Data Visualization using Python

August 17, 2021

1 Introduction

Data Visualisation refers to the graphical or visual representation of information and data using visual elements like charts, graphs, maps, etc.

1.0.1 MATPLOTLIB LIBRARY

It is a 2D plotting Library which produces publication quality figures. **PyPlot is a module** Of matplotlib library containing collection of methods which allow users to create 2D plots and graphs easily and interactively.

[1]:

```python
from matplotlib import pyplot as plt   #Import Statement
```

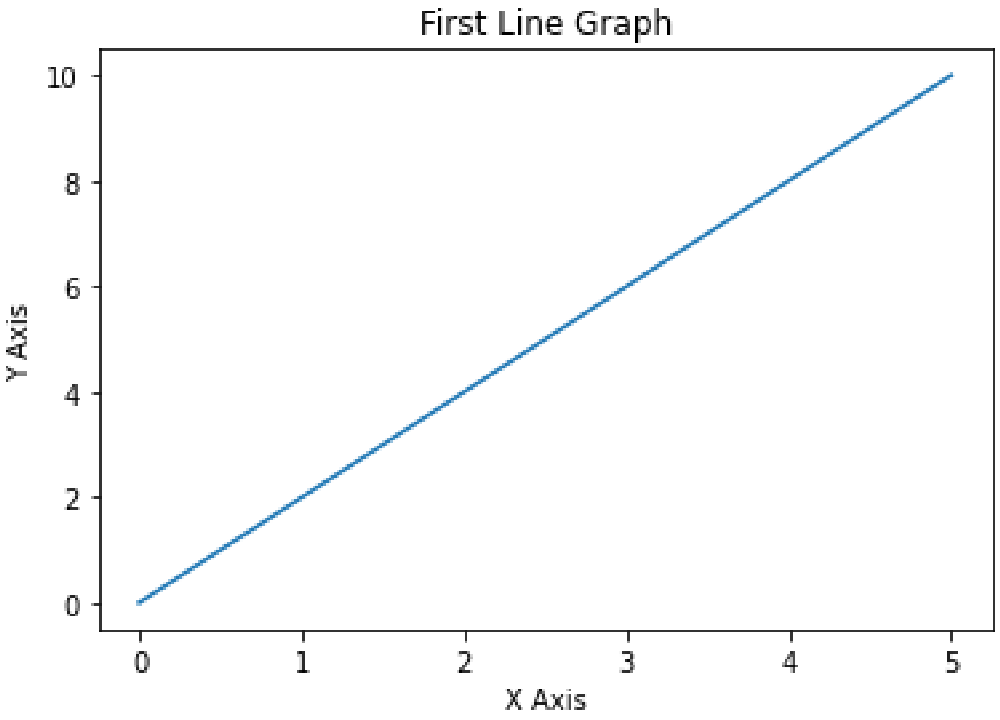
1.0.2 PLOT

A Plot is graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables. Lets look at an example

[3]:

```python
#Step 1: Import the Module
from matplotlib import pyplot as plt

#Step 2: Create the Lists
x = [0,1,2,3,4,5]
y = [0,2,4,6,8,10]

#Step 3: Not Required
#Step 4: Plotting the Graph --- For Line graph, the method we use is known as␣
 ↪plot()
plt.plot(x,y)

#Step 5: Detailing the Plot
#---------------------------
#Detail 1: Name of X Axis --> xlabel()
plt.xlabel('X Axis')
#Detail 2: Name of Y Axis --> ylabel()
plt.ylabel("Y Axis")
#Detail 3: Name of the Graph  --> title()
plt.title('First Line Graph')
```

```
#Step 6: Saving a Graph/Plot
plt.savefig('Line1.png')  # png, jpeg, pdf, svg

#Step 7: Display the Plot.
plt.show()
```



1.0.3 TYPES OF VISUALIZATION

- **LINE GRAPH**
- **BAR GRAPH**
- **HISTOGRAM**
- PIE CHART (Not In Syllabus)
- SCATTER PLOT (Not in Syllabus)
- FREQUENCY POLYGON (Not in Syllabus)
- BOX PLOT ETC. (Not in Syllabus)

1.1 TIP: General Steps to be followed for Plotting any Graph

1. Import the necessary modules (Ex. matplotlib.pyplot and numpy)
2. Create the Arrays/Lists to be plotted into a graph
3. Plot the Graph using the proper lists and mention the details (Ex. color, width, align, legend etc.)

4. Provide the necessary Details for the Graph (Ex. Title, XLabel, YLabel, XTicks, YTicks, Show Legend, etc)
5. [Optional - When Required] Save the Plot
6. Display the Plot

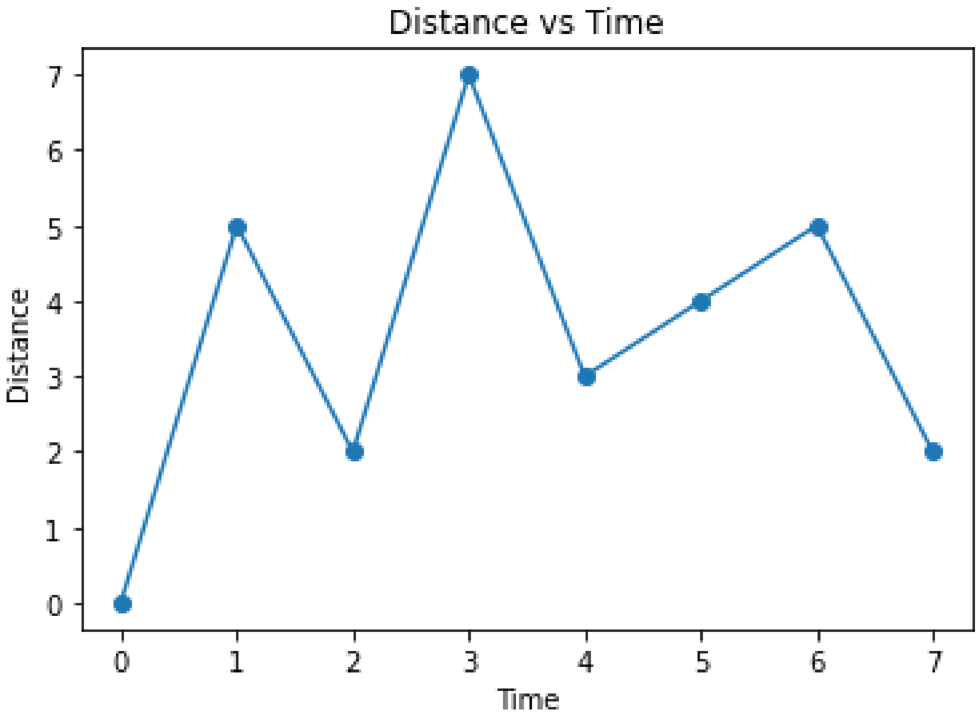
1.1.1 Line Graph

It is used to visualise data which has some kind of sequence. *Example: How is Distance changing with time Example: How many animals in forest residing against temperature of place.*

SYNTAX: plt.plot(data_x, data_y)

[49]:

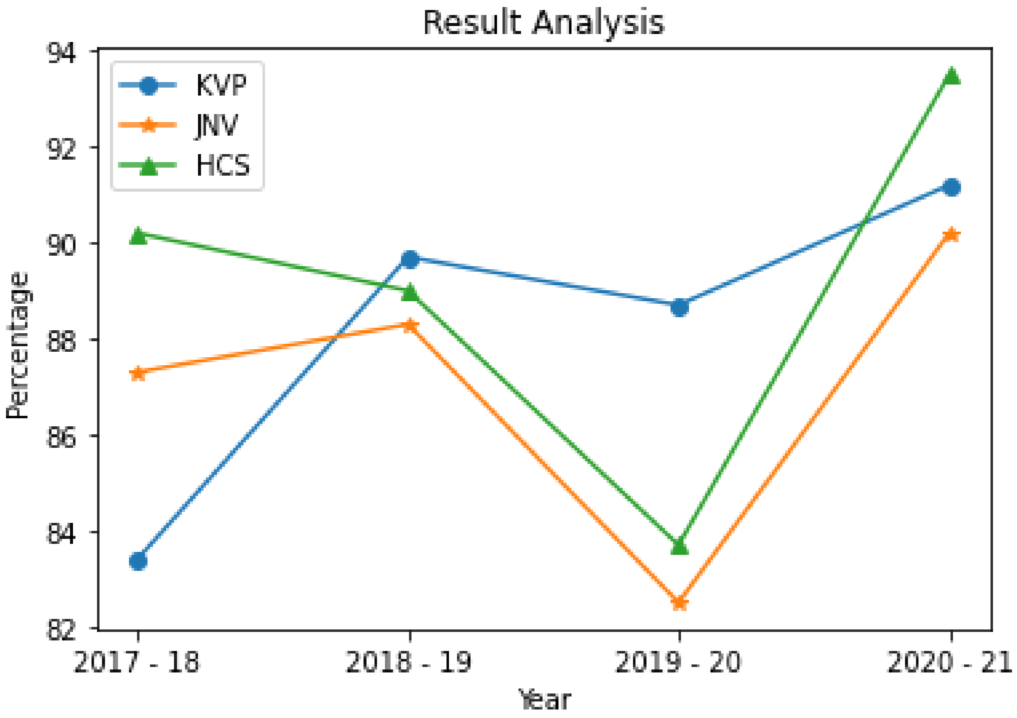
```
#Step 1: Import the Modules
from matplotlib import pyplot as plt

#Step 2: Create the Lists/Arrays
d = [0,5,2,7,3,4,5,2]
t = [0,1,2,3,4,5,6,7]

#Step 3: Plot the Graph
plt.plot(t,d, linestyle = '-',marker = 'o')

#Step 4: Provide the Details
plt.title("Distance vs Time")
plt.xlabel("Time")
plt.ylabel("Distance")

#Step 5: Save the Plot
plt.savefig("speed.png")

#Step 6: Display the Plot
plt.show()
```



[59]:

```
#Step 1: Import the Modules
from matplotlib import pyplot as plt

#Step 2: Create the Arrays
year = ['2017 - 18','2018 - 19','2019 - 20','2020 - 21']
kvp = [83.4,89.7,88.7,91.2]
jnv = [87.3,88.3,82.5,90.2]
hcs = [90.2,89.0,83.7,93.5]

#Step 3: Plot The Graphs
plt.plot(year, kvp, marker = 'o', label = 'KVP')
plt.plot(year, jnv, marker = '*', label = 'JNV')
plt.plot(year, hcs, marker = '^', label = 'HCS')

#Step 4: Provide the Details
plt.title("Result Analysis")
plt.xlabel("Year")
plt.ylabel("Percentage")
plt.legend()

#Step 5: Save the Graph.
plt.savefig('Result.png')

#Step 6: Display the Graph
```

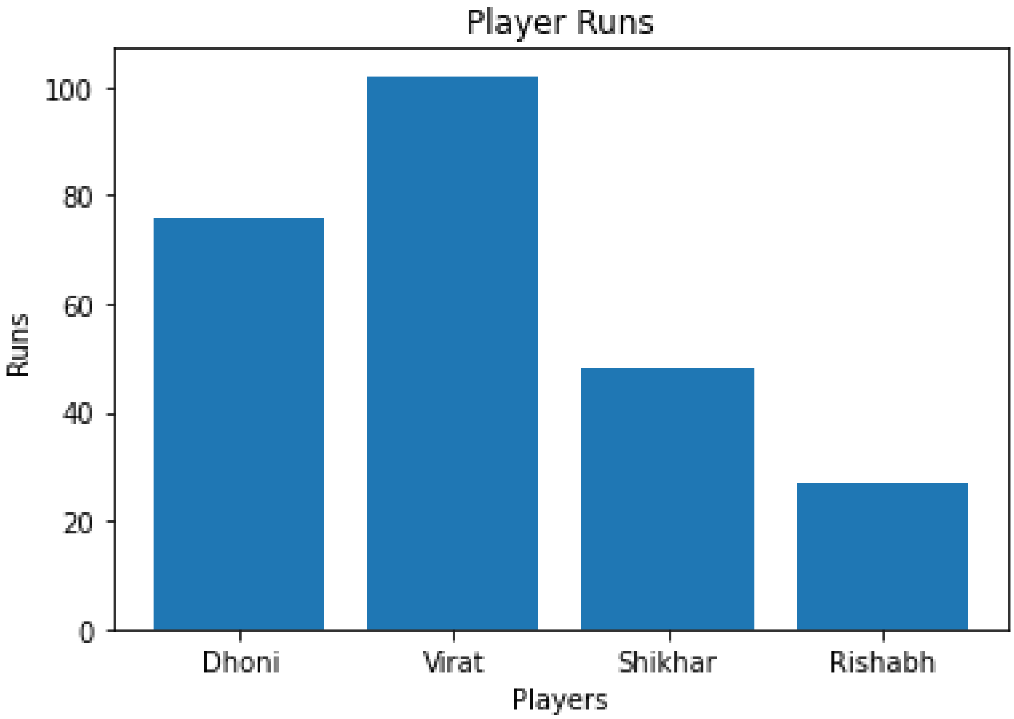
```
plt.show()
```



1.2 Bar Graph

[2]:

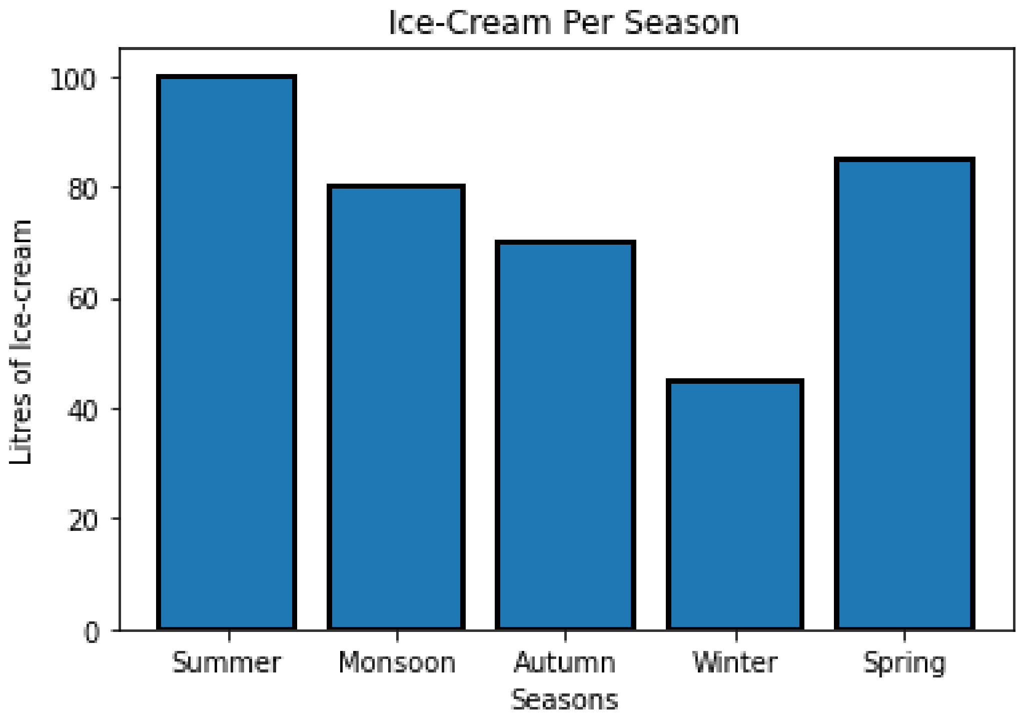
```
#Step 1: Import the Module
from matplotlib import pyplot as plt

#Step 2: Create the Arrays

players = ['Dhoni','Virat','Shikhar','Rishabh']
runs = [76,102,48,27]

#Step 3: Plot the Graph
plt.bar(players,runs)

#Step 4:
plt.title("Player Runs")
plt.xlabel('Players')
plt.ylabel('Runs')

#Step 5

plt.show()
```



#Assignment: Create the following Bar Graph

[2]:

```
from matplotlib import pyplot as plt

seasons = ['Summer','Monsoon','Autumn','Winter','Spring']
ice_cream = [100,80,70,45,85]

plt.bar(seasons,ice_cream, linewidth = 2, edgecolor = 'black')

plt.title('Ice-Cream Per Season')
plt.xlabel('Seasons')
plt.ylabel('Litres of Ice-cream')

plt.show()
```



[1]:
```python
from matplotlib import pyplot as plt

plt.bar?
```

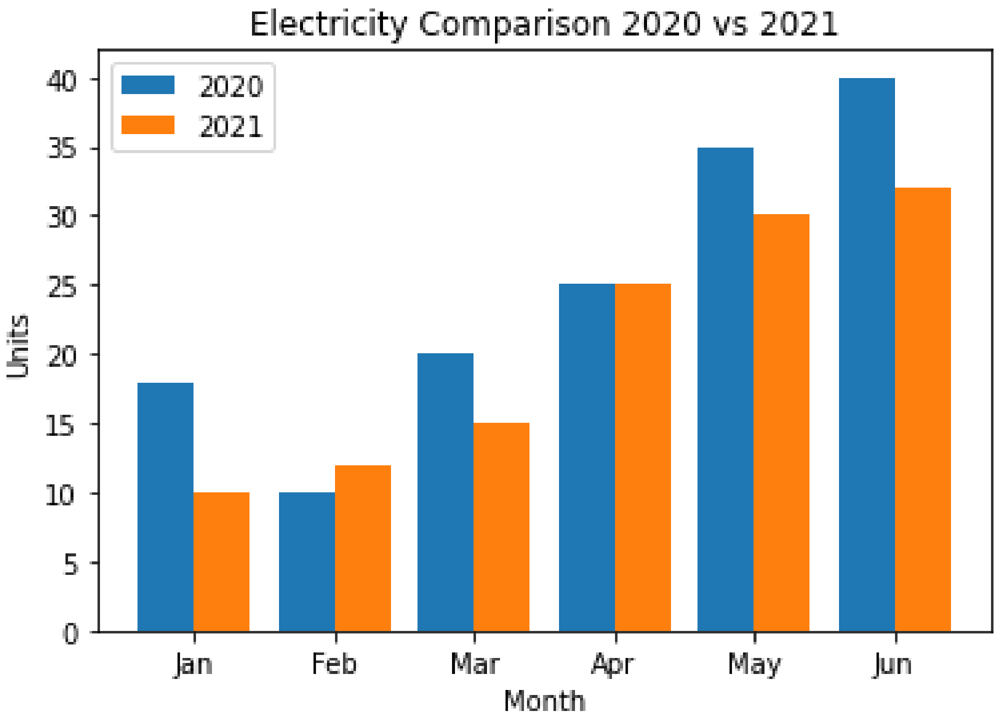
```
[1;31mSignature:[0m
[0mplt[0m[1;33m.[0m[0mbar[0m[1;33m([0m[1;33m
[0m    [0mx[0m[1;33m,[0m[1;33m
[0m    [0mheight[0m[1;33m,[0m[1;33m
[0m    [0mwidth[0m[1;33m=[0m[1;36m0.8[0m[1;33m,[0m[1;33m
[0m    [0mbottom[0m[1;33m=[0m[1;32mNone[0m[1;33m,[0m[1;33m
[0m    [1;33m*[0m[1;33m,[0m[1;33m
[0m    [0malign[0m[1;33m=[0m[1;34m'center'[0m[1;33m,[0m[1;33m
[0m    [0mdata[0m[1;33m=[0m[1;32mNone[0m[1;33m,[0m[1;33m
[0m    [1;33m**[0m[0mkwargs[0m[1;33m,[0m[1;33m
[0m[1;33m)[0m[1;33m[0m[1;33m[0m[0m
[1;31mDocstring:[0m
Make a bar plot.

The bars are positioned at *x* with the given *align*\ment. Their
dimensions are given by *height* and *width*. The vertical baseline
is *bottom* (default 0).

Many parameters can take either a single value applying to all bars
or a sequence of values, one for each bar.
```

```
Parameters
----------
x : float or array-like
    The x coordinates of the bars. See also *align* for the
    alignment of the bars to the coordinates.

height : float or array-like
    The height(s) of the bars.

width : float or array-like, default: 0.8
    The width(s) of the bars.

bottom : float or array-like, default: 0
    The y coordinate(s) of the bars bases.

align : {'center', 'edge'}, default: 'center'
    Alignment of the bars to the *x* coordinates:

    - 'center': Center the base on the *x* positions.
    - 'edge': Align the left edges of the bars with the *x* positions.

    To align the bars on the right edge pass a negative *width* and
    ``align='edge'``.

Returns
-------
`.BarContainer`
    Container with all the bars and optionally errorbars.

Other Parameters
----------------
color : color or list of color, optional
    The colors of the bar faces.

edgecolor : color or list of color, optional
    The colors of the bar edges.

linewidth : float or array-like, optional
    Width of the bar edge(s). If 0, don't draw edges.

tick_label : str or list of str, optional
    The tick labels of the bars.
    Default: None (Use default numeric labels.)

xerr, yerr : float or array-like of shape(N,) or shape(2, N), optional
    If not *None*, add horizontal / vertical errorbars to the bar tips.
    The values are +/- sizes relative to the data:
```

```
    - scalar: symmetric +/- values for all bars
    - shape(N,): symmetric +/- values for each bar
    - shape(2, N): Separate - and + values for each bar. First row
      contains the lower errors, the second row contains the upper
      errors.
    - *None*: No errorbar. (Default)

    See :doc:`/gallery/statistics/errorbar_features`
    for an example on the usage of ``xerr`` and ``yerr``.

ecolor : color or list of color, default: 'black'
    The line color of the errorbars.

capsize : float, default: :rc:`errorbar.capsize`
   The length of the error bar caps in points.

error_kw : dict, optional
    Dictionary of kwargs to be passed to the `~.Axes.errorbar`
    method. Values of *ecolor* or *capsize* defined here take
    precedence over the independent kwargs.

log : bool, default: False
    If *True*, set the y-axis to be log scale.

**kwargs : `.Rectangle` properties

Properties:
    agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, and re
    alpha: scalar or None
    animated: bool
    antialiased or aa: unknown
    capstyle: `.CapStyle` or {'butt', 'projecting', 'round'}
    clip_box: `.Bbox`
    clip_on: bool
    clip_path: Patch or (Path, Transform) or None
    color: color
    contains: unknown
    edgecolor or ec: color or None or 'auto'
    facecolor or fc: color or None
    figure: `.Figure`
    fill: bool
    gid: str
    hatch: {'/', '\\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}
    in_layout: bool
    joinstyle: `.JoinStyle` or {'miter', 'round', 'bevel'}
    label: object
    linestyle or ls: {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
    linewidth or lw: float or None
    path_effects: `.AbstractPathEffect`
    picker: None or bool or float or callable
    rasterized: bool
    sketch_params: (scale: float, length: float, randomness: float)
    snap: bool or None
    transform: `.Transform`
    url: str
    visible: bool
    zorder: float

See Also
--------
barh : Plot a horizontal bar plot.

Notes
-----
Stacked bars can be achieved by passing individual *bottom* values per
bar. See :doc:`/gallery/lines_bars_and_markers/bar_stacked`.

.. note::
    In addition to the above described arguments, this function can take
    a *data* keyword argument. If such a *data* argument is given,
    every other argument can also be string ``s``, which is
    interpreted as ``data[s]`` (unless this raises an exception).

    Objects passed as **data** must support item access (``data[s]``) and
    membership test (``s in data``).
[1;31mFile:[0m      c:\users\asus\anaconda3\lib\site-packages\matplotlib\pyplot.py
[1;31mType:[0m      function
```

[17]:

```python
from matplotlib import pyplot as plt
month = ['Jan','Feb','Mar','Apr','May','Jun']
Year2021 = [10,12,15,25,30,32]
Year2020 = [18,10,20,25,35,40]
plt.bar(month,Year2020, width = -0.4, align = 'edge', label = '2020') #Negetive␣
 ↪width shifts the graph to the left
plt.bar(month,Year2021, width = 0.4, align = 'edge', label = '2021') #Positive␣
 ↪width shifts the graph to the right

plt.title("Electricity Comparison 2020 vs 2021")
plt.xlabel("Month")
plt.ylabel("Units")
plt.legend()
plt.show()
```



1.3 HISTOGRAM

This graph is usually used to display the frequency of each item in the data. What is required for such a representation is buckets/bins of the range (10-20,20-30,30-40…) ***Bins can be mentioned in either of two ways*** *- A list [10,20,30,40,50,…] - An integer depicting the no of bins required. The Bins will then be generated by equally distributing the total range of the frequency data.*

By Default the Bin has a integer value of 10. #### There are 2 techniques for getting Data for Histogram. 1. Use the actual Frequency data as a list/array of values and use that to plot the histogram. 2. Separate the Data and the Frequency of Occurance into 2 Lists and use both to plot the histogram. Both techniques can be used depending on the requirement of the question.

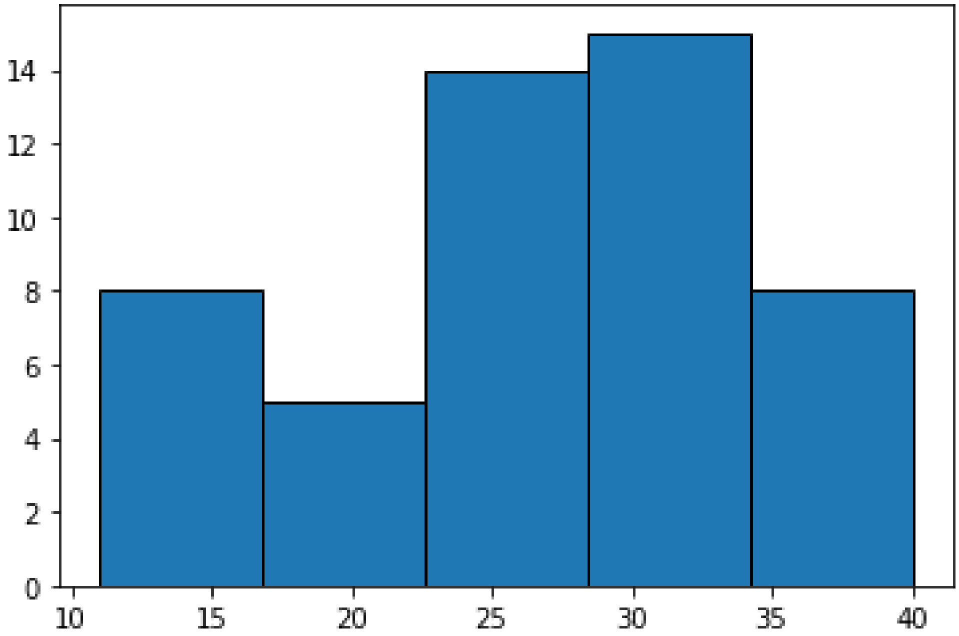
SYNTAX

- When actual frequency data is used:- **SYNTAX:** plt.hist(data,bins)
- When Data and Frequency of Occurance are in different variables:- **SYNTAX:** plt.hist(data,bins,weight)

Histogram with Actual Data and Integer Bins

[4]:
```
from matplotlib import pyplot as plt
import numpy as np
```

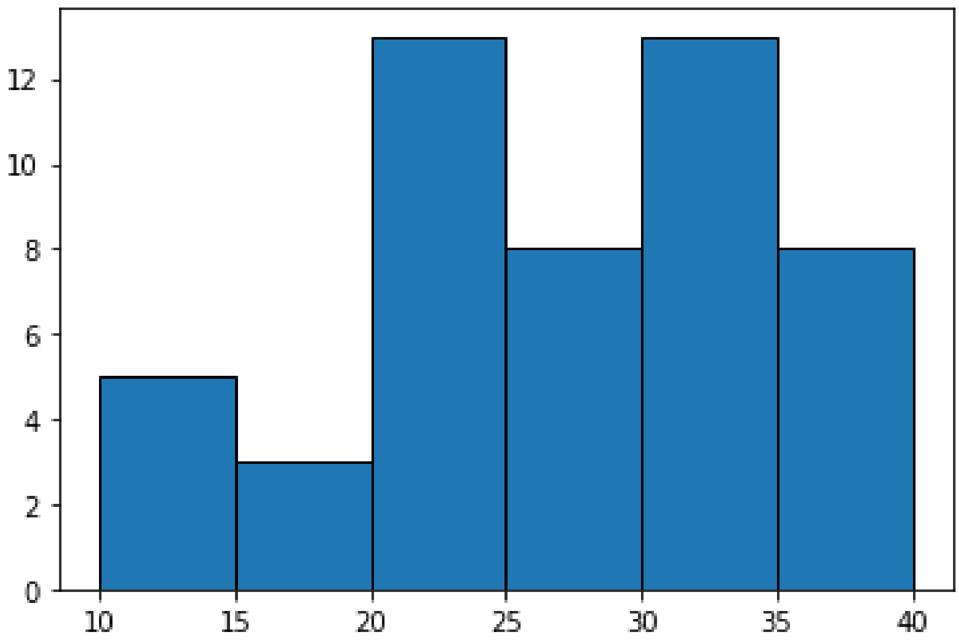
```
data = np.array([24,23,24,23,21,25,24,21,11,16,15,30,
                 31,34,35,35,34,31,32,35,13,21,34,21,
                 31,25,34,26,12,15,14,23,38,40,14,22,
                 27,39,24,29,29,27,24,34,35,27,32,39,34,34])
plt.hist(data, bins = 5, edgecolor = 'black')

plt.show()
```



Histogram with actual data and bins as list

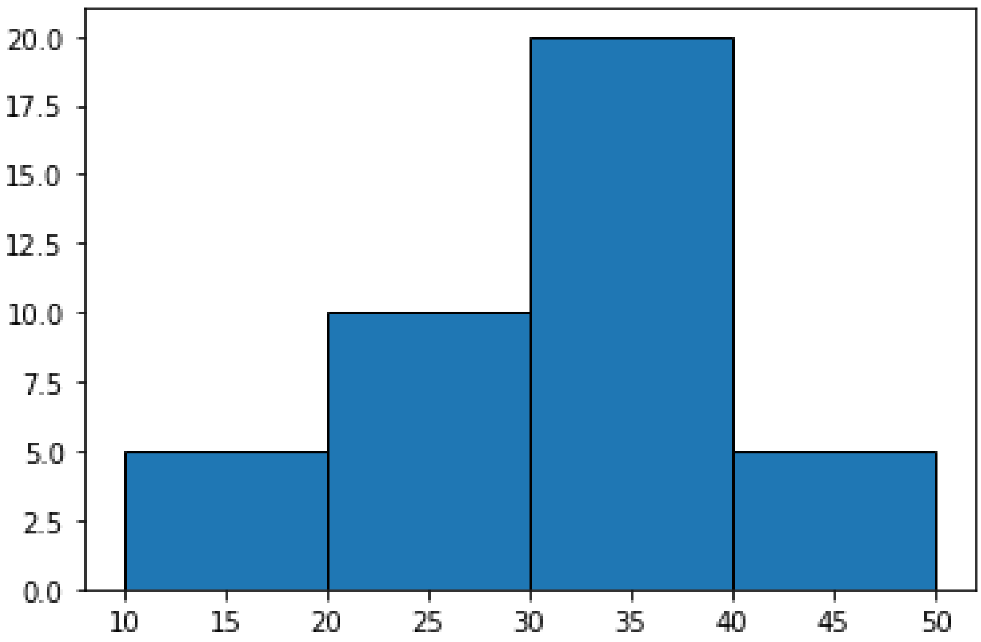
[10]:

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([24,23,24,23,21,25,24,21,11,16,15,30,
                 31,34,35,35,34,31,32,35,13,21,34,21,
                 31,25,34,26,12,15,14,23,38,40,14,22,
                 27,39,24,29,29,27,24,34,35,27,32,39,34,34])
plt.hist(data, bins = [10,15,20,25,30,35,40],edgecolor = 'black')

plt.show()
```



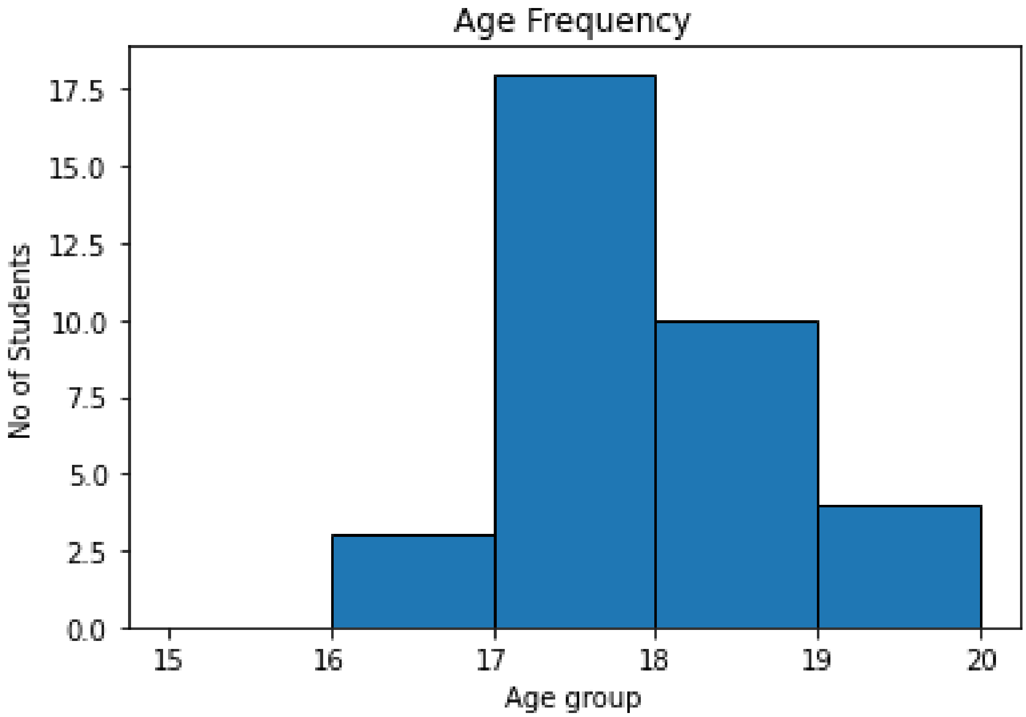
Histogram with Frequency Groups and Bins as Lists

[9]:
```
from matplotlib import pyplot as plt

mark_group=[15,25,35,45] # Class Marks - (lower point + upper point) /2
Frequency = [5,10,20,5]

plt.hist(mark_group, bins = [10,20,30,40,50], weights = Frequency,edgecolor=␣
 ↪'black')
plt.show()
```



[11]:

```
# Age 16, 17, 18, 19
from matplotlib import pyplot as plt

Age = [16,17,18,19]
Freq = [3, 18, 10, 4]

plt.hist(Age, bins = [15,16,17,18,19,20], weights = Freq, edgecolor = 'black')
plt.title('Age Frequency')
plt.xlabel('Age group')
plt.ylabel('No of Students')
plt.show()
```



1.3.1 PRACTICE QUESTIONS

1. Plot a line graph to display growth in population in the past 7 decades. Use the following Table Data for this purpose:-

Census Year	Population
1951	361,088,000
1961	439,235,000
1971	548,160,000
1981	683,329,000
1991	846,387,888
2001	1,028,737,436
2011	1,210,726,932

2. Plot a line graph to show Sin Curve. (HOTS) ***Hint:*** *Numpy has a function, numpy.sin() to find the sin values.*

3. Plot a line graph to show Cos Curve. (HOTS) ***Hint:*** *Numpy has a function, numpy.cos() to find the cos values.*

4. Plot a Bar Graph to show the number of boys in each class 6- 12. Data should be imagined by student.

5. Plot a Bar Graph for Marks scored in different subjects. Data should be imagined.

6. Plot a Histogram to find the number of employees coming to office between 7am to 12noon. Use bins as 1 hr gaps.