serve the purpose, then you have created the right type of chart. If it doesn't, then you need to try other options to make the message of the chart very clear.

You can apply your learnings from this segment in the practice notebook attached below. Along with the exercise, you can also find a notebook with solved questions. Although the solutions to these problems are given, it is recommended that you try to solve them on your own and then cross-check with the solutions. When you do look at the solution, try to understand the use of each line of code, understanding the usage of each line will help you write your own code later.

Exercise Notebook and Dataset

Download

Please try to solve the problems by yourself first. Once you are done trying to solve the problems, you can take a look at the solutions attached below.

Exercise Solution

Download

In the next segment, we will summarise all your learnings from this session.

# **Summary**

This session introduced you to the world of data visualisation. Throughout this session, you performed visualisations using Matplotlib. Now, let's summarise all that you learnt in this session.

Play Video

1569002

You started this session by understanding the importance of visualisation in interpreting data. Then you learnt about the different types of graphs and charts, namely:

* Bar chart
* Scatter plot
* Line graph
* Histogram
* Box plot

Further, you learnt about the elements that help display additional information about a plot. Also, you learnt how to fit several subplots inside a single plot object, which is useful for comparing different properties or elements of your data.

## Additional References:

You can refer to the links provided below to learn more about visualisation:

* You can go to this [link](https://gramener.com/posters/Winning-Parties.pdf) from Gramener to know how data visualisation can help you derive insights about the winning political parties in India.
* You can view this [link](https://blog.socialcops.com/open-data/aging-population) from SocialCops to see a creative visualisation on the ageing world population.
* You can refer to this [link](https://blog.hubspot.com/marketing/data-visualization-choosing-chart#sm.000vodl7ch6rf7e112a2q13r9cno0) to learn how to choose a graph plot for your data.
* You can refer to this [link](https://www.youtube.com/watch?v=UO98lJQ3QGI&list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB_) to find additional video tutorials by **Corey Shaffer** on Matplotlib
* You can refer to this [link](https://matplotlib.org/3.2.1/tutorials/index.html) to find **official Matplotlib Documentation**
* You can to more about choosing the right plot type by referring to this [link](https://blog.hubspot.com/marketing/types-of-graphs-for-data-visualization#sm.000vodl7ch6rf7e112a2q13r9cno0).
* You can refer to this [link](https://www.youtube.com/watch?v=Ercd-Ip5PfQ) to find out additional information about plotting real-time data.

In the next session, you will go through a case study which will make use of the tools you learnt in this session.