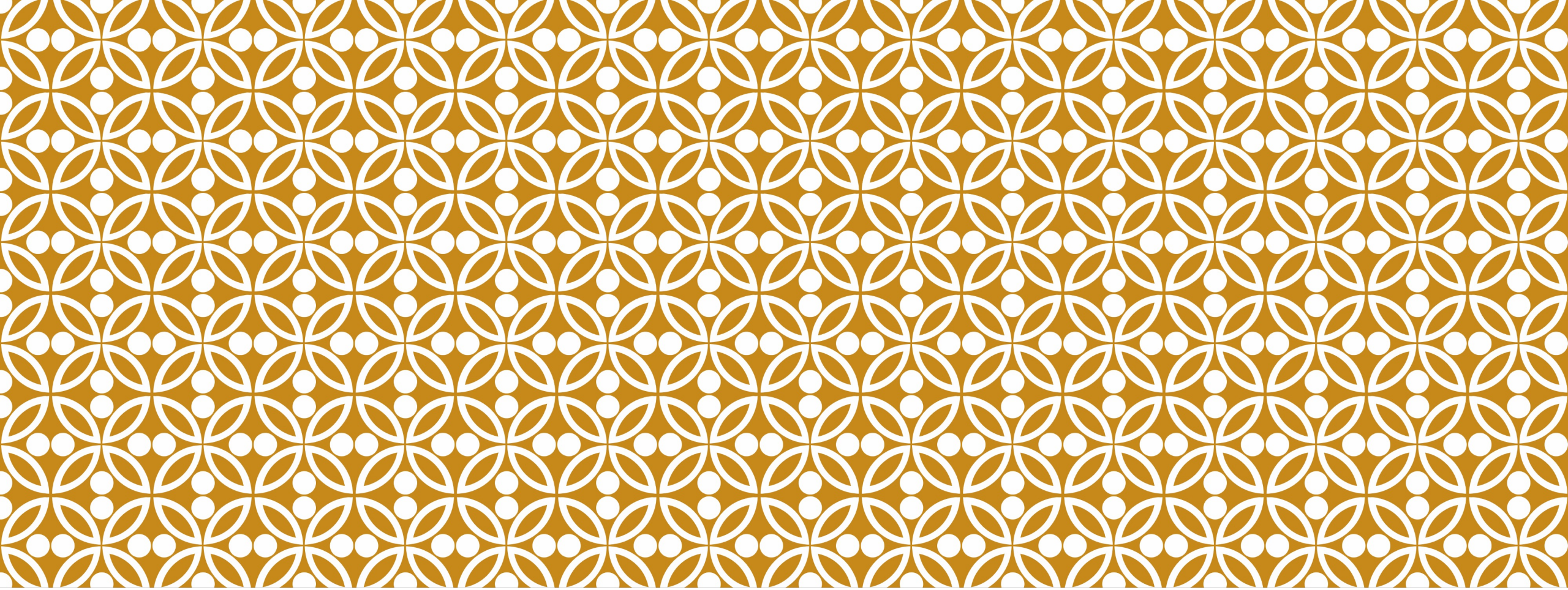


# PYTHON FOR DATA VISUALIZATION

Instructor: Somya Agrawal

# AGENDA

- Data visualization
- Python and data visualization
- Python libraries
  - Matplotlib

# ESSENTIAL LIBRARIES in Python

- NumPy
- Pandas
- Matplotlib
- Seaborn

# PREREQUISITES

- Before directly jumping to the libraries, we need to build a strong core knowledge of what datatypes, lists, dictionaries, mutable or immutable objects are.
- It’s also needed to practice looping, be able to write functions, lambda functions and other basic built in functions.

# KEY POINTS

- Python is a case sensitive language.
- Everything in Python is an object and objects have functions we call as methods that we can access using do notation.
- Every character in a string has a numeric representation in programming. You can use the built in function ord. So ord(“b”) is 98. So, 98 is the numeric representation of the letter b.

# EXECUTING PYTHON code

- Python needs an IDE to run its code. There are several options for doing so.
- In this class we will explore this using VSCODE and Jupyter notebook.

# VSCODE TO IDE

To make VSCode into an IDE install the following extensions:

1. Python
2. Pylint
3. Formatter autopep8

# STRUCTURE OF OOP

- Classes
- Objects
- Methods
- Attributes

https://www.youtube.com/watch?v=q2SGW2VgwAM

# FOUR PILLARS OF OOP

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Reference video: https://www.youtube.com/watch?v=pTB0EiLXUC8

# DATA VISUALIZATION THROUGH PYTHON

Lesson objective: To learn how to visualize data using Python.

Please can you think for a moment?

# WHY DO WE NEED DATA VISUALIZATION?

# WHY DO WE NEED DATA VISUALIZATION?

- Humans are visual creatures, and data visualization helps us quickly grasp complex datasets by presenting information in graphical or visual formats.
- It allows us to see patterns, trends, and relationships that might not be apparent from raw data alone.
- This helps in making informed decisions, predicting future outcomes, and discovering hidden insights that can drive business strategies or scientific research.

# WHY DO WE NEED DATA VISUALIZATION?

- Visualizations provide an effective means of communicating insights and findings to others, whether they are stakeholders, decision-makers, or the general public.
- A well-designed visualization can convey a message or tell a story more clearly and convincingly than a textual or tabular presentation of data.

# WHY DO WE NEED DATA VISUALIZATION?

- Data visualization allows us to explore relationships between different variables or dimensions of data.
- By plotting data points on graphs or charts, we can analyze how one variable affects another and gain insights into cause-and-effect relationships or dependencies.

# WHY DO WE NEED DATA VISUALIZATION?

- Visualizations make it easier to spot anomalies, outliers, or errors in the data that may require further investigation.
- For example, scatter plots or box plots can highlight data points that deviate significantly from the norm, helping to detect errors or unusual patterns in the data.

# WHY DO WE NEED DATA VISUALIZATION?

- Data visualization facilitates data-driven decision-making by providing decision-makers with clear, actionable insights derived from data analysis.
- Visualizations help stakeholders understand complex issues, evaluate different scenarios, and make informed choices based on evidence rather than intuition or guesswork.

# WHY DO WE NEED DATA VISUALIZATION?

- Visualizations enhance storytelling by making presentations more engaging, memorable, and persuasive.
- Whether it's in business meetings, academic presentations, or journalistic reports, visualizations help captivate the audience's attention and convey information in a compelling manner.

# In summary

*“Data visualization is crucial for understanding data, communicating insights, identifying trends, exploring relationships, detecting anomalies, making decisions, and enhancing storytelling.*

*It plays a vital role in data analysis, decision-making processes, and effective communication of information in various domains, including business, science, academia, and journalism.”*

# TYPES OF DATA VISUALIZATION

Plotting libraries:

- Matplotlib: low level, provides lots of freedom
- Pandas visualization: easy to use interface, built on Matplotlib
- Seaborn: high level interface, great default styles
- ggplot: based on R’s ggplot2
- Plotly: can create interactive plots

# USING MATPLOTLIB LIBRARY

In this lesson we will learn how to use Matplotlib library in Python to visualize data.

# USING MATPLOTLIB LIBRARY

- Matplotlib is a popular open source 2D and 3D Python library used for creating static, interactive, and animated visualizations in Python. It was introduced by John Hunter in 2002.

- It provides a wide range of plotting functions and tools for generating various types of plots and charts, including line plots, scatter plots, bar plots, histograms, pie charts, and more.

Wikipedia page: https://en.wikipedia.org/wiki/John_D._Hunter

Read more: https://matplotlib.org/stable/users/project/history.html

# KEY FEATURES OF MATPLOTLIB LIBRARY

- **Flexible and Customizable:** Matplotlib allows users to customize nearly every aspect of their plots, including colors, labels, markers, axes, gridlines, and more. This flexibility enables users to create publication-quality plots tailored to their specific needs.
- **Support for Various Plot Types:** Matplotlib supports a wide range of plot types, making it suitable for visualizing diverse datasets and analysis tasks. Whether you need to plot time series data, compare categories, or visualize distributions, Matplotlib provides the tools to create the desired plot.

# KEY FEATURES OF MATPLOTLIB LIBRARY

- **Object-Oriented API:** Matplotlib provides both a procedural interface (similar to MATLAB) and an object-oriented interface for creating plots. The object-oriented API offers more control and flexibility, making it suitable for advanced users and complex plotting scenarios.
- **Extensive Documentation and Community Support:** Matplotlib has comprehensive documentation with examples, tutorials, and API references, making it easy for users to get started and learn new features. Additionally, Matplotlib has a large and active community of users and developers who contribute to the library, provide support, and share resources.

# MATPLOTLIB LIBRARY

Overall, Matplotlib is a powerful and versatile library for creating high-quality visualizations in Python, making it a valuable tool for data analysis, scientific computing, exploratory data visualization, and presentation of results.

# DIFFERENT TYPES OF GRAPHS

Matplotlib offers a wide range of plot types for visualizing data as follows.

- **Line Plot:** Used to visualize trends over time or relationships between variables.
- **Scatter Plot:** Used to display the relationship between two continuous variables.
- **Bar Plot:** Used to compare different categories or groups.
- **Histogram:** Used to visualize the distribution of a continuous variable.
- **Pie Chart:** Used to represent parts of a whole or proportions.
- **Box Plot:** Used to visualize the distribution of a continuous variable and identify outliers.

# DIFFERENT TYPES OF GRAPHS

- These are just a few examples of the types of plots you can create with Matplotlib.
- The library offers many more plot types and customization options, allowing you to create virtually any type of graph to suit your data visualization needs.

# INSTALLING MATPLOTLIB

- Open the terminal/ command prompt window depending on the OS.
- Use the following statement:

```
pip install matplotlib
```

# INSTALLING MATPLOTLIB

- You need to import this library in Python. Use the following statements.

```
import matplotlib.pyplot as plt
```

OR

```
from matplotlib import pyplot as plt
```

- Here plt is just an alias. If we don't use alias then we will have to type the whole statement.
- Similar concept to SQL.

Note: Here pyplot is a class which helps you to make graphs

# CLASSES IN PYTHON

- In Python, classes are a fundamental concept of object-oriented programming (OOP).
- They serve as a blueprint for creating objects, which are instances of the class.
- Classes encapsulate data (attributes) and behaviors (methods) into a single unit, allowing for modular and organized code.
- Example: Human, characteristics of human beings

# CLASSES IN PYTHON

- **Class Definition:** A class is defined using the `class` keyword, followed by the class name and a colon. Inside the class definition, you define attributes and methods.

```
class MyClass:
    # Class attributes and methods are defined here
    pass
```

# CLASS ATTRIBUTES IN PYTHON

**Attributes:** Attributes are variables that hold data associated with a class or its objects. They represent the state of an object. Attributes can be either class attributes (shared among all instances) or instance attributes (unique to each instance).

```
class Person:
    # Class attribute
    species = "Homo sapiens"


    def __init__(self, name, age):
        # Instance attributes
        self.name = name
        self.age = age
```

# Dictionaries in Python

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values.
- Dictionary items are ordered, changeable, and do not allow duplicates.
- Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Reference: https://www.w3schools.com/python/python_dictionaries.asp

# METHODS IN PYTHON

**Methods:** Methods are functions defined within a class.

They define the behavior of the class and can access and modify its attributes.

Methods can be instance methods, class methods, or static methods.

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    @classmethod
    def square(cls, side_length):
        return cls(side_length, side_length)

    @staticmethod
    def is_square(rect):
        return rect.width == rect.height
```

# JUPYTER NOTEBOOK

- Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.
- It supports various programming languages, but it's particularly popular in the Python community.
- Jupyter Notebook provides an interactive computing environment where you can write, execute, and document code in an interactive and collaborative manner.

# JUPYTER NOTEBOOK

- **Support for Multiple Programming Languages:** While commonly used with Python, Jupyter Notebook supports over 40 different programming languages, including R, Julia, and Scala. Each notebook can have a kernel associated with a specific language, allowing you to write code in that language.
- **Visualization Capabilities:** Jupyter Notebook integrates with various visualization libraries, such as Matplotlib, Seaborn, Plotly, and Bokeh, allowing you to create interactive plots, charts, and graphs directly within your notebooks. This makes data exploration and presentation more convenient and effective.

# JUPYTER NOTEBOOK

- **Notebook Sharing and Collaboration:** You can share Jupyter Notebooks with others by exporting them as HTML, PDF, or other formats, or by hosting them on platforms like GitHub, JupyterHub, or Jupyter Notebooks Viewer. Additionally, Jupyter supports collaborative editing, allowing multiple users to work on the same notebook simultaneously.

# INSTALLING JUPYTER NOTEBOOK

To install Jupyter notebook, we will use Anaconda environment.

Steps:

1. Install Python
2. Check it on terminal if/ which version of Python exists
3. Download Anaconda
4. Launch Jupyter notebook (it will run in your default browser)

Reference video link for how to install Anaconda and then launching Jupyter notebook: https://www.youtube.com/watch?v=WUeBzT43JyY

Reference link for downloading Anaconda: https://www.anaconda.com/download

# INTRODUCING ANACONDA

- Anaconda is a popular open-source distribution of the Python and R programming languages for scientific computing, data science, and machine learning.
- It includes a collection of over 1,500 packages and libraries commonly used in these fields, along with tools for managing environments, dependencies, and packages.
- Anaconda simplifies the process of setting up and managing software environments for data analysis, allowing users to focus on their work rather than worrying about software compatibility and installation issues.

# INTRODUCING ANACONDA

- Anaconda provides access to a vast collection of packages and libraries commonly used in scientific computing, data analysis, machine learning, and related fields.
- This includes popular libraries such as NumPy, pandas, Matplotlib, SciPy, scikit-learn, TensorFlow, and PyTorch, among others.

# INTRODUCING ANACONDA

- **Cross-Platform Support:** Anaconda is available for Windows, macOS, and Linux operating systems, making it suitable for a wide range of users and environments.
- **Integrated Development Environments (IDEs):** Anaconda can be used with various integrated development environments (IDEs) and text editors, including Jupyter Notebook, JupyterLab, Spyder, and VSCode. These environments provide features such as code editing, debugging, and interactive visualization, enhancing the productivity of users.

# CHECKLIST:

1. Install Python
2. Check it on terminal if/ which version of Python exists
3. Download Anaconda
4. Launch Jupyter notebook (it will run in your default browser)

Reference video link for how to install Anaconda and then launching Jupyter notebook: https://www.youtube.com/watch?v=WUeBzT43JyY

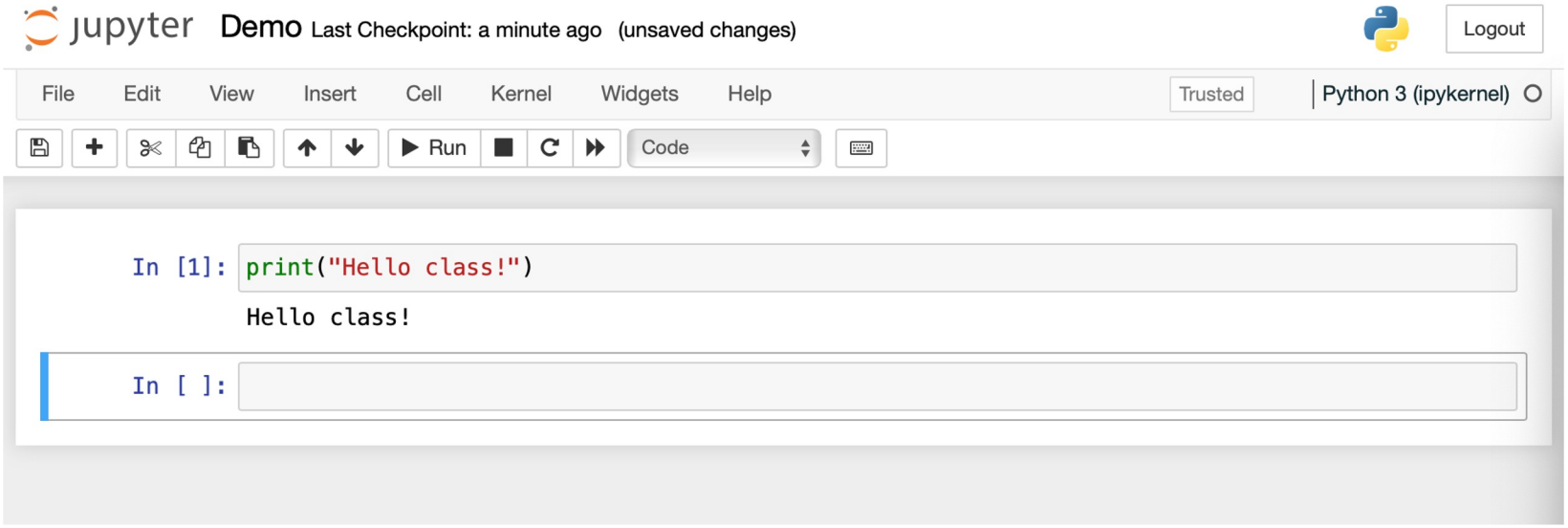
Reference link for downloading Anaconda: https://www.anaconda.com/download

# GETTING STARTED WITH JUPYTER NOTEBOOK

- Click on “New” on the right hand side of the Jupyter interface.
- A new notebook will be created where you will write the Python code in the cells.
- Rename the file. Maybe call it “Demo”.
- Explore the interface.
- Please note, to run a Python code, an interpreter is needed.
- The interpreter in Jupyter notebook is called “Kernel”.

# GETTING STARTED WITH JUPYTER NOTEBOOK

Write the following piece of code to test if the notebook is working!



# GETTING STARTED WITH JUPYTER NOTEBOOK

- Click on the ‘+’ sign to add new cells to write code.
- The number beside **In** in every cell represents the number of times you have run the file.

# INSTALLING MATPLOTLIB LIBRARY

- There are several ways of installing Matplotlib library on the computer.
- We will install Matplotlib library on Jupyter notebook using Anaconda.
- Steps:
  1. Create a new notebook called “install_matplotlib” on Jupyter. Save it.
  2. In the first cell type: **!pip install matplotlib**
  3. Click on run.
  4. If successful, then you will get an interface as follows.

Note: To install any Python library package, we need pip.

Reference video link: https://youtu.be/yR0wtOxjx-U?si=QF5wvAS2qa9eztqH

# INSTALLING MATPLOTLIB LIBRARY

```
In [1]: !pip install matplotlib

Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.11/site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.0.
5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (4.2
5.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.
4.4)
Requirement already satisfied: numpy>=1.20 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/python3.11/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.7->mat
plotlib) (1.16.0)
```

Reference video link: https://youtu.be/yR0wtOxjx-U?si=QF5wvAS2qa9eztqH

# INSTALLING MATPLOTLIB LIBRARY

- There are several ways of installing Matplotlib library on the computer.
- We will install Matplotlib library on Jupyter notebook using Anaconda.
- Steps:
  1. Create a new notebook called “install_matplotlib” on Jupyter. Save it.
  2. In the first cell type: **!pip install matplotlib**
  3. Click on run.
  4. Then we will import matplotlib and use the pyplot submodule. Most of the Matplotlib utilities work under pyplot. The instruction will look something like this.

```
In [2]: import matplotlib.pyplot as plt
```

Reference video link: https://youtu.be/yR0wtOxjx-U?si=QF5wvAS2qa9eztqH

# INSTALLING MATPLOTLIB LIBRARY

- There are several ways of installing Matplotlib library on the computer.
- We will install Matplotlib library on Jupyter notebook using Anaconda.
- Steps:
  1. Create a new notebook called “install_matplotlib” on Jupyter. Save it.
  2. In the first cell type: **!pip install matplotlib**
  3. Click on run.
  4. Then we will import matplotlib and use the pyplot submodule. Most of the Matplotlib utilities work under pyplot. The instruction will look something like this. After writing this statement, click on run.

```
In [2]: import matplotlib.pyplot as plt
```

Reference video link: https://youtu.be/yR0wtOxjx-U?si=QF5wvAS2qa9eztqH

# RUNNING THE FIRST PLOT

- We will create a line using x and y coordinates.
- We will also be using numpy to use the array method inside. Don't forget to click on run.
- The code looks something as below:

```
In [3]: import numpy as np

xpts = np.array([0,5])
ypts = np.array([0,150])
plt.plot(xpts, ypts)
plt.show()
```

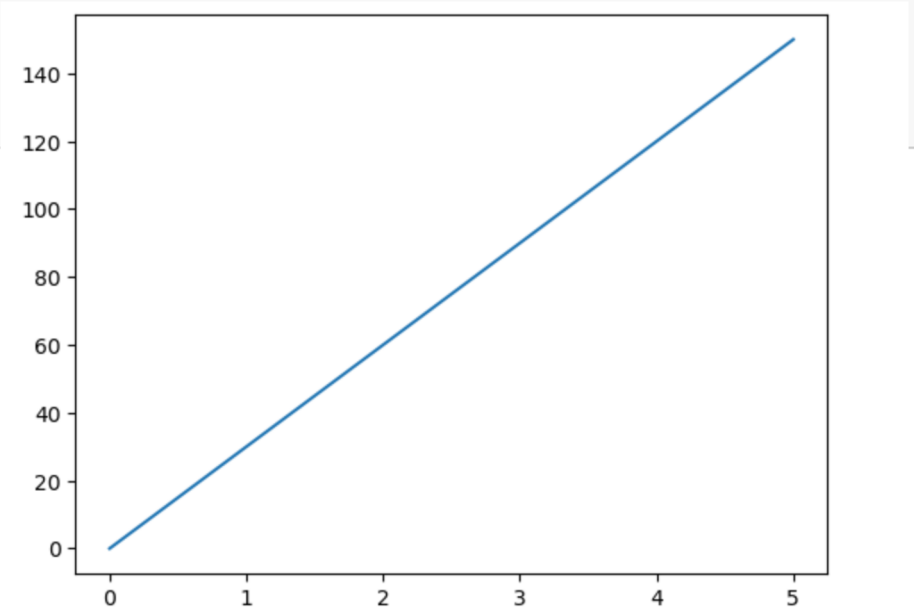
NumPy, which stands for Numerical Python, is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

NumPy is the foundation for many other scientific computing libraries in Python, making it an essential tool for data manipulation, analysis, and visualization.

# RUNNING THE FIRST PLOT

- The code below creates a line graph as shown on this slide. Here, we created x and y points.

```
In [3]: import numpy as np

xpts = np.array([0,5])
ypts = np.array([0,150])
plt.plot(xpts, ypts)
plt.show()
```




- We used the plot method to set the coordinates x and y.
- Then we used the show method to show the line graph.
- You can right click on the graph and save it as an image or download the file in other formats.

# CAVEAT OF USING JUPYTER NOTEBOOK

- Once you save a Jupyter notebook, you will never get the extension as ‘.py’ directly.
- To achieve that you need to go to File → ‘Download as’ and select that option.

# CLASS EXERCISE

Plot a line graph with the following coordinates.

x = [1, 2, 3, 4]

y = [2, 4, 6, 8]

# BAR PLOT using MATPLOTLIB

Basic syntax/ instructions:

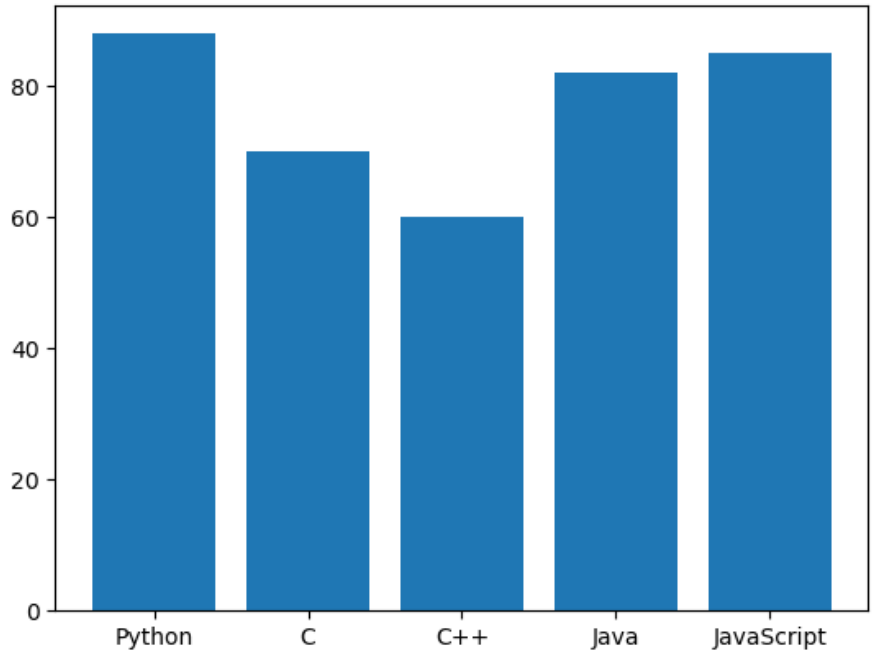
```
import matplotlib pyplot as plt
x = [] # x is one parameter
y = [] # y is second parameter
plt.plot(x, y) #inside this plot many other parameters are used which we will see
plt.show()
```

# BAR PLOT using MATPLOTLIB

Create a heading: Bar plot

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]

#plotting the graph

plt.bar(x,y)
plt.show()
```

Output graph

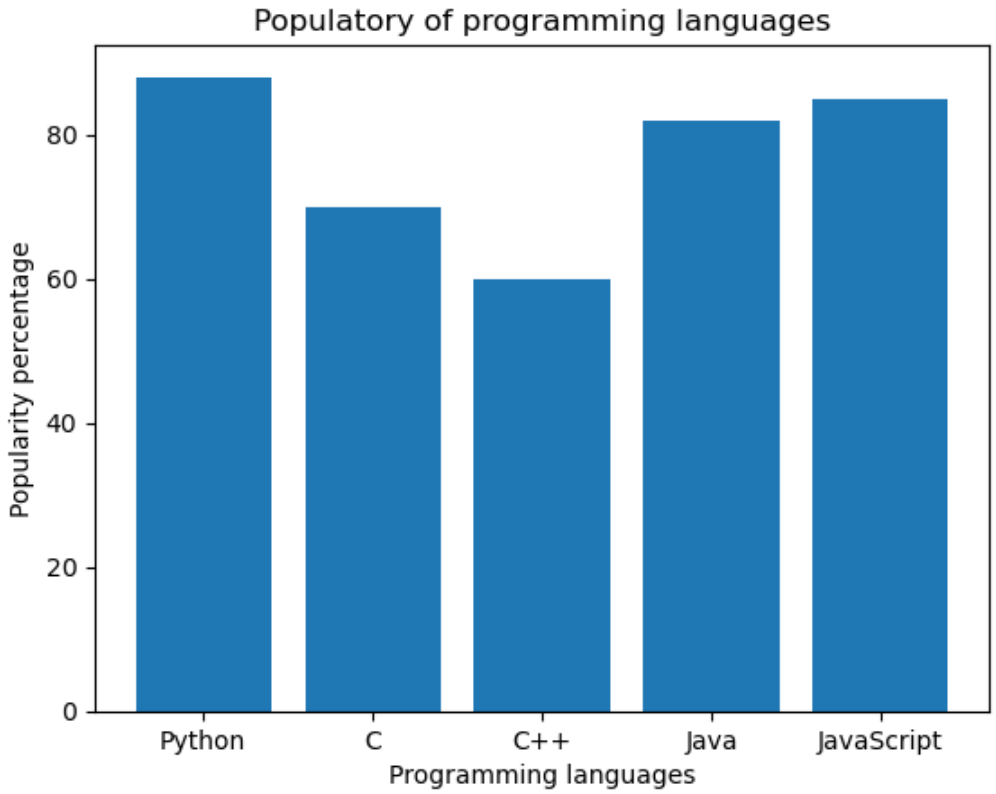



Reference video link:
https://youtu.be/RRHQ6Fs1b8w?si=hEDIEL3Hh88T6R3p

# BAR PLOT – labels, title

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages")
plt.ylabel("Popularity percentage")
plt.title("Populatory of programming languages")

#plotting the graph

plt.bar(x,y)
plt.show()
```

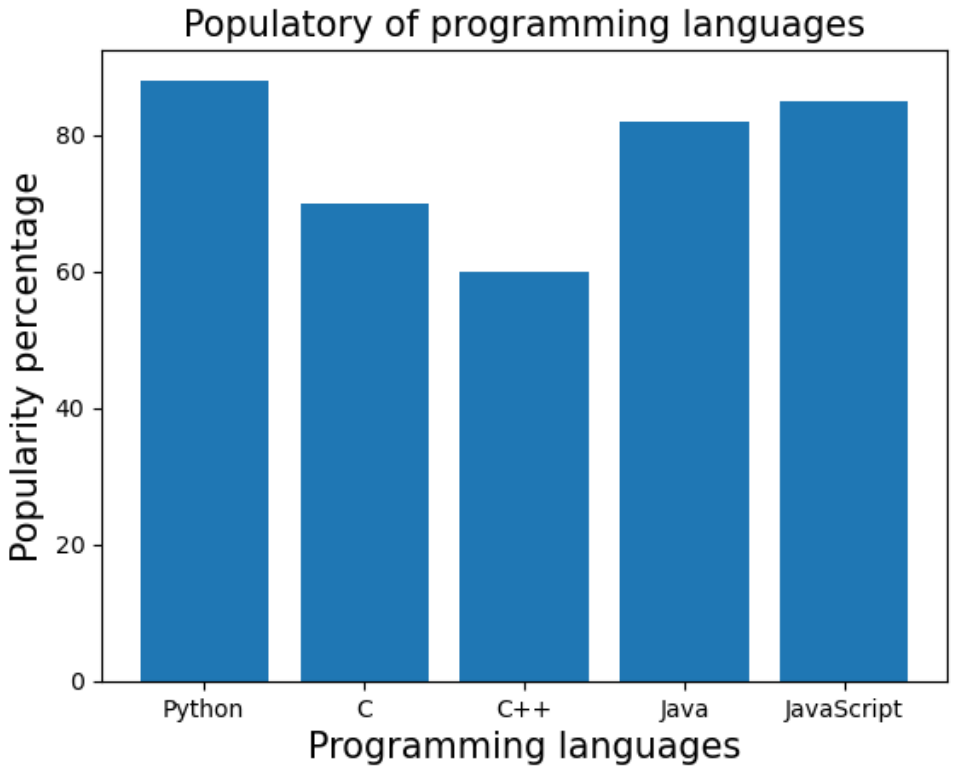
Output graph



# BAR PLOT - fontsize

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)

#plotting the graph

plt.bar(x,y)
plt.show()
```

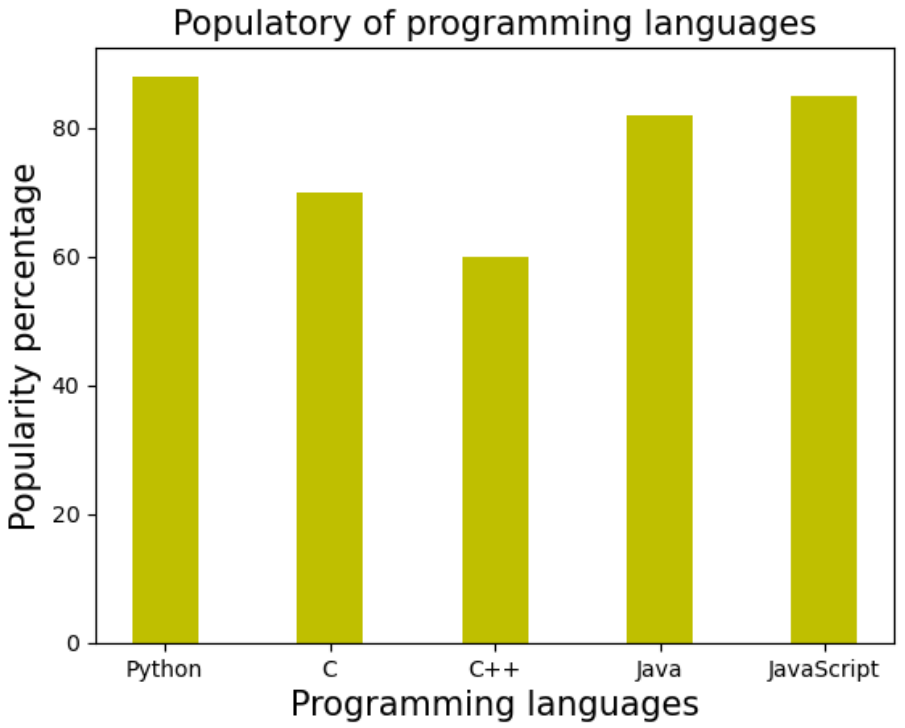
Output graph



# BAR PLOT – width, color

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)

#plotting the graph

plt.bar(x,y, width=0.4, color ='y')
plt.show()
```

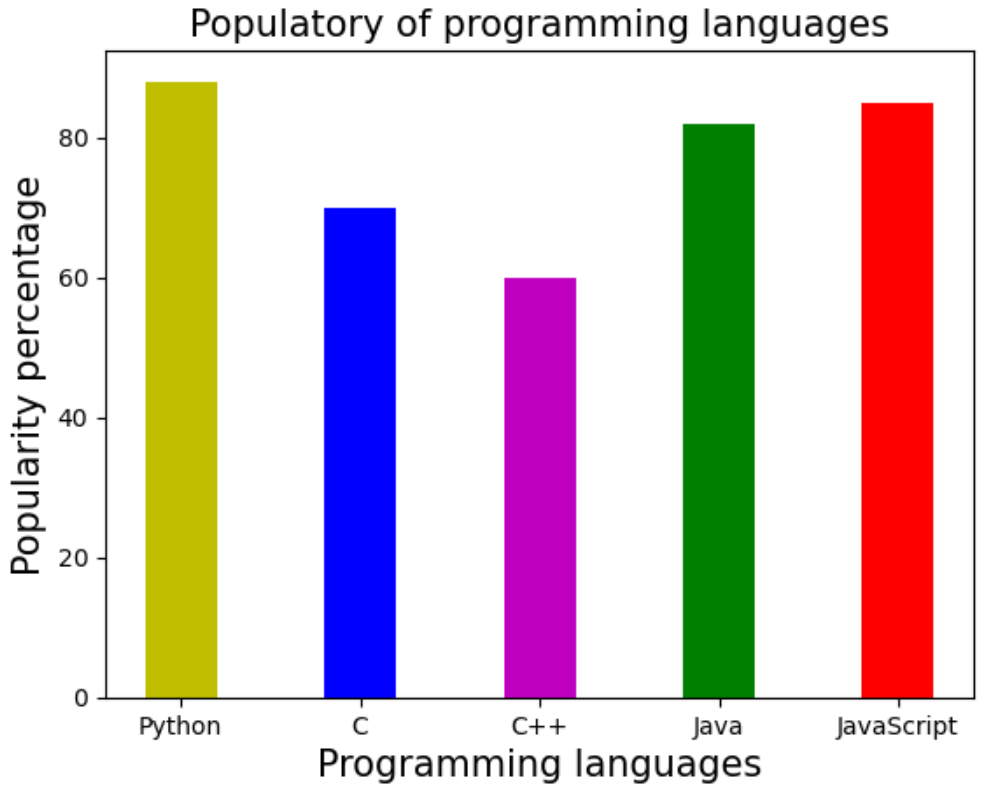
Output graph



# BAR PLOT – multicolor

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c)
plt.show()
```

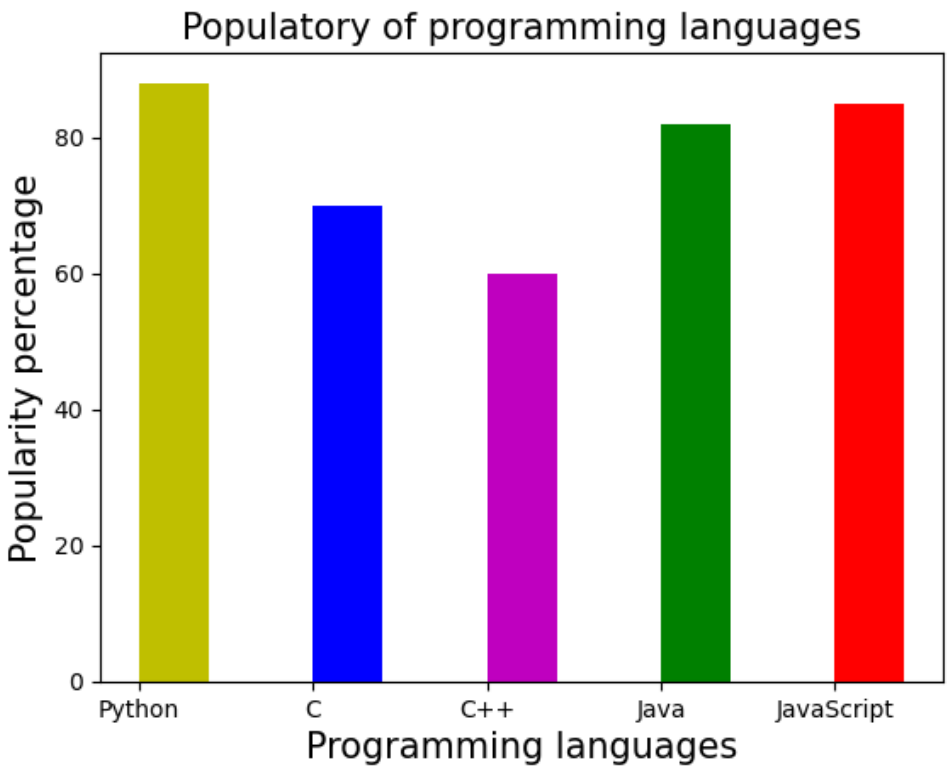
Output graph



# BAR PLOT – edge

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c, align=‘edge’)
plt.show()
```

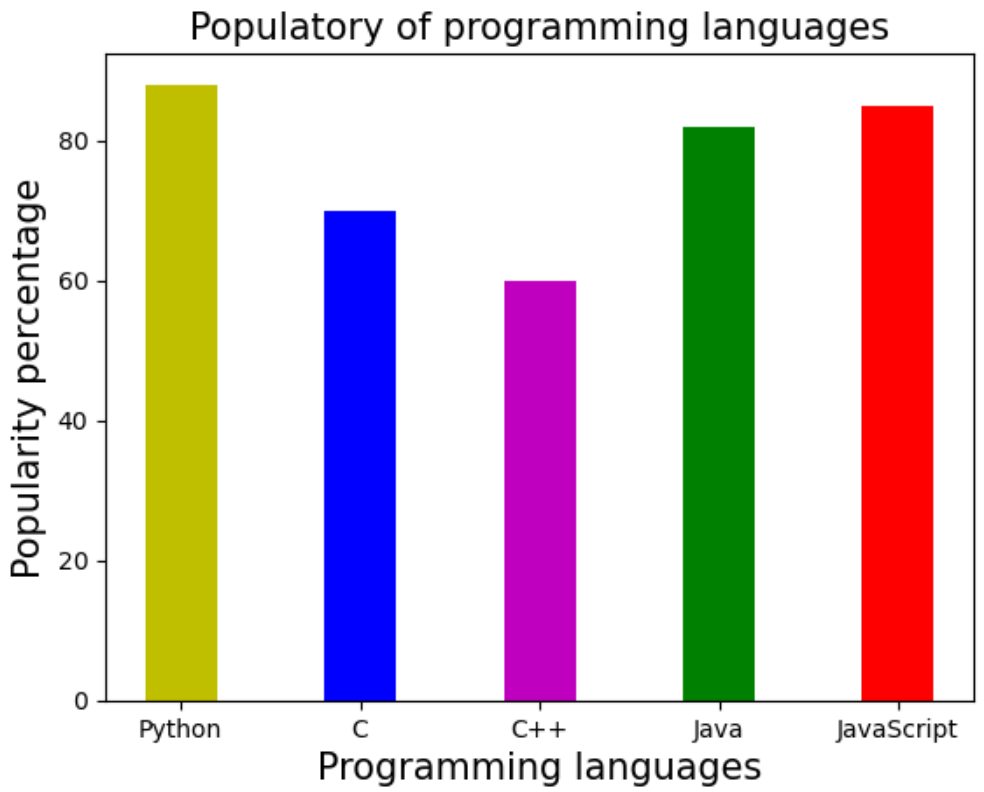
Output graph



# BAR PLOT – center

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c, align=‘center’)
plt.show()
```

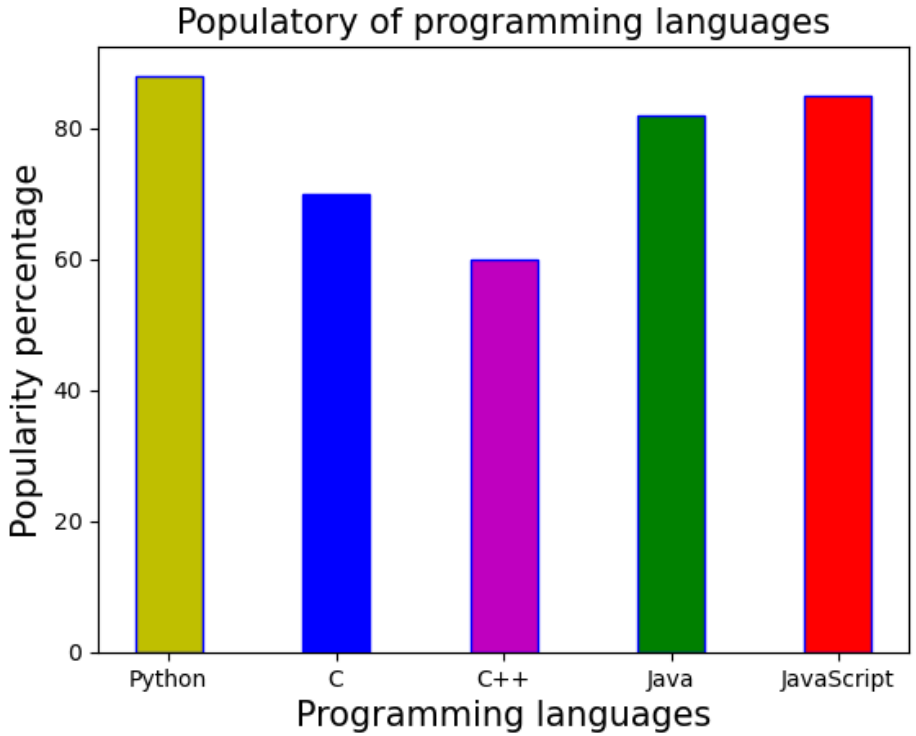
Output graph



# BAR PLOT – edgecolor

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c, edgecolor=‘b’)
plt.show()
```

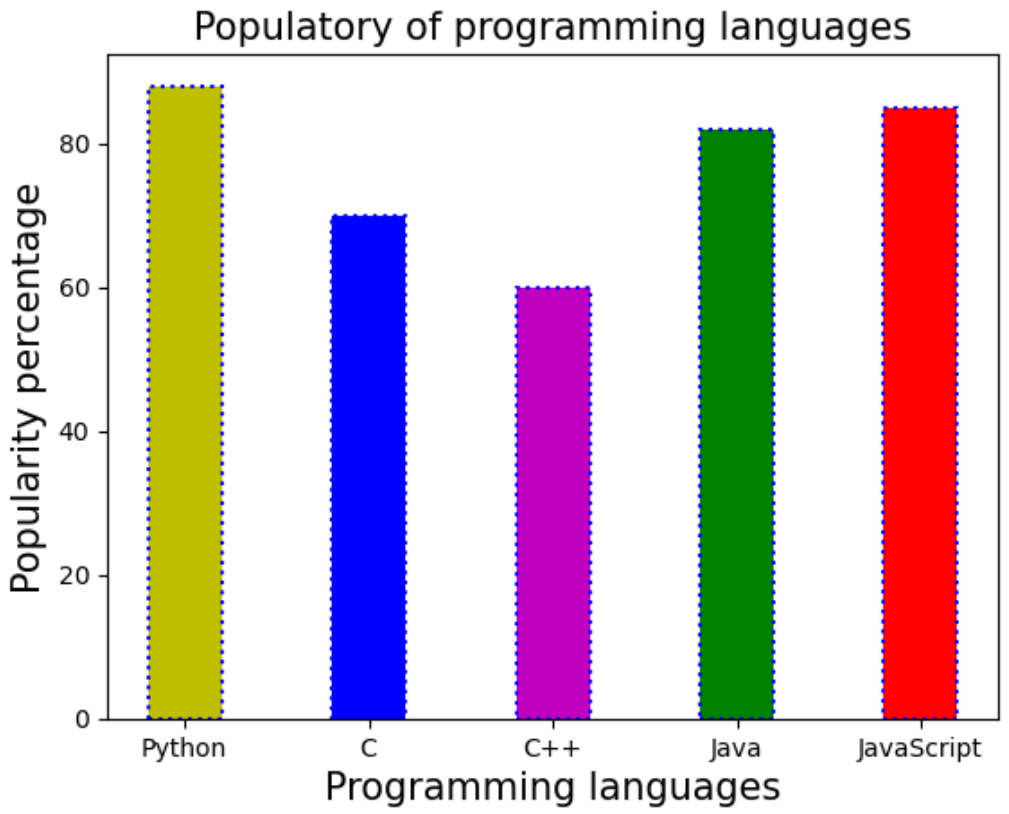
Output graph



# BAR PLOT – line border related

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c, edgecolor=‘b’, linewidth=1.5, linestyle='dotted')
plt.show()
```

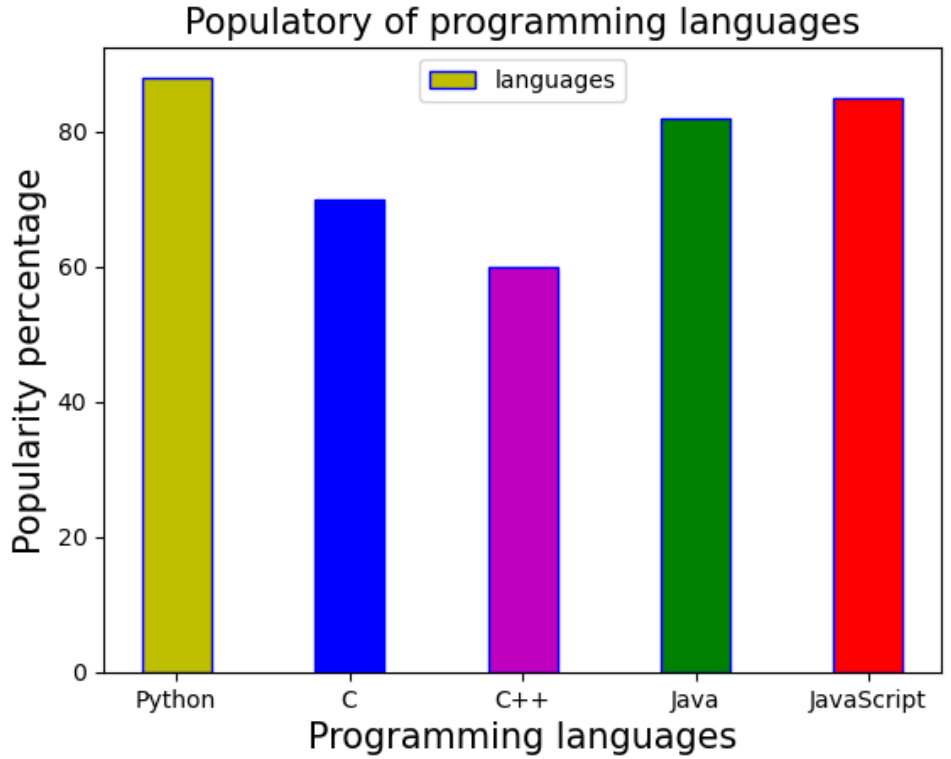
Output graph



# BAR PLOT – legend

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
plt.xlabel("Programming languages”, fontsize = 15)
plt.ylabel("Popularity percentage”, fontsize = 15)
plt.title("Populatory of programming languages”, fontsize = 15)
c=['y','b','m','g','r’]

#plotting the graph

plt.bar(x,y, width=0.4, color =c, edgecolor=‘b’, label='languages’)
plt.legend()
plt.show()
```

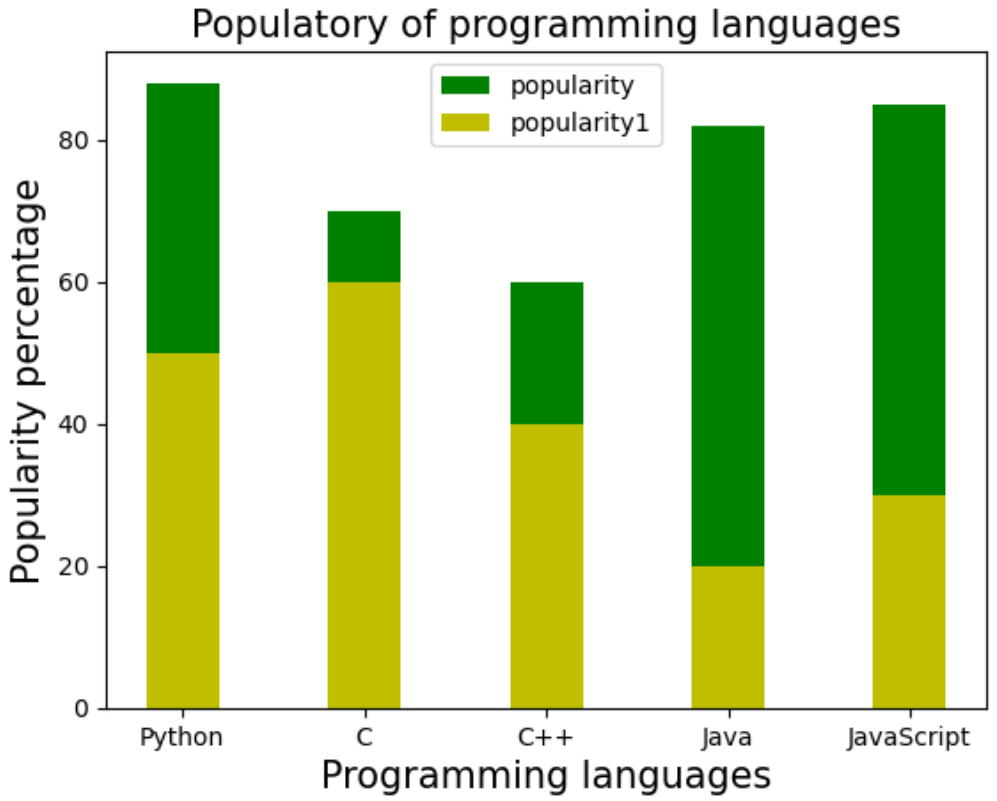
Output graph



# BAR PLOT – overlap bargraphs

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
z = [90, 89, 70, 85, 78]
plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)
c=['g']
c1=['y']

#plotting the graph
plt.bar(x,y, width=0.4, color =c, label='popularity')
plt.bar(x,z, width=0.4, color =c, label='popularity1')
plt.legend()
plt.show()
```

Output graph

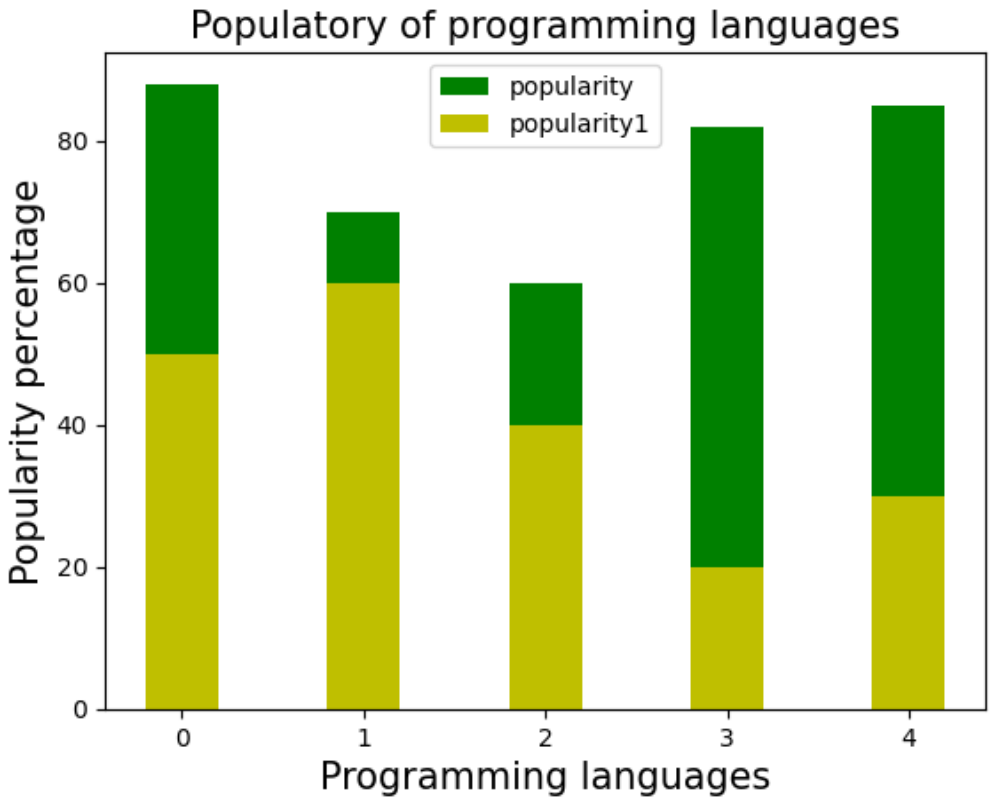


# BAR PLOT – separate bargraphs

- For this first, we need a list. In that list you need to pass the index numbers of the data values of x.
- But instead of creating a list directly such as p=[0,1,2,3,4], we will do it using another library.
- So for that we will import and use numpy library in python.
- It helps to create an array. It provides an arrange function which allows us create arrays.
- We will create an array of length x and use it to perform our action.
- And then change the letter x to p where we are plotting.
- You will see that the bars are still overlapping. To solve that, we will create a new variable p1.
- Inside p1, we will create a list of all the values that we got from p.
- We will also add width in the array that gets created which will be just beside the first width that we already have.
- For that I will create a new variable j and will add width to it. Thereafter, I will use a for loop for j inside p.

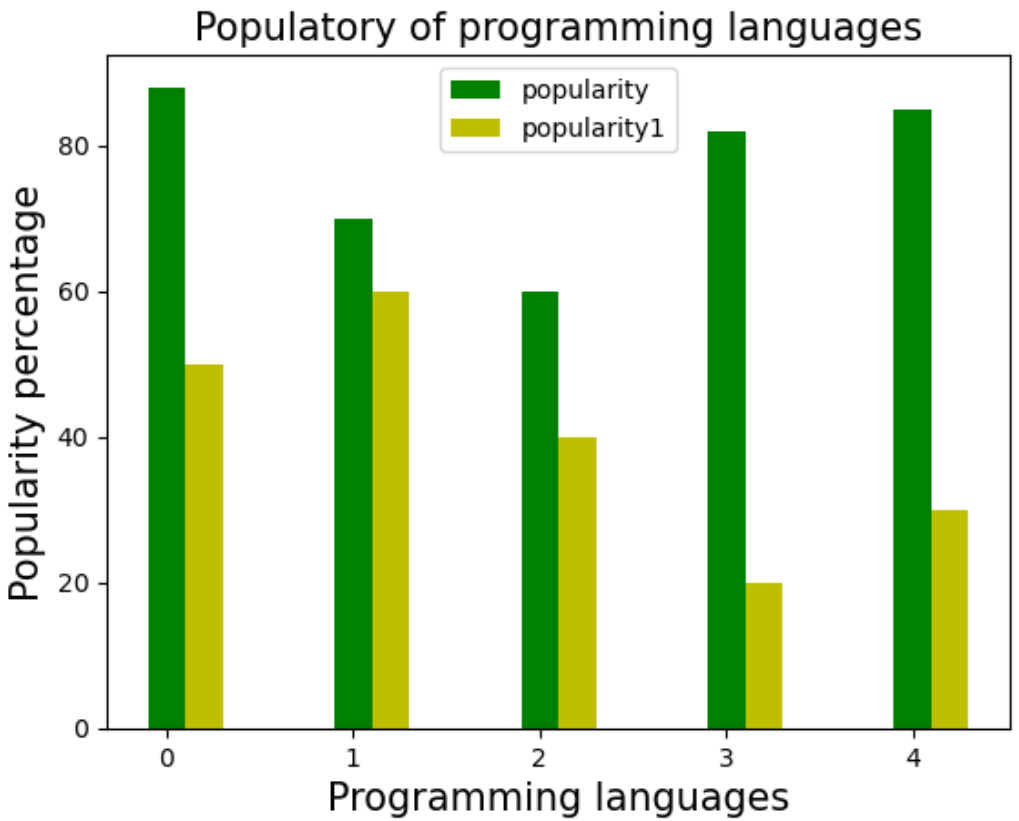
```
import matplotlib.pyplot as plt #here pyplot is a class
import numpy as np #used for creating arrays

#giving x parameter and their values
x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages
y = [88, 70, 60, 82, 85]
z = [50, 60, 40, 20, 30]

p=np.arange(len(x)) #this will create an array of length of x which we can use

plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)
c=['g']
c1=['y']

#plotting the graph

plt.bar(p,y, width=0.4, color =c, label='popularity')
plt.bar(p,z, width=0.4, color =c1, label='popularity1')
plt.legend()
plt.show()
```

Output graph



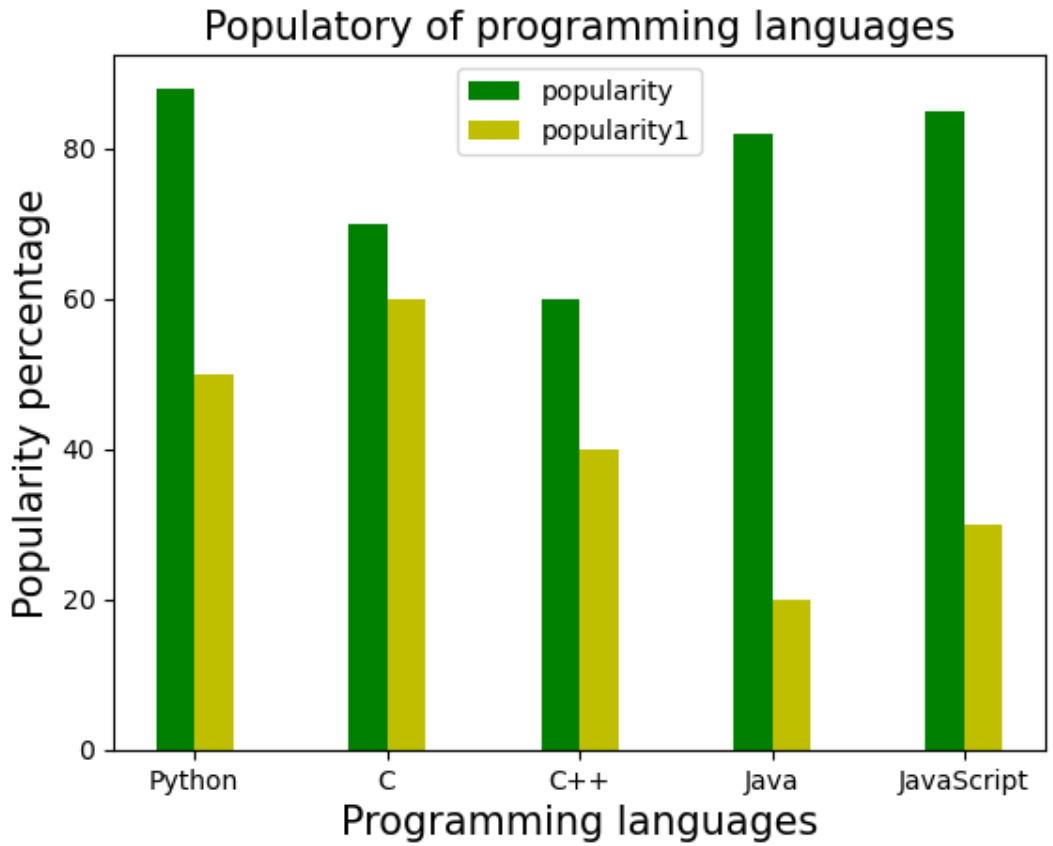
```
import matplotlib.pyplot as plt #here pyplot is a class
import numpy as np #used for creating arrays

#giving x parameter and their values
x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages
y = [88, 70, 60, 82, 85]
z = [50, 60, 40, 20, 30]

width = 0.2
p=np.arange(len(x)) #this will create an array of length of x which we can use
p1= [j+width for j in p]


plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)


#plotting the graph

plt.bar(p,y, width, color ='g', label='popularity')
plt.bar(p1,z, width, color ='y', label='popularity1')

plt.legend()
plt.show()
```

Output graph



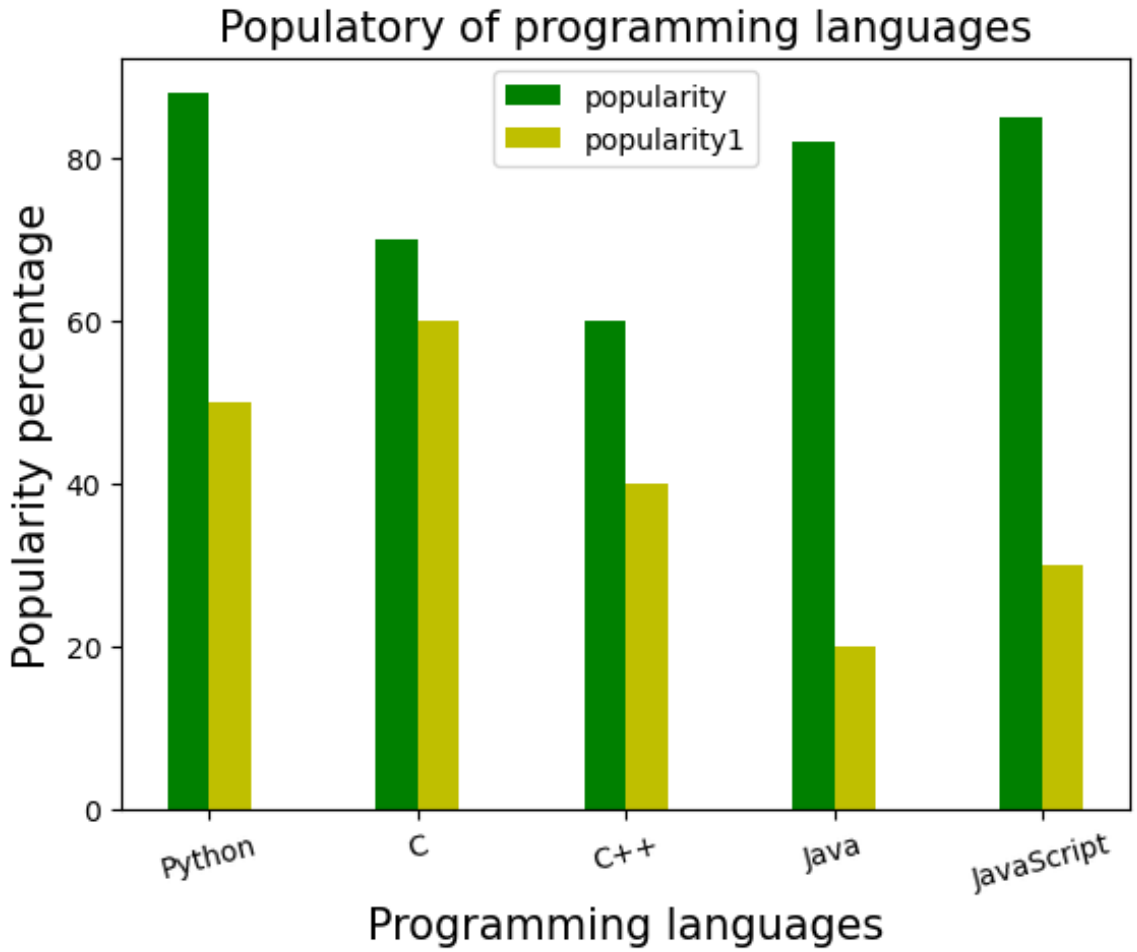
```
import matplotlib.pyplot as plt #here pyplot is a class
import numpy as np #used for creating arrays

#giving x parameter and their values
x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages
y = [88, 70, 60, 82, 85]
z = [50, 60, 40, 20, 30]

width = 0.2
p=np.arange(len(x)) #this will create an array of length of x which we can use
p1= [j+width for j in p]


plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)


#plotting the graph

plt.bar(p,y, width, color ='g', label='popularity')
plt.bar(p1,z, width, color ='y', label='popularity1')


plt.xticks(p+width/2, x) #here p+width is the position of the labels.
plt.legend()
plt.show()
```

Output graph



```
import matplotlib.pyplot as plt #here pyplot is a class
import numpy as np #used for creating arrays

#giving x parameter and their values
x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages
y = [88, 70, 60, 82, 85]
z = [50, 60, 40, 20, 30]

width = 0.2
p=np.arange(len(x)) #this will create an array of length of x which we can use
p1= [j+width for j in p]


plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)


#plotting the graph

plt.bar(p,y, width, color ='g', label='popularity')
plt.bar(p1,z, width, color ='y', label='popularity1')


plt.xticks(p+width/2, x, rotation=15)
plt.legend()
plt.show()
```

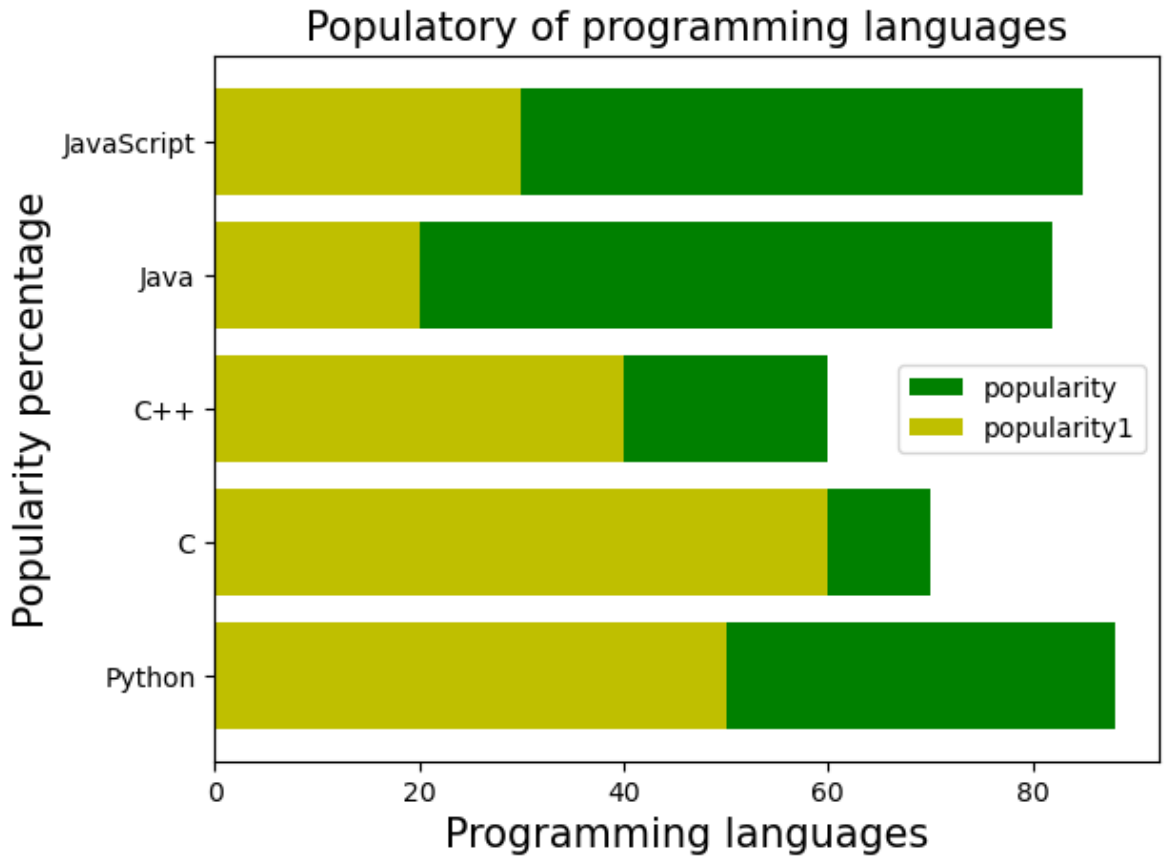
Output graph



# Horizontal bargraph

```
import matplotlib.pyplot as plt #here pyplot is a class

#giving x parameter and their values

x =["Python", "C", "C++", "Java", "JavaScript"]

#giving y parameter which explains the popularity of these languages

y = [88, 70, 60, 82, 85]
z = [50, 60, 40, 20, 30]

plt.xlabel("Programming languages", fontsize = 15)
plt.ylabel("Popularity percentage", fontsize = 15)
plt.title("Populatory of programming languages", fontsize = 15)

#plotting the graph

plt.barh(x,y, color ='g', label='popularity')
plt.barh(x,z, color ='y', label='popularity1')

plt.legend()
plt.show()
```

Output graph



# CHANGING AXIS using MATPLOTLIB

These are some functions that we can use to make changes to the axis.

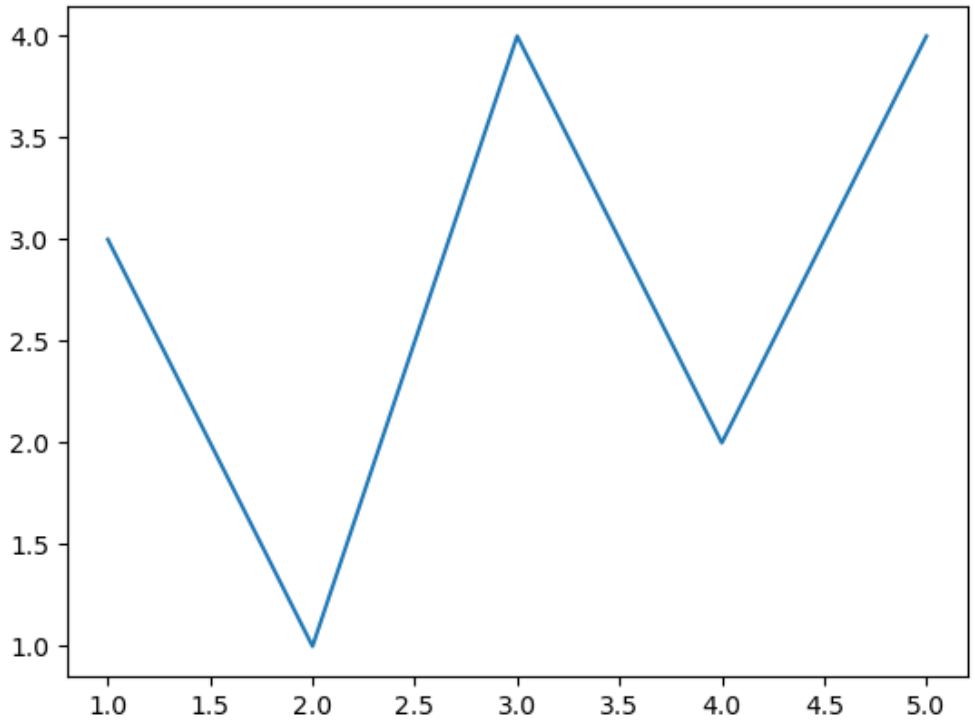
- xticks()
- yticks()
- xlim()
- ylim()
- axis()

Reference video link: https://youtu.be/yG_kr-CVi8s?si=yI2cbTMvkeo1P-60

# CHANGING AXIS using MATPLOTLIB

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,4,2,4]

plt.plot(x,y)

plt.show()
```

Output graph



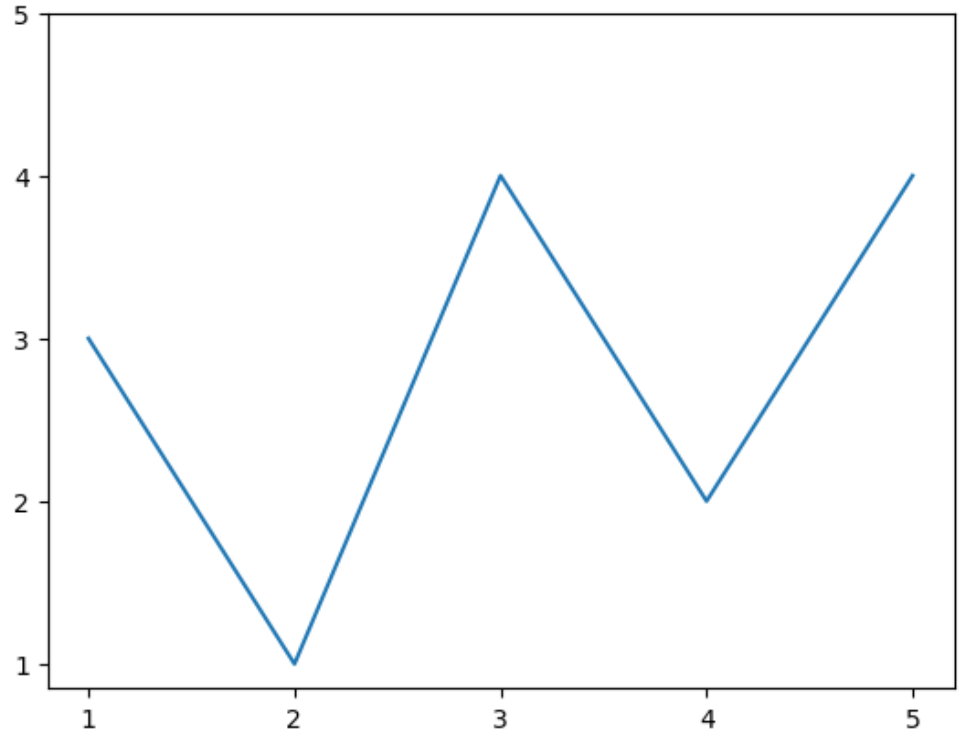

Reference video link:
https://youtu.be/yG_kr-CVi8s?si=yI2cbTMvkeo1P-60

# CHANGING AXIS using MATPLOTLIB

Let’s say we want to change the coordinates on x and y axis.

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,4,2,4]

plt.plot(x,y)

plt.xticks(x)
plt.yticks(x)

plt.show()
```

Output graph

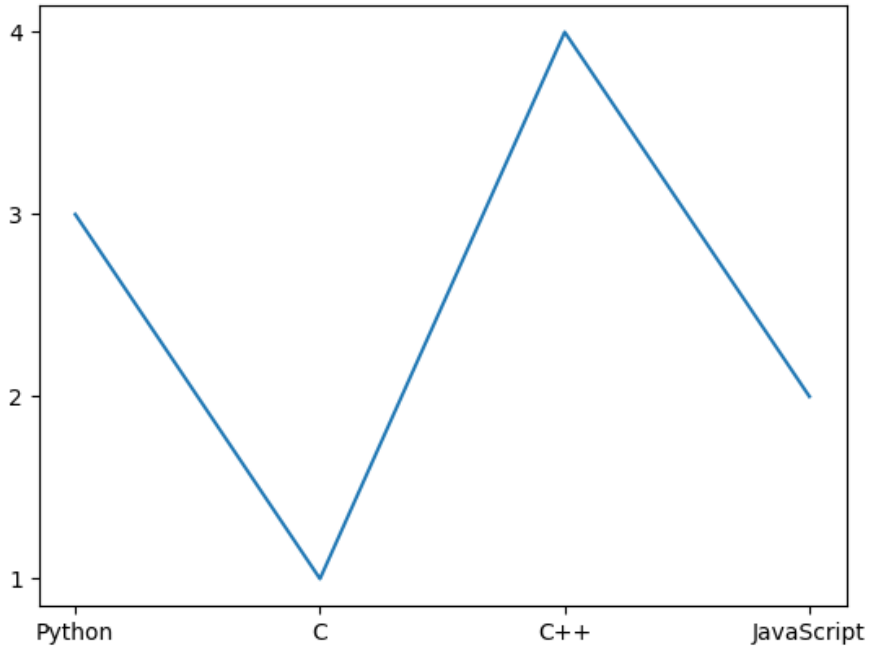


# CHANGING AXIS using MATPLOTLIB

Let’s add labels to the x axis.

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [3,1,4,2]

plt.plot(x,y)

plt.xticks(x, labels = ['Python', 'C', 'C++', 'JavaScript'])
plt.yticks(x)

plt.show()
```

Output graph

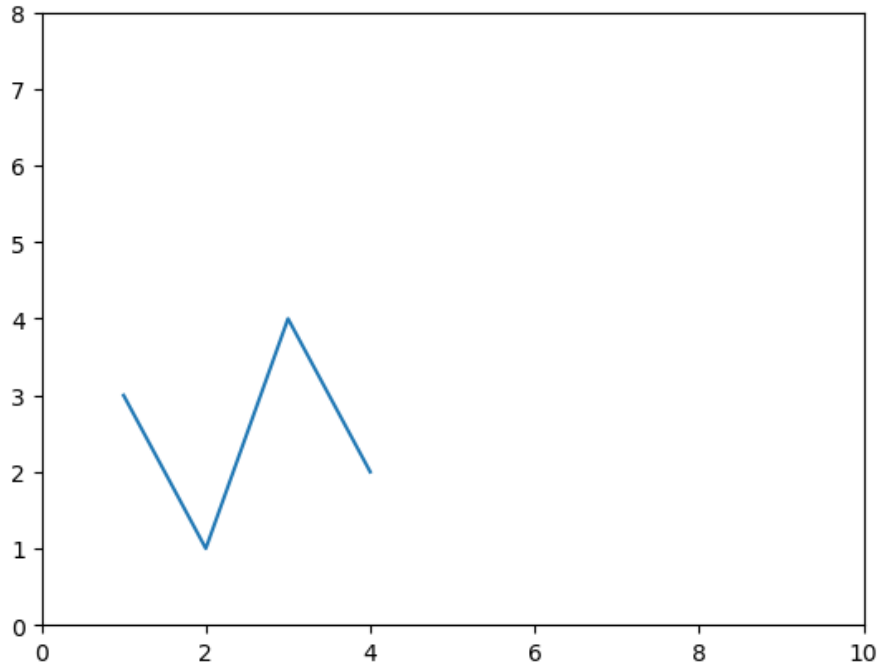


# CHANGING AXIS using MATPLOTLIB

Let’s change limit of the axis. (option 1)

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [3,1,4,2]

plt.plot(x,y)

#plt.xticks(x, labels = ['Python', 'C', 'C++', 'JavaScript'])
#plt.yticks(x)

plt.xlim(0,10)
plt.ylim(0,8)

plt.show()
```

Output graph

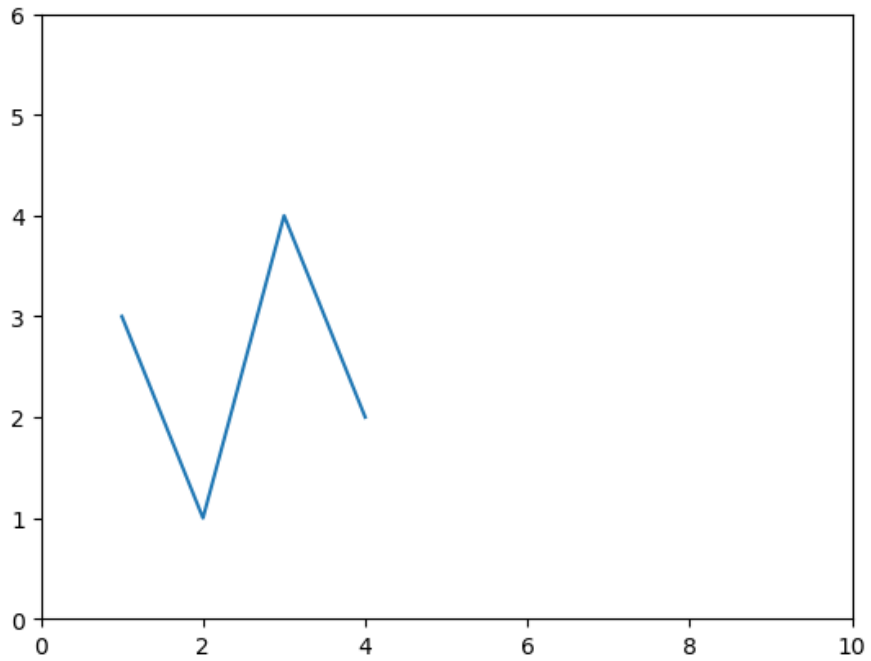


# CHANGING AXIS using MATPLOTLIB

Let’s change limit of the axis. (option 2)

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [3,1,4,2]

plt.plot(x,y)

#plt.xticks(x, labels = ['Python', 'C', 'C++', 'JavaScript'])
#plt.yticks(x)

#plt.xlim(0,10)
#plt.ylim(0,8)

plt.axis([0,10,0,6])

plt.show()
```

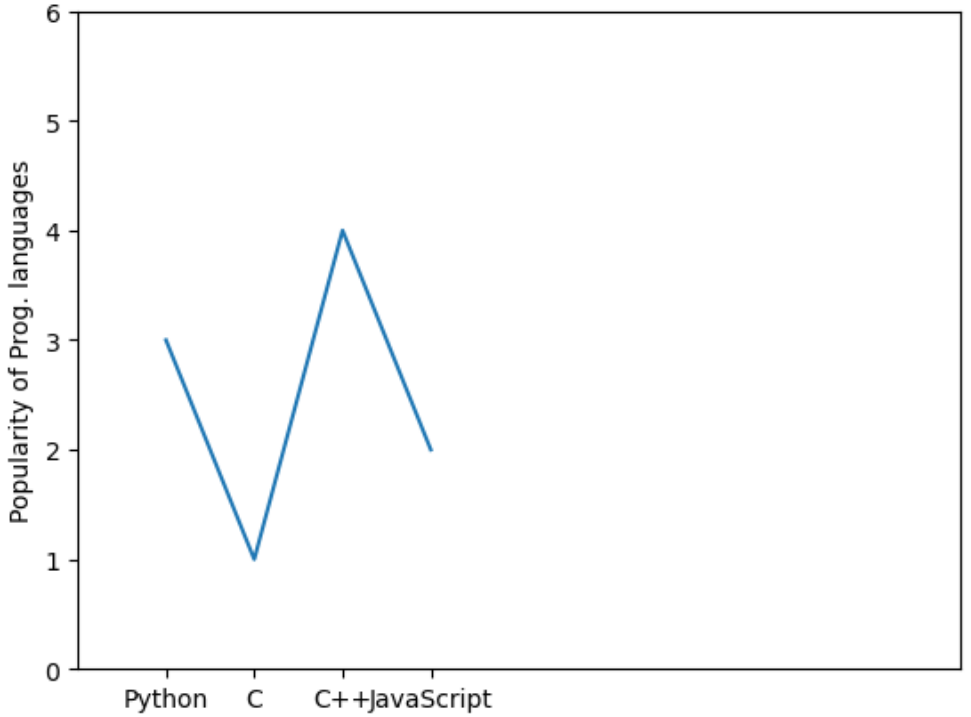
Output graph



# Adding Y-axis label

```
#let's make a graph first

import matplotlib.pyplot as plt

#x = [1,2,3,4]
#y = [3,1,4,2]

plt.plot(x,y)

plt.xticks(x, labels = ['Python', 'C', 'C++', 'JavaScript'])
plt.ylabel('Popularity of Prog. languages')

plt.axis([0,10,0,6])

plt.show()
```

Output graph



# Adding text to plots

Types of texts

- Adding text at an arbitrary location.
- Annotate: Adding an annotation with an optional arrow, at an arbitrary location.
- Xlabel: Add a label to the x axis.
- Ylabel: add a label to the y axis.
- Title: Add a title to the Axes.

Reference video link: https://youtu.be/WT8r4z7fBfU?si=4O4bFOFE1IIakQ8r

# Adding text to plots

Types of texts

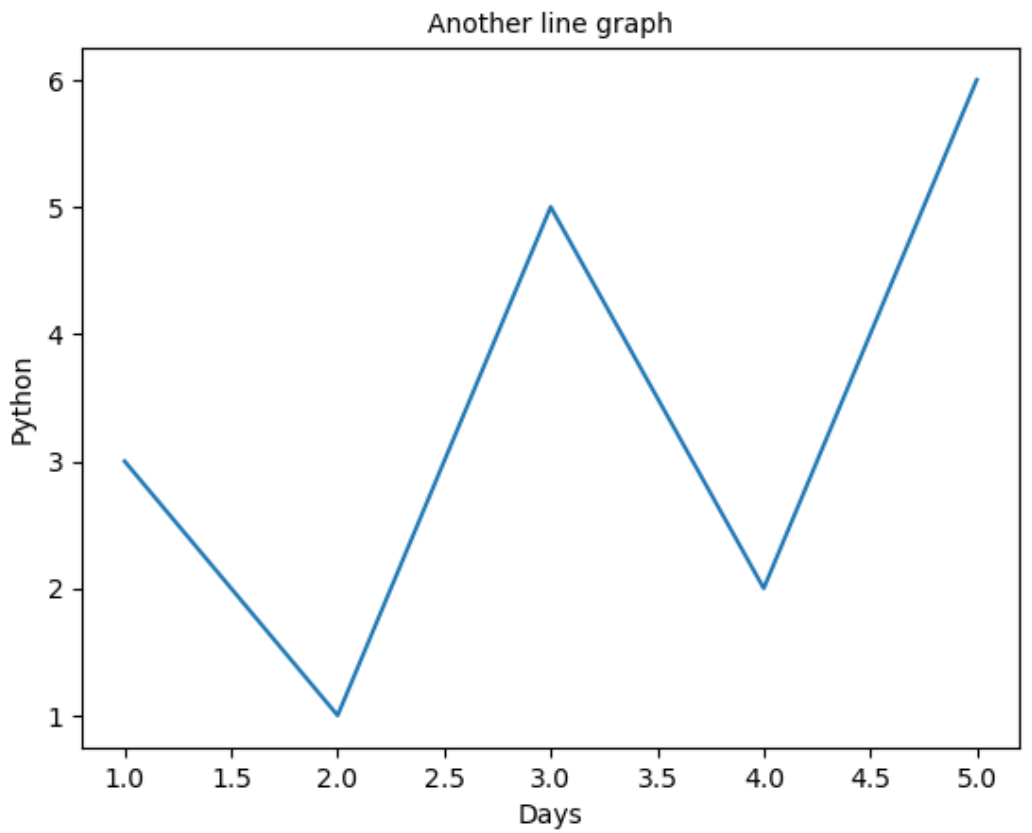
- Adding text at an arbitrary location.
- Annotate: Adding an annotation with an optional arrow, at an arbitrary location.
- Xlabel: Add a label to the x axis.
- Ylabel: add a label to the y axis.
- Title: Add a title to the Axes.

Reference video link: https://youtu.be/WT8r4z7fBfU?si=4O4bFOFE1IIakQ8r

# Adding text to plots

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,5,2,6]

plt.plot(x,y)

plt.title("Another line graph", fontsize=10)
plt.xlabel('Days', fontsize=10)
plt.ylabel('Python', fontsize=10)

plt.show()
```

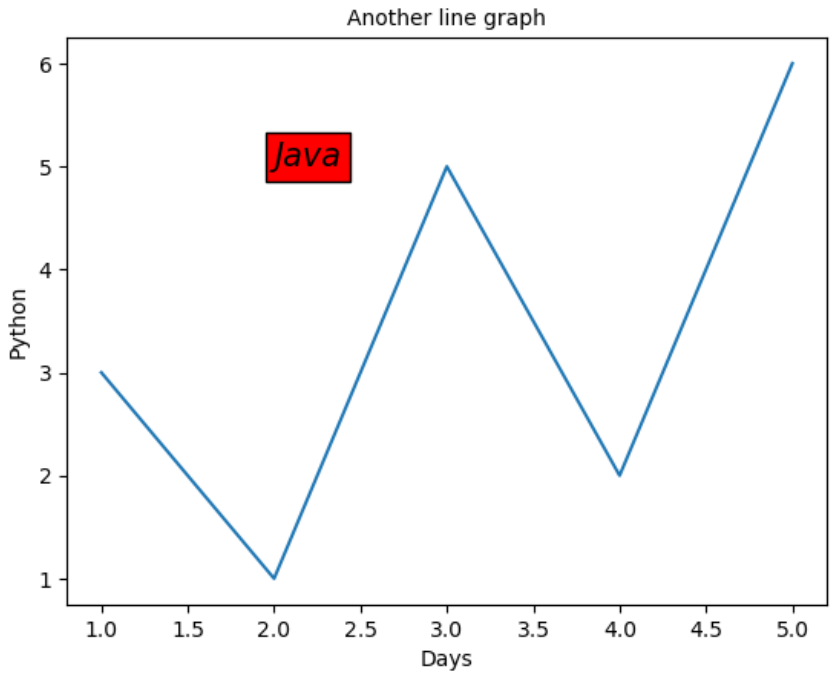
Output graph



# Adding text to plots

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,5,2,6]

plt.plot(x,y)

plt.title("Another line graph", fontsize=10)
plt.xlabel('Days', fontsize=10)
plt.ylabel('Python', fontsize=10)

#adding text inside the graph
#here bbbox is a dictionary as it has a lot of characteristics that can be defined

plt.text(2,5,"Java", fontsize=15,style='italic', bbox={"facecolor":'red'})

plt.show()
```

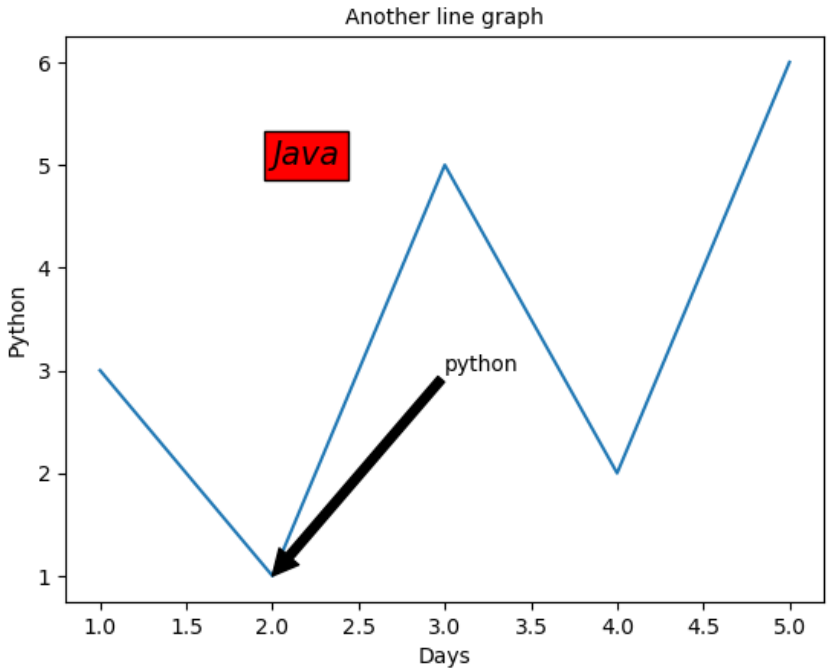
Output graph



# Adding text to plots

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,5,2,6]

plt.plot(x,y)

plt.title("Another line graph", fontsize=10)
plt.xlabel('Days', fontsize=10)
plt.ylabel('Python', fontsize=10)

#adding text inside the graph
#here bbox is a dictionary as it has a lot of characteristics that can be defined

plt.text(2,5,"Java", fontsize=15,style='italic', bbox={"facecolor":'red'})

#annotation
plt.annotate("python", xy=(2,1), xytext=(3,3),
arrowprops=dict(facecolor='black',shrink=100))

plt.show()
```

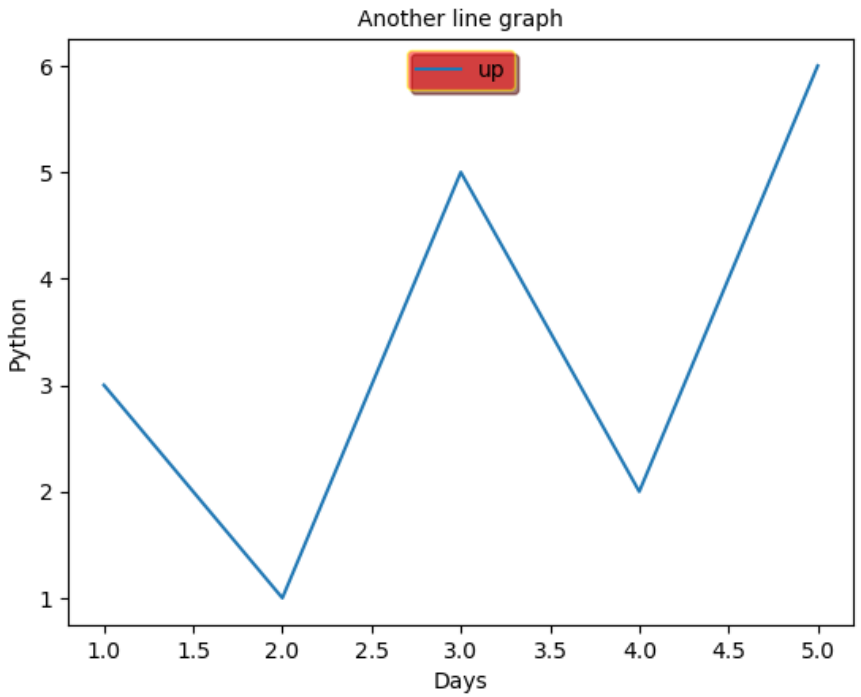
Output graph



# Adding text to plots

```
#let's make a graph first

import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,1,5,2,6]

plt.plot(x,y)

plt.title("Another line graph", fontsize=10)
plt.xlabel('Days', fontsize=10)
plt.ylabel('Python', fontsize=10)

#adding text inside the graph
#here bbbox is a dictionary as it has a lot of characteristics that can be defined
#plt.text(2,5,"Java", fontsize=15,style='italic', bbox={"facecolor":'red'})

#annotation
#plt.annotate("python", xy=(2,1), xytext=(3,3), arrowprops=dict(facecolor='black',shrink=100))

plt.legend(['up'], loc=9, facecolor='red', edgecolor='yellow', framealpha=0.5, shadow=True)

plt.show()
```

Output graph

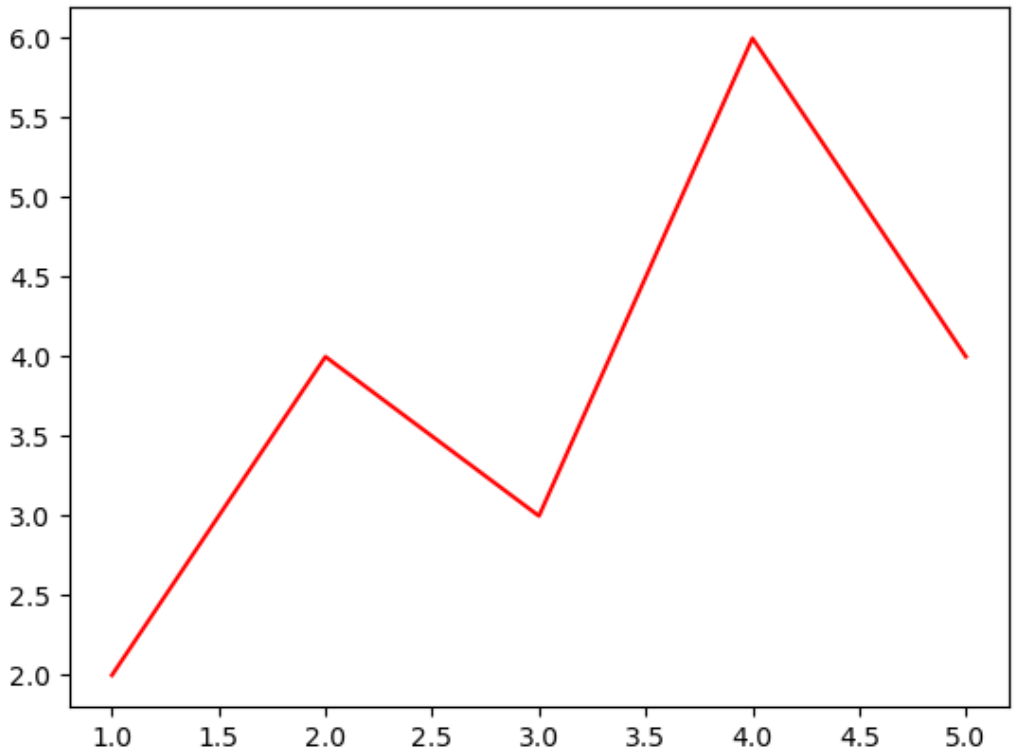


# SAVE FIG FUNCTION

- Create a new folder on desktop.
- Clear the link on the path and type cmd. #You'll be taken to the CLI.
- Type 'Jupyter Notebook' on the CLI. #It'll open the Jupyter notebook
- This is where you can type your code and save it in the new folder you created on desktop.
- Add dpi parameter to enhance the image quality.

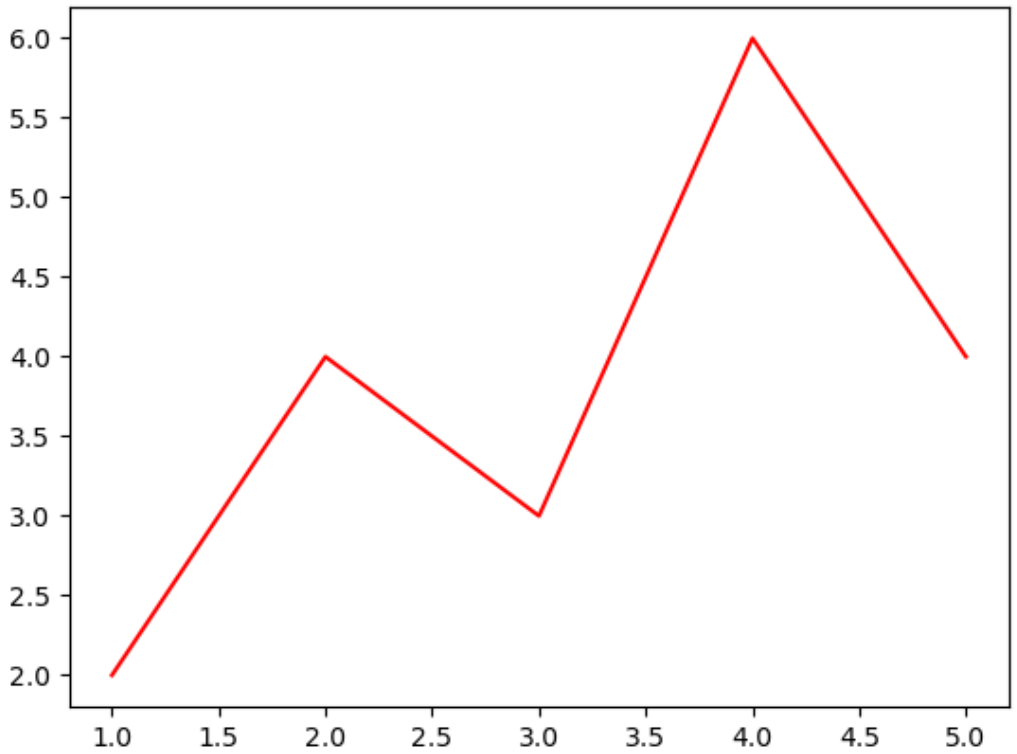
# SAVE FIG FUNCTION (dpi)

```
import matplotlib.pyplot as plt

xpt=[1, 2, 3, 4, 5]
ypt=[2, 4, 3, 6, 4]

plt.plot(xpt,ypt, color='r')

plt.savefig("Line plot-graph", dpi=2000)

plt.show()
```



# SAVE FIG FUNCTION (facecolor)

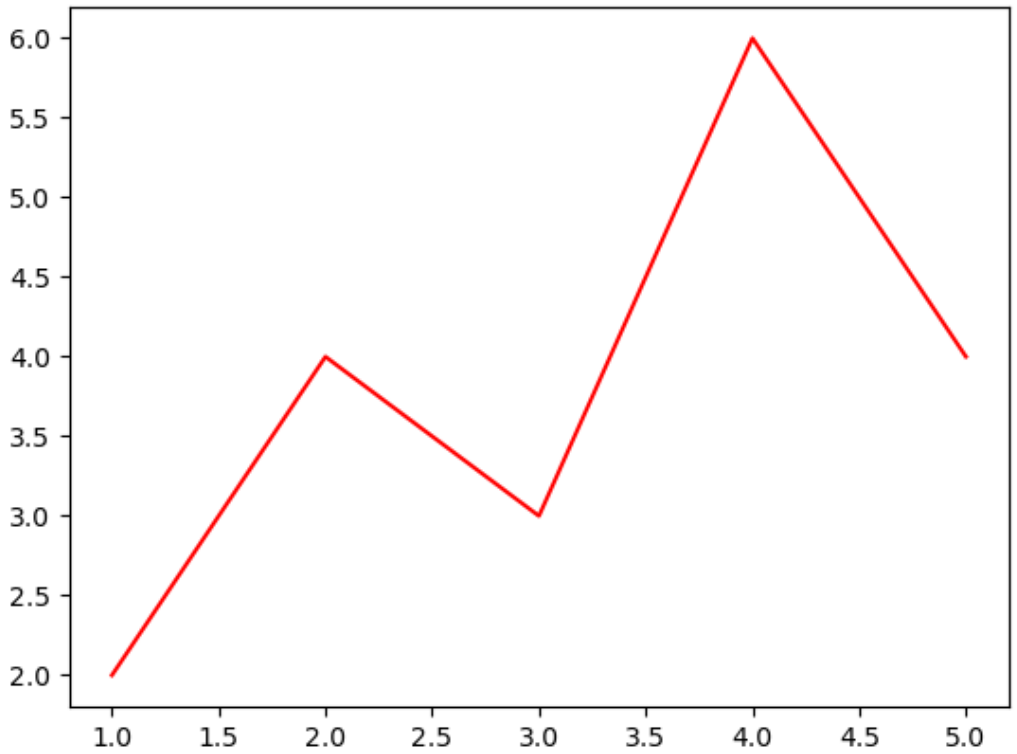
```
import matplotlib.pyplot as plt

xpt=[1, 2, 3, 4, 5]
ypt=[2, 4, 3, 6, 4]

plt.plot(xpt,ypt, color='r')

plt.savefig("Line plot-graph", dpi=2000, facecolor='g')
#this creates a green padding in the saved image file

plt.show()
```



# SAVE FIG FUNCTION (saving in pdf/jpg format)

```
import matplotlib.pyplot as plt

xpt=[1, 2, 3, 4, 5]
ypt=[2, 4, 3, 6, 4]

plt.plot(xpt,ypt, color='r')

plt.savefig("Line plot-graph.pdf", dpi=2000, facecolor='g') #this saves the image file in pdf format

plt.show()
```

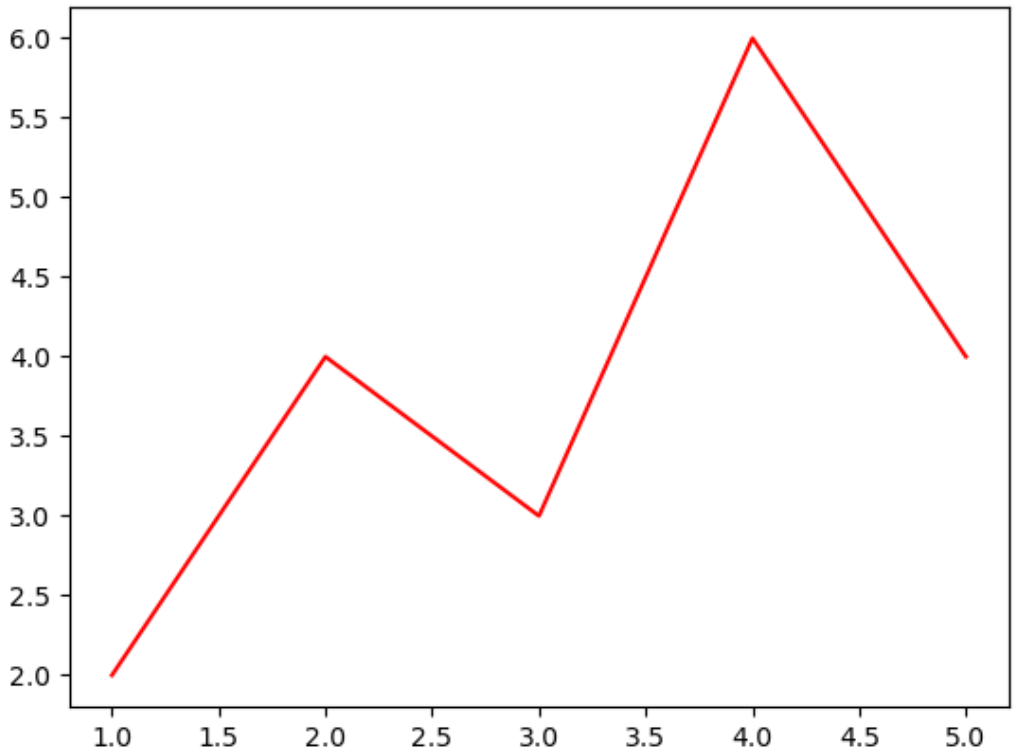
Note: .png is the default format of the file in which it is saved



# SAVE FIG FUNCTION (saving in transparent format)

```
import matplotlib.pyplot as plt

xpt=[1, 2, 3, 4, 5]
ypt=[2, 4, 3, 6, 4]

plt.plot(xpt,ypt, color='r')

plt.savefig("Line plot-graph.pdf", dpi=2000, facecolor='g',
transparent=True)

plt.show()
```

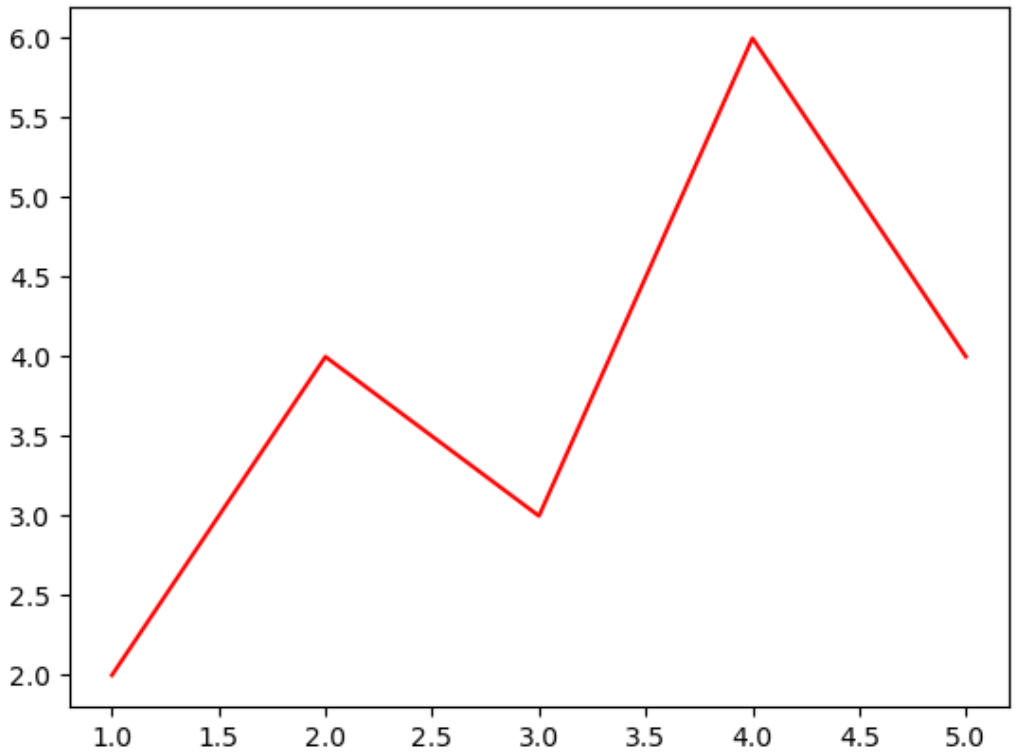
Note: .png is the default format of the file in which it is saved



# SAVE FIG FUNCTION (saving in bbox format)

```
import matplotlib.pyplot as plt

xpt=[1, 2, 3, 4, 5]
ypt=[2, 4, 3, 6, 4]

plt.plot(xpt,ypt, color='r')

plt.savefig("Line plot-graph.pdf", dpi=2000, facecolor='g',
transparent=False, bbox_inches='tight')
plt.show()
```

Note: .png is the default format of the file in which it is saved



# BOX VS WHISKER PLOT using MATPLOTLIB

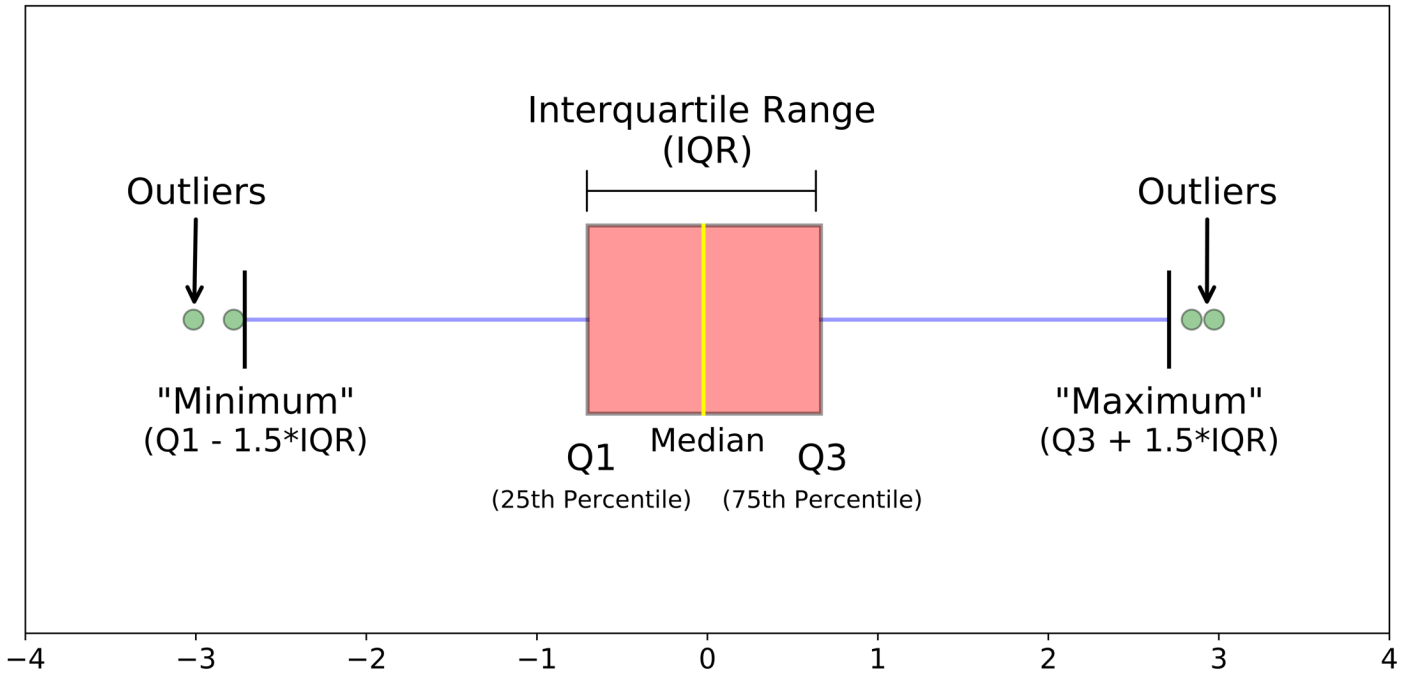
- Create a heading: Box Vs Whisker plot
- Basic syntax/ instructions:

```
import matplotlib pyplot as plt
x = [] # x is one parameter
plt.boxplot(x)
plt.show()
```

- It all starts with x = [10, 20, 30, 40, 50, 60, 70]

Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

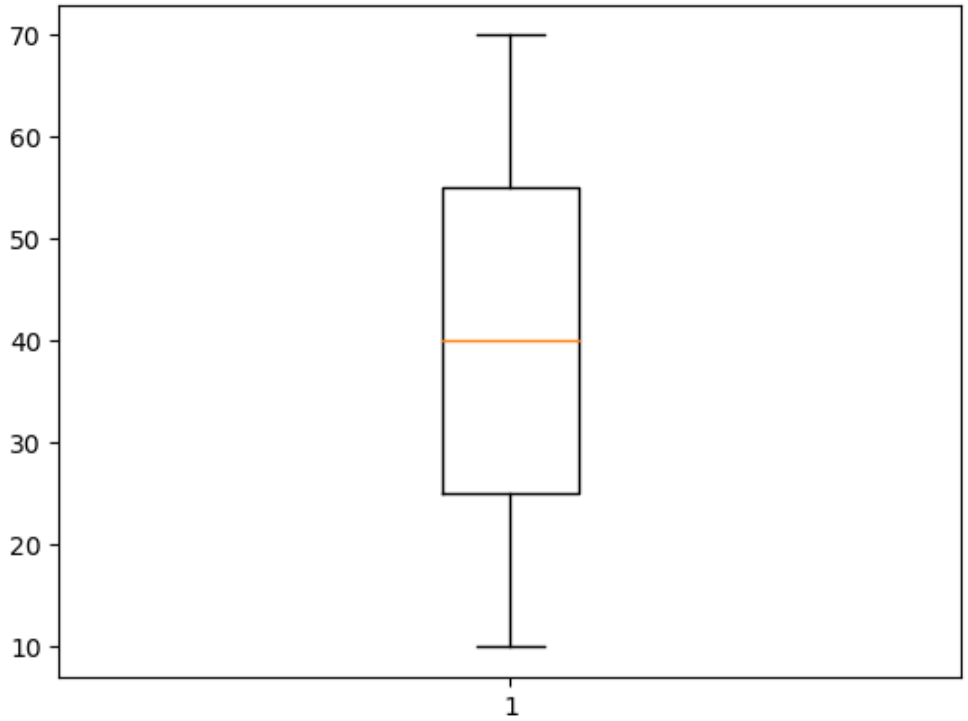
- It all starts with x = [10, 20, 30, 40, 50, 60, 70]
- A linear plot is created first.

Example: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review

Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

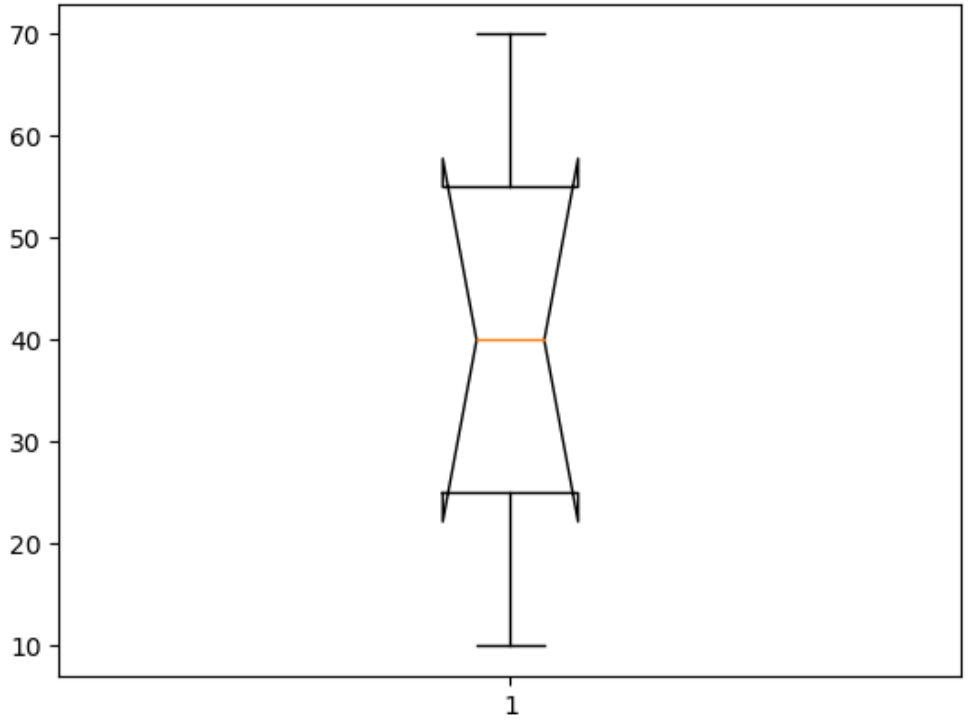
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

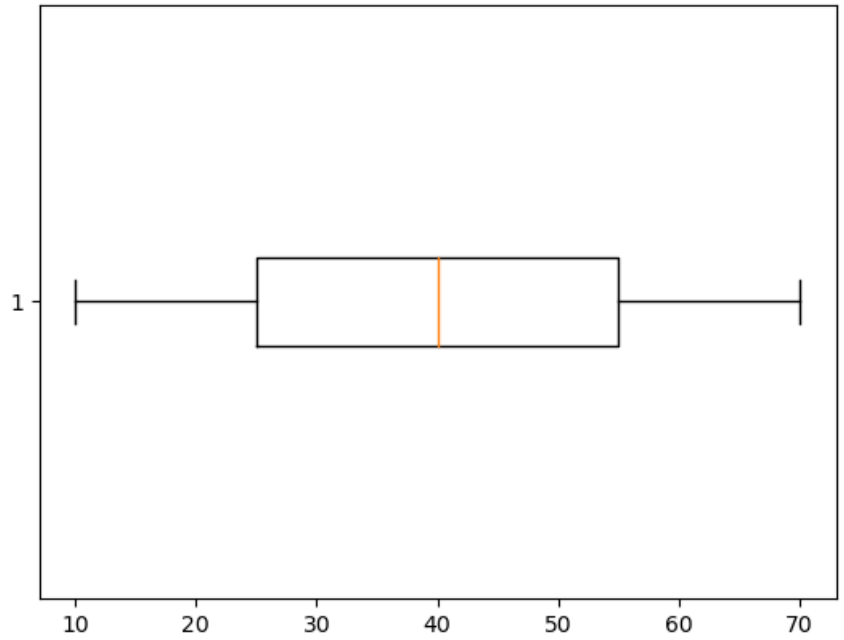
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x, notch=True)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

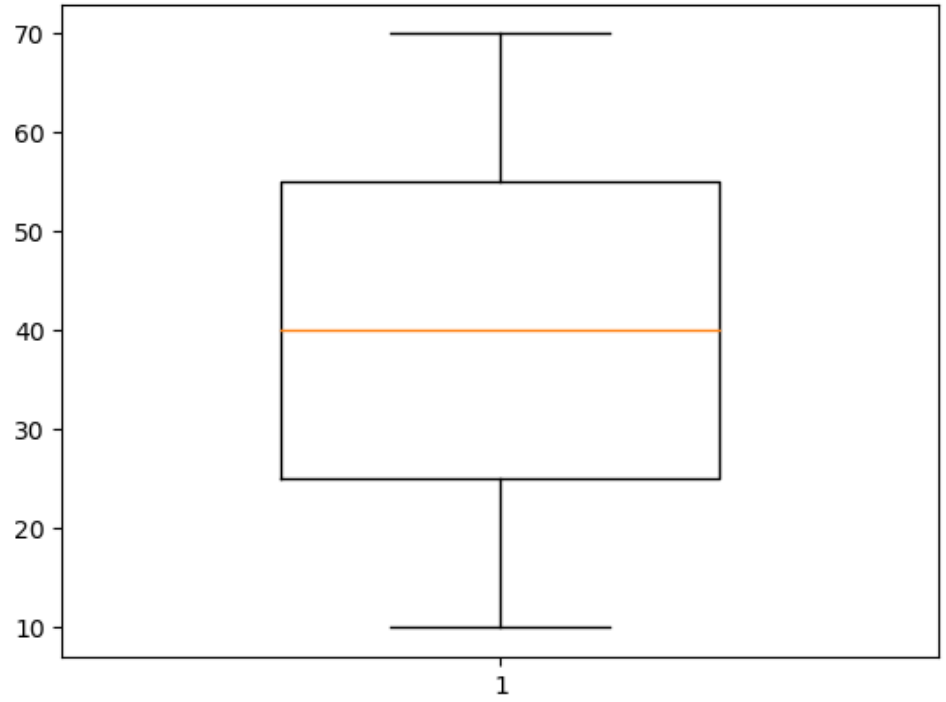
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

#horizontal boxplot

plt.boxplot(x, vert=False)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

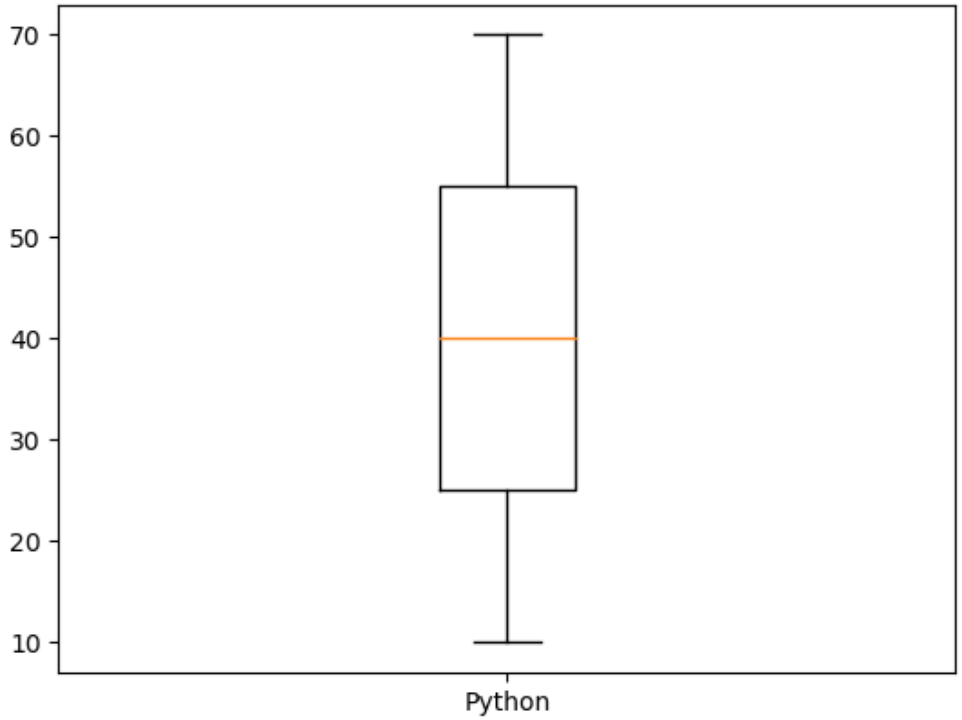
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x, widths=0.5)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

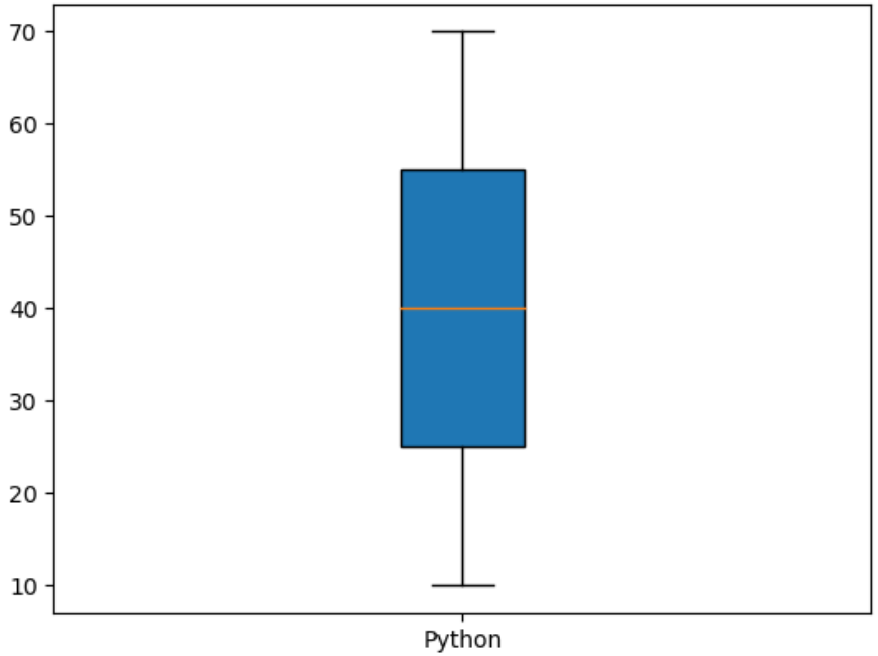
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x, labels=['Python'])

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

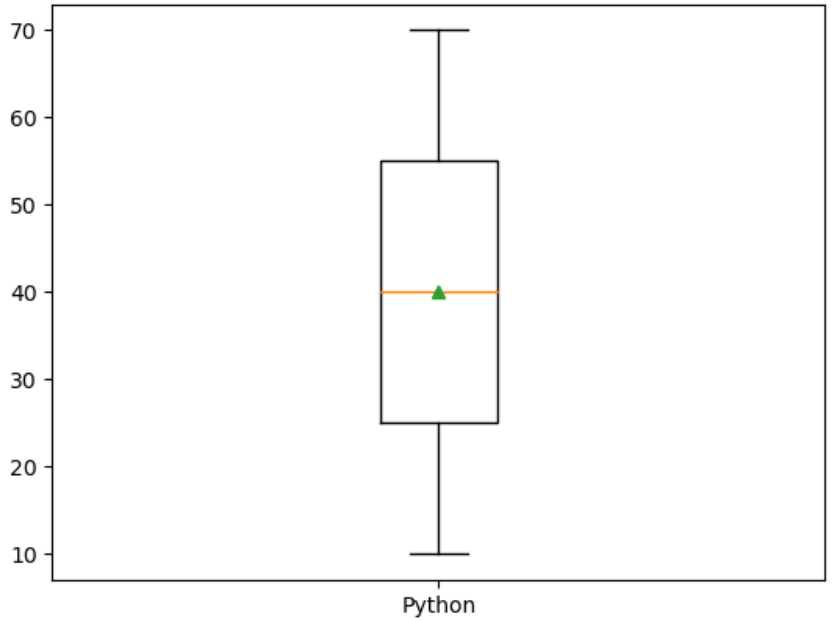
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x, labels=['Python'], patch_artist=True)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

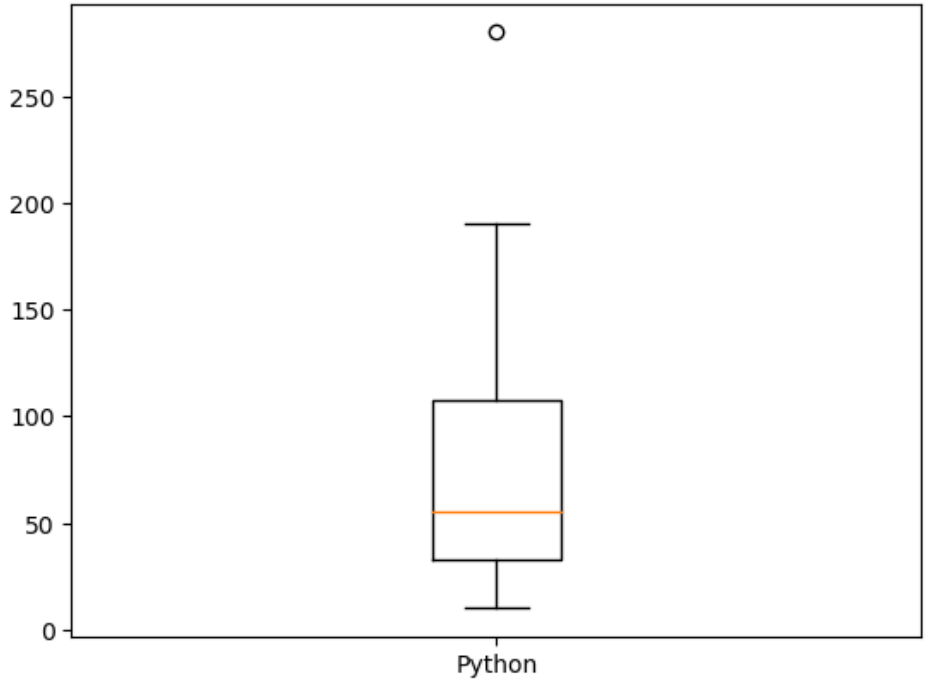
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70]

plt.boxplot(x, labels=['Python'], showmeans=True)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

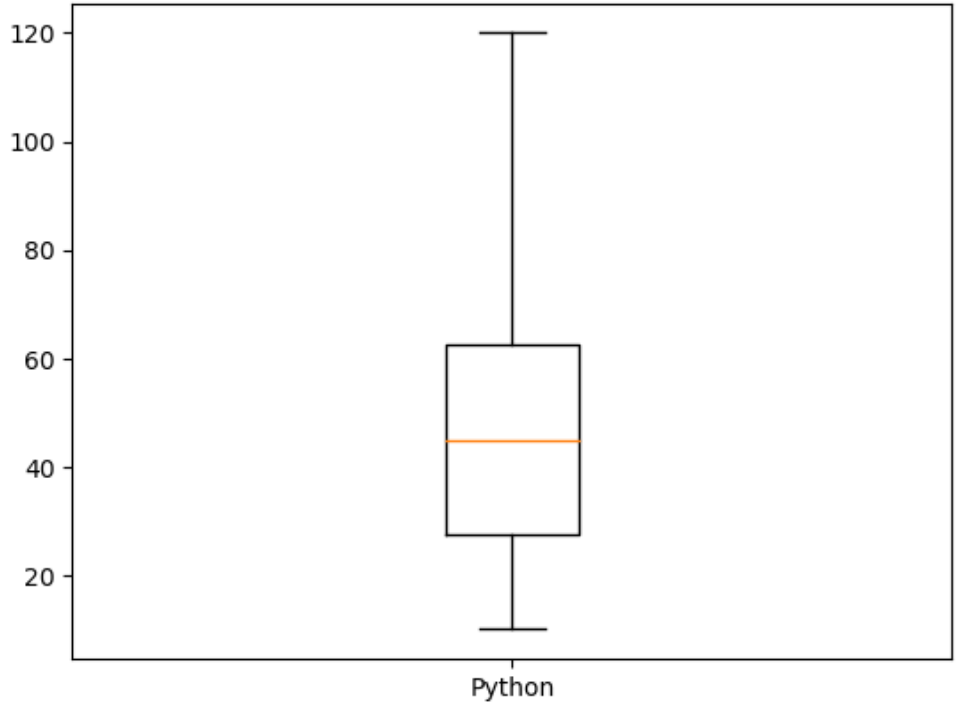
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120, 190, 280]

plt.boxplot(x, labels=['Python'])

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

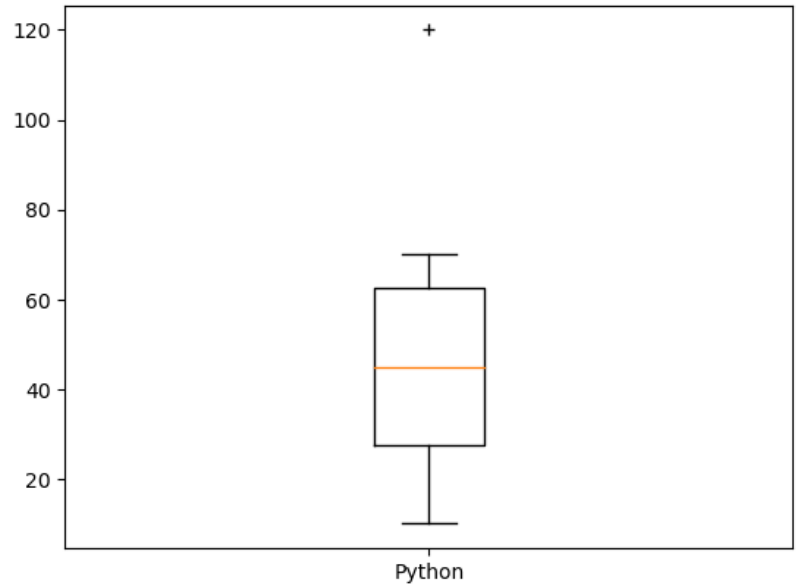
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot

plt.boxplot(x, labels=['Python'], whis=2)

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

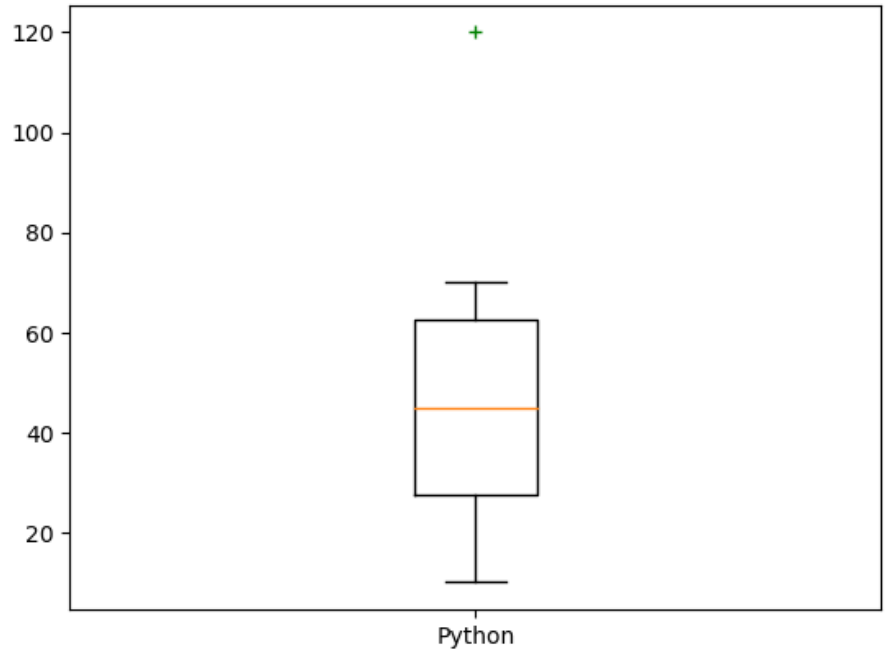
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot and changing the notation

plt.boxplot(x, labels=['Python'], sym='+')

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

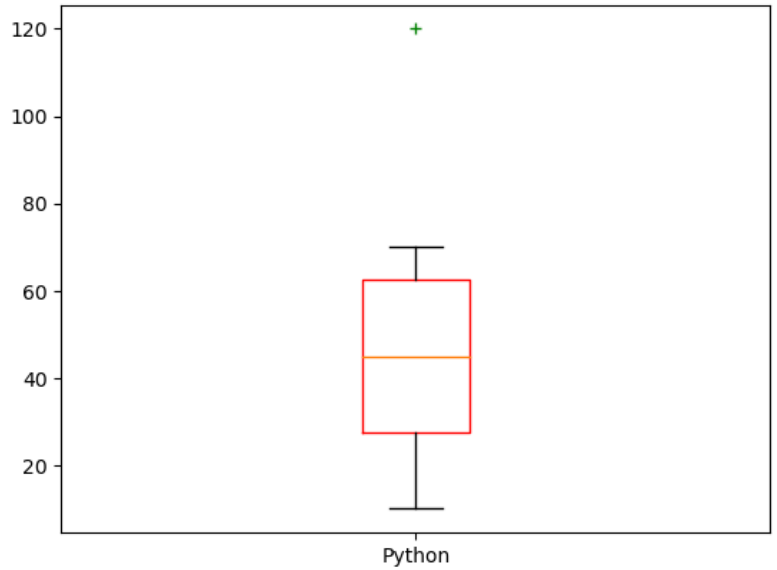
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot and changing the notation

plt.boxplot(x, labels=['Python'], sym='g+')

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

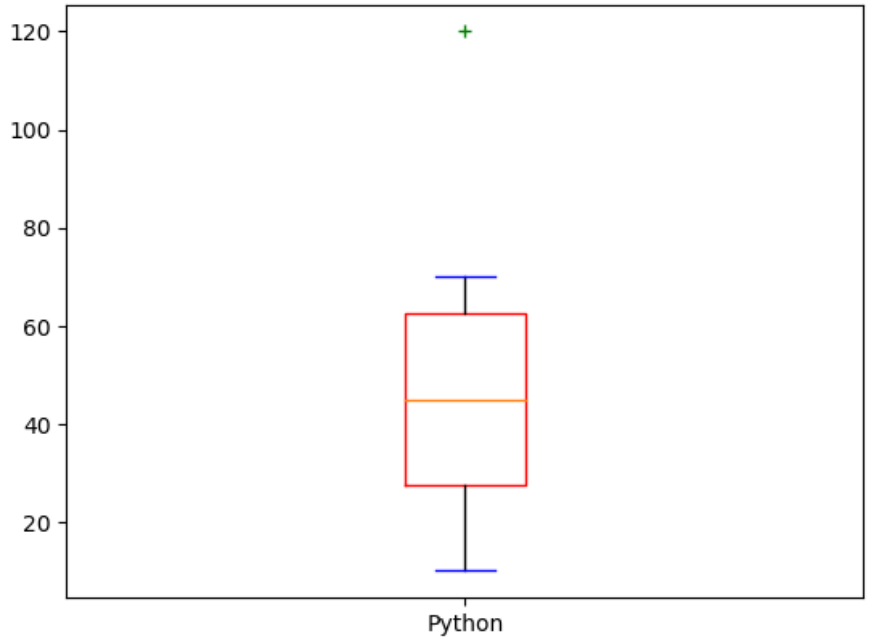
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot and changing the notation

plt.boxplot(x, labels=['Python'], sym='g+', boxprops=dict(color='r'))

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

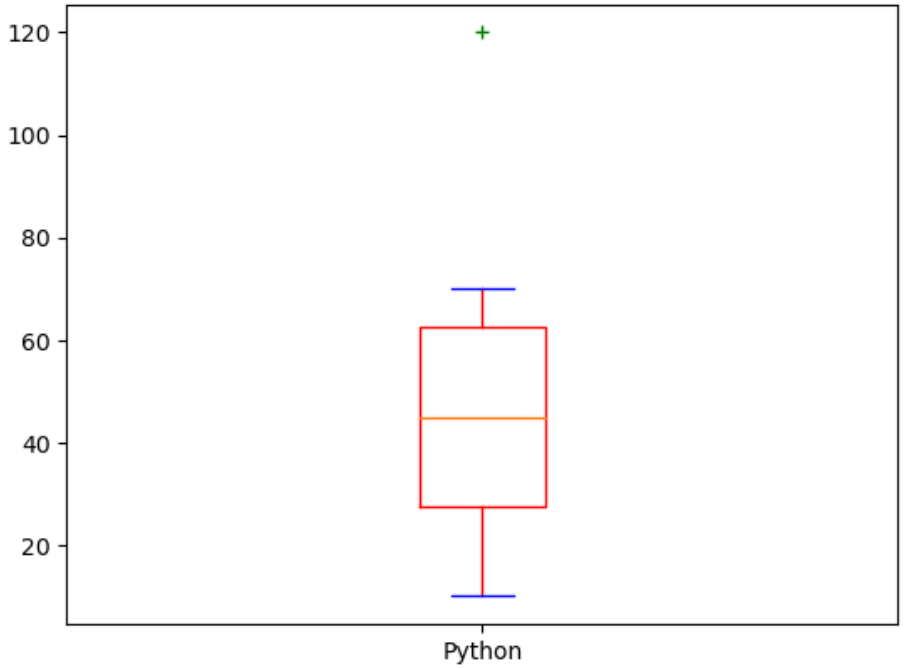
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot and changing the notation

plt.boxplot(x, labels=['Python'], sym='g+’,
boxprops=dict(color='r'), capprops=dict(color='b'))

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

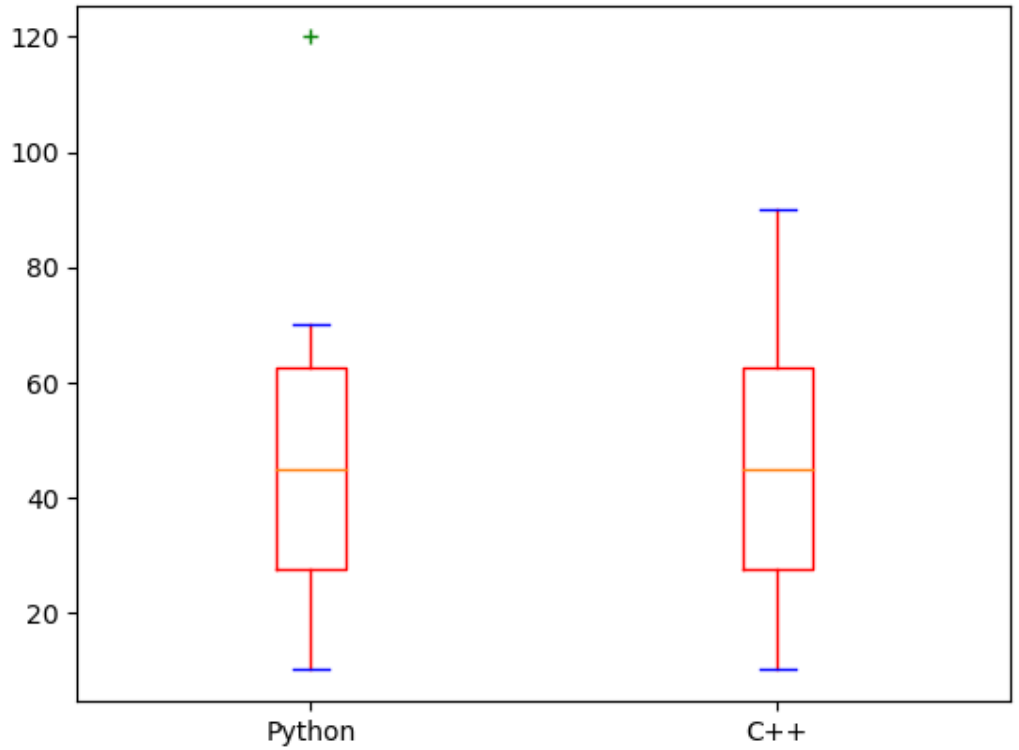
```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]

#joining the whisker plot and changing the notation

plt.boxplot(x, labels=['Python'], sym='g+’,
boxprops=dict(color='r'), capprops=dict(color='b’),
whiskerprops=dict(color='r'))

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# BOX VS WHISKER PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40, 50, 60, 70, 120]
x1= [10, 20, 30, 40, 50, 60, 70, 90]

y=[x,x1]

#joining the whisker plot and changing the notation

plt.boxplot(y,  labels=['Python', 'C++'], sym='g+',
boxprops=dict(color='r'), capprops=dict(color='b'),
whiskerprops=dict(color='r'))

plt.show()
```




Reference video link: https://youtu.be/rweaW76GV_Y?si=7bRZQrwObtuIUdfr

# PIE CHART/ PLOT using MATPLOTLIB

Create a heading: Pie chart/ plot

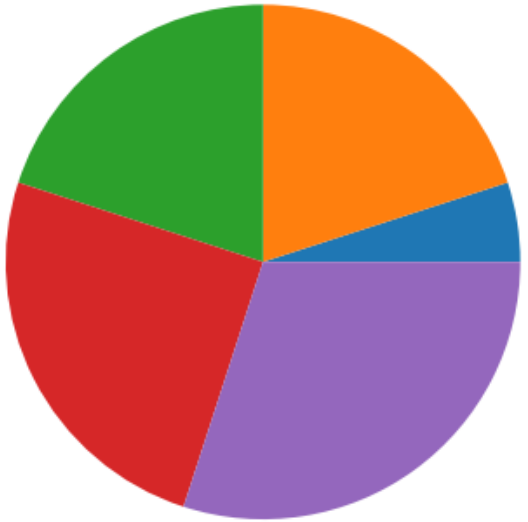
Basic syntax:

```
import matplotlib.pyplot as plt
x = []
plt.pie(x)
plt.show()
```

Reference video link:
https://youtu.be/PSji21jUNO0?si=Lks5-d4GUUMBukzH

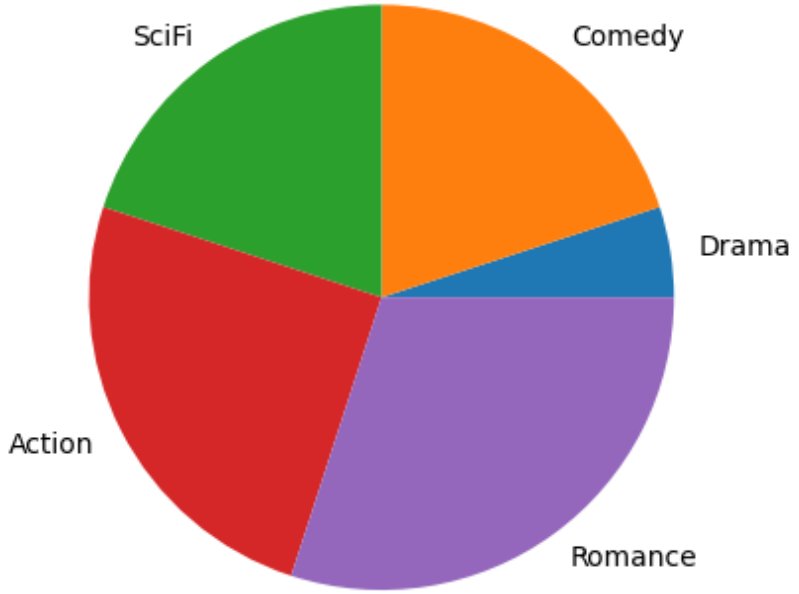
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt
#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance'] #these act as l
plt.pie(x)
plt.show()
```



# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt
#giving two parameters
x=[5,20,20,25,30] #the total is 100
#these act as labels
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
plt.pie(x, labels=y)
plt.show()
```

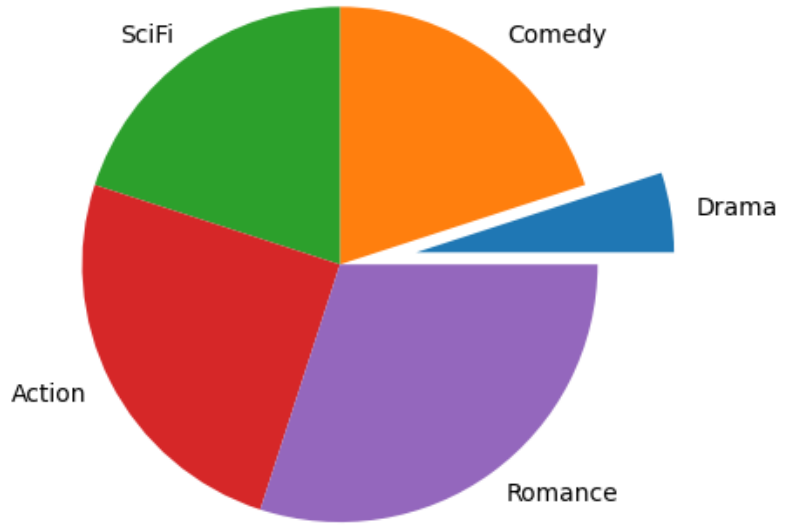


# Explode function

- Maybe you want one of the wedges to stand out?
- The explode parameter allows you to do that.
- The explode parameter, if specified, and not None, must be an array with one value for each wedge.

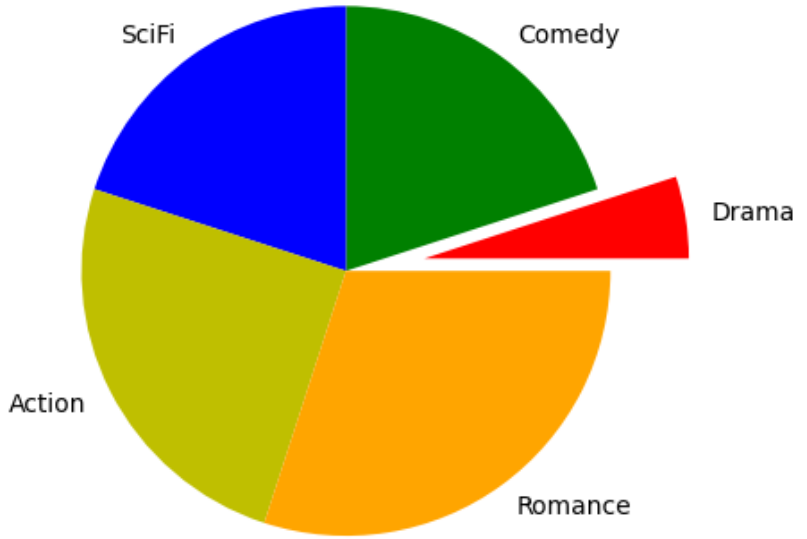
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
plt.pie(x, labels=y, explode=ex)
plt.show()
```



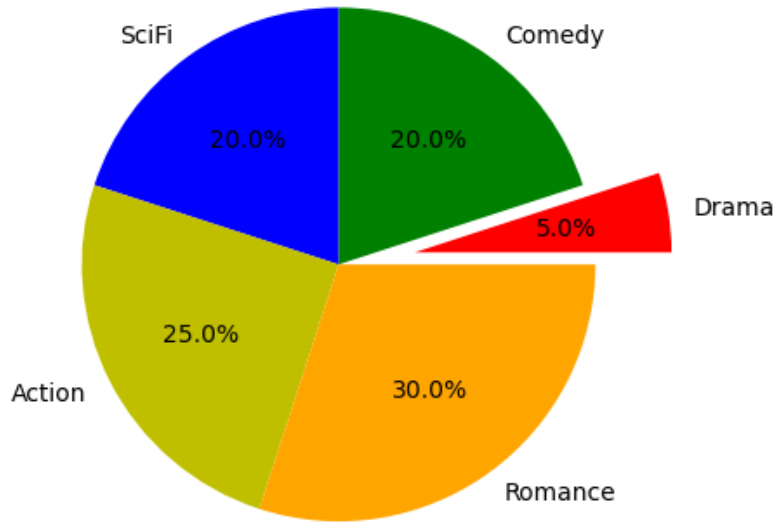
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c)
plt.show()
```



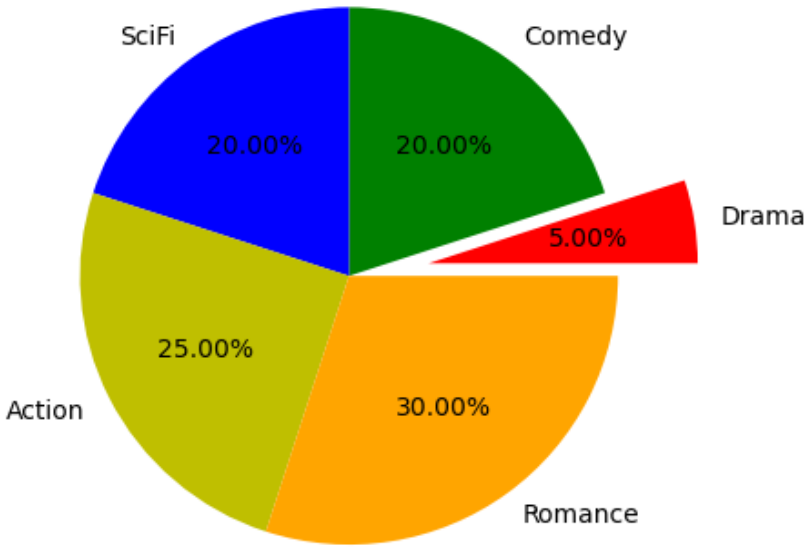
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c, autopct="%0.1f%%")
plt.show()
```



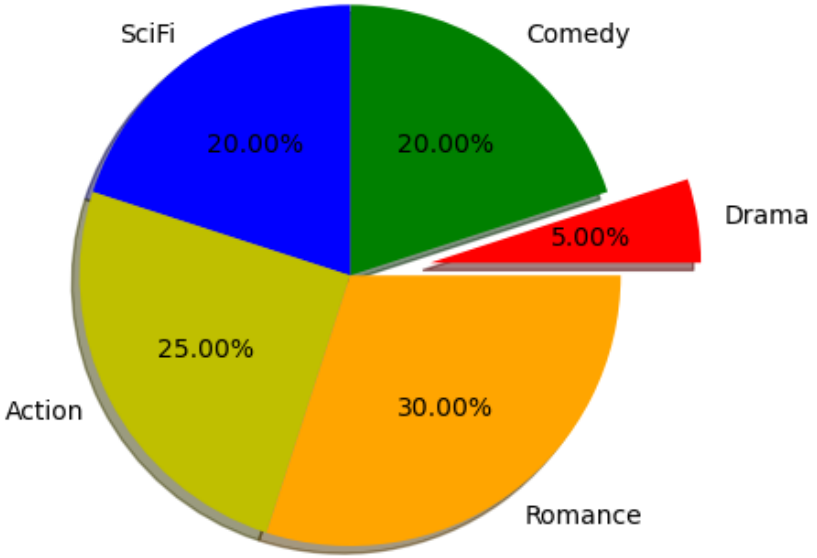
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c, autopct="%0.2f%%")
plt.show()
```



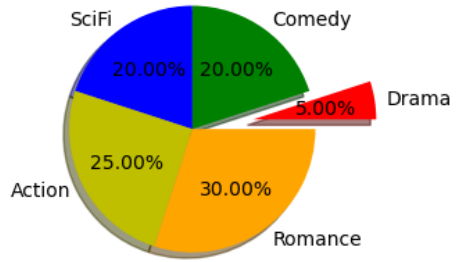
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c, autopct="%0.2f%%", shadow=True)
plt.show()
```



# PIE CHART/ PLOT using MATPLOTLIB

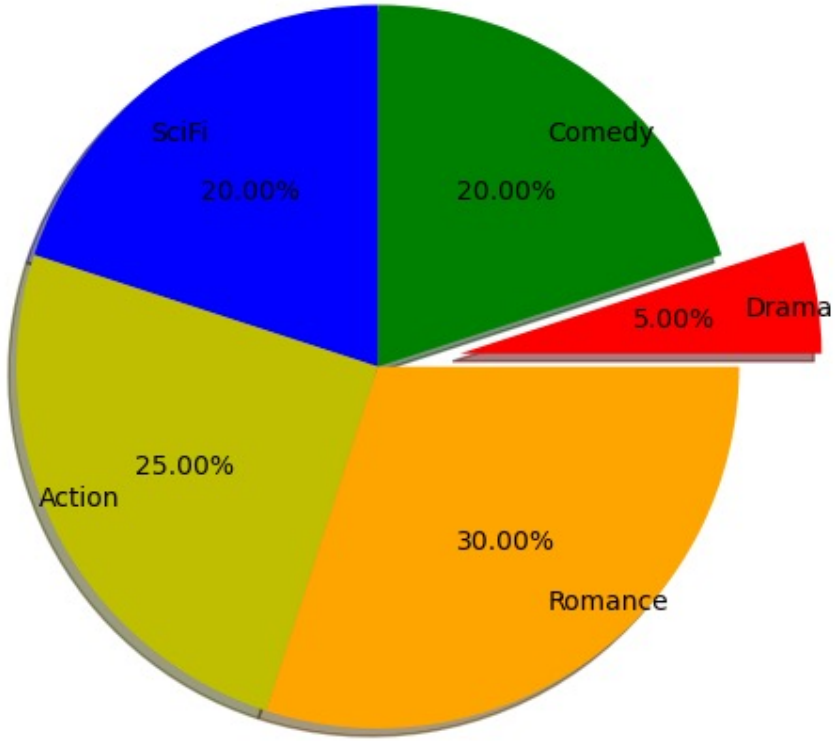
```
import matplotlib.pyplot as plt
```




```
#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c, autopct=”%0.2f%%”, shadow=True, radius=0.6)
plt.show()
```

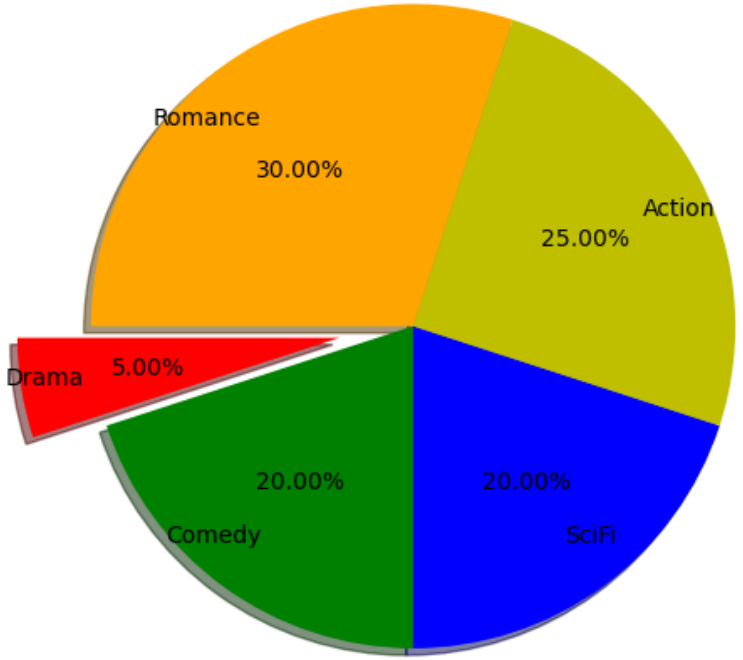
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c, autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8)
plt.show()
```

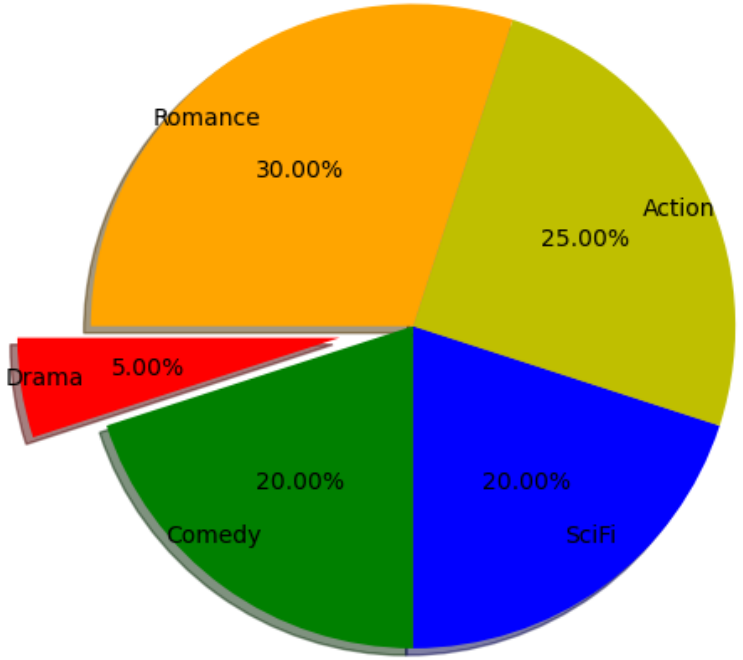


# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8,
startangle = 180)
plt.show()
```

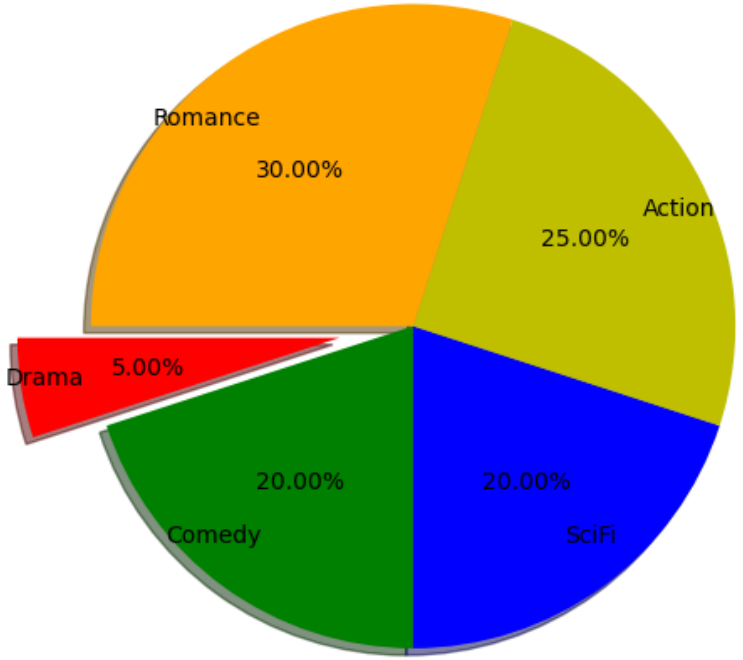


# PIE CHART/ PLOT using MATPLOTLIB




```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8,
startangle = 180, textprops={'fontsize':15})
plt.show()
```

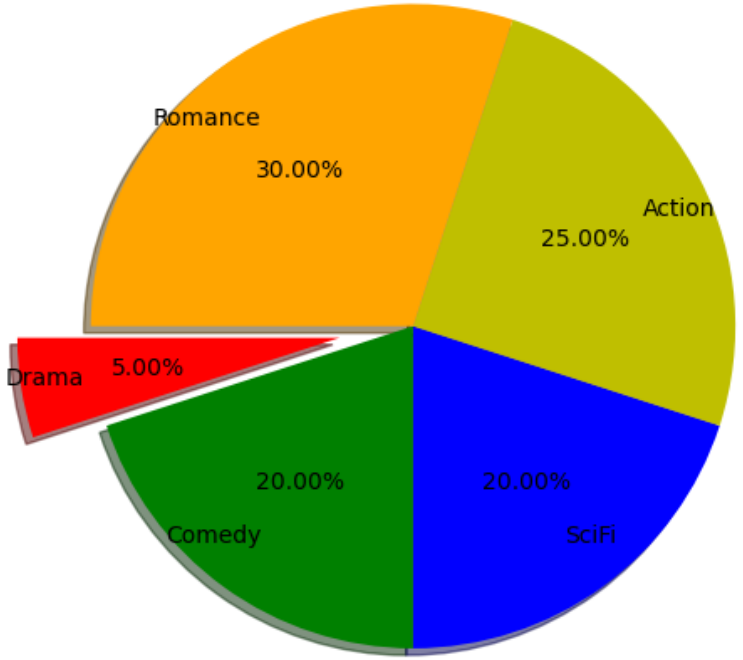
# PIE CHART/ PLOT using MATPLOTLIB



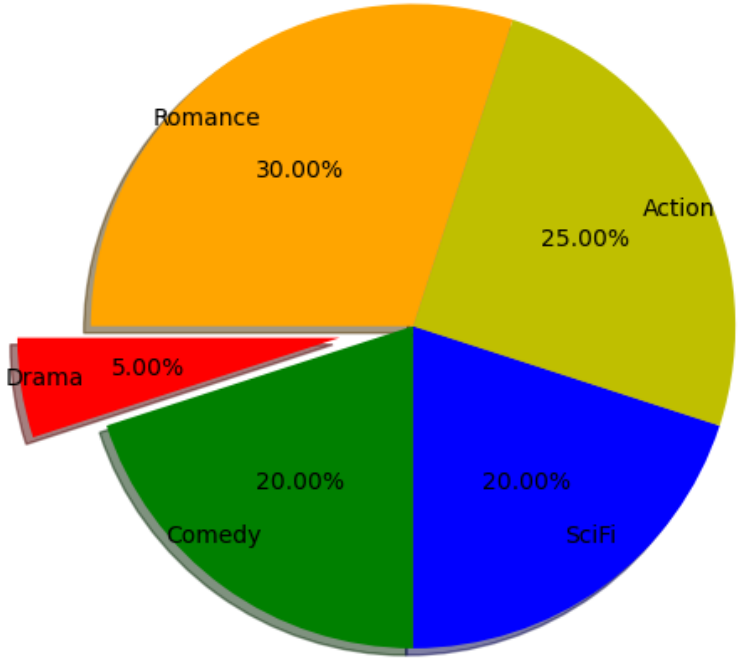

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8, startangle = 180, textprops={'fontsize':15}, counterclock=False)
plt.show()
```

# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8, startangle = 180, textprops={'fontsize':15}, counterclock=False)
plt.show()
```



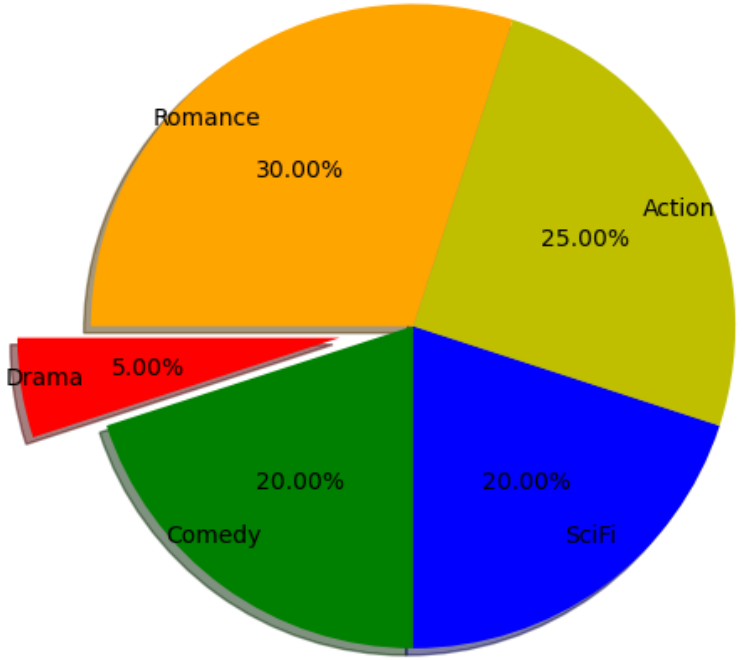
# PIE CHART/ PLOT using MATPLOTLIB



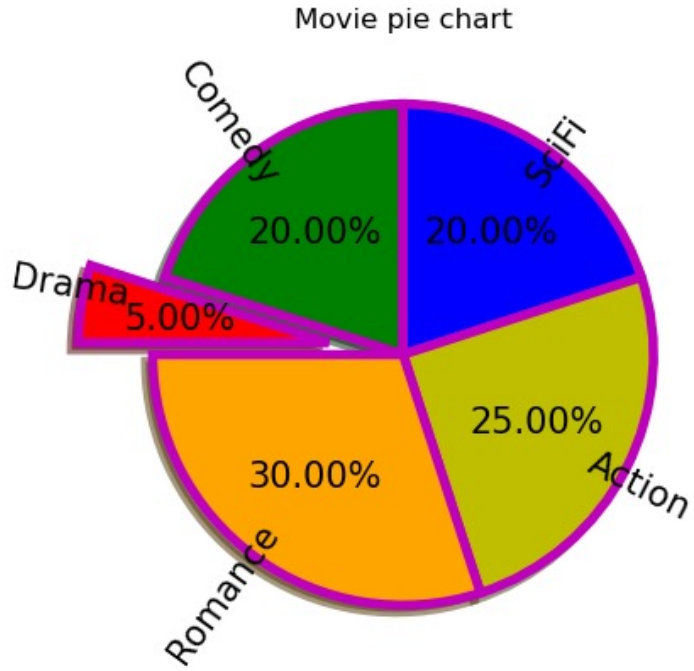

```
import matplotlib.pyplot as plt


#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8,
startangle = 180, textprops={'fontsize':15}, counterclock=False, wedgeprops={'linewidth':4, 'edgecolor':'m'})
plt.show()
```

# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt


#giving two parameters

x=[5,20,20,25,30] #the total is 100

y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']

ex=[0.3,0.0,0.0,0.0,0.0]

c=['r','g', 'b', 'y', 'orange']

plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1.3, labeldistance=0.8, startangle = 180, textprops={‘fontsize’:15}, counterclock=False, wedgeprops={'linewidth’:4, 'edgecolor':'m’}, rotatelabels=True)

plt.show()
```



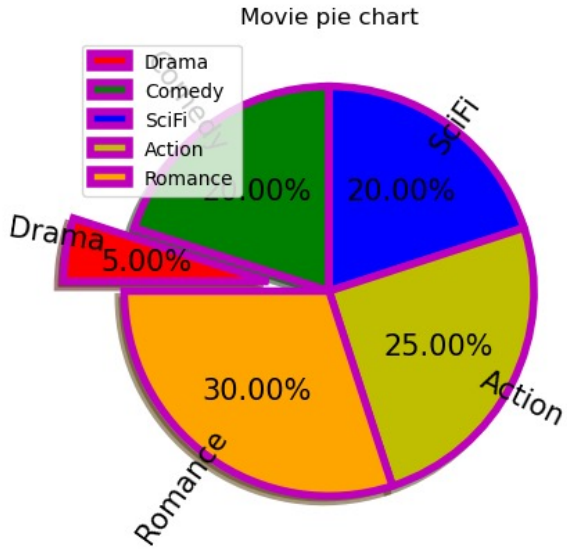
# PIE CHART/ PLOT using MATPLOTLIB




```
import matplotlib.pyplot as plt

#giving two parameters

x=[5,20,20,25,30] #the total is 100

y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']

ex=[0.3,0.0,0.0,0.0,0.0]

c=['r','g', 'b', 'y', 'orange']

plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1, labeldistance=0.8, startangle = 180, textprops={'fontsize':15}, counterclock=False, wedgeprops={'linewidth':4, 'edgecolor':'m'}, rotatelabels=True)

plt.title('Movie pie chart')

plt.show()
```

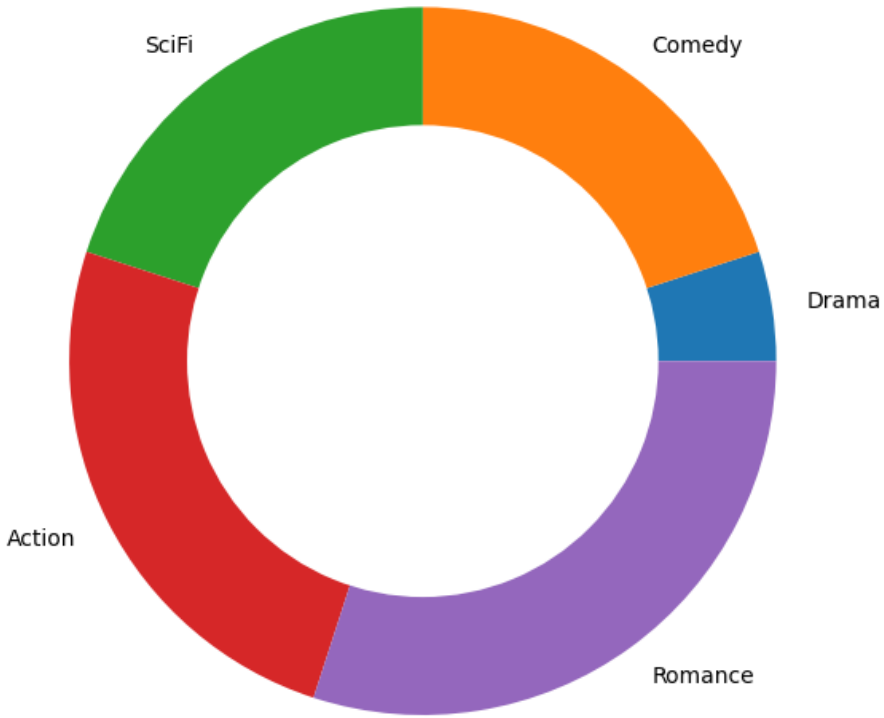
# PIE CHART/ PLOT using MATPLOTLIB

```
import matplotlib.pyplot as plt
#giving two parameters
x=[5,20,20,25,30] #the total is 100
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']
ex=[0.3,0.0,0.0,0.0,0.0]
c=['r','g', 'b', 'y', 'orange']
plt.pie(x, labels=y, explode=ex, colors=c,autopct="%0.2f%%", shadow=True, radius=1, labeldistance=0.8, startangle = 180, textprops={'fontsize':15}, counterclock=False, wedgeprops={'linewidth':4, 'edgecolor':'m'}, rotatelabels=True)
plt.title('Movie pie chart')
plt.legend(loc=2)
plt.show()
```



# Donut piechart

```
import matplotlib.pyplot as plt

#giving two parameters
x=[5,20,20,25,30] #the total is 100
x1=[40,30,20,10,5]
y=['Drama', 'Comedy', 'SciFi', 'Action', 'Romance']

c=['r','g', 'b', 'y', 'orange']

plt.pie(x, labels=y, radius=1.5)
plt.pie([1], colors='w')

plt.show()
```



# SCATTER PLOT using MATPLOTLIB

Create a heading: Scatter plot in matplotlib

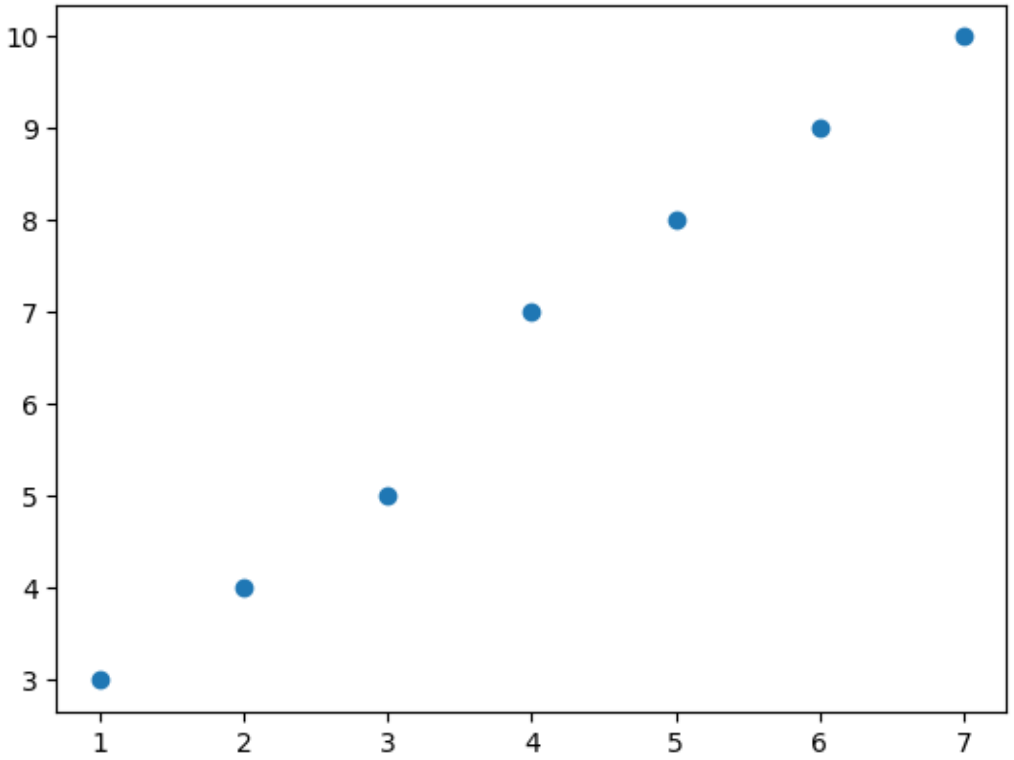
- Basic syntax/ instructions:

```
import matplotlib pyplot as plt
x = []
y = []
plt.scatter(x, y)
plt.show()
```

Reference video link: https://youtu.be/7ABCuhWO9II?si=uKVWLcd2V2C7aWfP

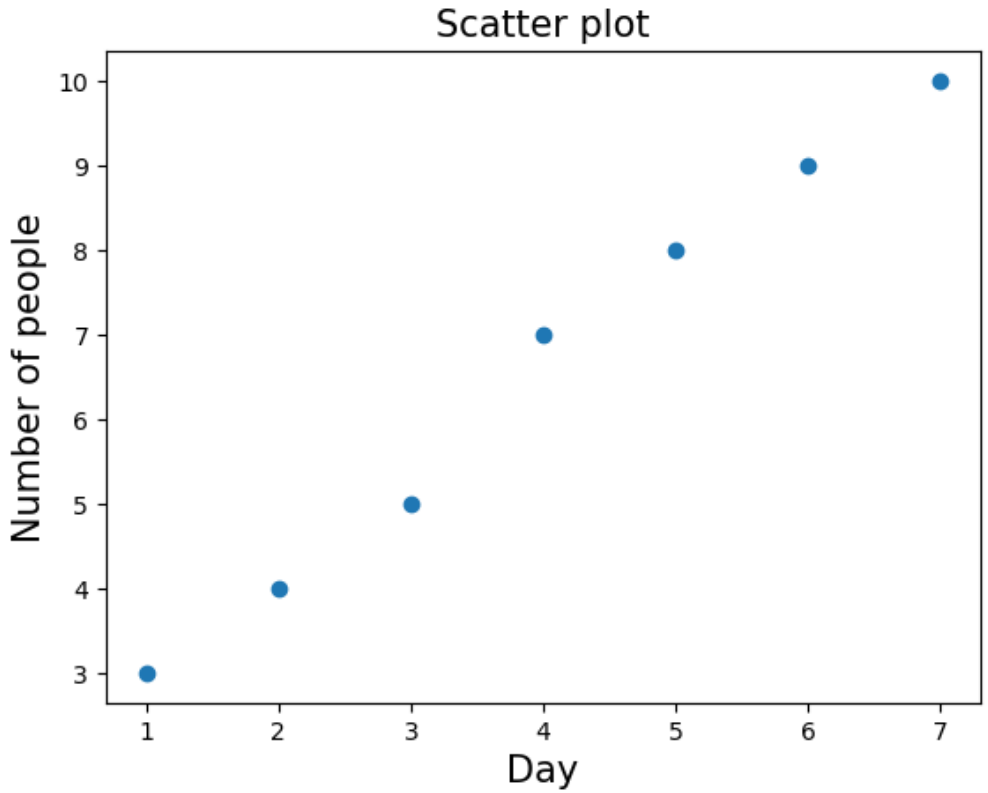
# SCATTER PLOT

```
import matplotlib.pyplot as plt
x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 8, 9, 10]
plt.scatter(x,y)
plt.show()
```



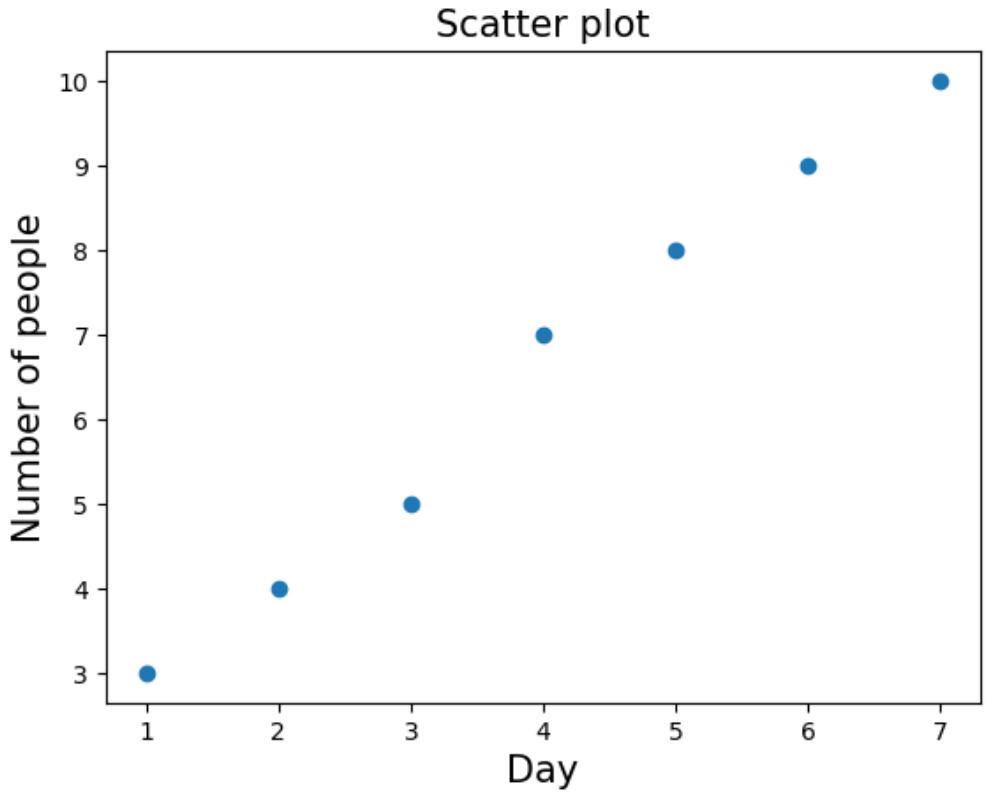
# SCATTER PLOT

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]


plt.scatter(day, number)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)


plt.show()
```



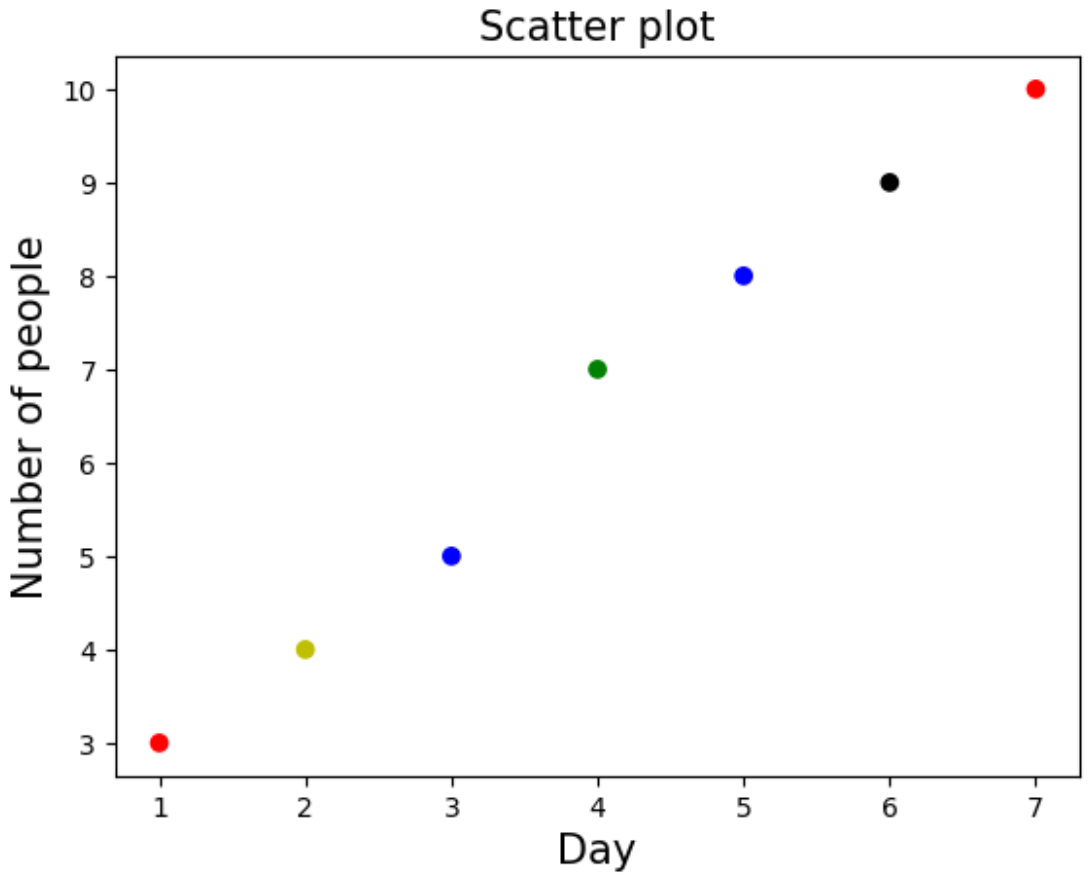
# SCATTER PLOT (color)

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]


plt.scatter(day, number, color="r")
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)


plt.show()
```



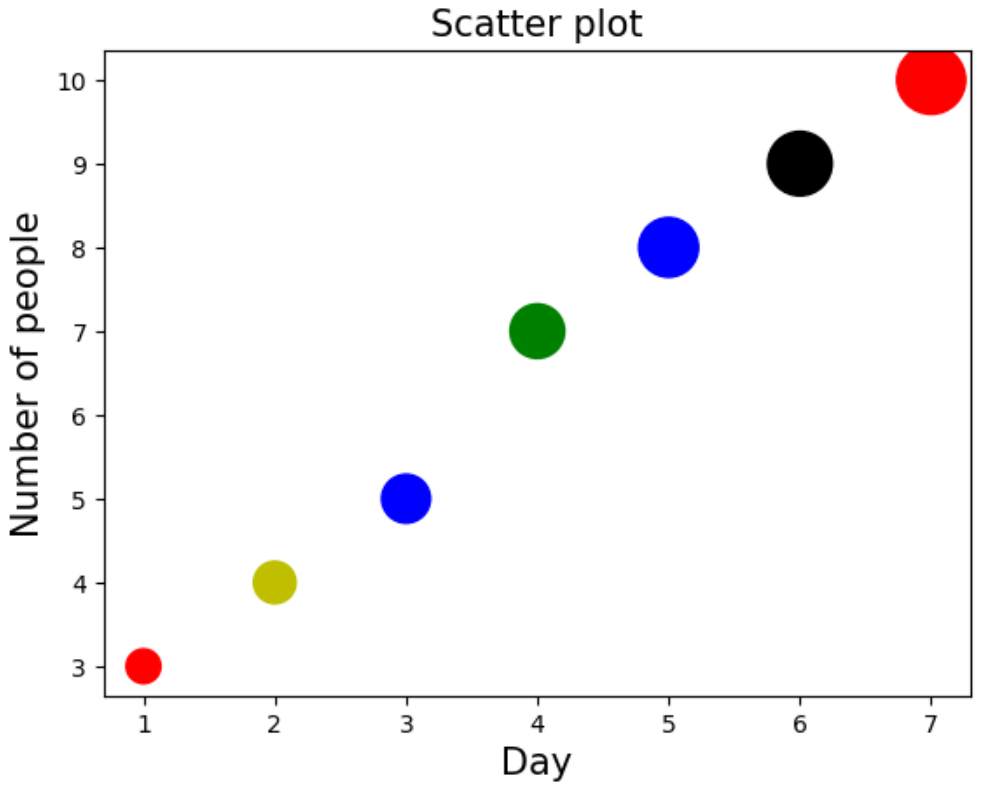
# SCATTER PLOT (multicolor)

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
colors=["r","y","b", "g", "b", "k", "r"]


plt.scatter(day, number, c=colors)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)


plt.show()
```



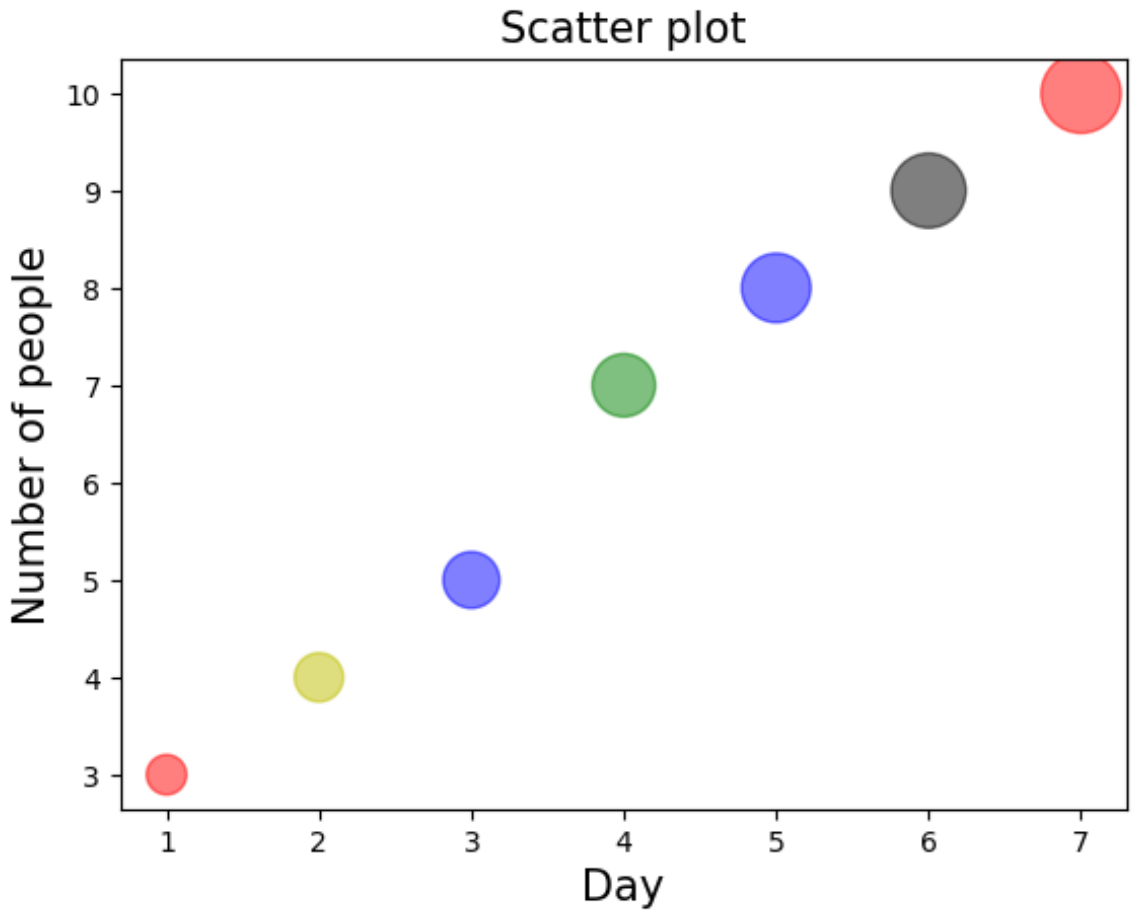
# SCATTER PLOT (dot size)

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
colors=["r","y","b", "g", "b", "k", "r"]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



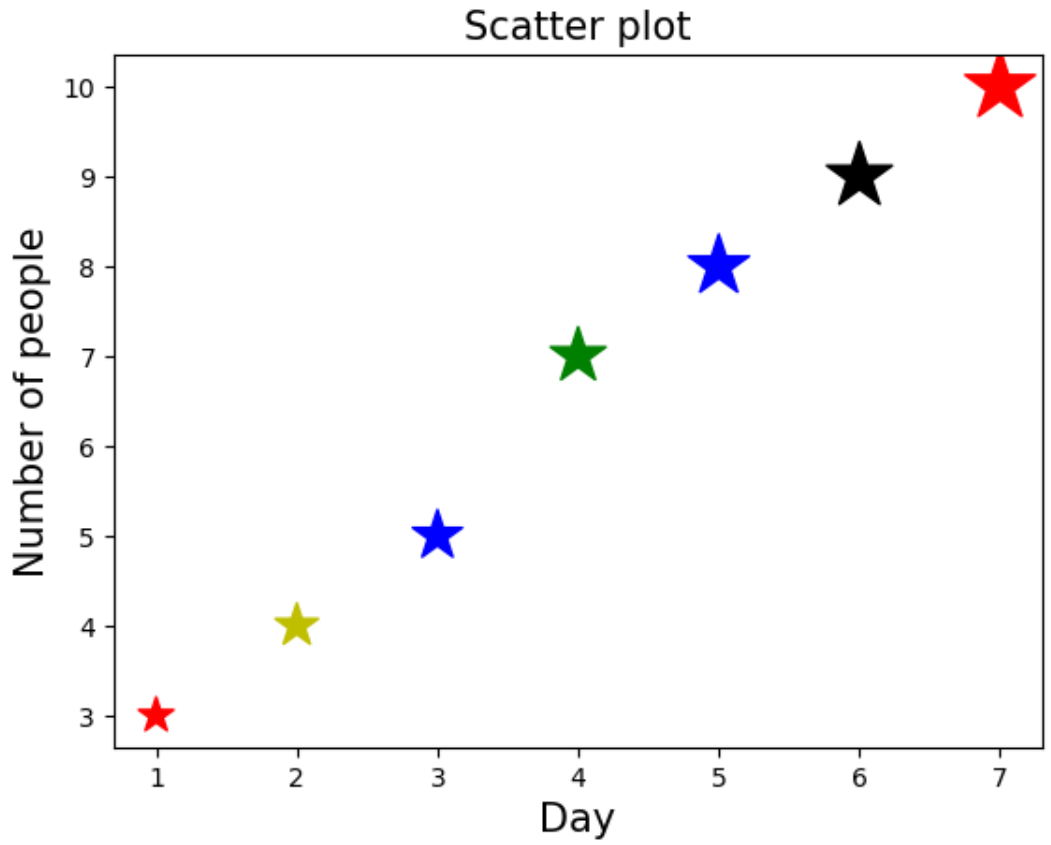
# SCATTER PLOT (transparency)

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
colors=["r","y","b", "g", "b", "k", "r"]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, alpha=0.5)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



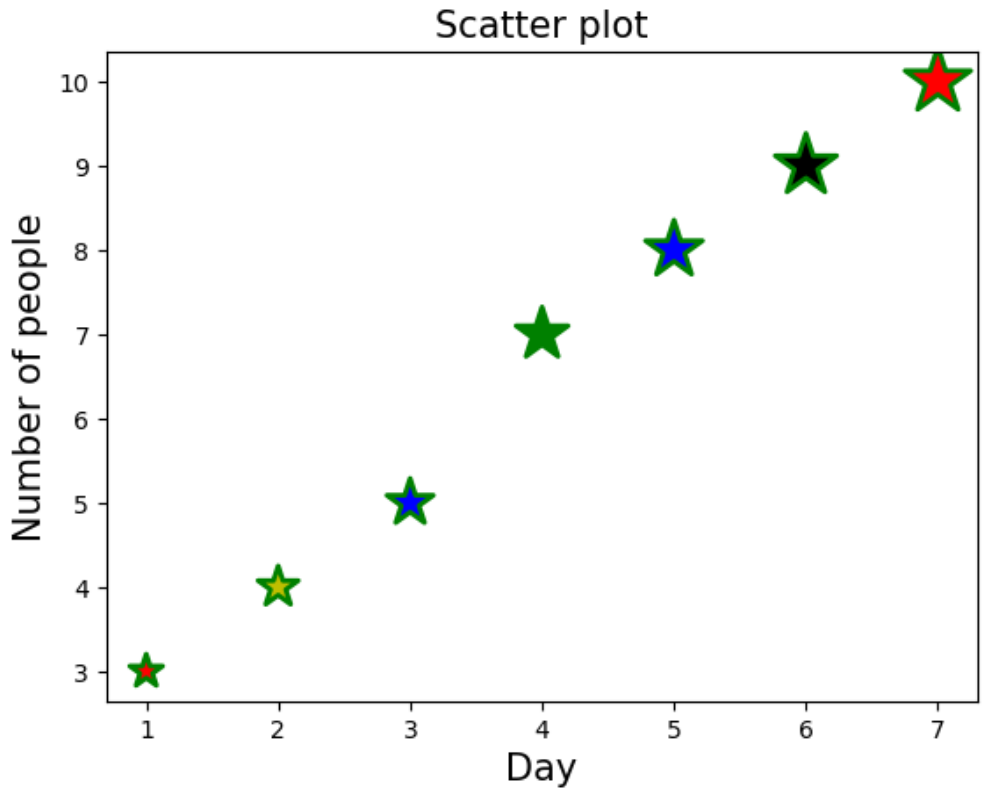
# SCATTER PLOT (dot to stars)

```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
colors=["r","y","b", "g", "b", "k", "r"]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, marker="*")
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



# SCATTER PLOT (edgecolor)

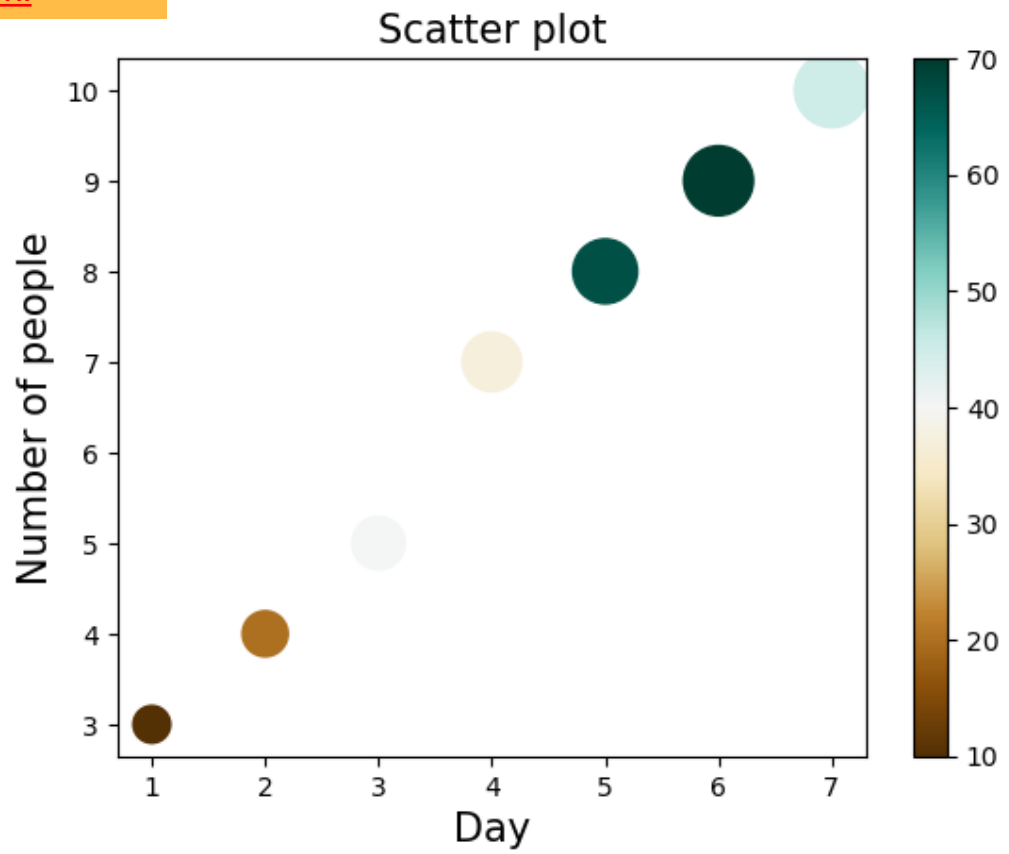
```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
colors=["r","y","b", "g", "b", "k", "r"]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, marker="*”, , edgecolor=‘g’,
linewidth=2)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



# SCATTER PLOT (Cmap property)

List: https://matplotlib.org/stable/users/explain/colors/colormaps.html

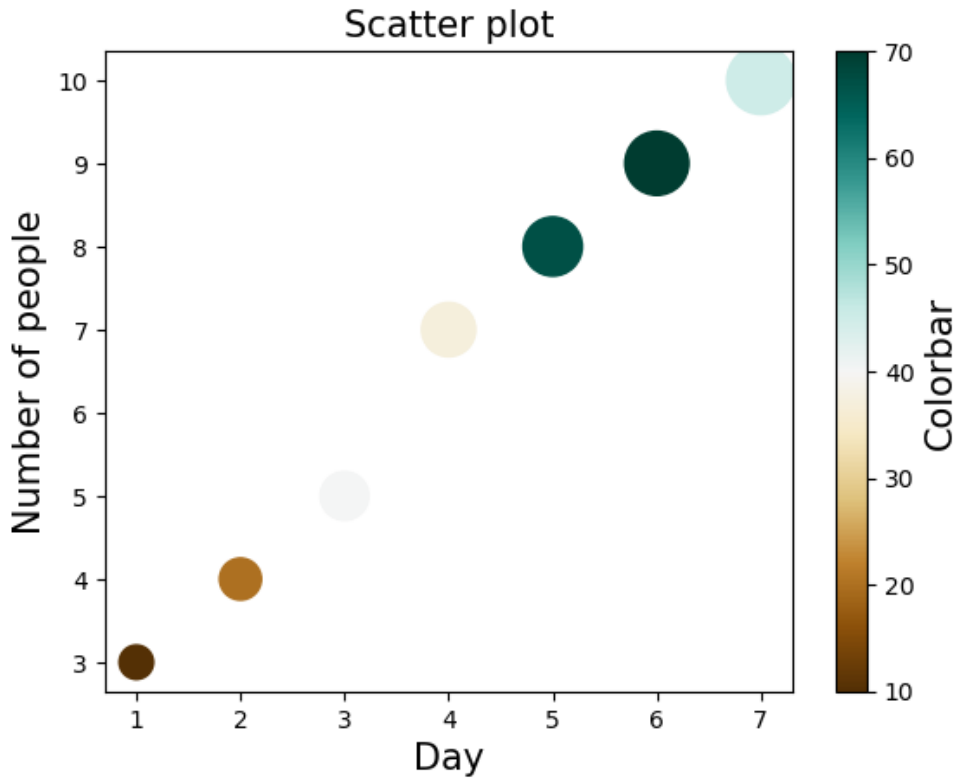
```
import matplotlib.pyplot as plt


day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
#colors=[10,20,40,37,67,70,45]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, cmap="BrBG")
plt.colorbar()
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



# SCATTER PLOT (Cmap property)

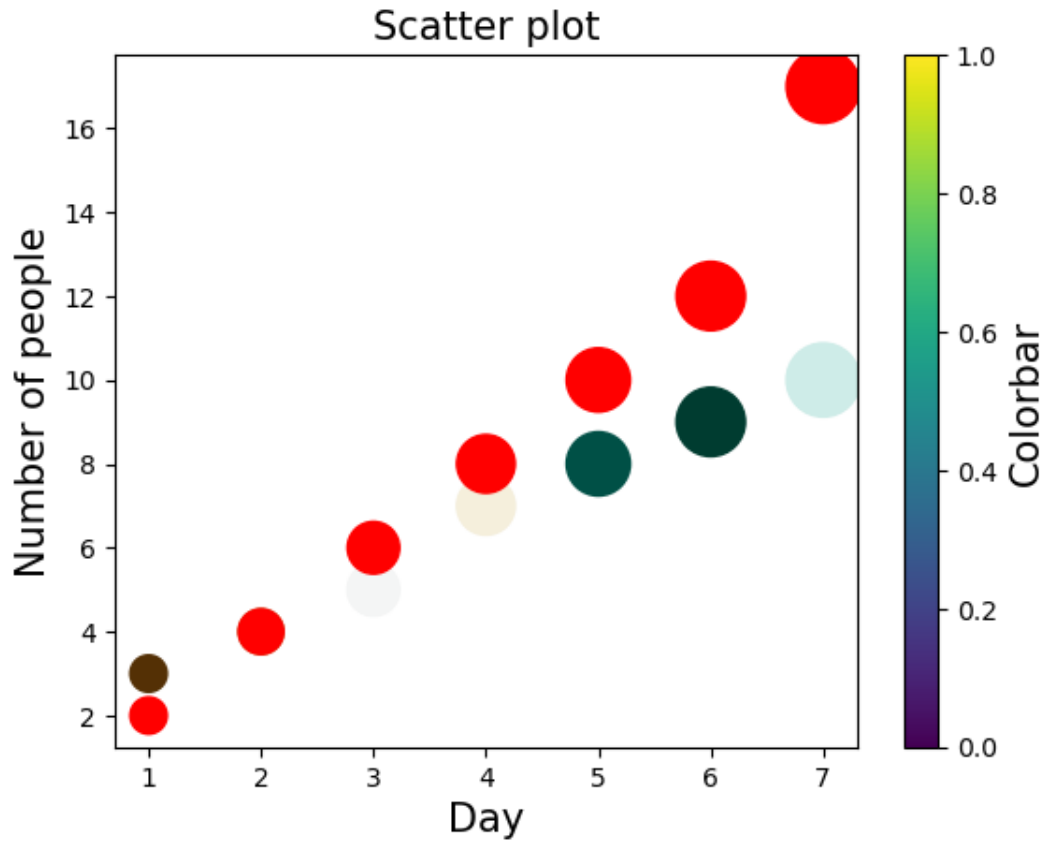
List: https://matplotlib.org/stable/users/explain/colors/colormaps.html

```
import matplotlib.pyplot as plt
day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
#colors=[10,20,40,37,67,70,45]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, cmap="BrBG")
t=plt.colorbar()
t.set_label("Colorbar", fontsize=15)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



# SCATTER PLOT (several scatterplots)

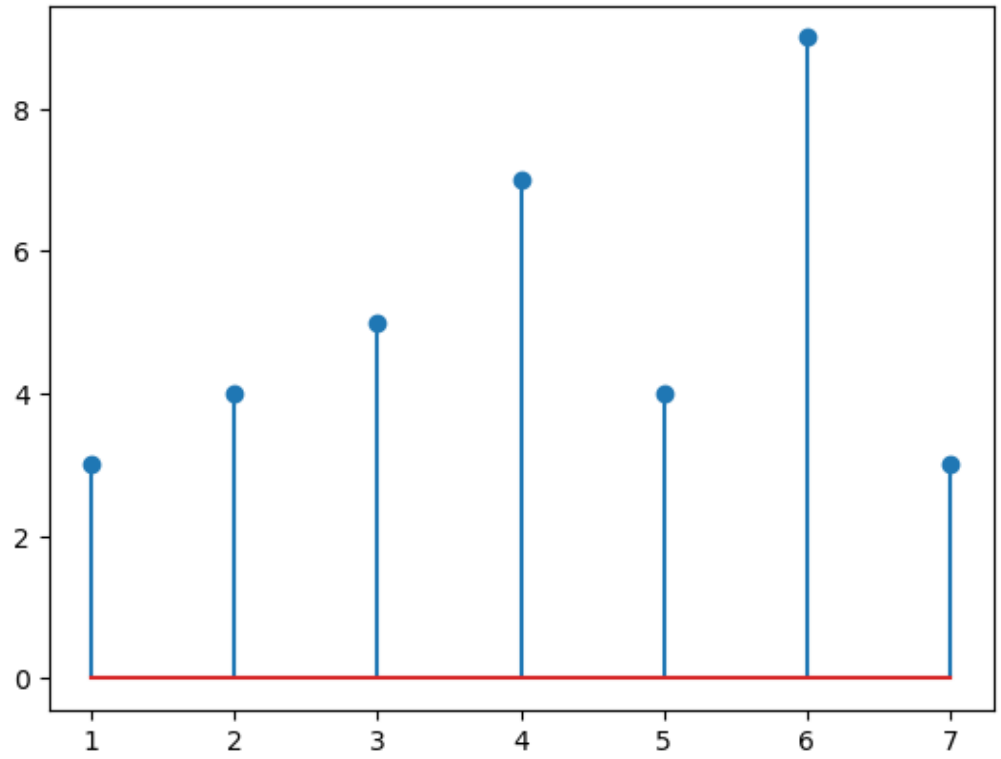
```
import matplotlib.pyplot as plt
day=[1, 2, 3, 4, 5, 6, 7]
number=[3, 4, 5, 7, 8, 9, 10]
number2=[2,4,6,8,10,12,17]
sizes=[200,300,400,500,600,700,800]
plt.scatter(day, number, c=colors, s=sizes, cmap="BrBG")
plt.scatter(day, number2, color='r', s=sizes)
t=plt.colorbar()
t.set_label("Colorbar", fontsize=15)
plt.title("Scatter plot", fontsize=15)
plt.xlabel("Day", fontsize=15)
plt.ylabel("Number of people", fontsize=15)
plt.show()
```



# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y)

plt.show()
```



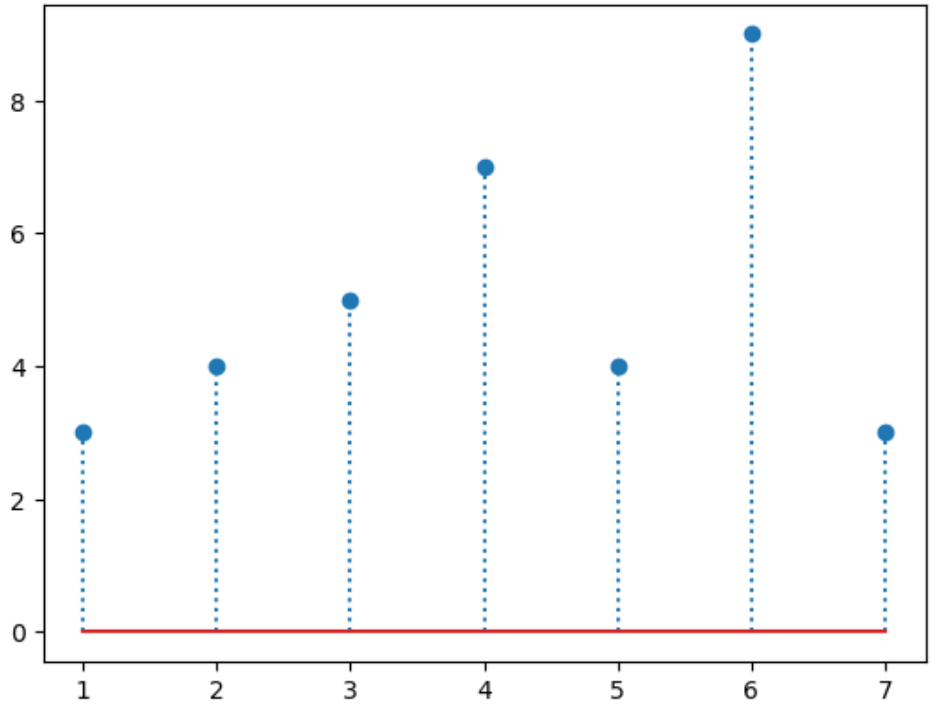

Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

| character | line style |
| --- | --- |
| ``'-'`` | Solid line |
| ``'--'`` | Dashed line |
| ``'-.'`` | Dash-dot line |
| ``':'`` | Dotted line |

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":")

plt.show()
```

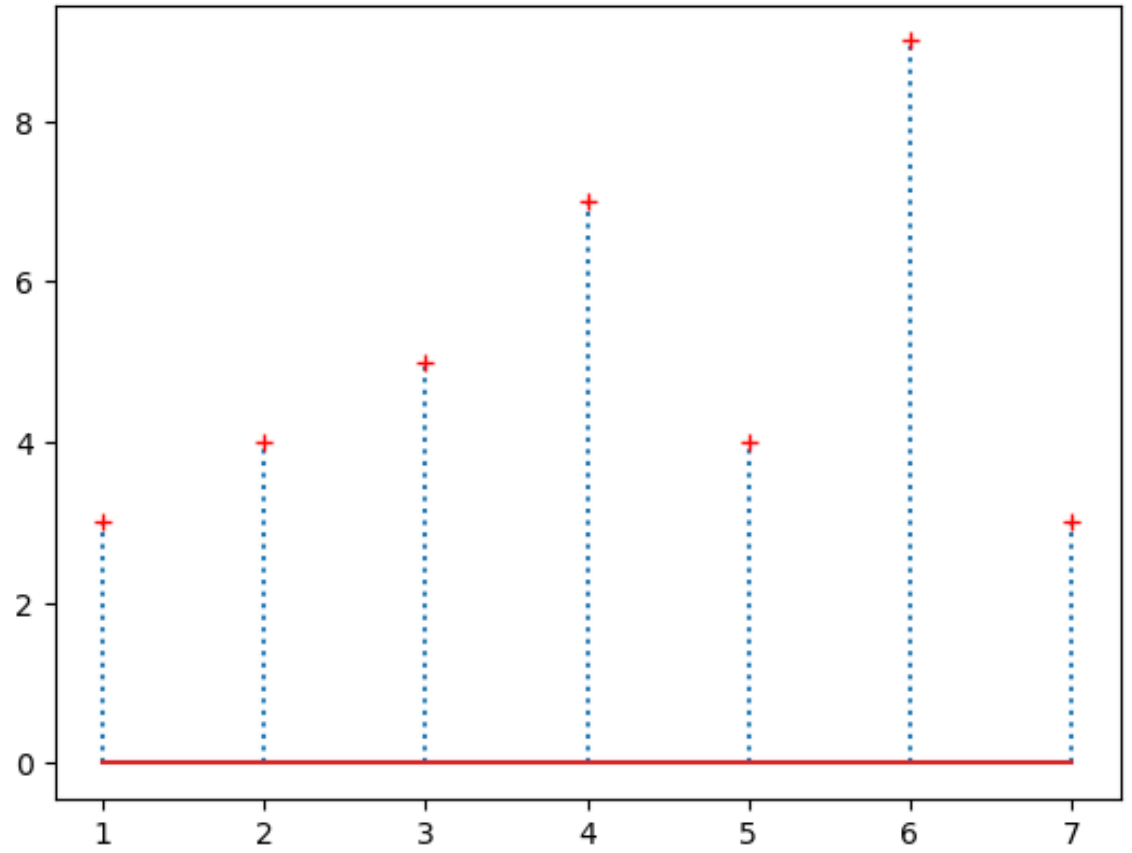



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="r+")

plt.show()
```

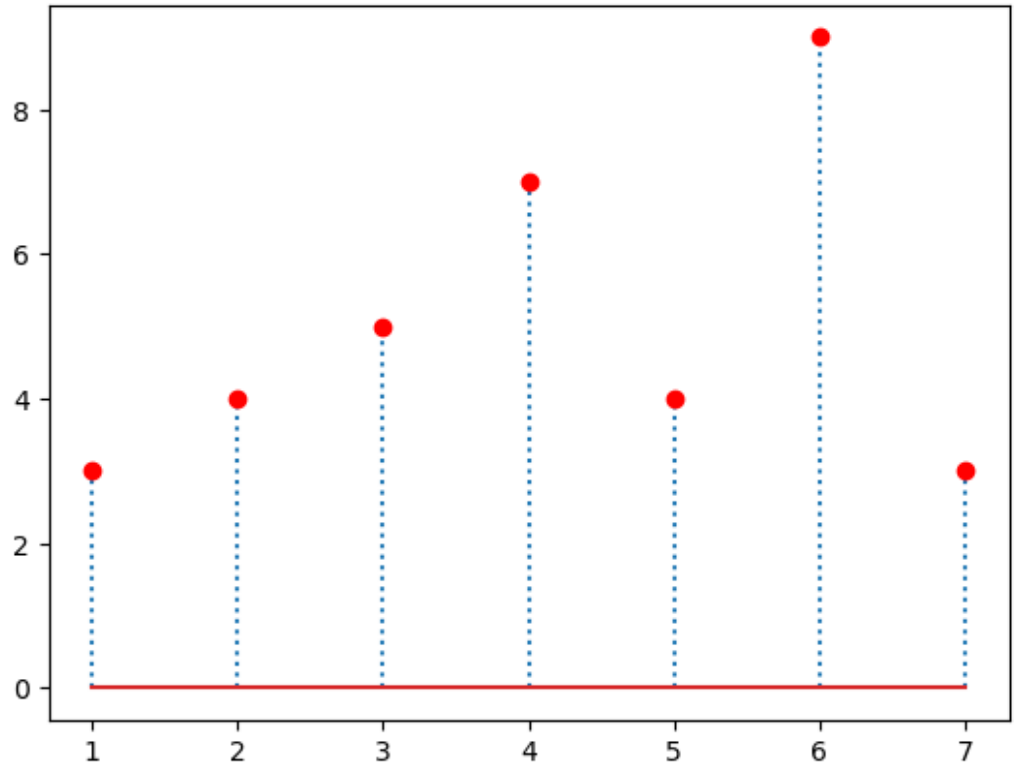



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="ro")

plt.show()
```

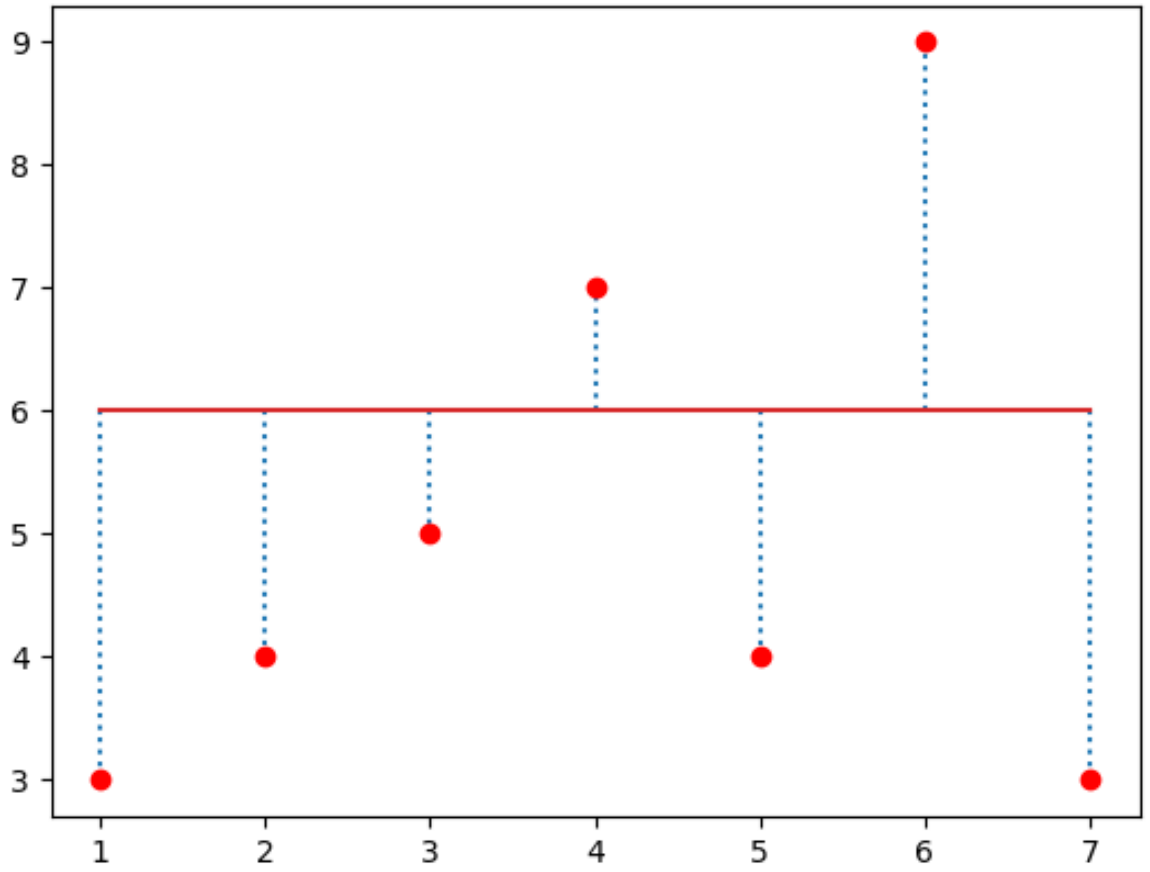



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="ro", bottom=6)

plt.show()
```

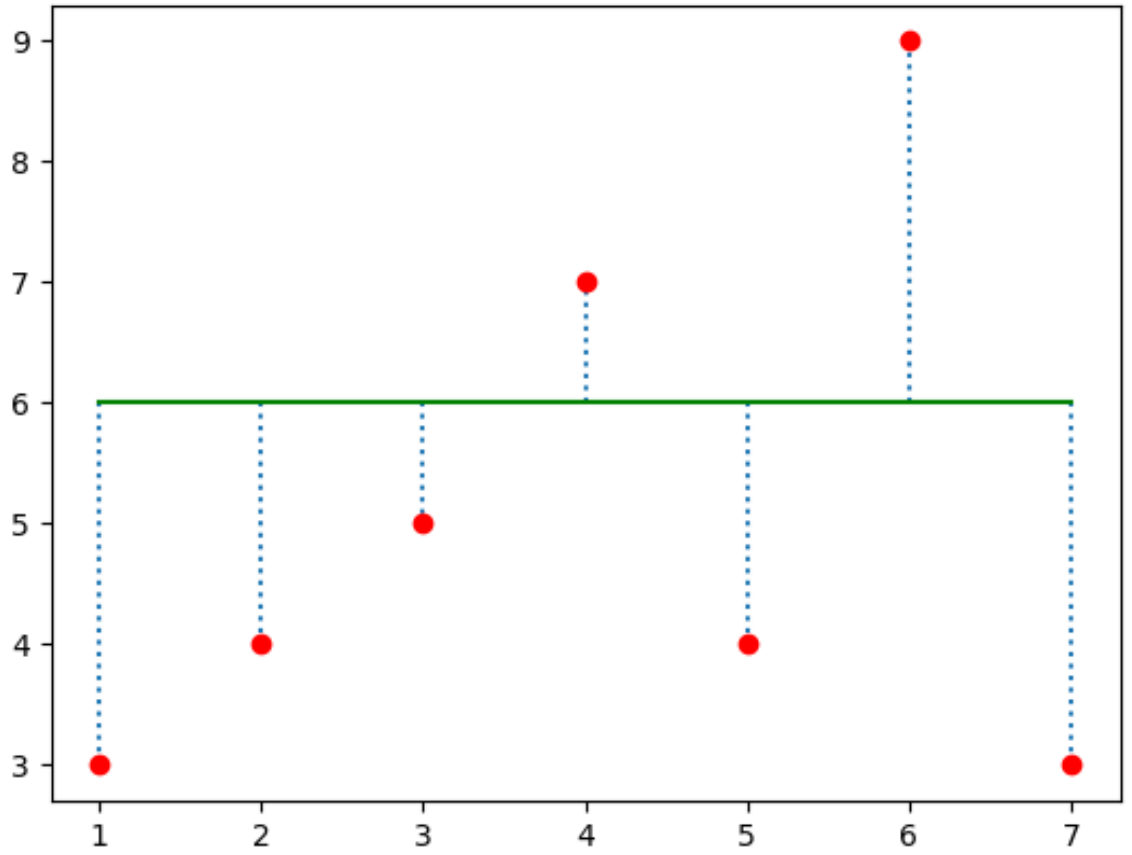



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="ro", bottom=6, basefmt='g')

plt.show()
```

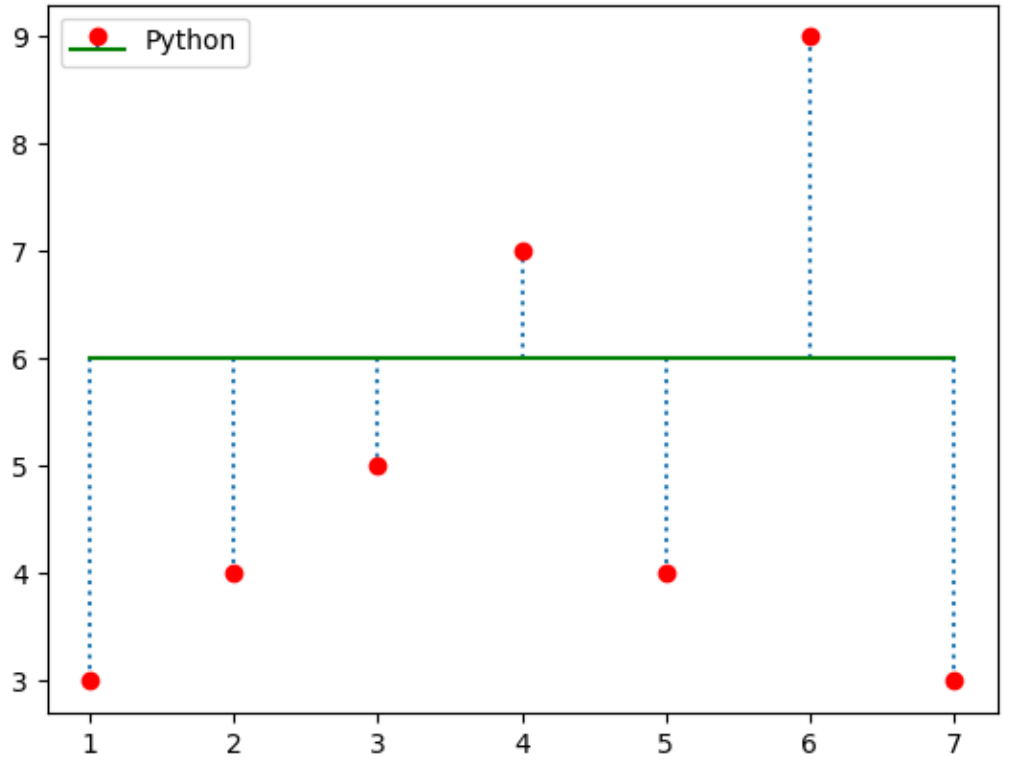



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="ro", bottom=6, basefmt='g',
label="Python")
plt.legend()

plt.show()
```

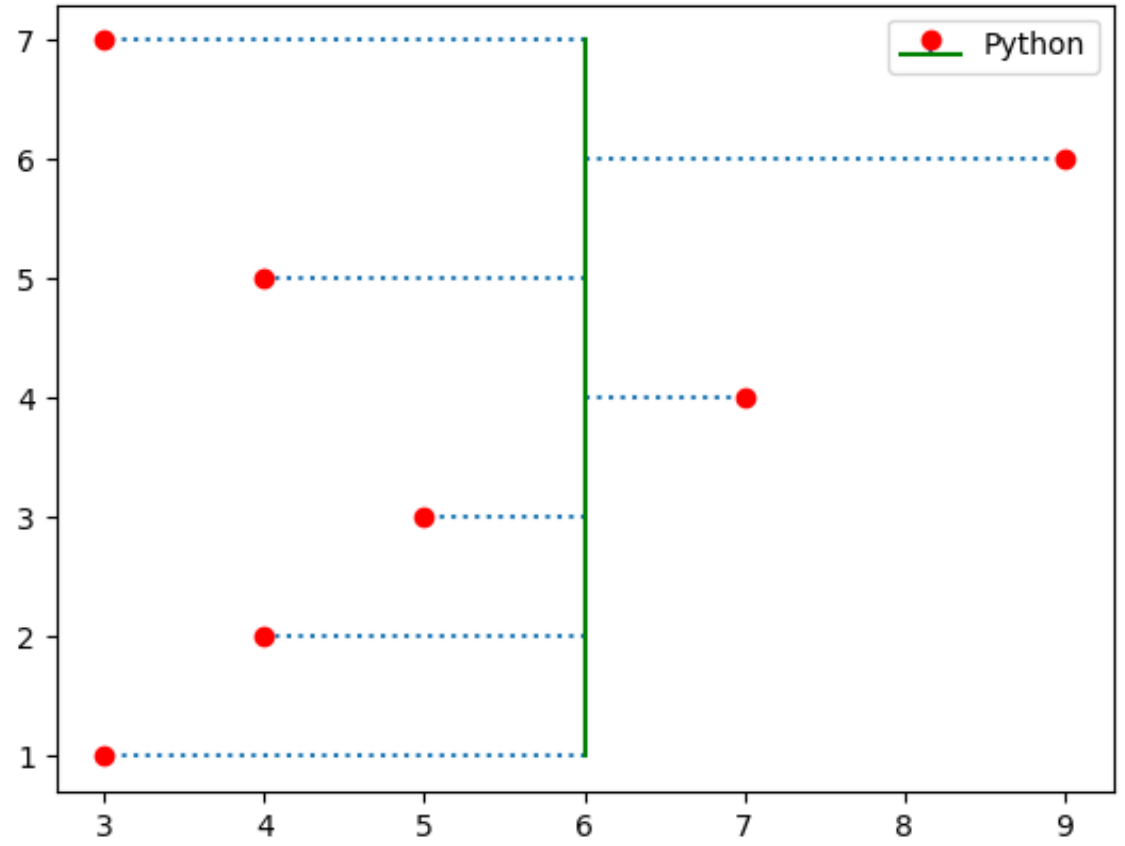



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# STEM PLOT

Create a heading: Stem plot

```
import matplotlib.pyplot as plt

x=[1, 2, 3, 4, 5, 6, 7]
y=[3, 4, 5, 7, 4, 9, 3]

plt.stem(x,y, linefmt=":", markerfmt="ro", bottom=6, basefmt='g',
label="Python", orientation='horizontal')
plt.legend()

plt.show()
```

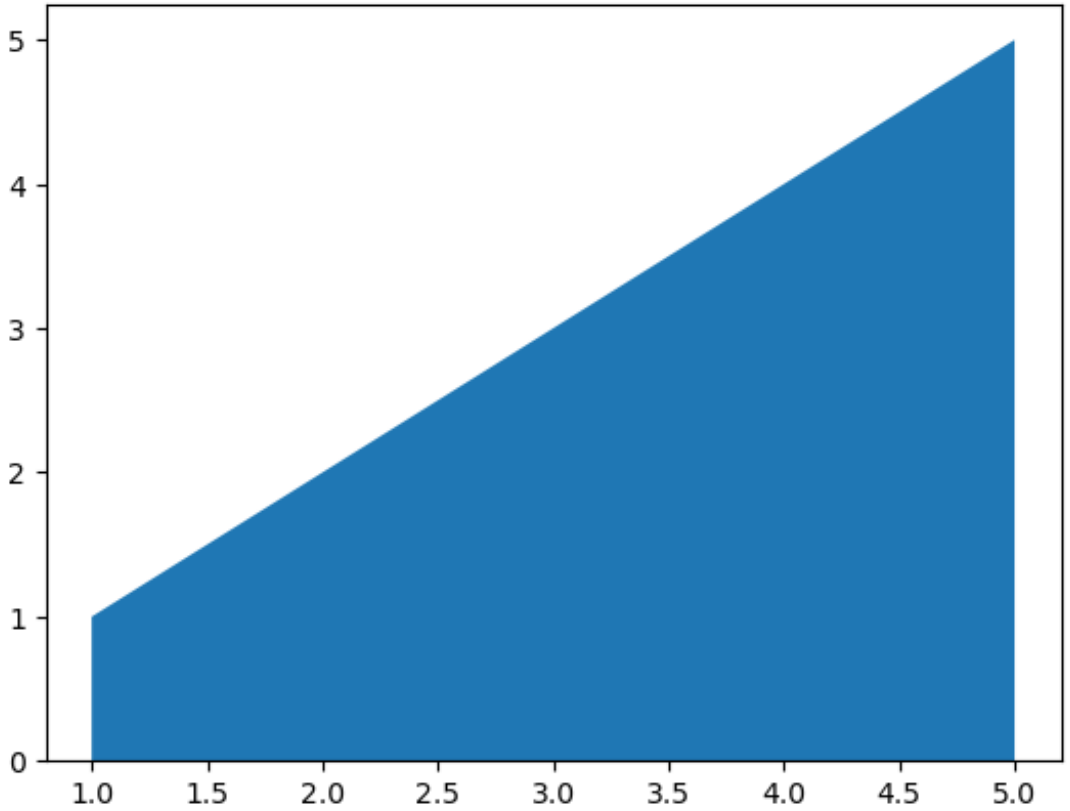



Reference video link: https://youtu.be/8GmcizWbHb0?si=Hx5CDBy_bPiMYIex

# AREA VS STACK PLOT

Create a heading: Area Vs Stack plot

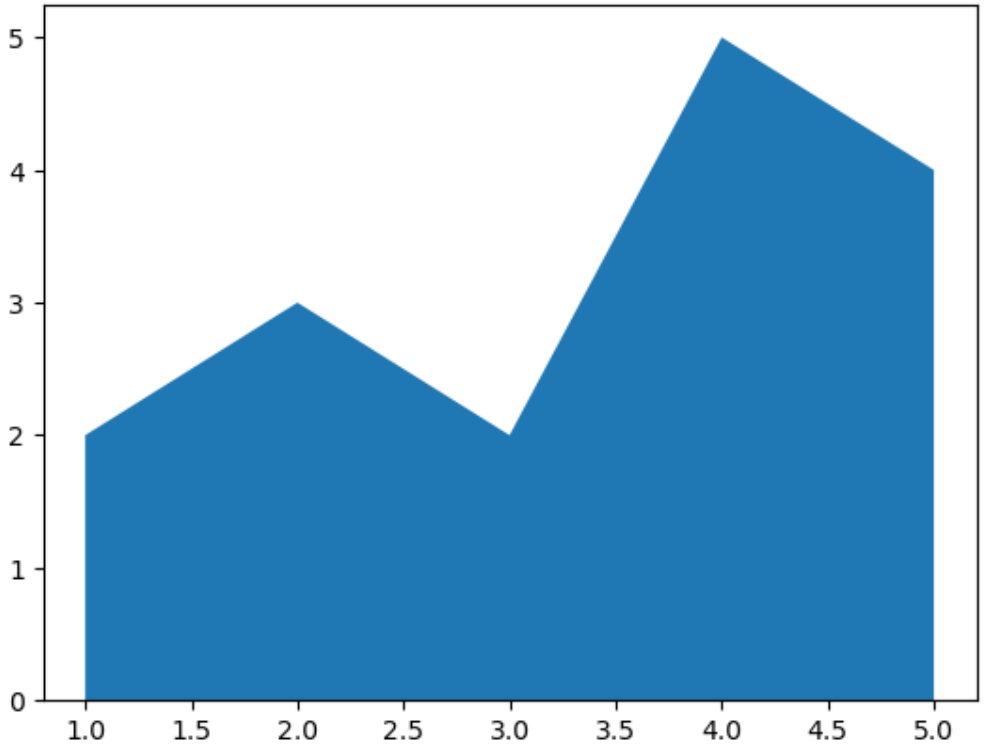
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5]
area=[1,2,3,4,5]
plt.stackplot(x,area)
plt.show()
```




Reference video link: https://youtu.be/bwJUdNHRf9g?si=ADPMrg2eLTAbyq6v

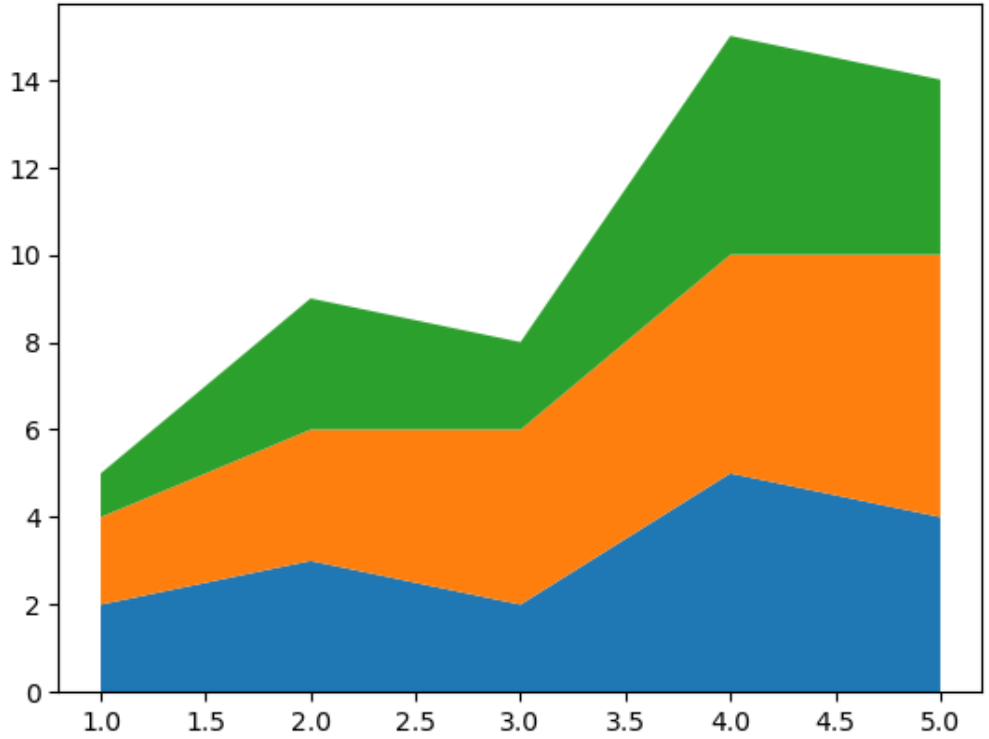
# AREA VS STACK PLOT

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area=[2,3,2,5,4]

plt.stackplot(x,area)

plt.show()
```



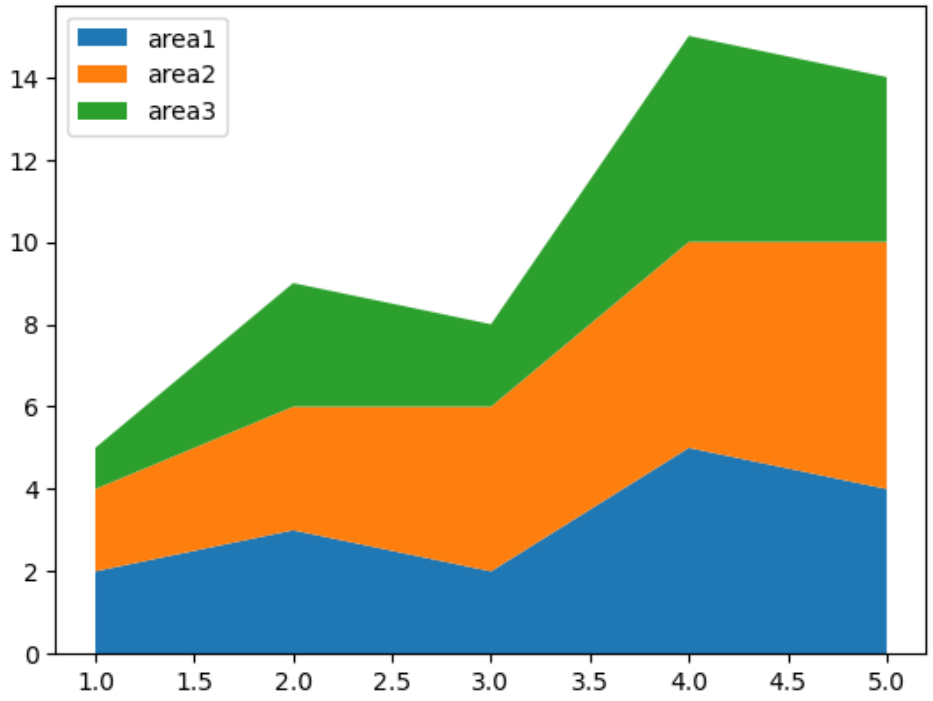
# AREA VS STACK PLOT

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]

plt.stackplot(x, area1, area2, area3)

plt.show()
```

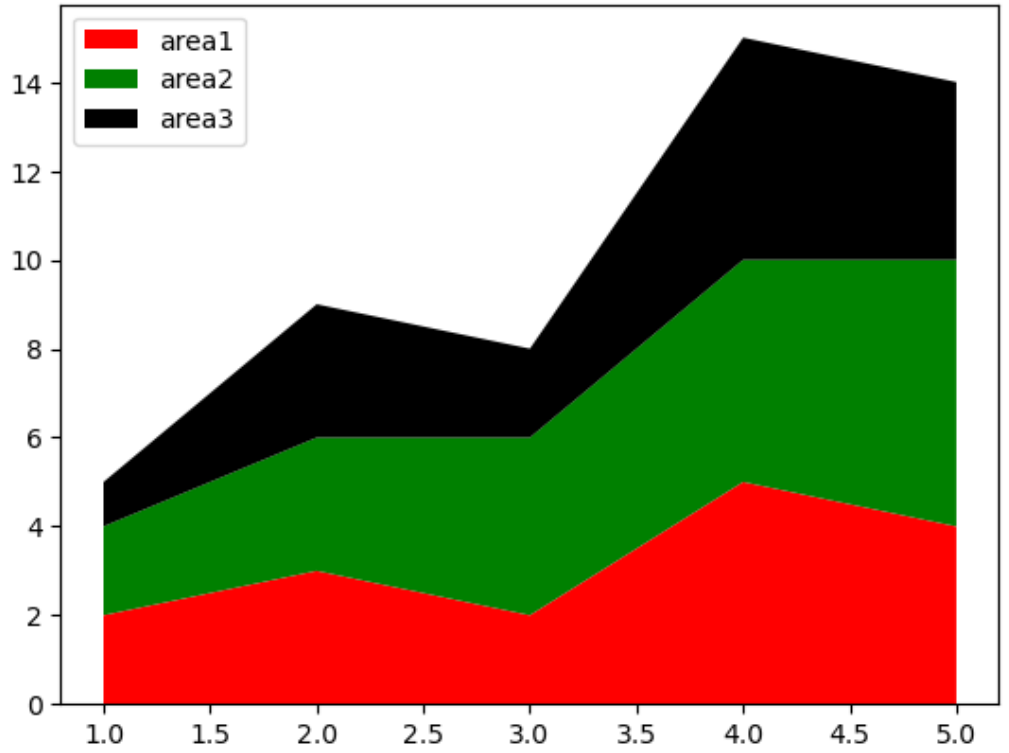


# AREA VS STACK PLOT

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l)
plt.legend(loc=2) #labels need to have legend()

plt.show()
```

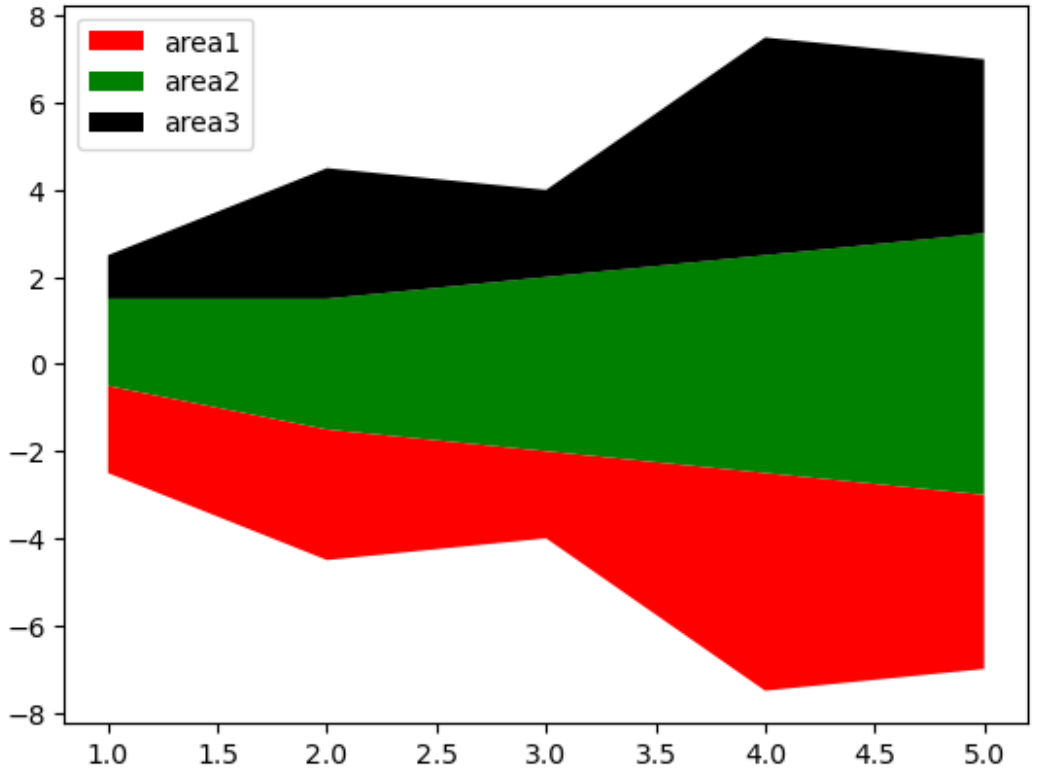


# AREA VS STACK PLOT



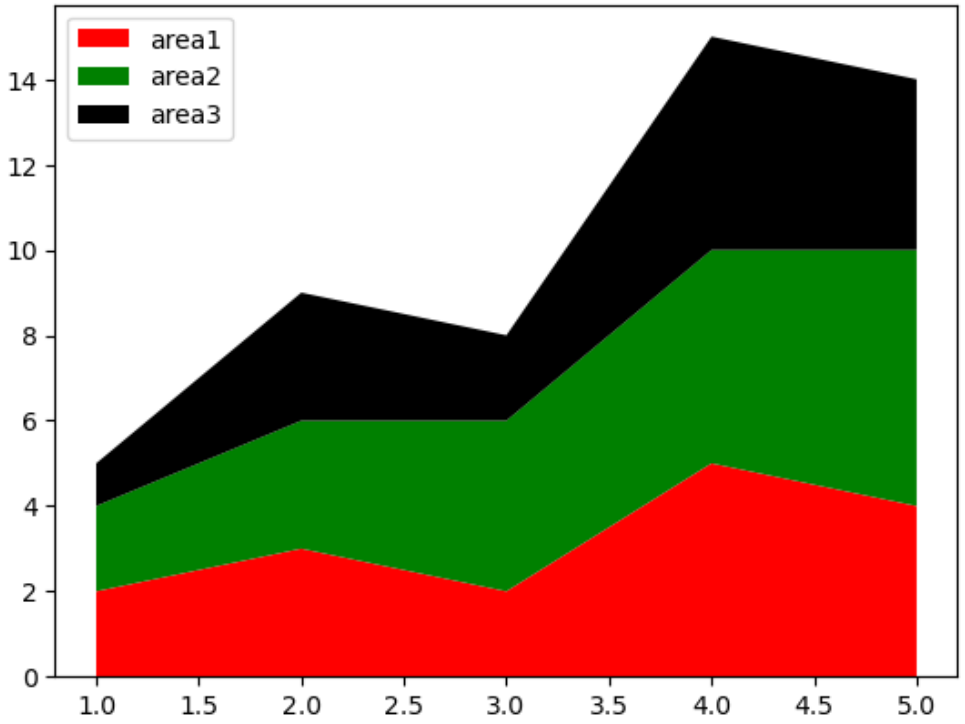

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l, colors=["r", "g", "k"])
plt.legend(loc=2)

plt.show()
```

# AREA VS STACK PLOT



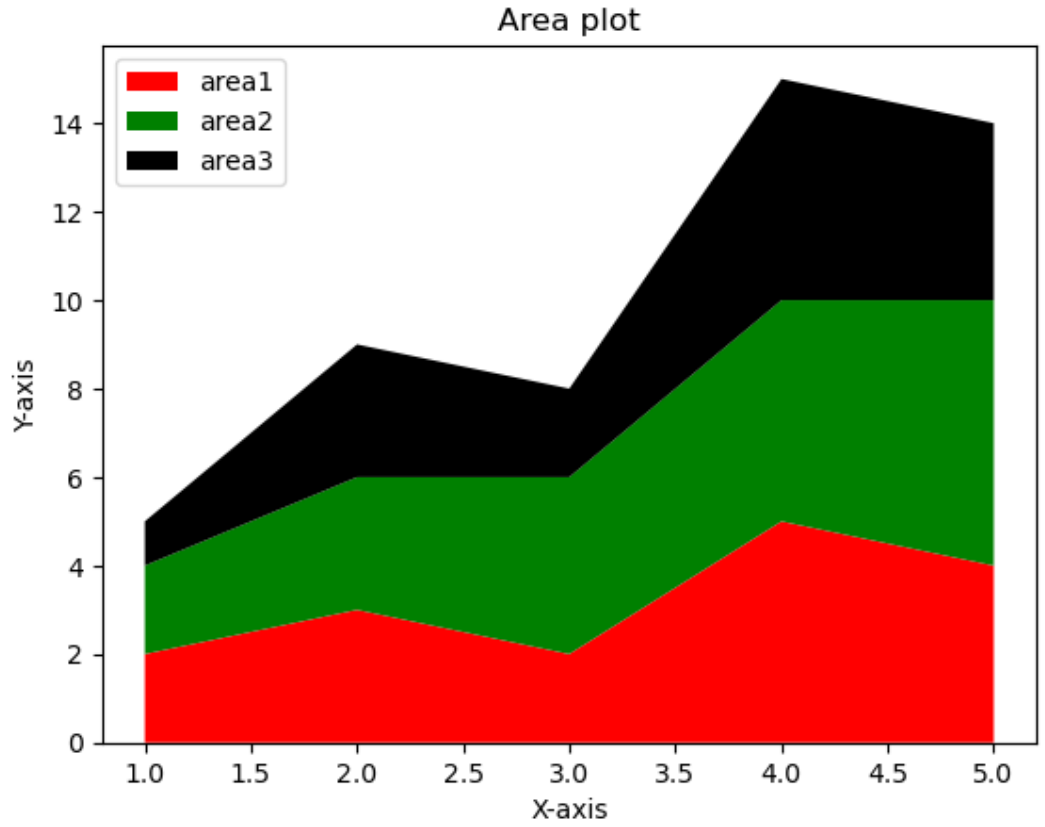

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l, colors=["r", "g", "k"], baseline='sym')
plt.legend(loc=2)

plt.show()
```

# AREA VS STACK PLOT



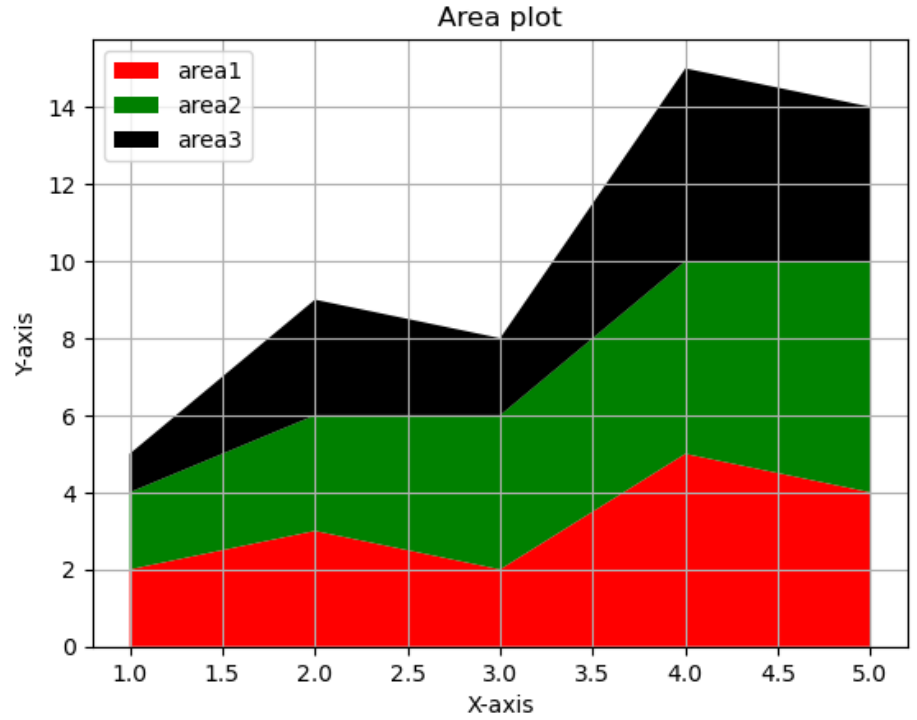

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l, colors=["r", "g", "k"], baseline='zero')
plt.legend(loc=2)

plt.show()
```

# AREA VS STACK PLOT




```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l, colors=["r", "g", "k"], baseline='zero')

plt.title("Area plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend(loc=2)

plt.show()
```

# AREA VS STACK PLOT

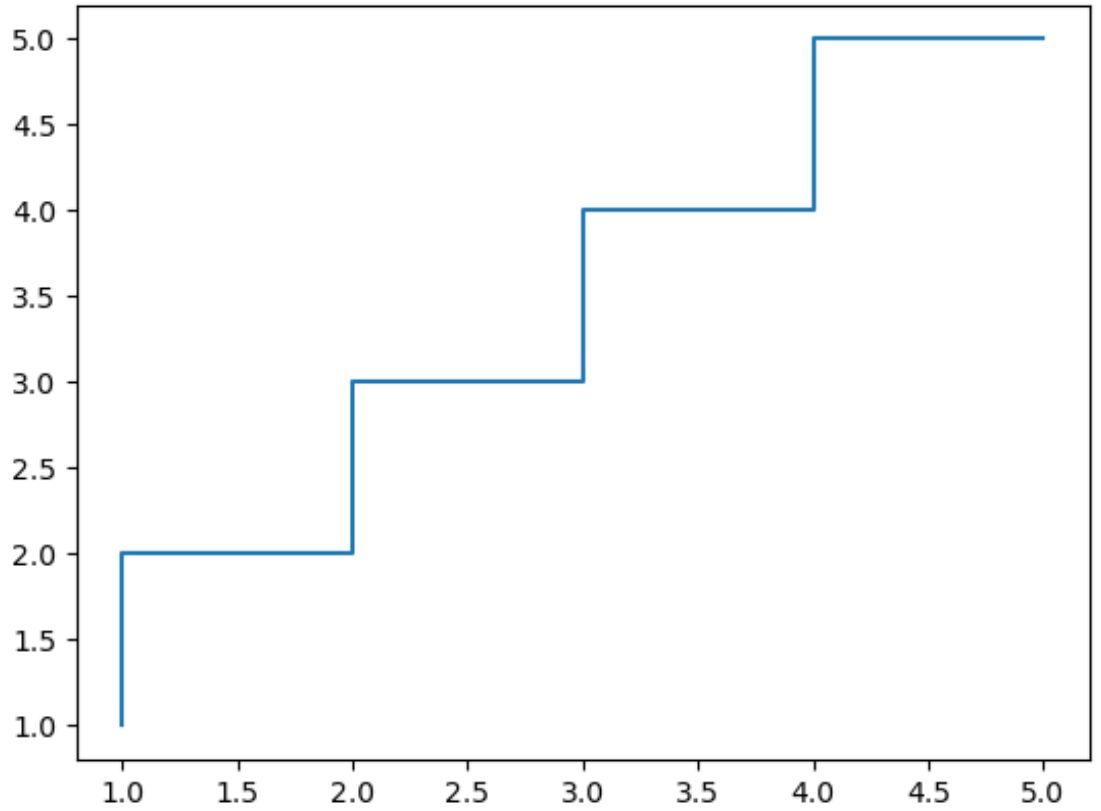



```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
area1=[2,3,2,5,4]
area2=[2,3,4,5,6]
area3=[1,3,2,5,4]
l=["area1", "area2", "area3"]

plt.stackplot(x, area1, area2, area3, labels=l, colors=["r", "g", "k"], baseline='zero')

plt.title("Area plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend(loc=2)
plt.grid()

plt.show()
```

# STEP PLOT

Create a heading: Step plot

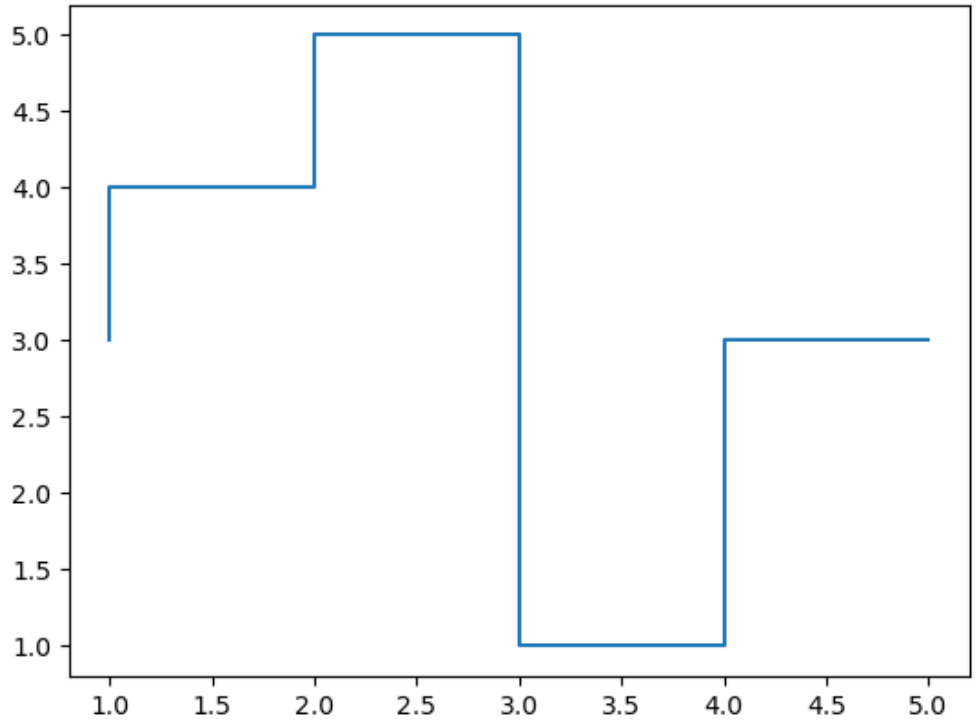
```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [1,2,3,4,5]

plt.step(x,y)

plt.show()
```




Reference video link: https://youtu.be/kOz0hVRA4lw?si=nzm1P8-myA04eL1v

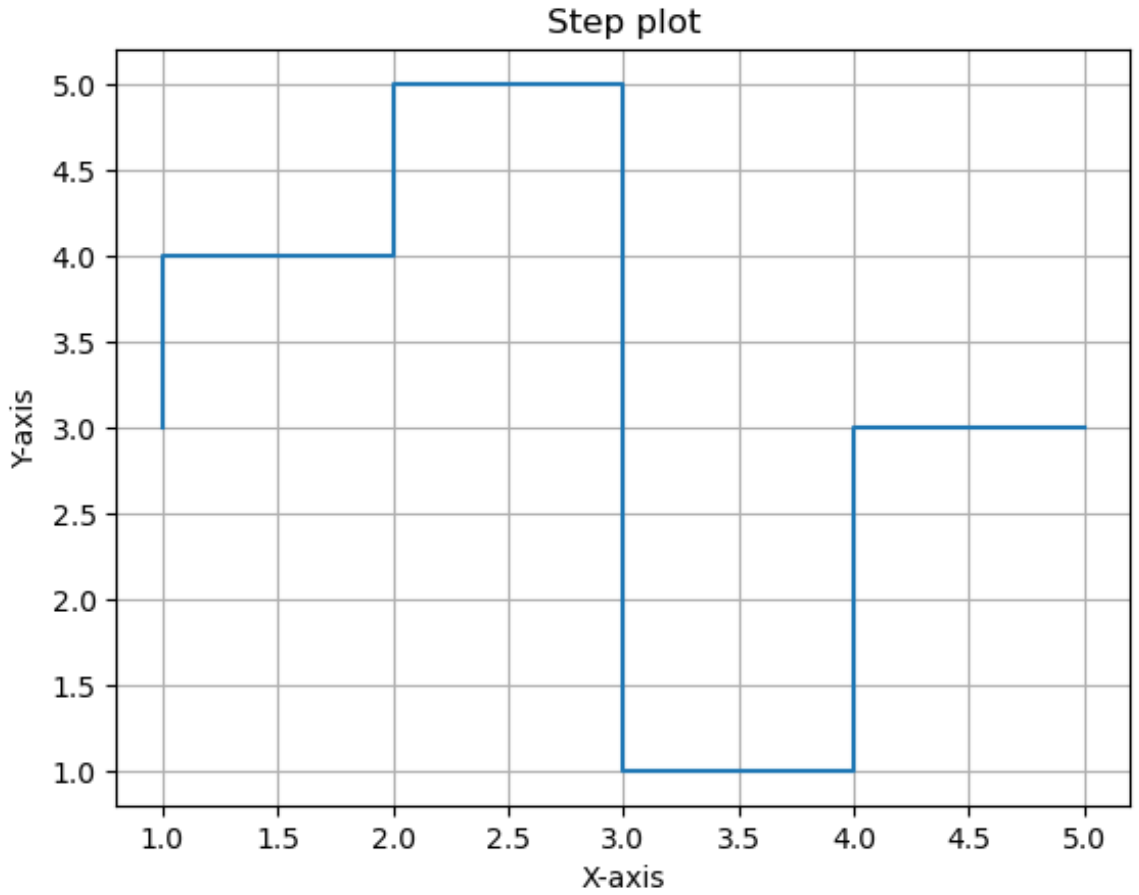
# STEP PLOT

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y)

plt.show()
```



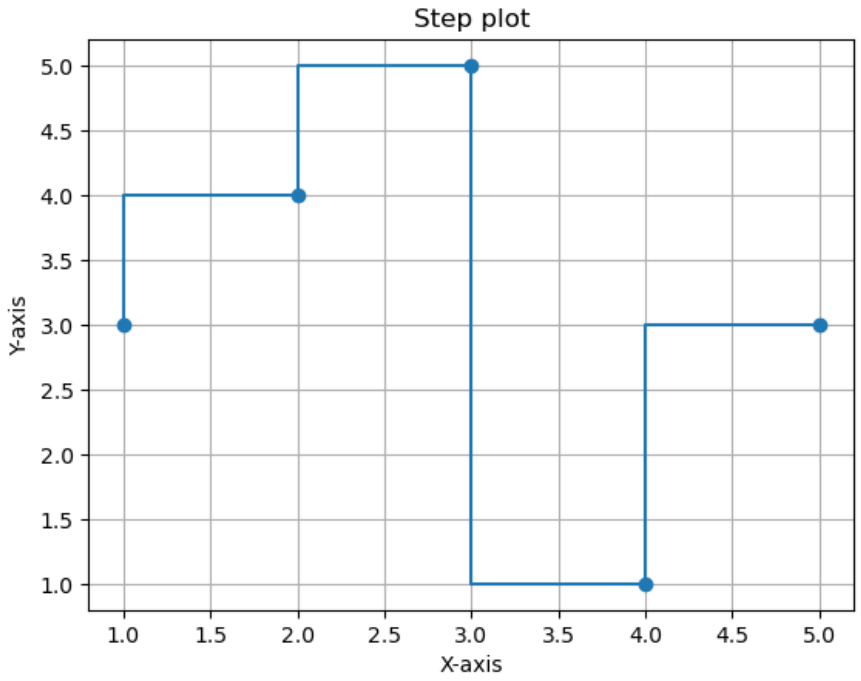
# STEP PLOT

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y)
plt.title("Step plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.grid()

plt.show()
```



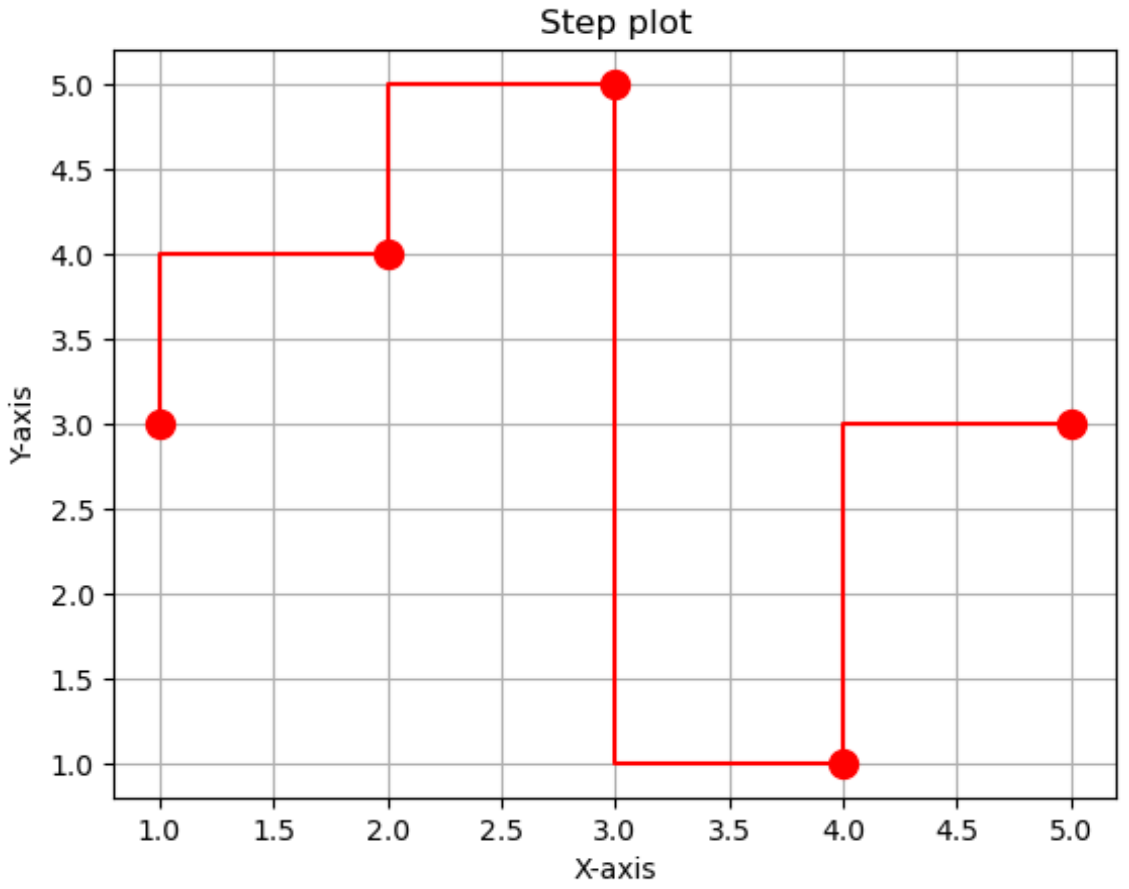
# STEP PLOT

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y, marker='o')

plt.title("Step plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.grid()

plt.show()
```



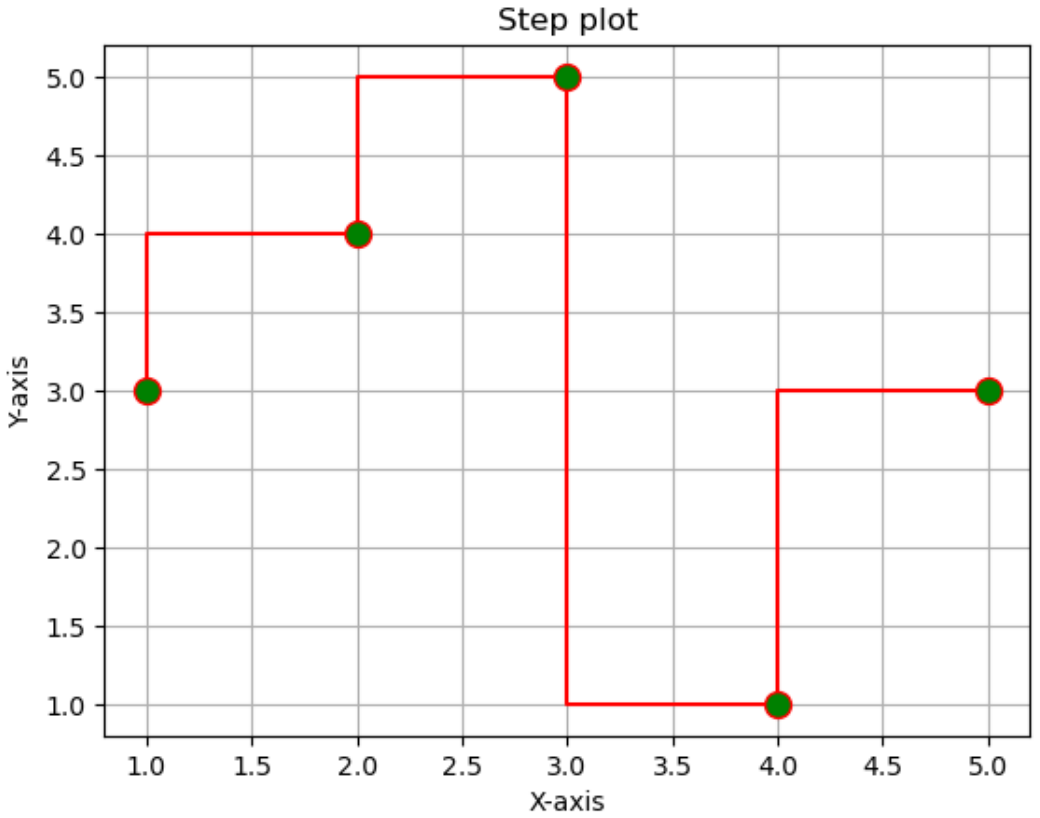
# STEP PLOT

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y, color='r', marker='o', ms=10)

plt.title("Step plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.grid()

plt.show()
```



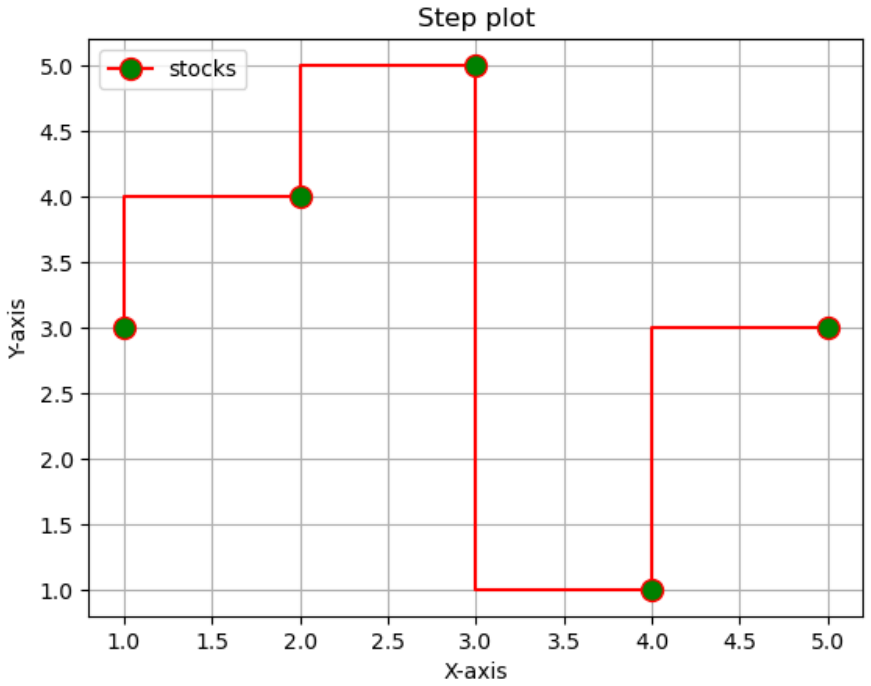
# STEP PLOT

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y, color='r', marker='o', ms=10, mfc='g')

plt.title("Step plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.grid()

plt.show()
```



# STEP PLOT

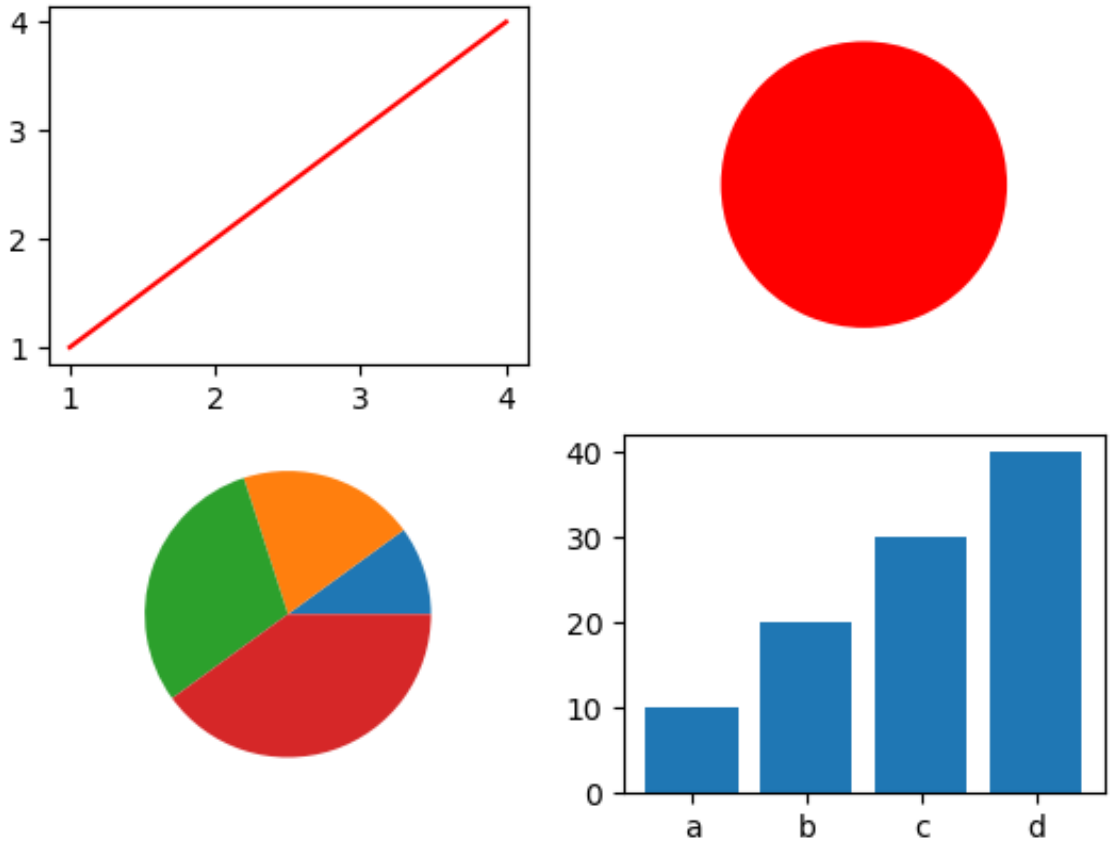
```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [3,4,5,1,3]

plt.step(x,y, color='r', marker='o', ms=10, mfc='g',
label='stocks')

plt.title("Step plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.grid()
plt.legend(loc=2)
plt.show()
```



# SUB PLOT

Create a heading: Sub plot

```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [1,2,3,4]
plt.subplot(2,2,1)
plt.plot(x,y, color='r')

plt.subplot(2,2,2)
plt.pie([1], colors='r')

x1 = [10,20,30,40]
plt.subplot(2,2,3)
plt.pie(x)

x2=["a", "b", "c", "d"]
y2=[10,20,30,40]
plt.subplot(2,2,4)
plt.bar(x2,y2)

plt.show()
```

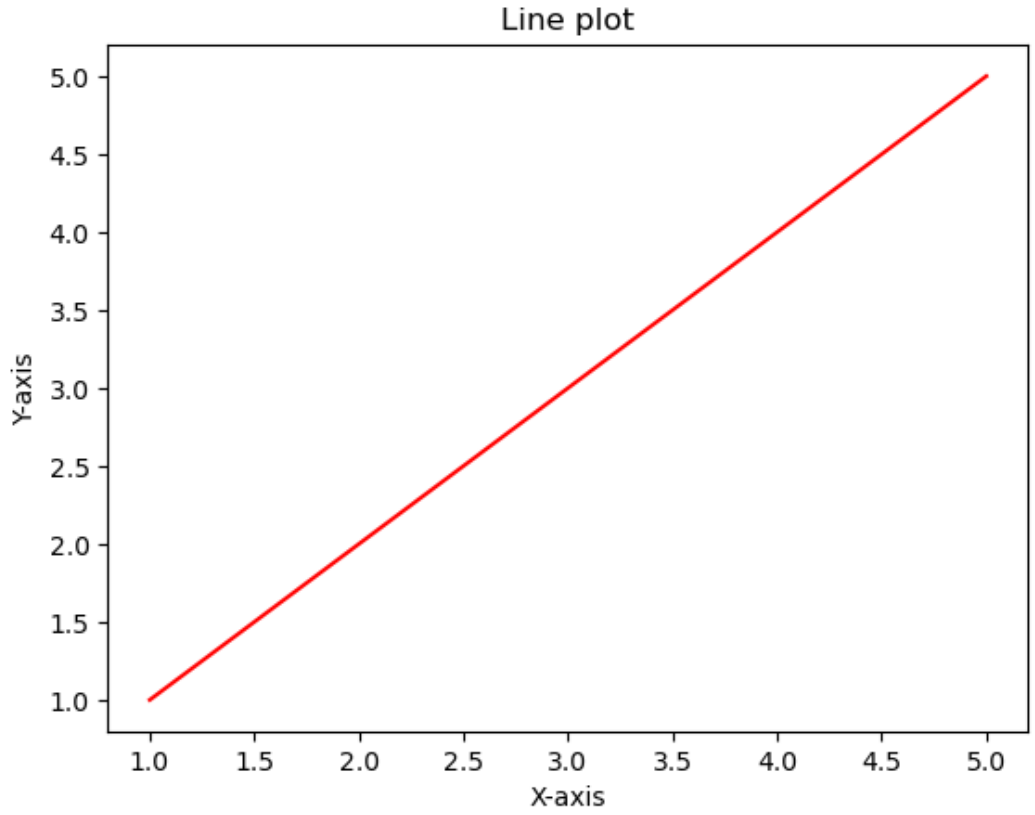



Reference video link: https://youtu.be/52tn54VjtDE?si=taFlG3lNPIrds41m

# FILL BETWEEN PLOT

Create a heading: Fill between plot

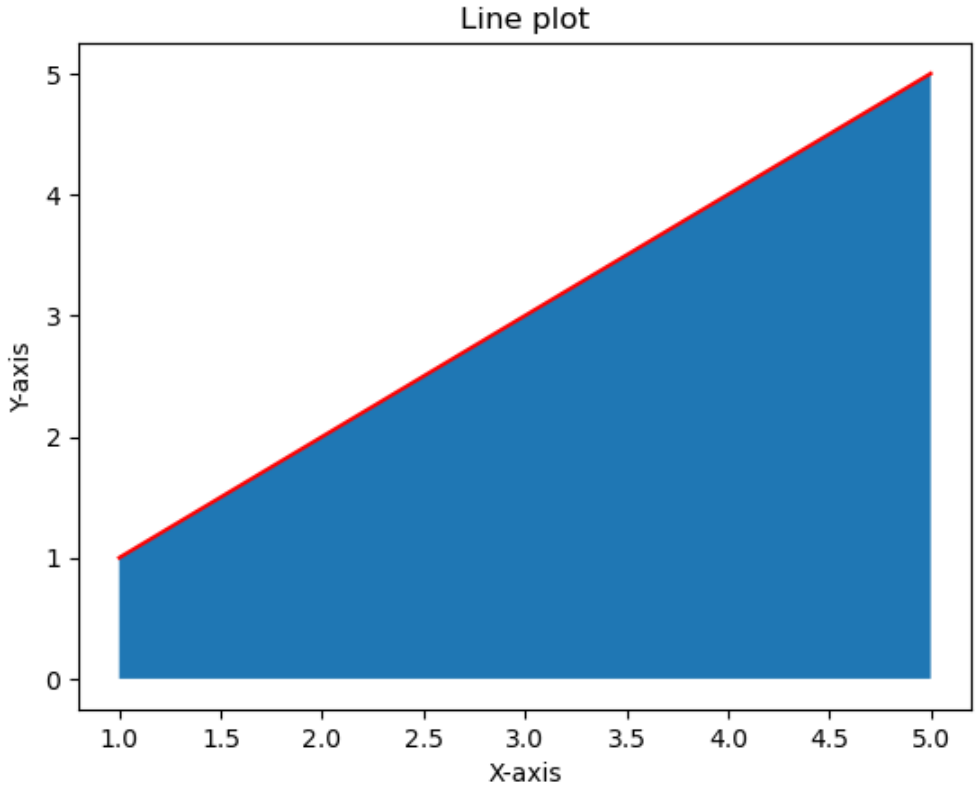
```
import matplotlib.pyplot as plt


x=[1,2,3,4,5]
y=[1,2,3,4,5]
plt.plot(x,y,color='r')
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

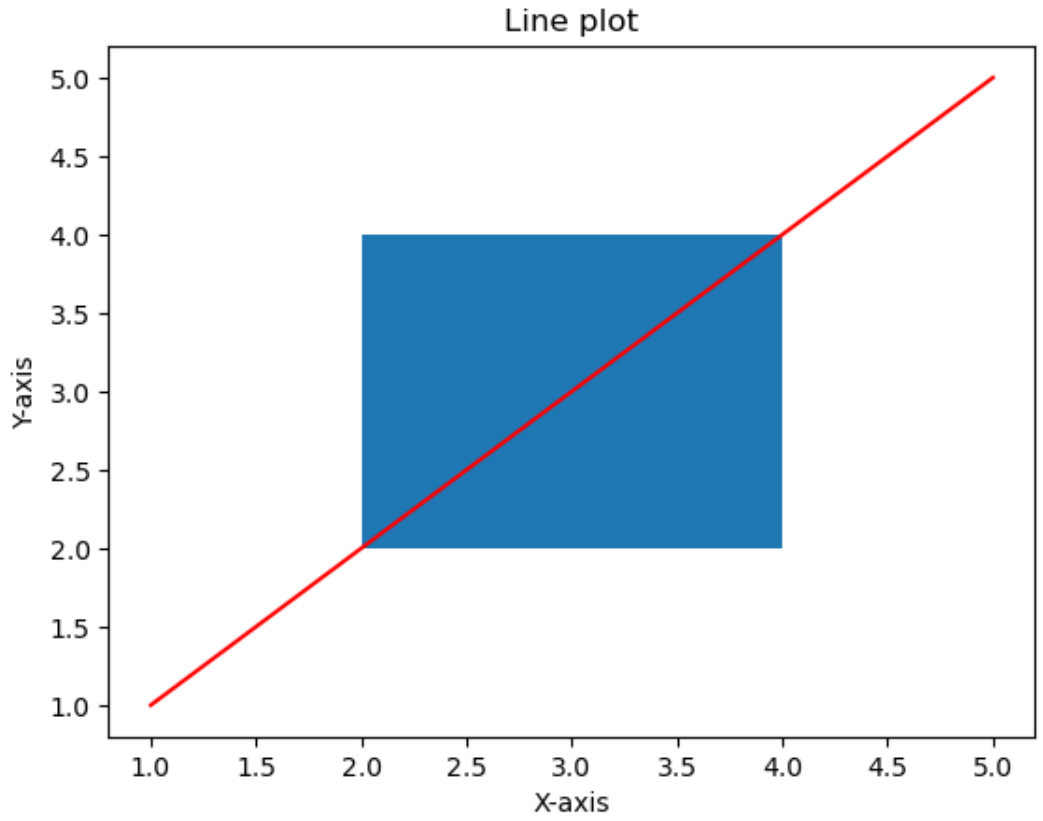
```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[1,2,3,4,5]

plt.plot(x,y,color='r')

plt.fill_between(x,y)
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

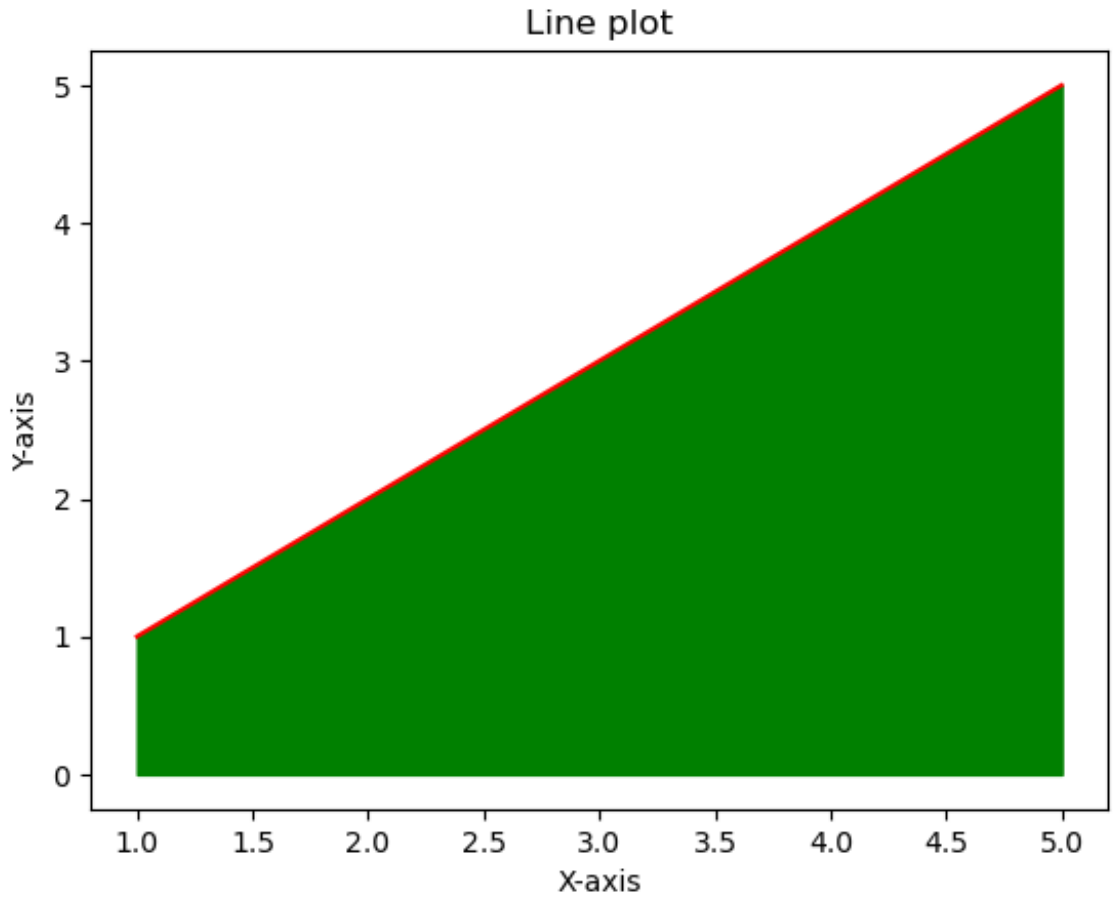
```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[1,2,3,4,5]

plt.plot(x,y,color='r')

plt.fill_between(x=[2,4],y1=2,y2=4)
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[1,2,3,4,5]

plt.plot(x,y,color='r')

plt.fill_between(x,y,color='g')
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[1,2,3,4,5]

plt.plot(x,y,color='r')

plt.fill_between(x,y,color='g', where=(x>=2)
& (x<=4))
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 8
      4 y=[1,2,3,4,5]
      6 plt.plot(x,y,color='r')
----> 8 plt.fill_between(x,y,color='g', where=(x>=2) & (x<=4))
      9 plt.title("Line plot")
     10 plt.xlabel("X-axis")

TypeError: '>=' not supported between instances of 'list' and 'int'
```

Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

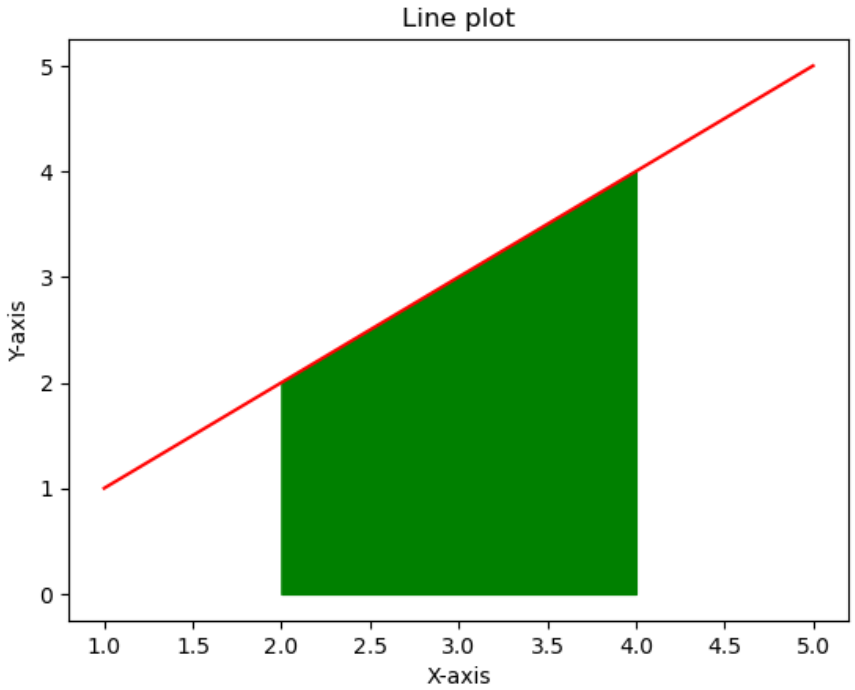
```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[1,2,3,4,5]

plt.plot(x,y,color='r')

plt.fill_between(x,y,color='g', where=(x>=2)
& (x<=4))
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 8
      4 y=[1,2,3,4,5]
      6 plt.plot(x,y,color='r')
----> 8 plt.fill_between(x,y,color='g', where=(x>=2) & (x<=4))
      9 plt.title("Line plot")
     10 plt.xlabel("X-axis")

TypeError: '>=' not supported between instances of 'list' and 'int'
```

Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

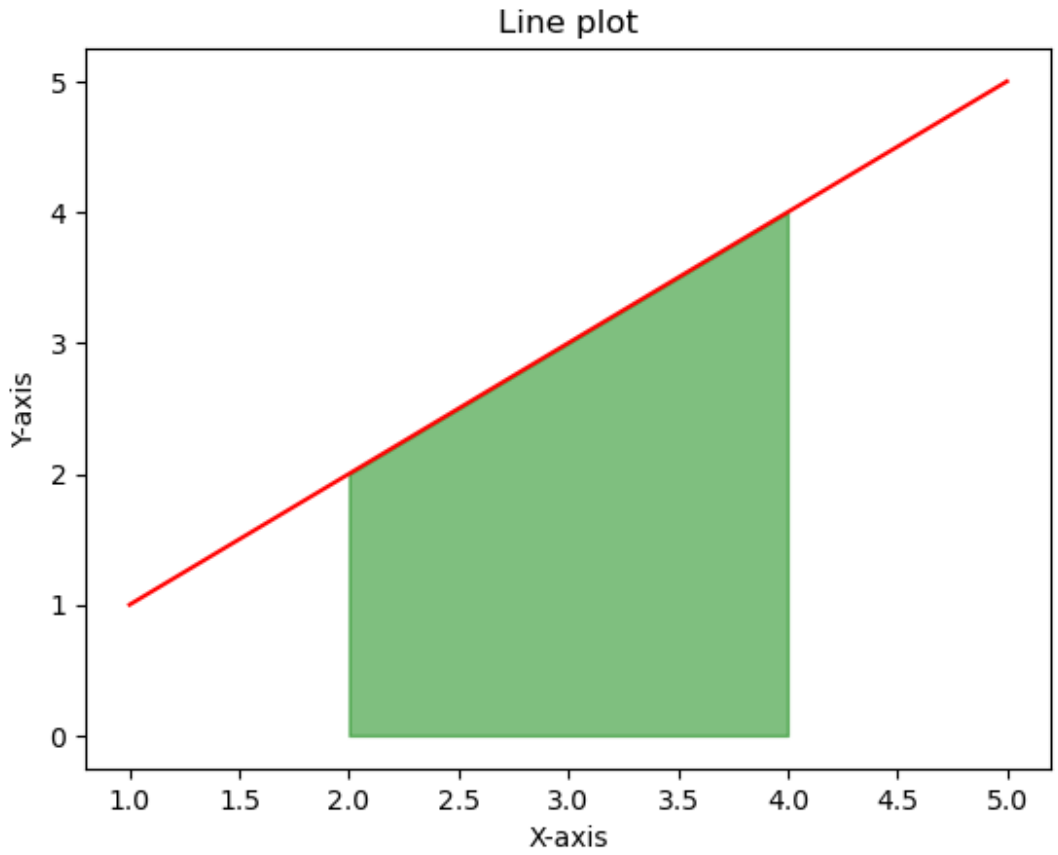
```
import matplotlib.pyplot as plt
import numpy as np #numpy array works like a number

x=np.array([1,2,3,4,5])
y=np.array([1,2,3,4,5])

plt.plot(x,y,color='r')

plt.fill_between(x,y,color='g', where=(x>=2) & (x<=4))
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# FILL BETWEEN PLOT

```
import matplotlib.pyplot as plt
import numpy as np #numpy array works like a number

x=np.array([1,2,3,4,5])
y=np.array([1,2,3,4,5])

plt.plot(x,y,color='r')

plt.fill_between(x,y,color='g', where=(x>=2) & (x<=4),
alpha=0.5)
plt.title("Line plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```




Reference video link: https://youtu.be/9rps_k4xi1M?si=xZaoHJR28lhR4wb3

# HISTOGRAM using MATPLOTLIB

```
import matplotlib.pyplot as plt
import numpy as np #using this to create data array
import random #this will give us random numbers


#generate fifty random numbers between the range 10 to 60
x=np.random.randint(10,60,(50))
print(x)
```

```
[21 37 15 15 55 20 53 56 58 22 29 10 48 48 55 25 36 59 41 53 21 34 52 15
 22 30 24 59 43 36 37 13 33 44 15 58 36 49 44 20 20 17 49 58 17 53 29 56
 49 21]
```

Reference video: https://youtu.be/Qk7caotaQUQ?si=M24XAOI-3Zu6D2vP

# HISTOGRAM

```
import matplotlib.pyplot as plt
import numpy as np #using this to create data array
import random #this will give us random numbers


#generate fifty random numbers between the range 10 to 60
x=np.random.randint(10,60,(50))
print(x)
```

```
[21 37 15 15 55 20 53 56 58 22 29 10 48 48 55 25 36 59 41 53 21 34 52 15
 22 30 24 59 43 36 37 13 33 44 15 58 36 49 44 20 20 17 49 58 17 53 29 56
 49 21]
```

Reference video: https://youtu.be/Qk7caotaQUQ?si=M24XAOI-3Zu6D2vP

# HISTOGRAM

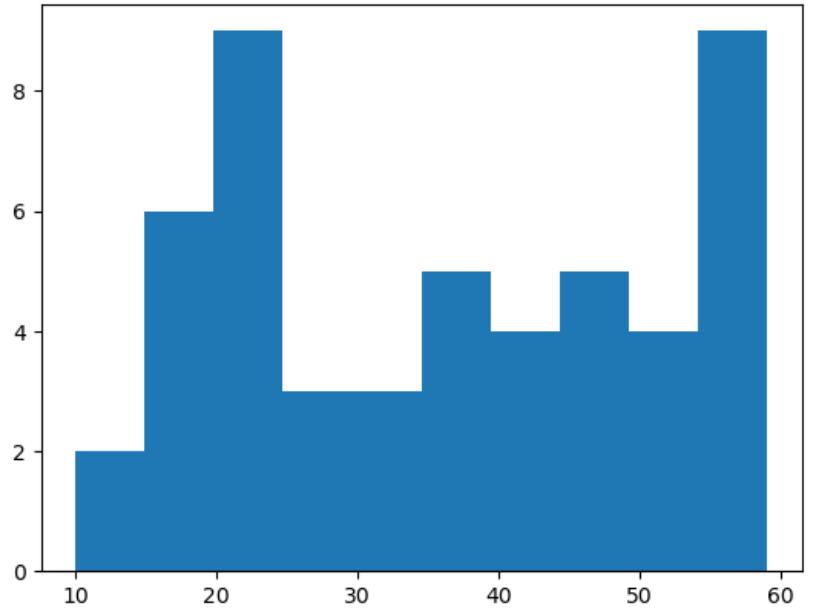
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15,

22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56,

49, 21] #separated the data using commas

plt.hist(number)

plt.show()

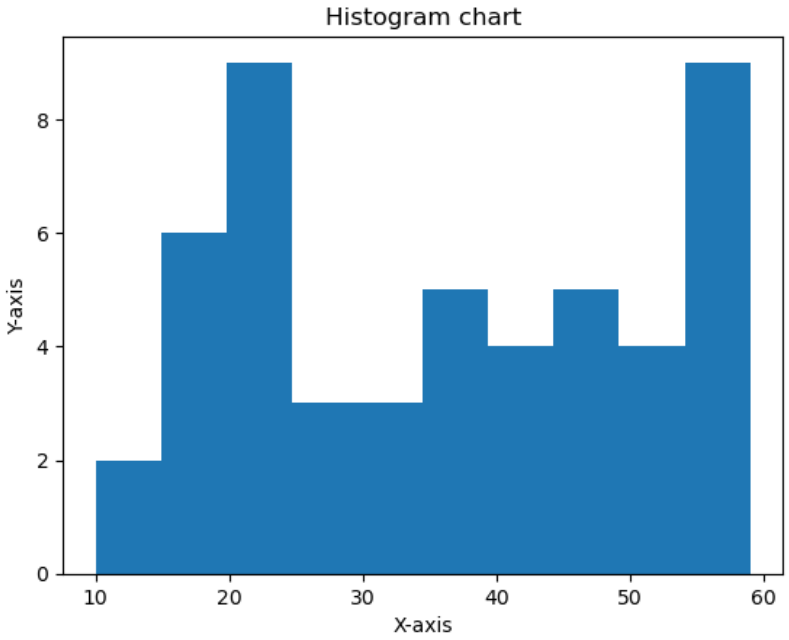



Reference video: https://youtu.be/Qk7caotaQUQ?si=M24XAOI-3Zu6D2vP

# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

```
plt.hist(number)

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")


plt.show()
```

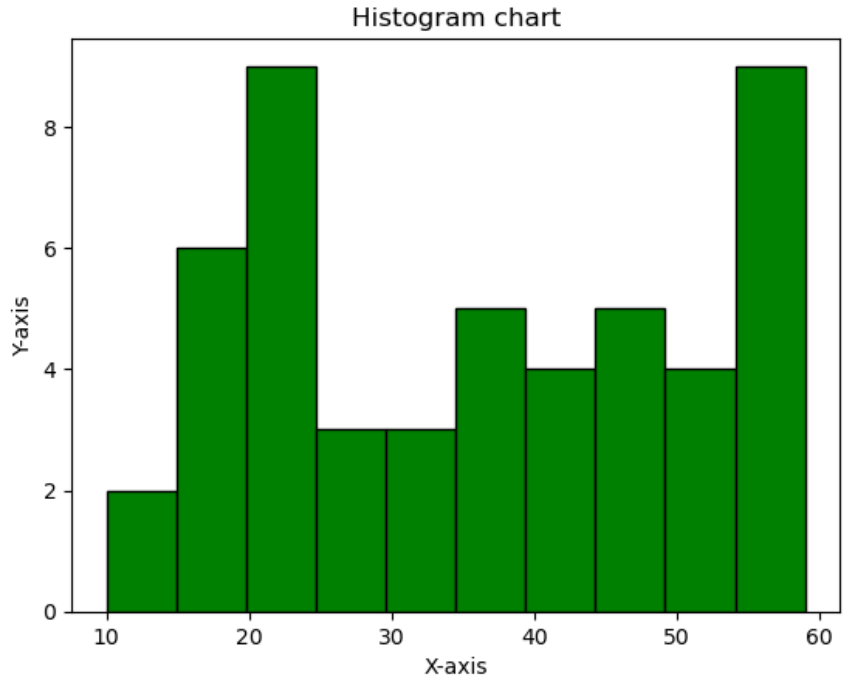



Reference video: https://youtu.be/Qk7caotaQUQ?si=M24XAOI-3Zu6D2vP

# HISTOGRAM

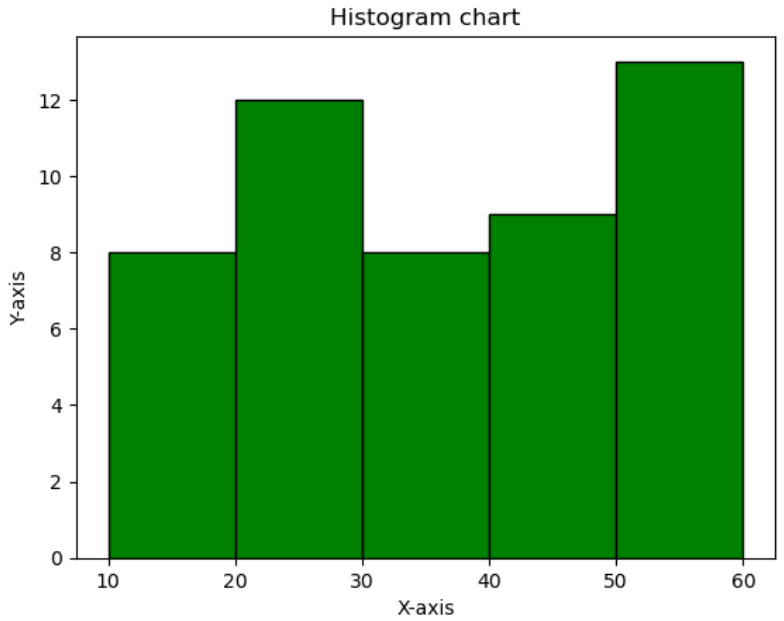
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

```
plt.hist(number, color='g', edgecolor='k')
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



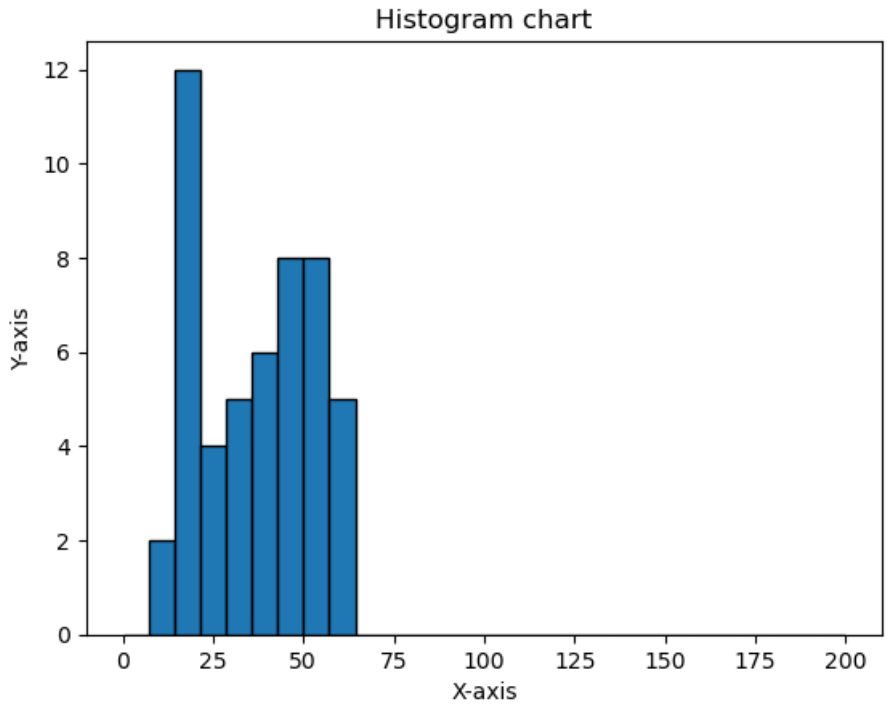
# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15,
 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56,
 49, 21] #separated the data using commas
l=[10,20,30,40,50,60]

plt.hist(number, color='g', bins=l, edgecolor='k')
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



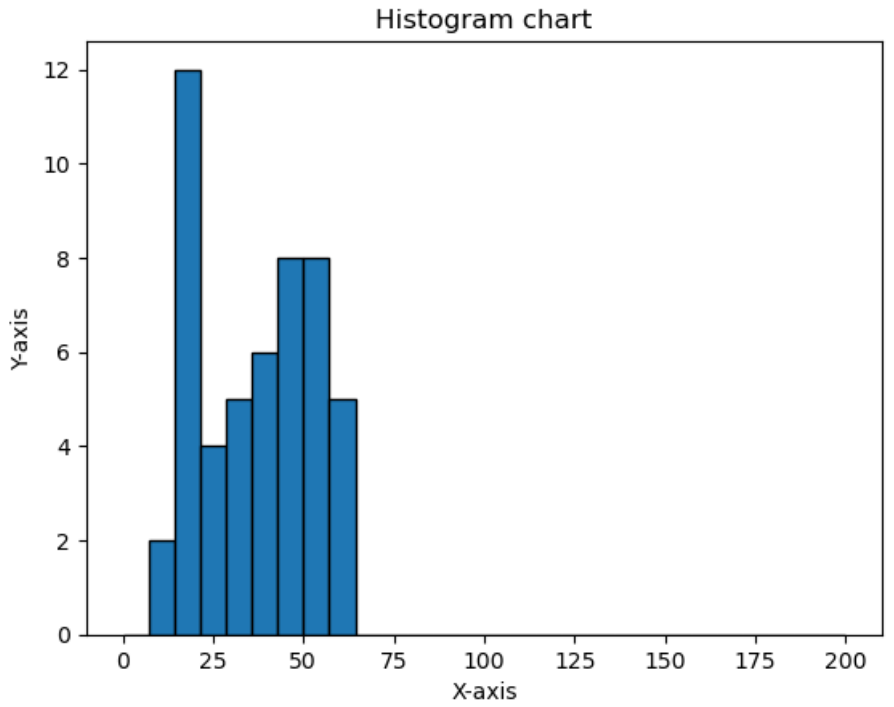
# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15,
 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56,
 49, 21] #separated the data using commas


l=[10,20,30,40,50,60]


plt.hist(number, "auto", (0,200), edgecolor='k') #auto parameter needs to come second
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")



plt.show()
```



# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15,
 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56,
 49, 21] #separated the data using commas


l=[10,20,30,40,50,60]


plt.hist(number, "auto", (0,200), edgecolor='k') #auto parameter needs to come second
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")



plt.show()
```



# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

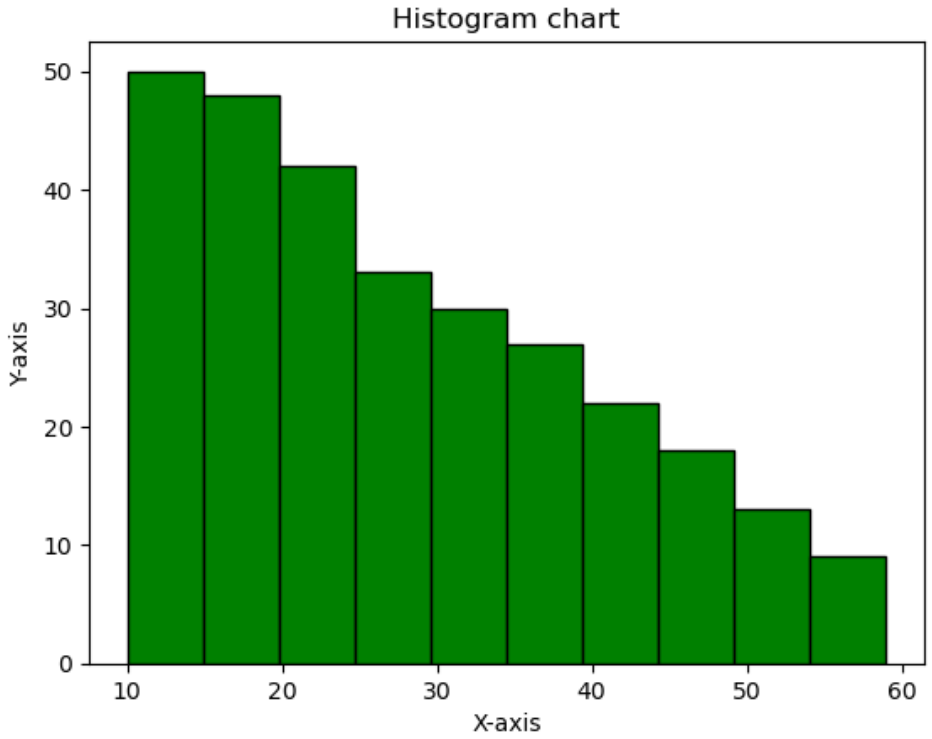
plt.hist(number, color='g', edgecolor='k', cumulative=-1)

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

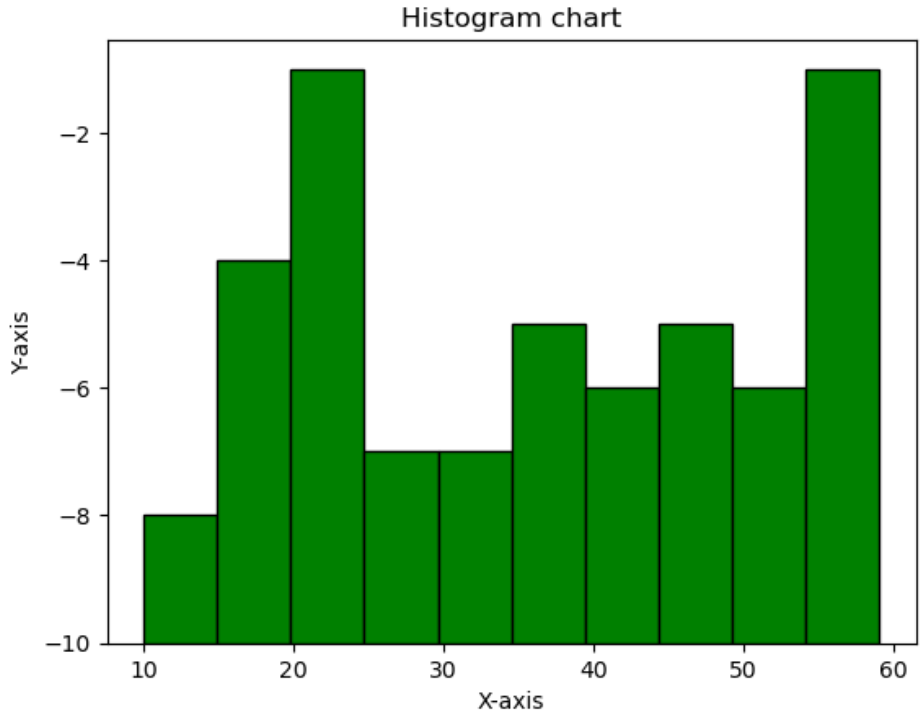
plt.show()



# HISTOGRAM

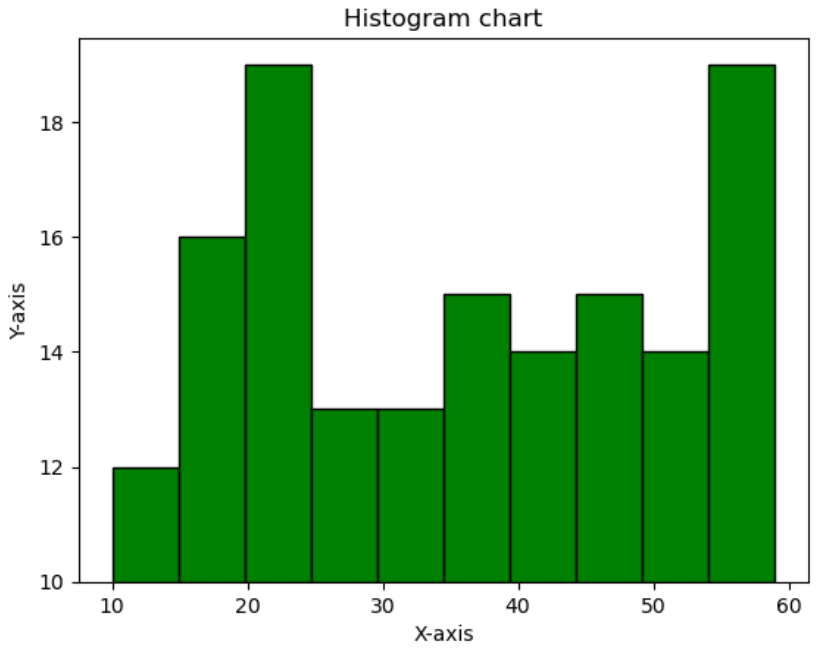
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

```
plt.hist(number, color='g', edgecolor='k', bottom=-10)
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



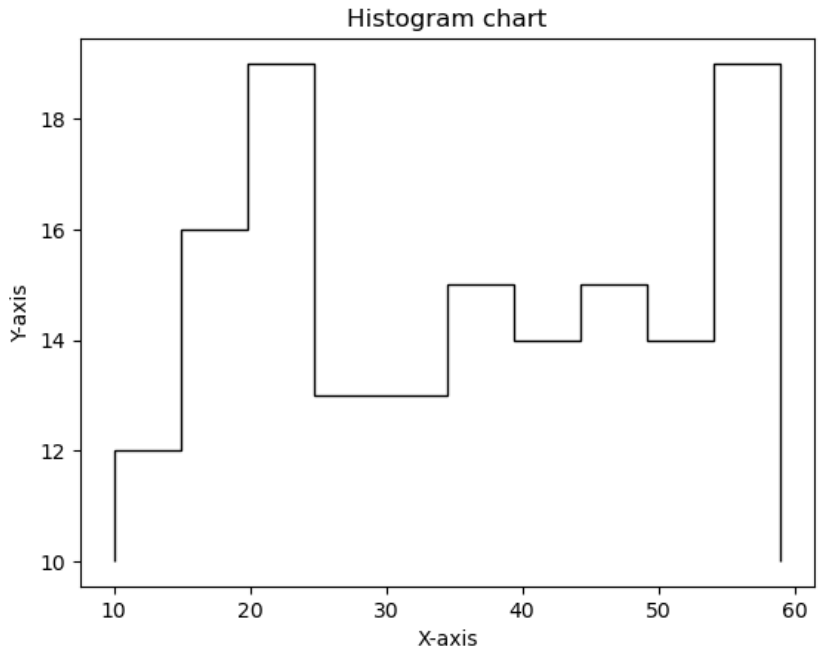
# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

#alignment has three options, mid, left, right

plt.hist(number, color='g', edgecolor='k', bottom=10, align='mid')

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")


plt.show()
```



# HISTOGRAM

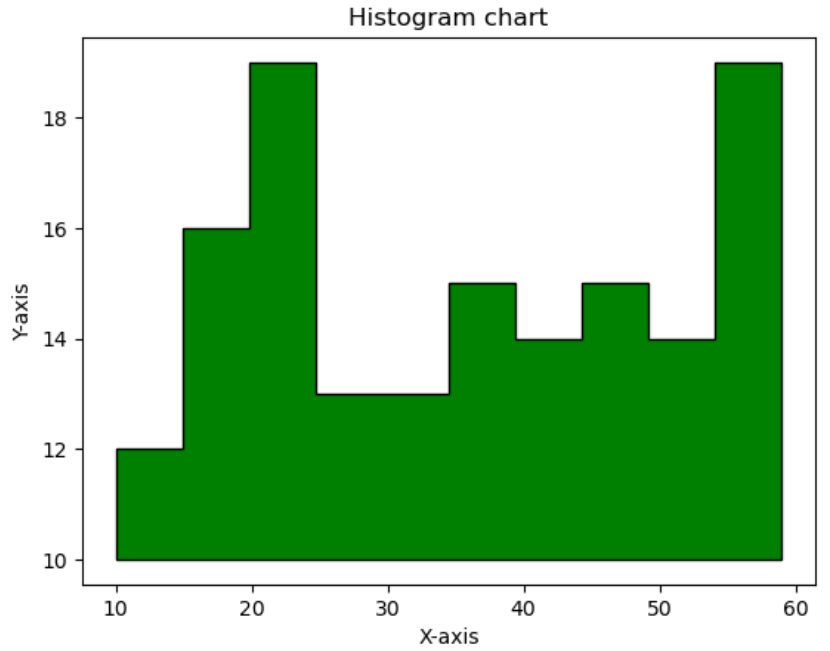
```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas


plt.hist(number, color='g', edgecolor='k', bottom=10, histtype="step")

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")


plt.show()
```



# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

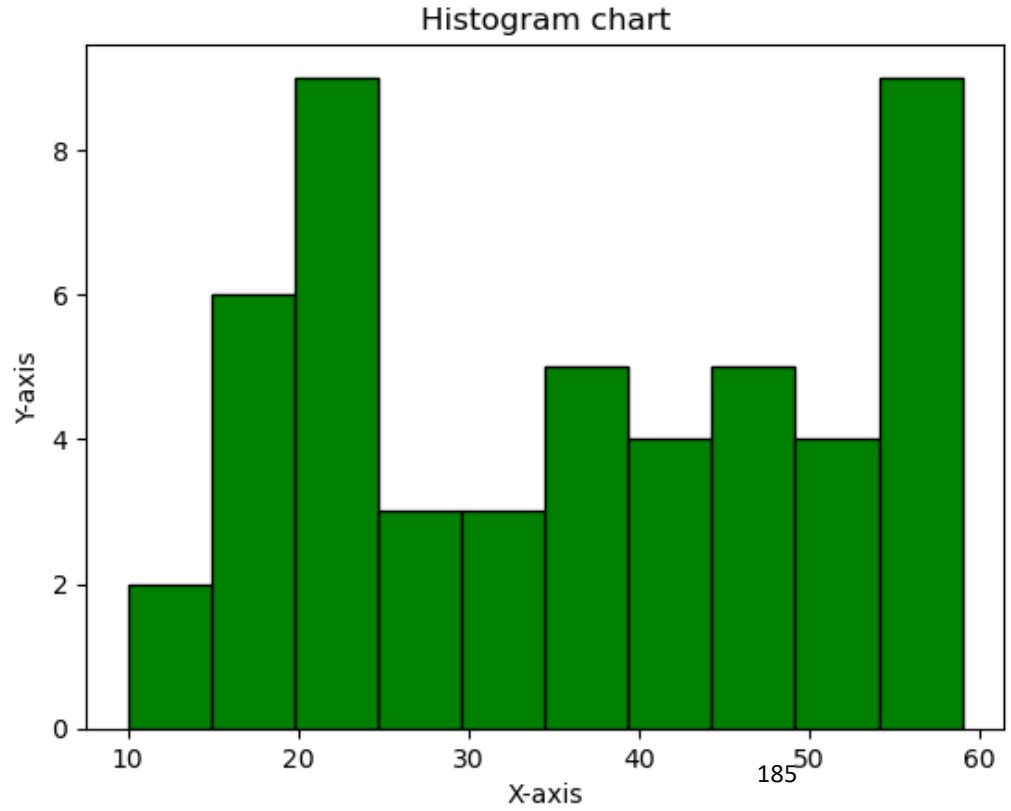
```
plt.hist(number, color='g', edgecolor='k', bottom=10, histtype="stepfilled")
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

```
plt.hist(number, color='g', edgecolor='k', histtype="barstacked")
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

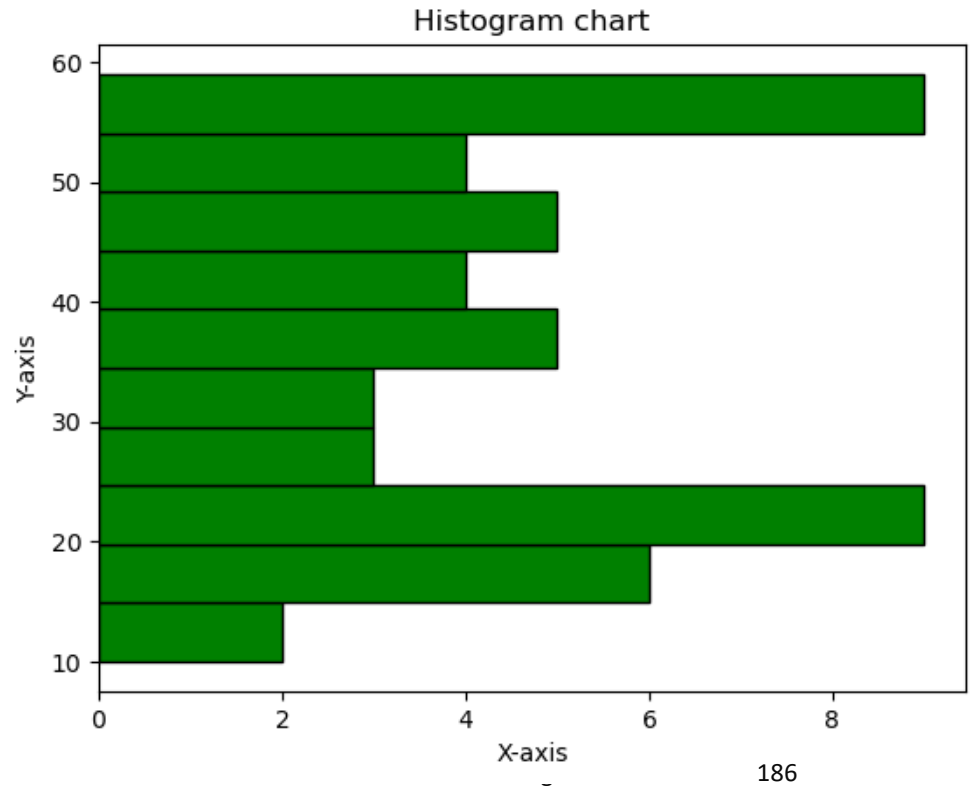
plt.hist(number, color='g', edgecolor='k', orientation='horizontal')

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

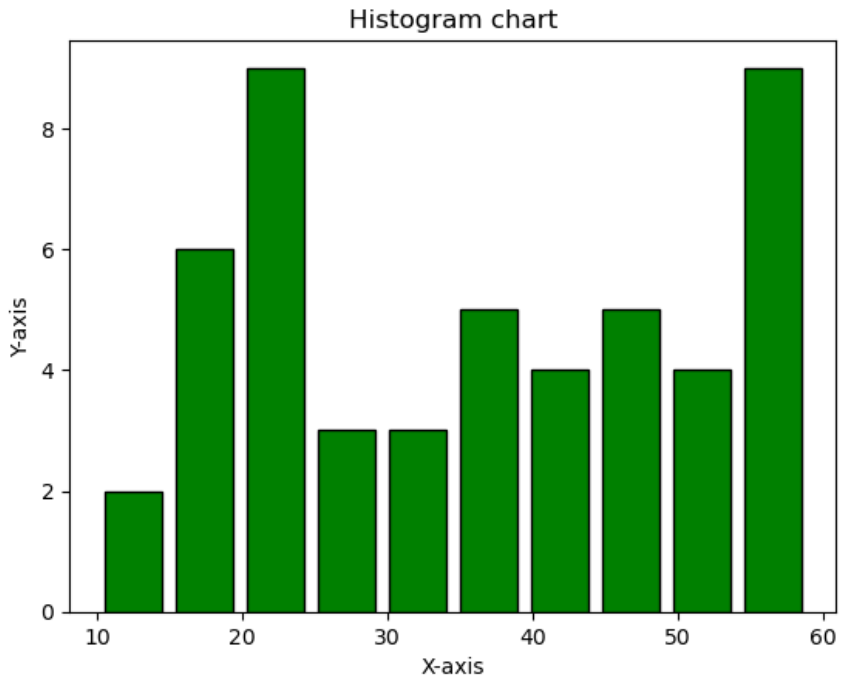
plt.show()



# HISTOGRAM

number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

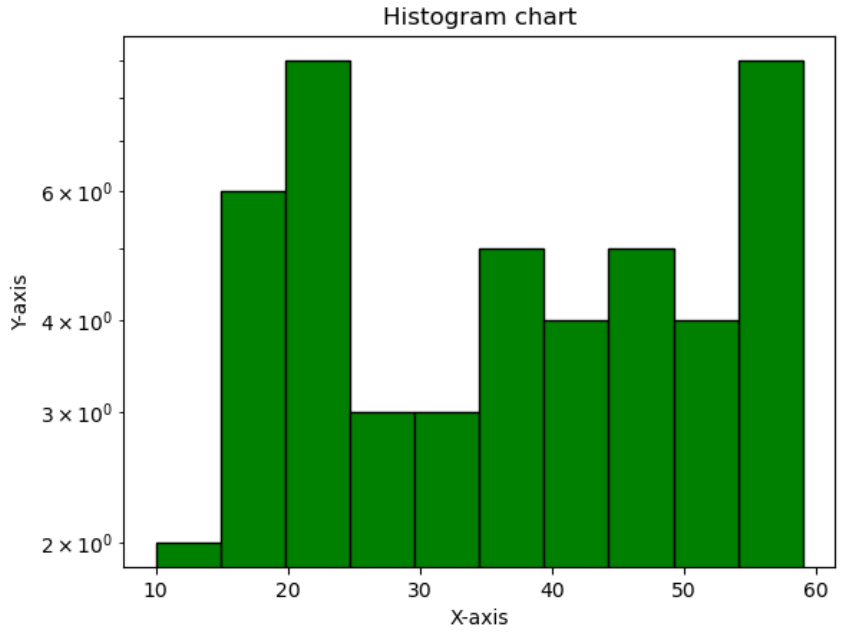
```
plt.hist(number, color='g', edgecolor='k', rwidth=0.8)
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



# HISTOGRAM

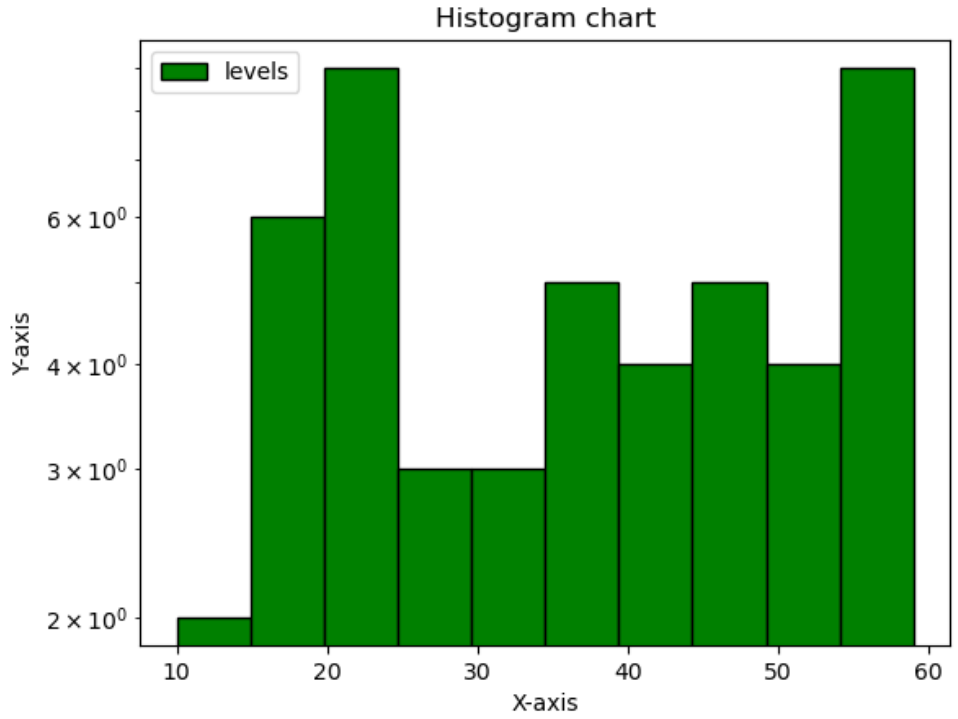
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas

```
plt.hist(number, color='g', edgecolor='k', log=True)
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```



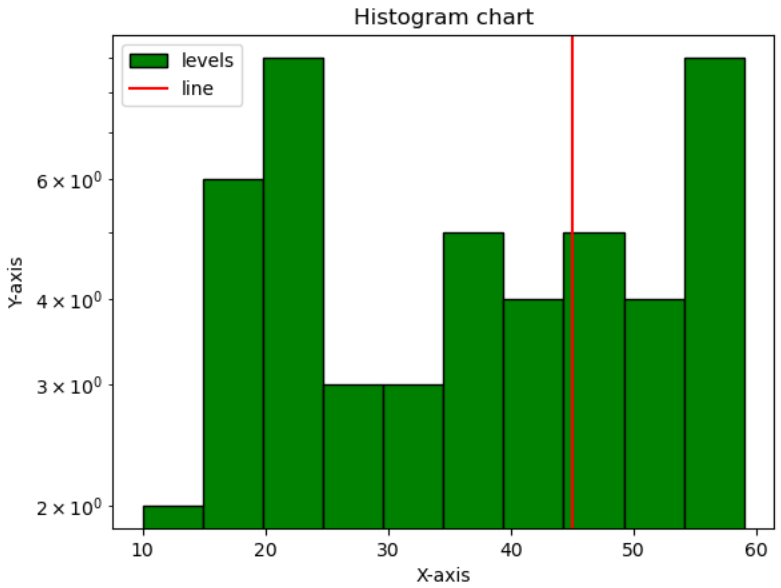
# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41,
53, 21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20,
17, 49, 58, 17, 53, 29, 56, 49, 21] #separated the data using commas


plt.hist(number, color='g', edgecolor='k', log=True, label='levels')
plt.title("Histogram chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend(loc=2)


plt.show()
```



# HISTOGRAM

```
number=[21, 37, 15, 15, 55, 20, 53, 56, 58, 22, 29, 10, 48, 48, 55, 25, 36, 59, 41, 53,
21, 34, 52, 15, 22, 30, 24, 59, 43, 36, 37, 13, 33, 44, 15, 58, 36, 49, 44, 20, 20, 17, 49,
58, 17, 53, 29, 56, 49, 21] #separated the data using commas

plt.hist(number, color='g', edgecolor='k', log=True, label='levels')

plt.title("Histogram chart")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")


plt.axvline(45,color='r', label='line')

plt.legend(loc=2)


plt.show()
```



# Move the Legend Outside the Plot

To move the legend outside the plot, we'll use the bbox_to_anchor and loc parameters of the legend function.

```
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
```

The bbox_to_anchor parameter specifies the legend's position. The tuple (1.05, 1) positions the legend just outside the plot's right edge. The loc parameter determines where the legend's anchor point should be. 'upper left' means the upper left corner of the legend will be at the position specified by bbox_to_anchor.

# Adjust the Plot’s Size

Finally, we may need to adjust the plot’s size to ensure the legend doesn’t overlap with it. We can do this using the subplots_adjust function.

```
plt.subplots_adjust(right=0.7)
```

The right parameter specifies the right side of the subplot’s position in the figure. By setting it to 0.7, we leave enough space for the legend on the right.

# DISPLAY COLUMNS

The display.max_columns option controls the number of columns to be printed. It receives an int or None, the latter used to print all the columns):

```
pd.set_option('display.max_columns', None)
fifa.head()
```

Reference: https://builtin.com/data-science/pandas-show-all-columns

# Practice projects using MATPLOTLIB

Have a look at the video below for reference:

https://www.youtube.com/watch?v=0P7QnIQDBJY

For more clarity please refer to the MATPLOTLIB documentation:
https://matplotlib.org/stable/index.html

Also, refer to the Github profile to refer to more datasets and practice visualizing them using MATPLOTLIB library in Python.

Github profile: https://github.com/asomya87

Thank you all!
See you in next class… :)