Contents

[DATA VISUALIZATION 2](#_Toc210472204)

[MATPLOTLIB 2](#_Toc210472205)

[FUNCTIONAL BASED METHOD APPROACH FOR PLOT 2](#_Toc210472206)

[OOP BASED METHOD APPROACH FOR PLOT 2](#_Toc210472207)

[FIGURE OBJECT 2](#_Toc210472208)

[SUBPLOTS 8](#_Toc210472209)

# DATA VISUALIZATION

# MATPLOTLIB

* Matplotlib is a library in Python that helps us draw graphs and charts.
* There are 2 approaches to create plots
  + Functional Based Methods
  + OOP (Object Oriented) Method

|  |  |
| --- | --- |
| INSTALL | pip install marplotlib |
| IMPORT | import matplotlib.pyplot as plt |

## FUNCTIONAL BASED METHOD APPROACH FOR PLOT

|  |  |
| --- | --- |
| X values | x = np.arange(10)  [0 1 2 3 4 5 6 7 8 9] |
| Y values | y = 2 \* x  [ 0 2 4 6 8 10 12 14 16 18] |
| PLOT | plt.plot(x,y)  plt.show() 🡨 Not required for Jupiter Notebook. Required for .py scripts |
| TITLE | plt.title("Simple Line Plot") |
| X & Y LABEL | plt.xlabel("X-axis")  plt.ylabel("Y-axis") |
| SHOW | plt.show() 🡨 Not required for Jupiter Notebook. Required for .py scripts |

A line graph with numbers and symbols

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| The plot are saved on different format like PNG files, jpeg files | **plt.savefig("line\_plot.png")** |

## OOP BASED METHOD APPROACH FOR PLOT

### FIGURE OBJECT

* **Figure object** is the **top-level container** for all plot elements.
* Think of it as the **canvas** on which everything is drawn — like a blank sheet of paper where we can add one or more plots (called **Axes**).
* A **Figure** is created using plt.figure() or plt.subplots().
* It can contain **multiple Axes** (i.e., subplots).
* It also holds other elements like titles, legends, and colorbars.

|  |  |  |
| --- | --- | --- |
| **CREATING FIGURE OBJECT** | fig = plt.figure()   * This creates an empty canvas | |
| **ADDING AXES** | **axes = fig.add\_axes([0, 0, 1, 1])**   * The add\_axes() method in **Matplotlib** is used to manually add an **Axes** (i.e., a plot area) to a **Figure** at a specific position and size | |
| **axes = fig.add\_axes([0, 0, 1, 1])**  **What This Does?**   * fig.add\_axes() adds a new **Axes** (plot area) to the **Figure**. * The argument [0, 0, 1, 1] is a list that defines the **position and size** of the Axes **within the figure**.   **Format: [left, bottom, width, height]**  All values are **fractions of the figure size**:   * left = 0: starts at the **left edge** of the figure. * bottom = 0: starts at the **bottom edge** of the figure. * width = 1: spans the **entire width** of the figure. * height = 1: spans the **entire height** of the figure.   So this line creates an Axes that **fills the entire figure**. | | |
| **add\_axes() method** | | * It lets us precisely control where the plot appears on the canvas. * We specify the position and size using a list of four numbers: * [left, bottom, width, height] — **all values are in fraction of the figure size** (from 0 to 1). |
| **fig = plt.figure()**  **axes = fig.add\_axes([0, 0, 1, 1])**  **axes.plot([1, 2, 3], [4, 5, 6])**  **axes.set\_title("Full-Figure Axes")**  **plt.show()**  **The plot will start from left bottom as left=0, bottom=0**  **and height & width will take complete width and height of the canvas** | | **fig = plt.figure()**  **axes = fig.add\_axes([0, 0, 0.5, 0.5])**  **axes.plot([1, 2, 3], [4, 5, 6])**  **axes.set\_title("Full-Figure Axes")**  **plt.show()**  **The plot will start from left bottom as left=0, bottom=0**  **and height & width will 0.5 of the total canvas** |
|  | |  |

#### IMPLEMENTING FIGURE AND AXES

|  |  |
| --- | --- |
| import matplotlib.pyplot as plt  import numpy as np  fig = plt.figure()  a = np.linspace(0, 10, 11)  **## [ 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.]**  b = a\*\*4  axes = fig.add\_axes([0.1, 0.1, 0.8, 0.8])  axes.plot(a, b)  axes.set\_title("Power of 4")  axes.set\_xlabel("A")  axes.set\_ylabel("B")  plt.show() |  |

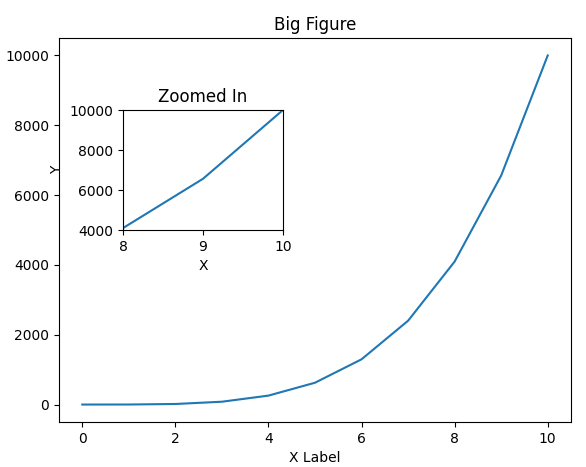
#### MULTIPLE AXES IN FIGURE

A graph of a line

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| import matplotlib.pyplot as plt  import numpy as np  fig = plt.figure()  a = np.linspace(0, 10, 11)  print(a)  b = a\*\*4  x = np.arange(0, 10)  y = 2 \* x  axes1 = fig.add\_axes([0.1, 0.1, 0.8, 0.8])  axes1.plot(a, b)  axes1.set\_title("Power of 4")  axes1.set\_xlabel("A")  axes1.set\_ylabel("B")  axes2 = fig.add\_axes([0.2, 0.4, 0.4, 0.4])  axes2.plot(x, y)  axes2.set\_title("Linear")  axes2.set\_xlabel("Z")  axes2.set\_ylabel("Y")  plt.show() | * This will create an insert plot with multiple axes in a figure |

**EXAMPLE -2 (ZOOMED PLOT)**



|  |  |
| --- | --- |
| import matplotlib.pyplot as plt  import numpy as np  fig = plt.figure()  a = np.linspace(0, 10, 11)  print(a)  b = a\*\*4  axes1 = fig.add\_axes([0.1, 0.1, 0.8, 0.8])  # Large figure  axes2 = fig.add\_axes([0.2, 0.5, 0.25, 0.25])  # Smaller figure  # Larger Figure Axes 1  axes1.plot(a, b)  # Use set\_ to add to the axes figure  axes1.set\_xlabel("X Label")  axes1.set\_ylabel("Y Label")  axes1.set\_title("Big Figure")  # Insert Figure Axes 2  axes2.plot(a, b)  axes2.set\_xlim(8, 10)  axes2.set\_ylim(4000, 10000)  axes2.set\_xlabel("X")  axes2.set\_ylabel("Y")  axes2.set\_title("Zoomed In")  plt.show() | **In this example – we will want to zoon in a specific section of bigger plot**  **axes1 = fig.add\_axes([0.1, 0.1, 0.8, 0.8])**   * Adds a large plot area. * [0.1, 0.1, 0.8, 0.8] means:   + Starts 10% from left and bottom.   + Covers 80% of width and height.   **axes2 = fig.add\_axes([0.2, 0.5, 0.25, 0.25])**   * Adds a smaller plot **inside** the main one. * [0.2, 0.5, 0.25, 0.25] places it at 20% from left, 50% from bottom, and 25% size.   **axes2.set\_xlim(8, 10)**  **axes2.set\_ylim(4000, 10000)**  Focuses on the range where a is between 8 and 10, and b is between 4000 and 10000. |

A graph with a line

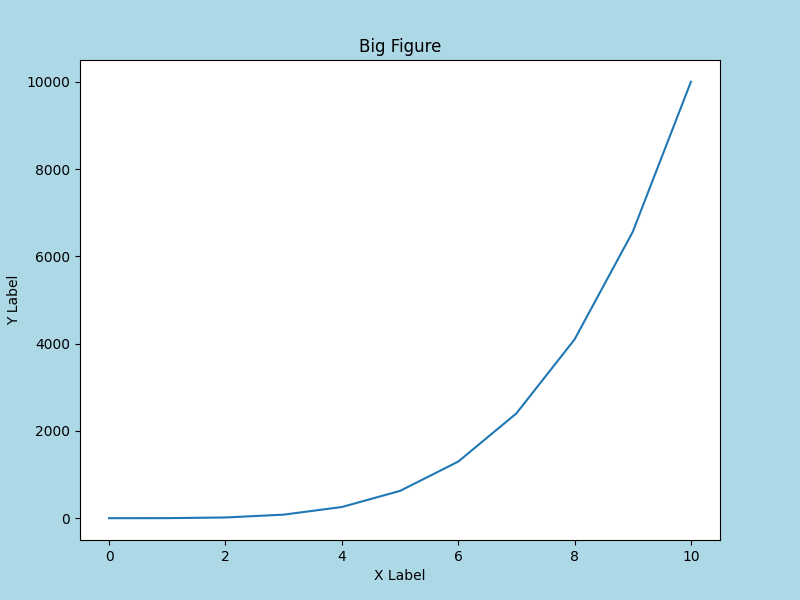
AI-generated content may be incorrect.

|  |  |
| --- | --- |
| import matplotlib.pyplot as plt  import numpy as np  fig = plt.figure()  a = np.linspace(0, 10, 11)  print(a)  b = a\*\*4  axes1 = fig.add\_axes([0.1, 0.1, 0.6, 0.6]) # Large figure  axes2 = fig.add\_axes([0.7, 0.7, 0.25, 0.25]) # Smaller figure  # Larger Figure Axes 1  axes1.plot(a, b)  # Use set\_ to add to the axes figure  axes1.set\_xlabel("X Label")  axes1.set\_ylabel("Y Label")  axes1.set\_title("Big Figure")  # Insert Figure Axes 2  axes2.plot(a, b)  axes2.set\_xlim(8, 10)  axes2.set\_ylim(4000, 10000)  axes2.set\_xlabel("X")  axes2.set\_ylabel("Y")  axes2.set\_title("Zoomed In")  plt.show() | * By changing the parameters of add\_axes() we can change the location of the inset plot |

#### FIGURE PARAMETERS

|  |  |
| --- | --- |
| Parameter | Description |
| num | An identifier for the figure (number or name). Useful when managing multiple figures. |
| figsize | Tuple (width, height) in inches. Controls the size of the figure. Example: figsize=(8, 6) |
| dpi | Dots per inch. Controls the resolution of the figure. Higher = sharper image. |
| facecolor | Background color of the figure. Example: 'lightgray' |
| edgecolor | Border color of the figure. |
| frameon | Boolean. If False, suppresses the figure frame. Default is True. |
| clear | If True, clears the figure if it already exists. |

|  |  |
| --- | --- |
| import matplotlib.pyplot as plt  fig = plt.figure(      num=1,      figsize=(8, 6),      dpi=100,      facecolor='lightblue',      edgecolor='black',      frameon=True  ) | This creates a figure:   * 8 inches wide, 6 inches tall * 100 DPI resolution * Light blue background * Black border |



### SUBPLOTS