DATA ANALYTICS

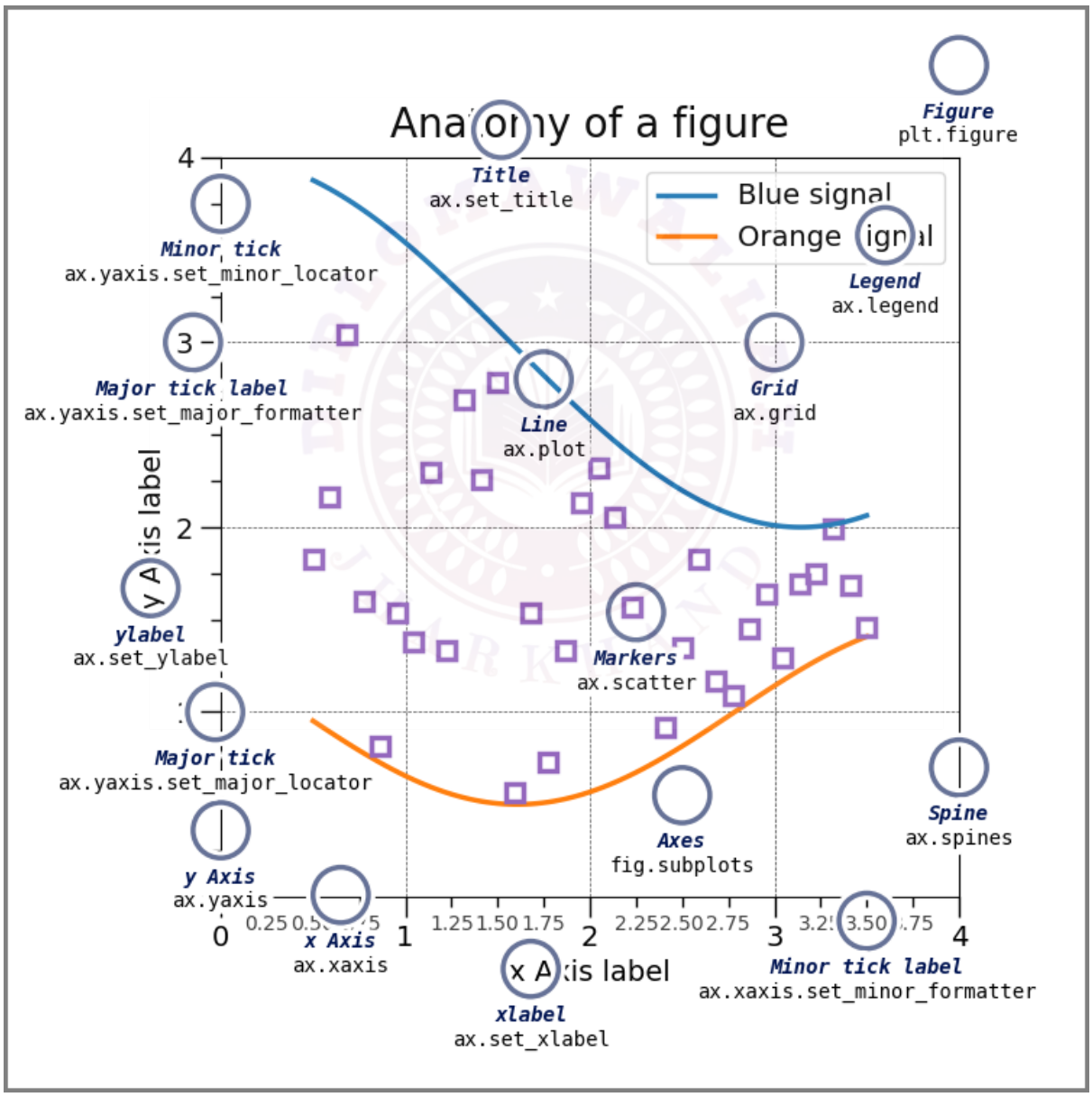
DIPLOMA WALLAH

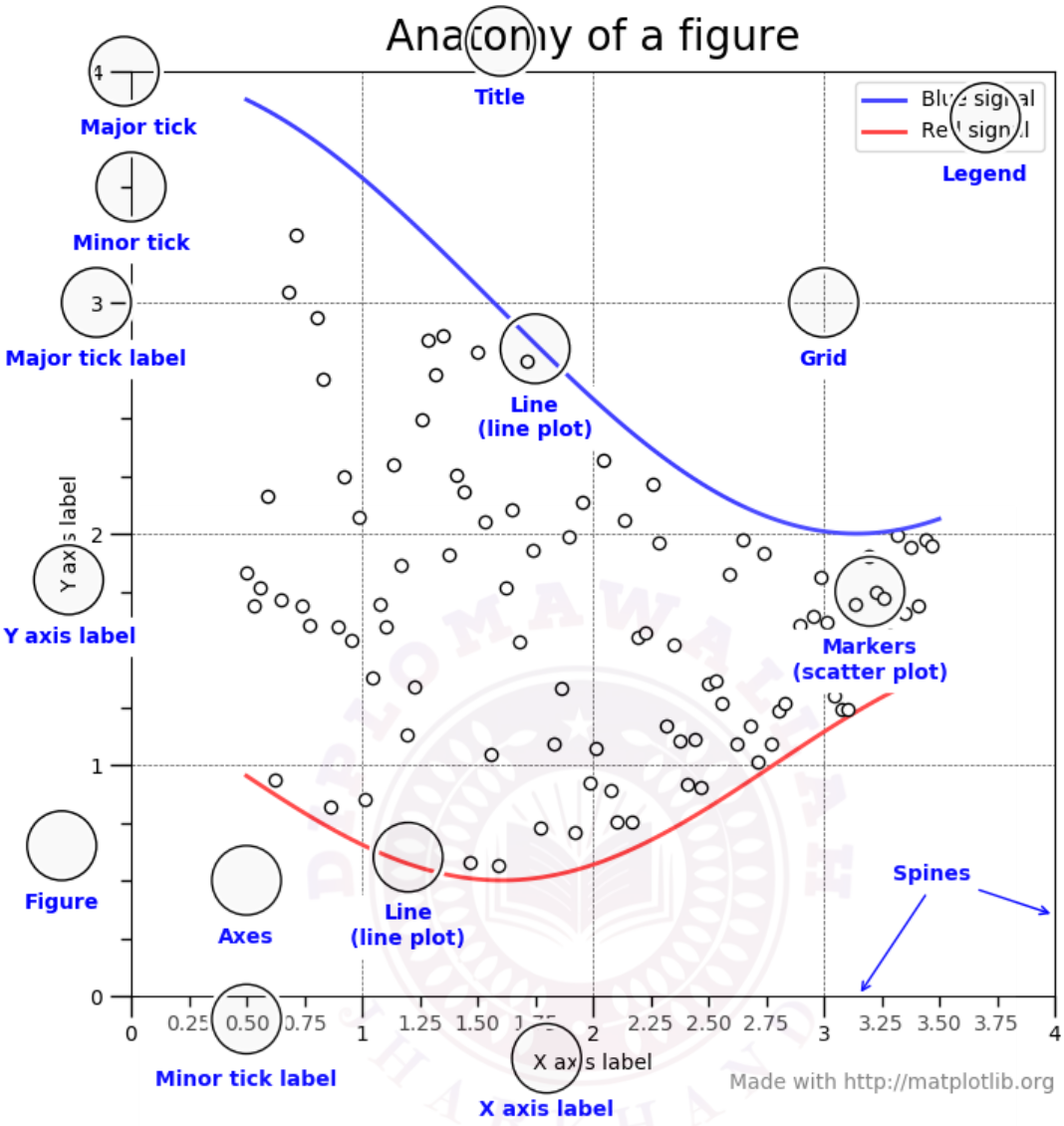
OPEN ELECTIVE

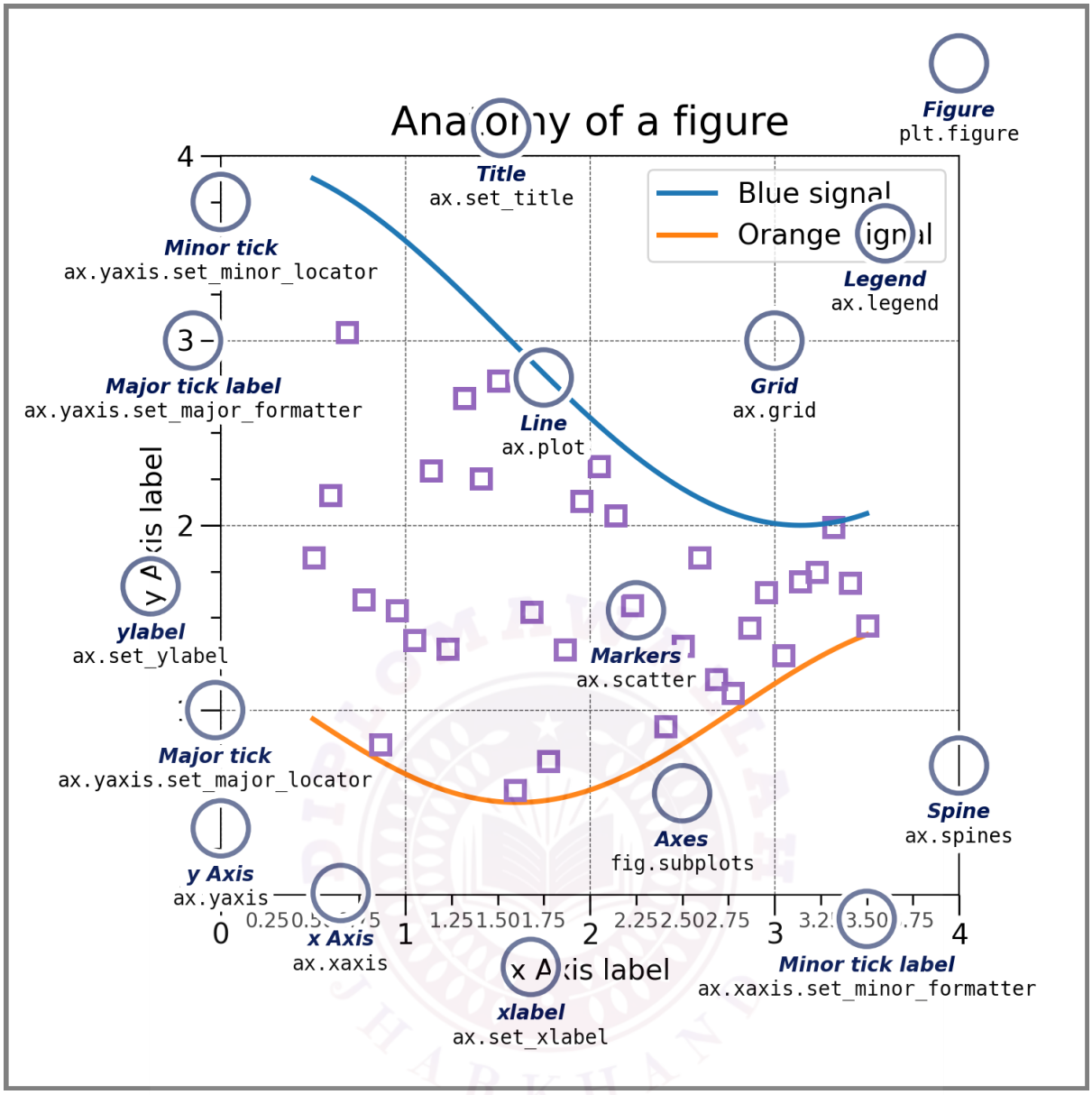
Jharkhand University Of Technology (JUT)

Unit V: Data Visualization using Python

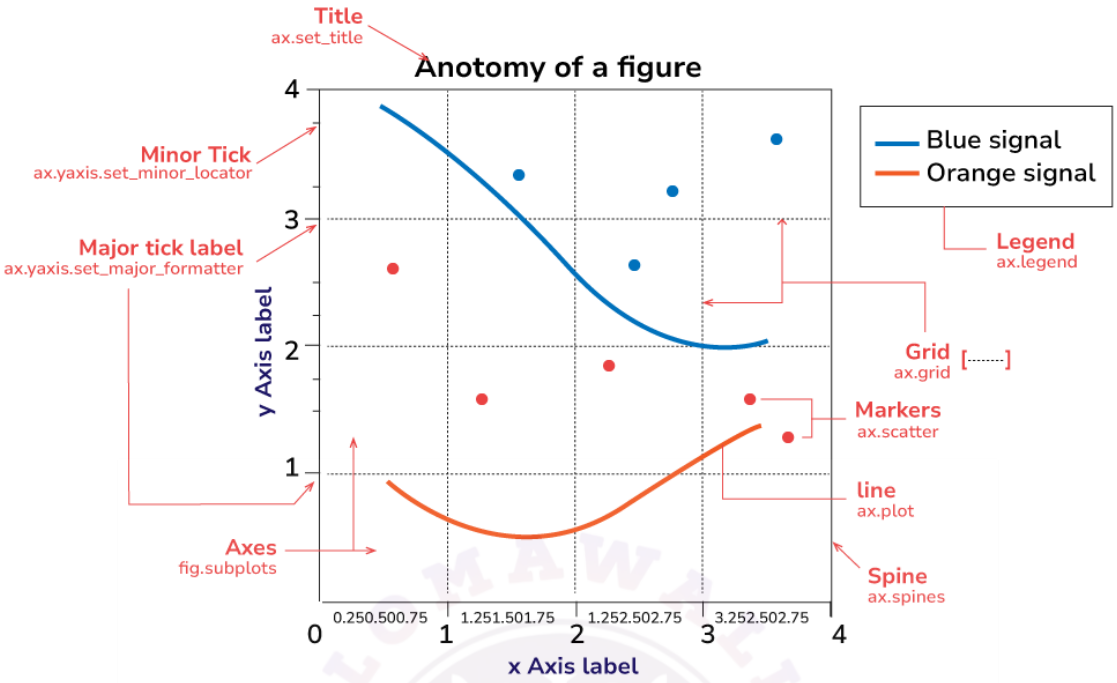
5.1 Overview of Matplotlib and Its Role in Data Visualization

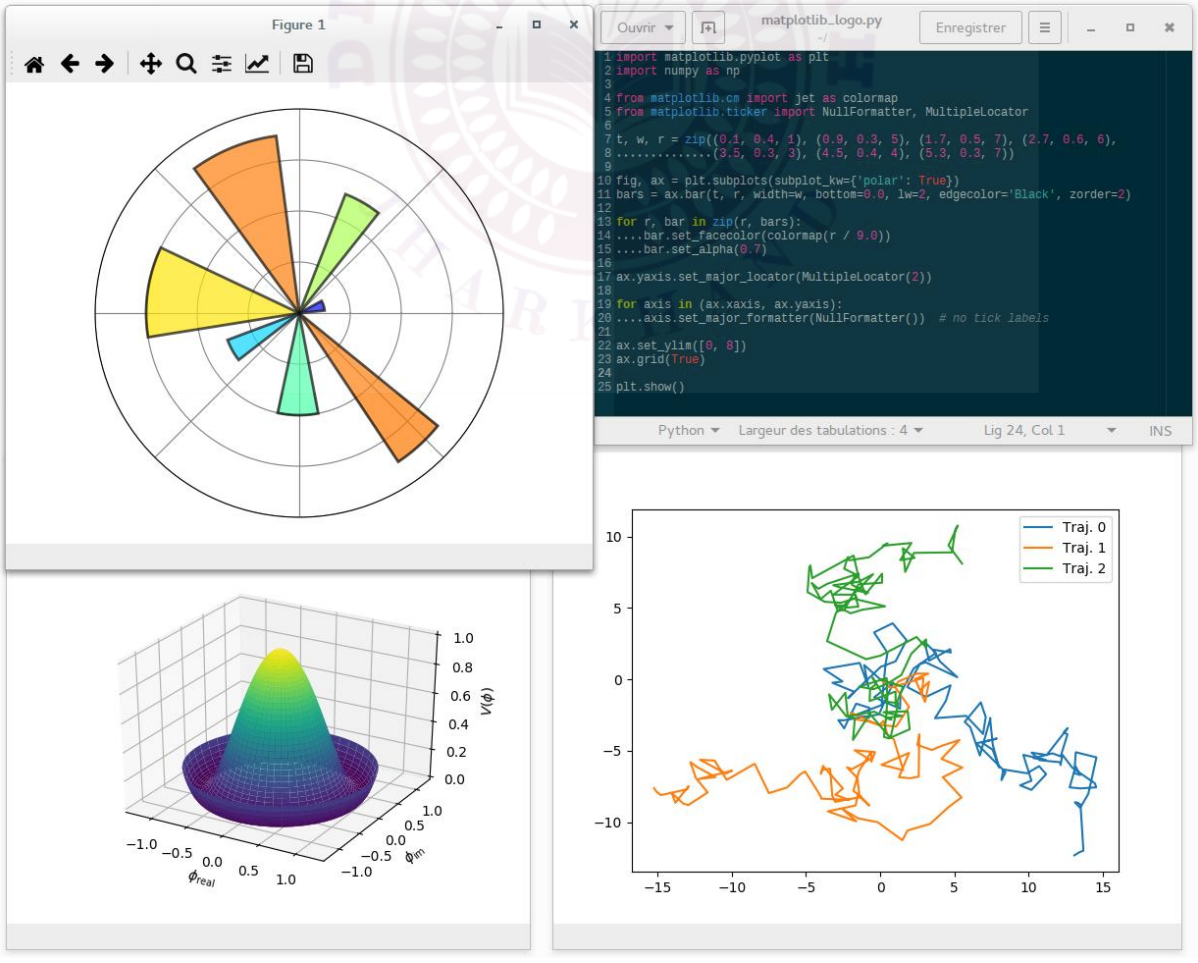






Matplotlib





What Matplotlib is:

- Matplotlib is a comprehensive Python library for creating static, animated and interactive visualisations. (Matplotlib)
- It was originally created by John D. Hunter and is distributed under a BSD-style license. (Wikipedia)
- It integrates well with NumPy and other scientific Python libraries and is foundational for data-science visualisation tasks. (GeeksforGeeks)

Role in Data Visualization:

- Converts raw numeric / categorical data into graphical form (plots, charts) so that trends, patterns, and relationships become visible rather than buried in tables.
- Enables creation of publication-quality figures, embedding into notebooks, GUIs, web applications. (Matplotlib)
- Forms the base for higher-level libraries (such as Seaborn) which rely on Matplotlib underneath.

Installation and Setup:

- Typical installation via pip:
- pip install matplotlib

(Matplotlib)

- In code you import like:
- import matplotlib.pyplot as plt

This gives the pyplot interface.

- Once installed and imported, you're ready to create plots. You may also use the object-oriented API for more advanced control (e.g., fig, ax = plt.subplots()).

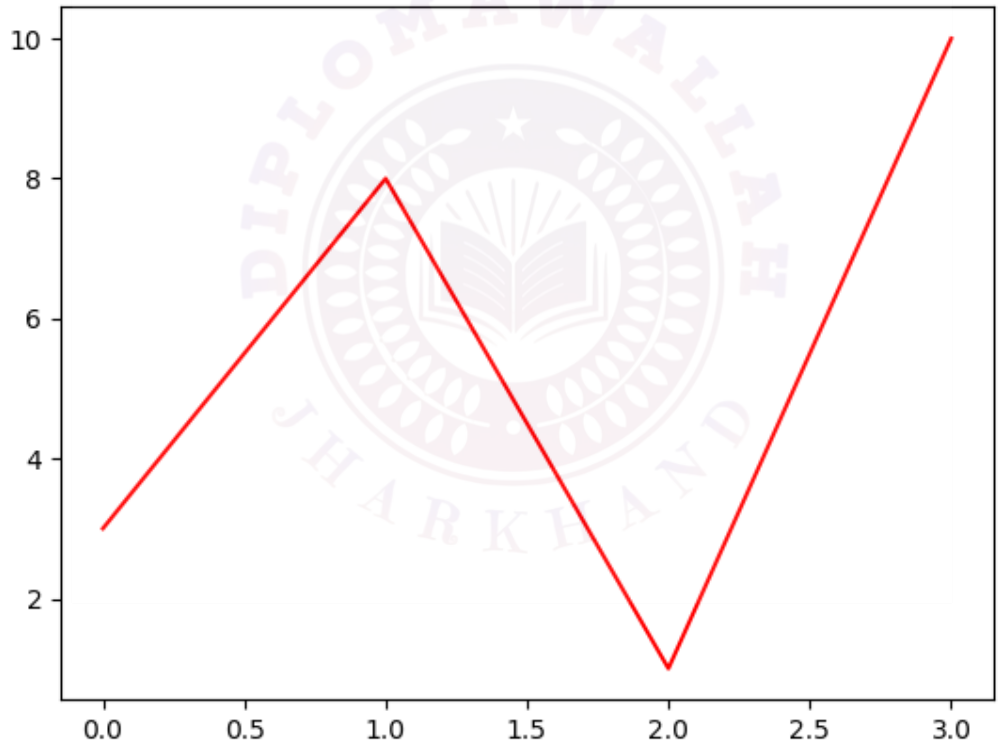
Key Theoretical Concepts:

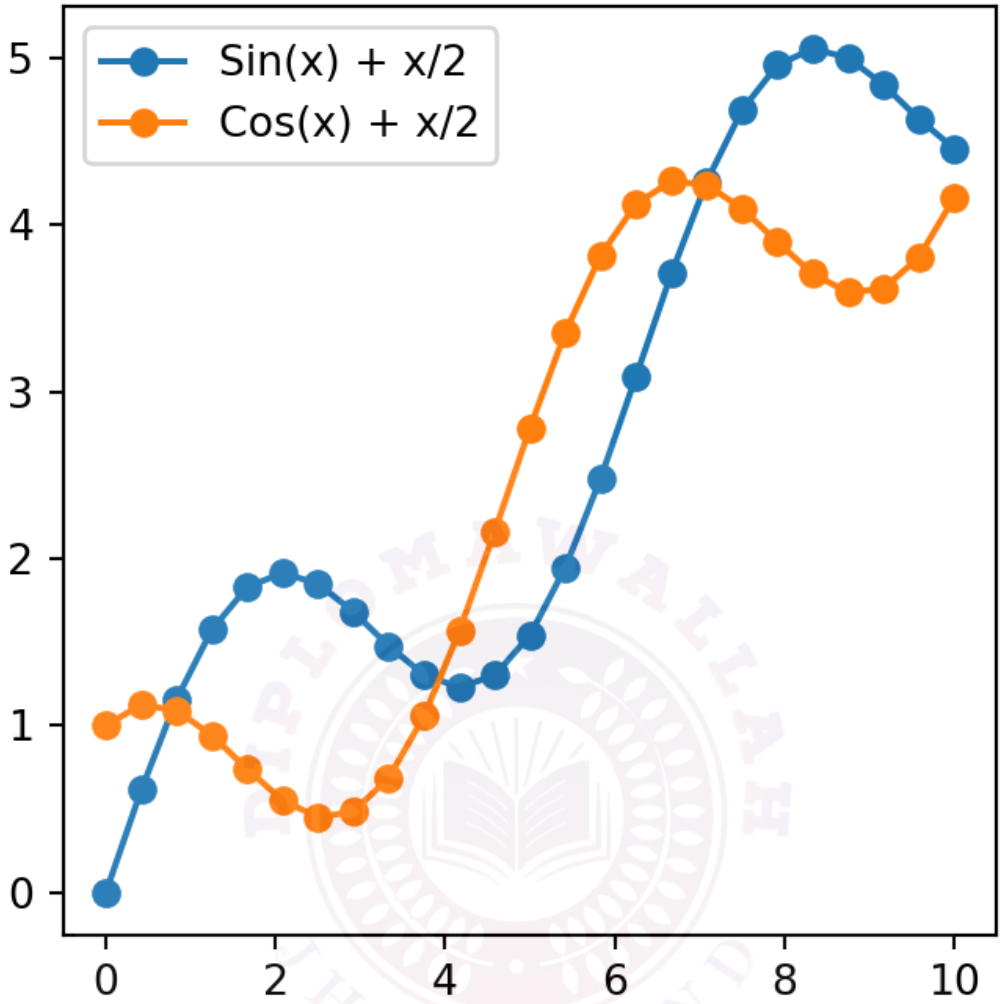
- **Figure**: the top-level container (canvas) for drawing plots.
- **Axes**: region inside a figure where data is plotted; each axes has x-axis and y-axis (and possibly more). (Matplotlib)
- **Artist**: any element you see on the plot (lines, text, markers).
- Understanding these components helps you reason about how plotting mechanisms work (which is important in a theory exam).

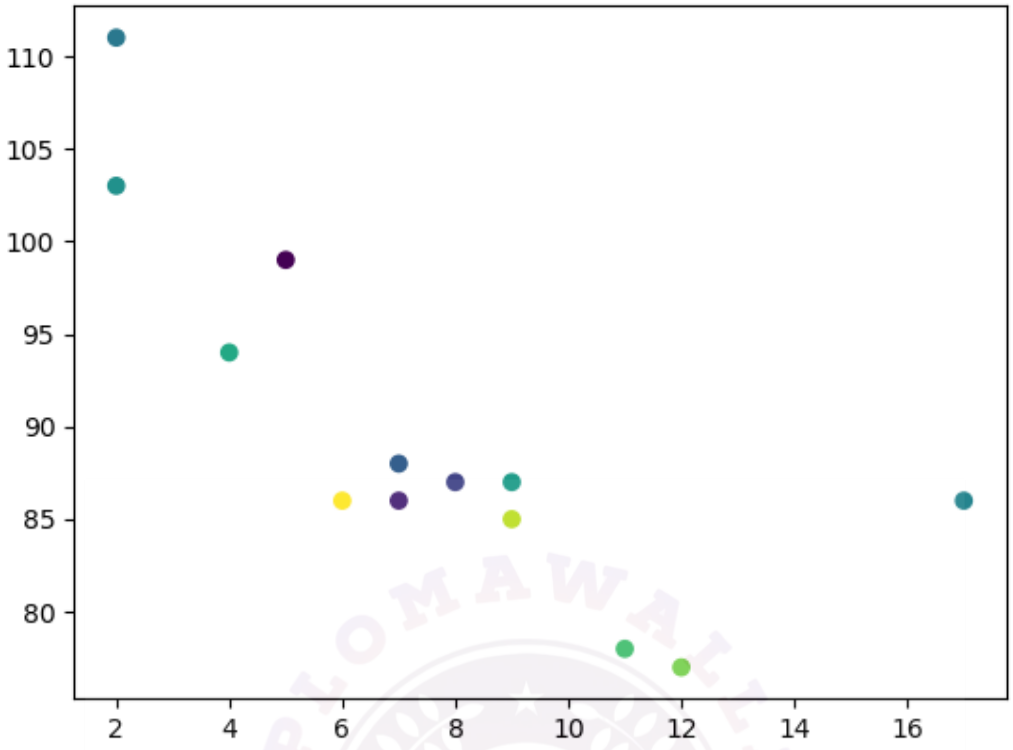
Exam Tips for Theory:

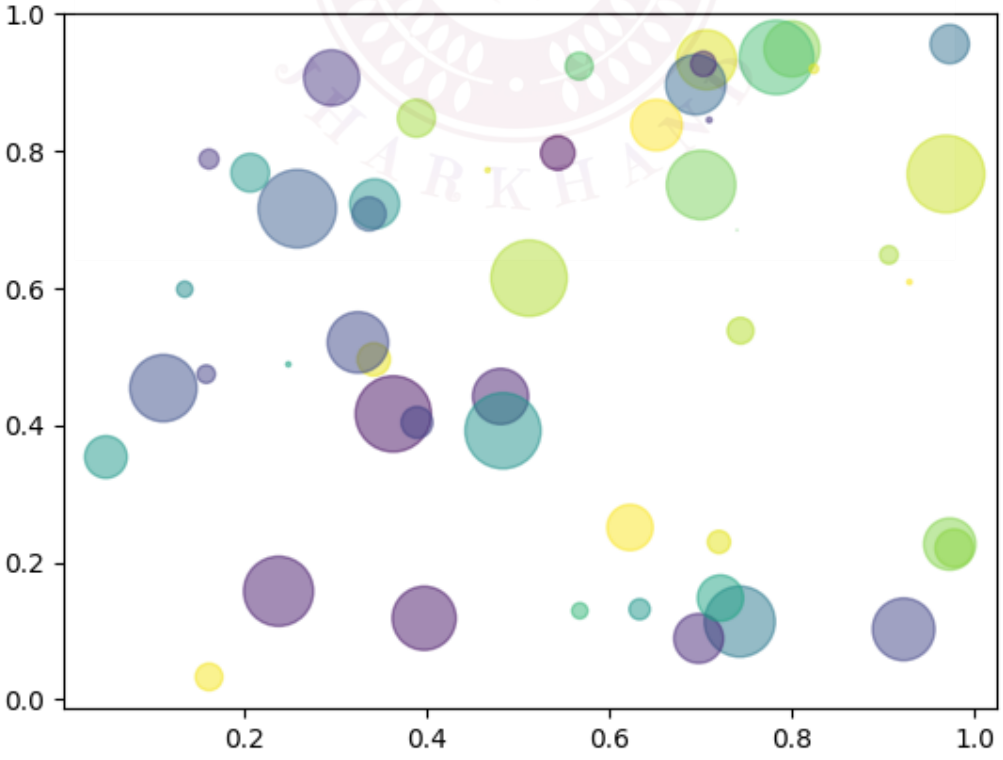
- Define Matplotlib clearly and mention its importance.
- Mention use-cases: static visuals, interactive visuals, scientific plots.
- Explain the installation and import steps.
- Given a figure, you should be able to label parts: figure, axes, title, legend, etc (use the diagrams above).
- You might be asked: *"What is the difference between Figure and Axes in Matplotlib?"* or *"List the major components of a Matplotlib plot."*

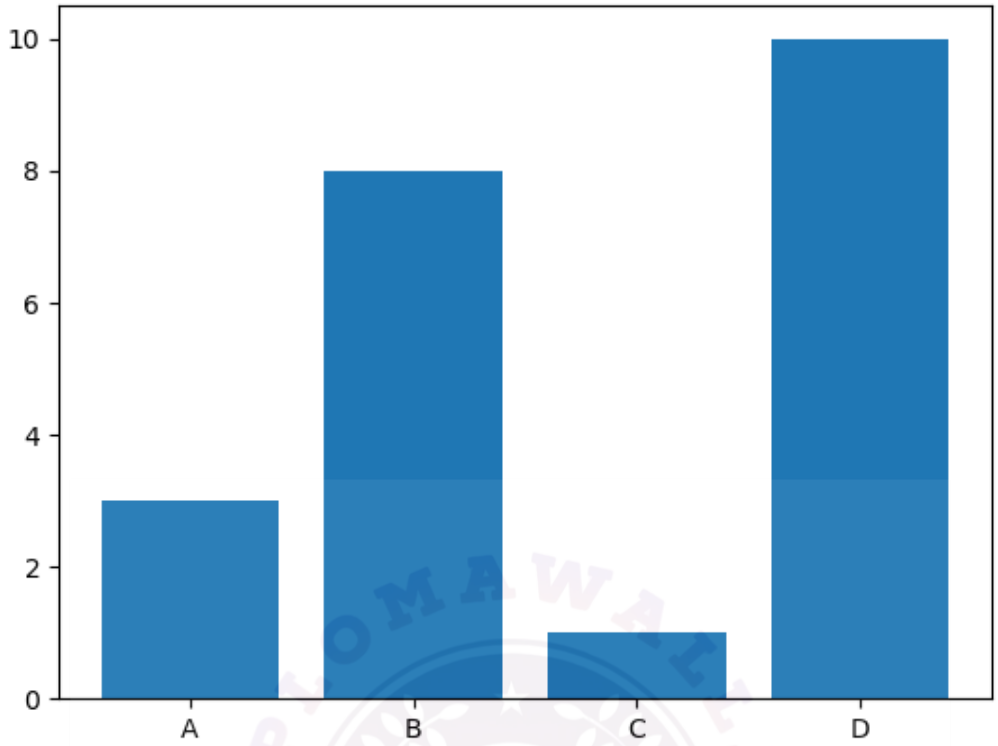
5.2 Basic Plotting with Matplotlib: Line Plot, Scatter Plot, Bar Chart, Histogram; Adding Titles, Labels & Legends

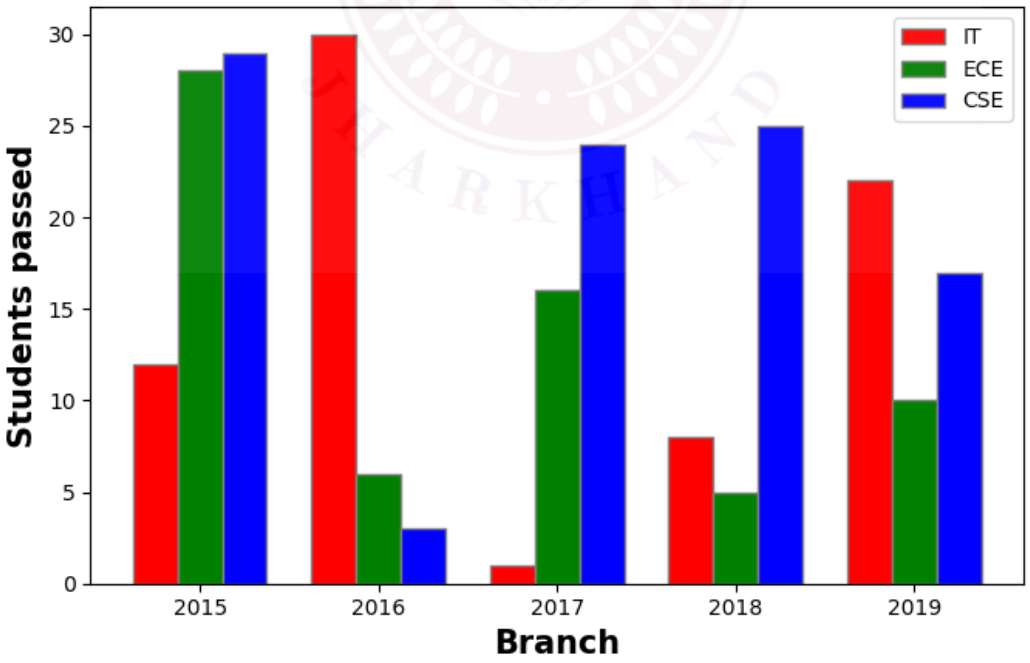


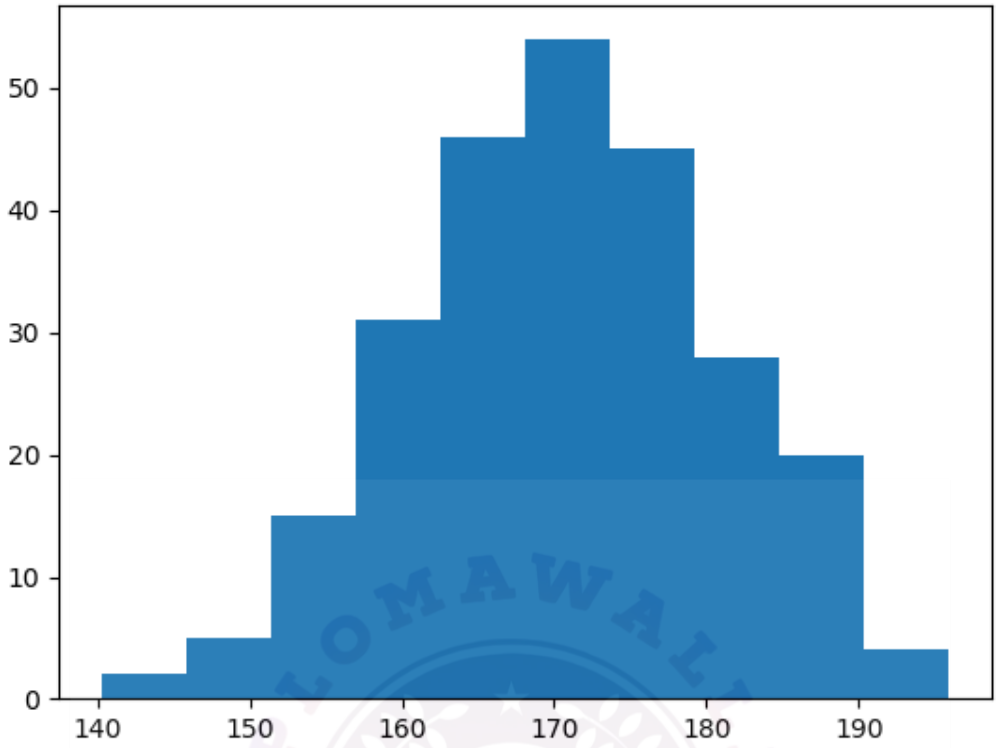


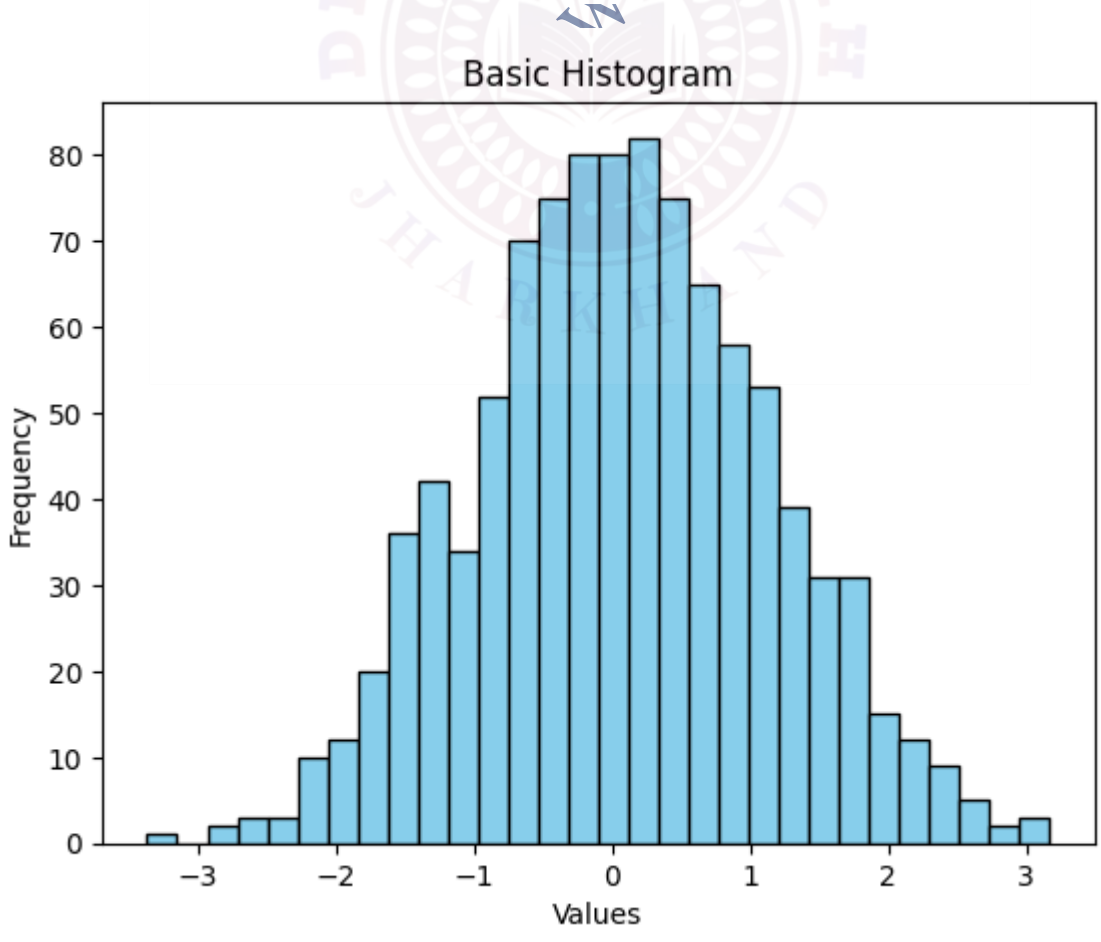












Plot Types – Theory & Use:

- **Line Plot**: Used when you have continuous data (often time series) and you want to show how a value changes.
- **Scatter Plot**: Shows relationship between two numeric variables (e.g., one on x-axis, one on y); useful to show correlation, clusters, outliers.
- **Bar Chart**: Good for comparing quantities across discrete categories (e.g., sales by region).
- **Histogram**: Used to show the *distribution* of a numeric variable by splitting into bins and showing frequency of each bin.

Adding Context – Titles / Labels / Legends:

- **Title**: Indicates the subject of the plot (e.g., "Monthly Sales Trend").
- **X-axis / Y-axis Labels**: Clearly state what each axis measures (with units if relevant).
- **Legend**: Necessary when multiple data series are plotted, so viewer can distinguish them.
  Without these, a plot may not be interpretable.

Theory Highlights for Exam:

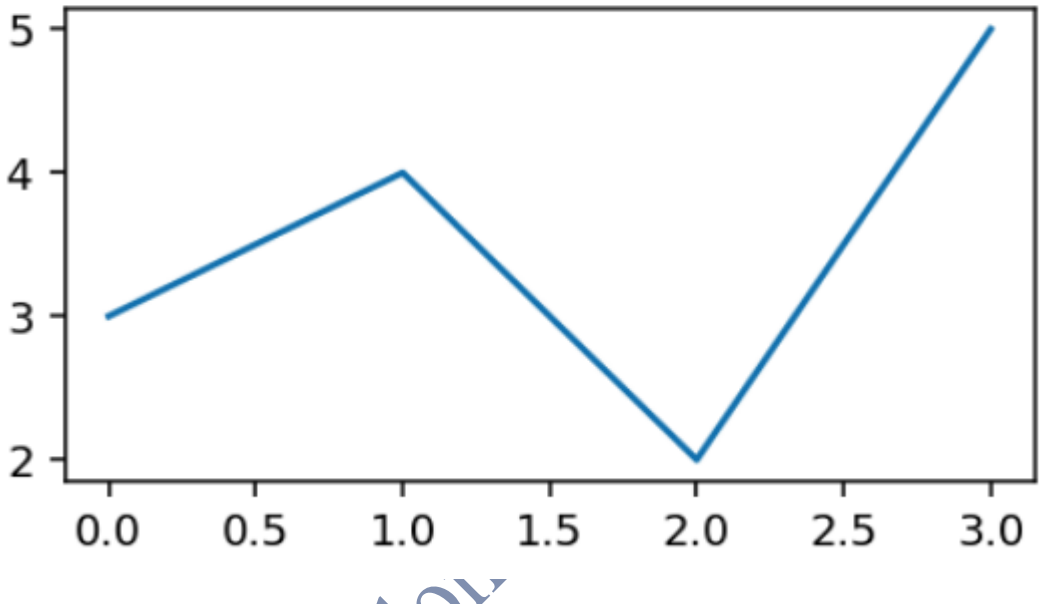
- Know when to use each plot type (purpose + data type). For example: *"Use a histogram when you want to examine the distribution of a continuous variable, not a bar chart."*
- Understand that adding labels and legends is part of making the plot *self-explanatory*—an important exam point.
- Recognise that plots alone are not enough — interpretation is key: e.g., a histogram skewed to the right implies most values are low but few are high.
- Understand difference between line plot and scatter: line shows trend, scatter focuses on relationship.
- Be able to write pseudo-code or mention the functions (though your focus is theory, mention knowledge of typical functions: plt.plot(), plt.scatter(), plt.bar(), plt.hist()).

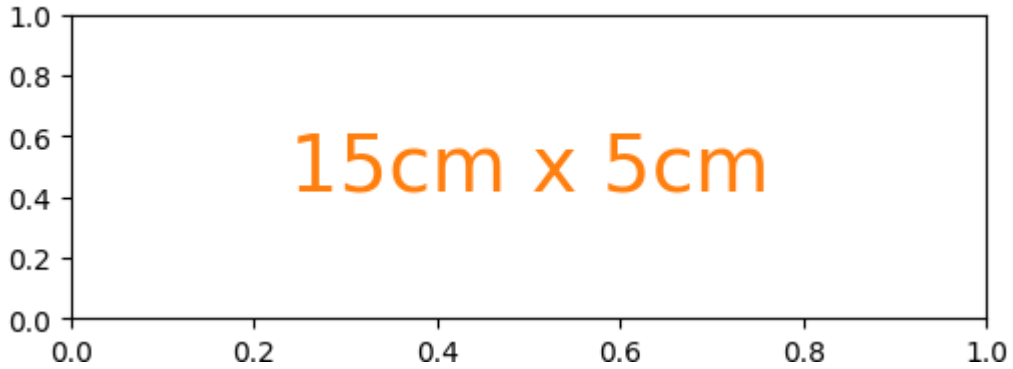
Example Explanation (Theory Format):

"In Matplotlib, a line plot is created via plt.plot(x, y). It is appropriate when you have an ordered x-axis (e.g., time). The axis labels and title should indicate what is being plotted. A scatter plot (plt.scatter(x, y)) is appropriate when exploring how two variables relate. A histogram (plt.hist(values, bins=…)) shows the distribution of a variable and may indicate skewness or spread."

5.3 Changing Figure Size & Aspect Ratio; Customising Axes (Limits, Ticks & Labels)

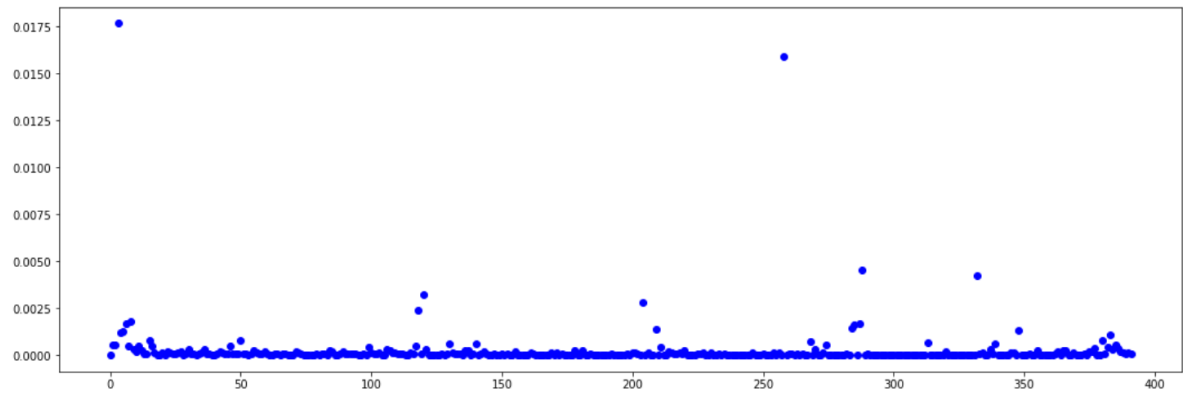
```
f = plt.figure()
f.set_figwidth(4)
f.set_figheight(2)
f.set_dpi(142)
_ = plt.plot([3,4,2,5])
```

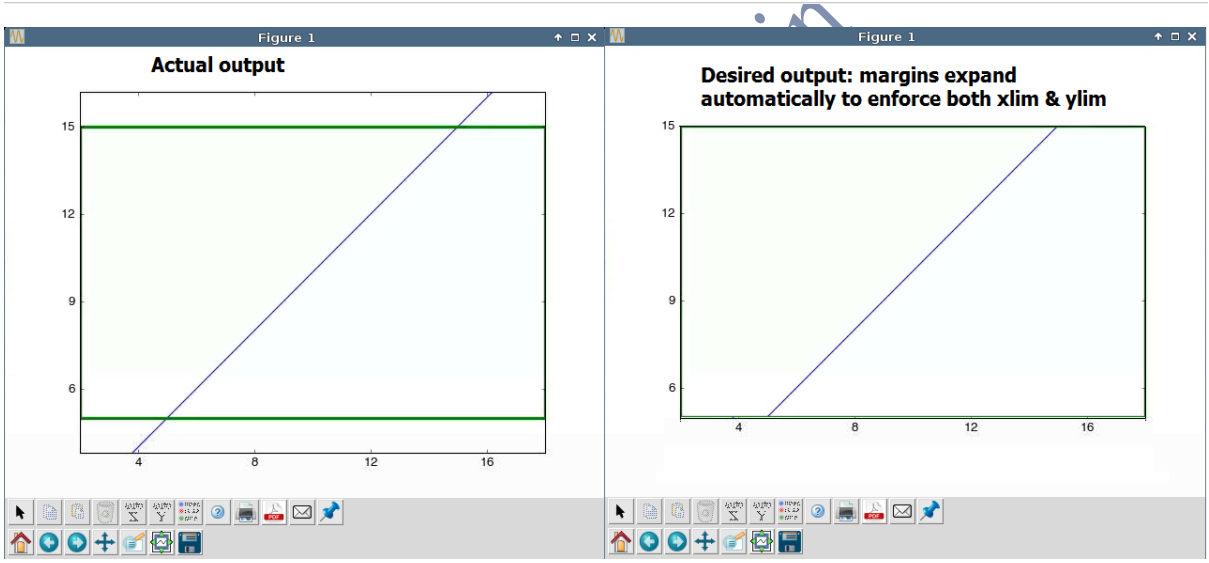


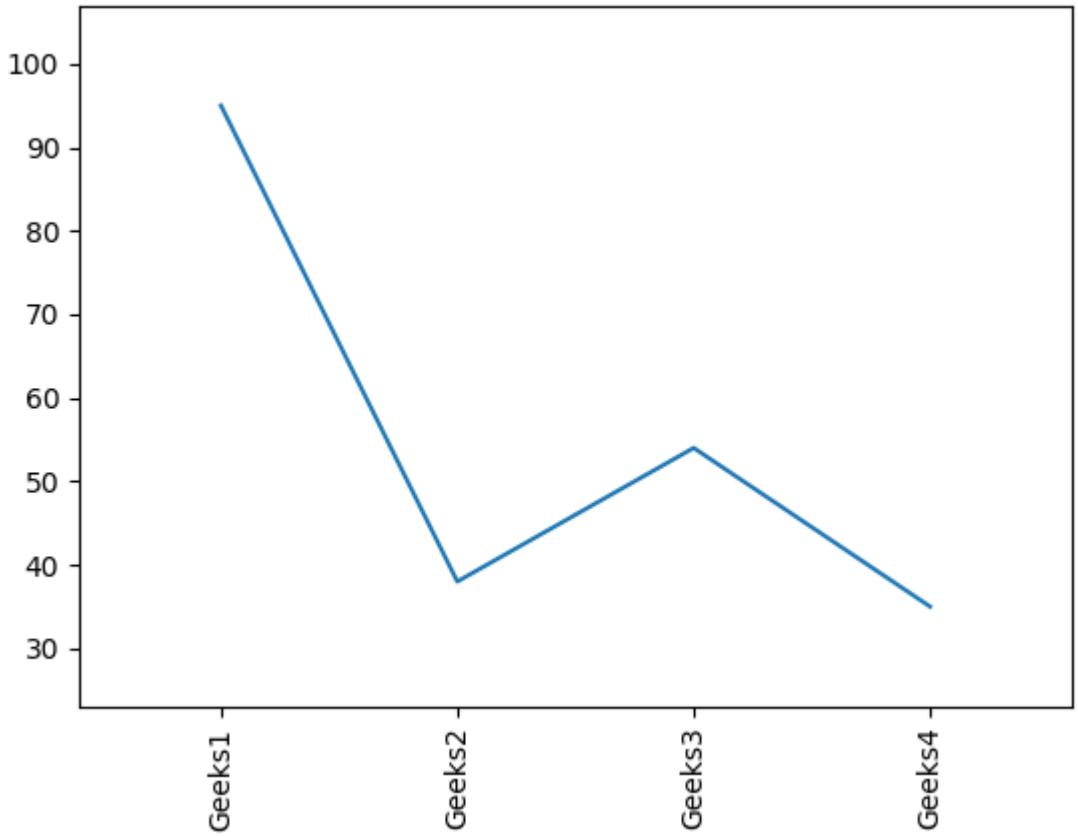


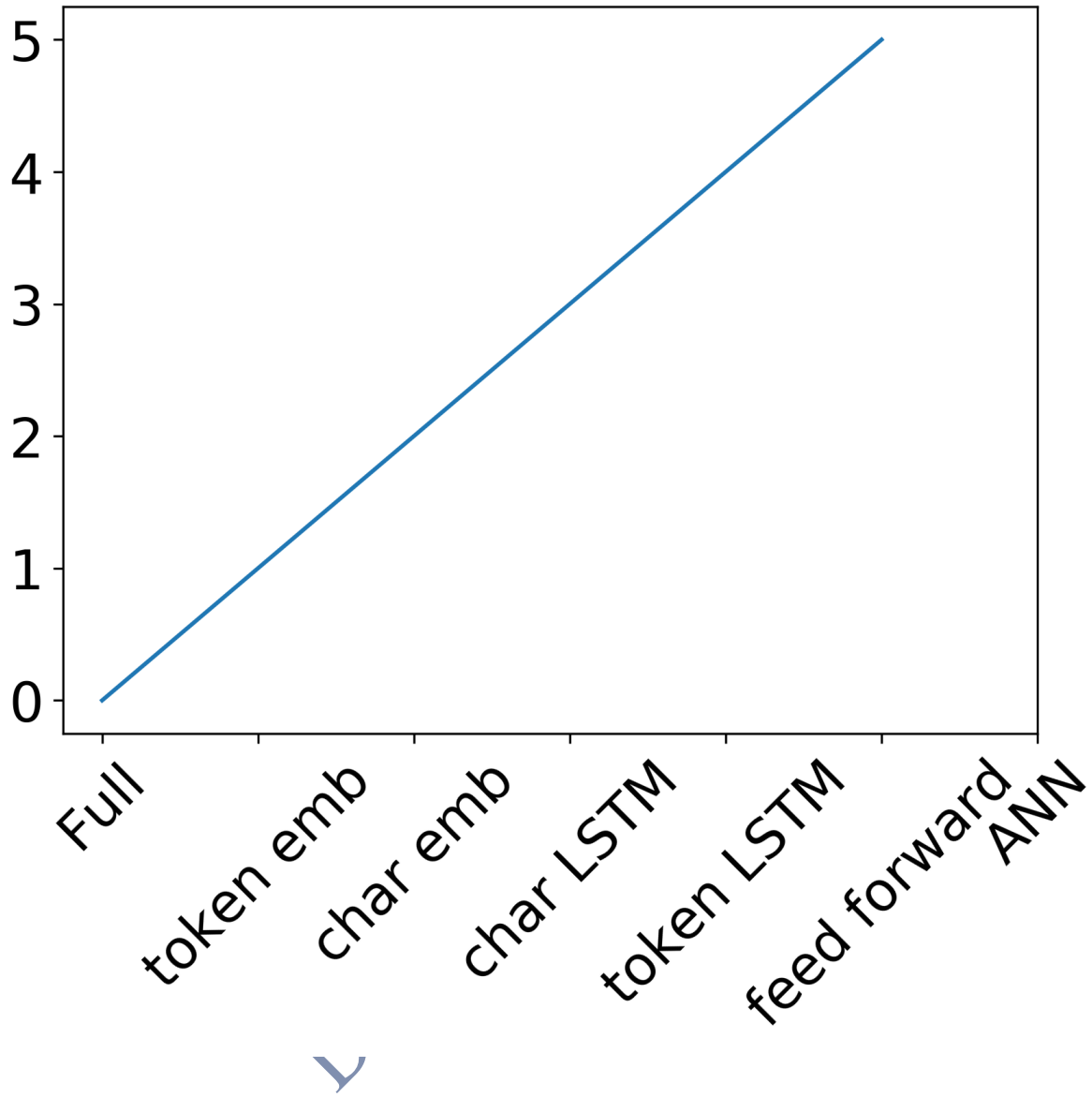
```
%matplotlib inline
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(18,6))
ax = fig.add_subplot(111)
#ax.autoscale(enable=False, axis='both')

plt.ylim=(0,0.0022)
plt.plot(list(range(elast_array.shape[0])), elast_array[:,1],'bo')
print(plt.ylim)
plt.show()
```

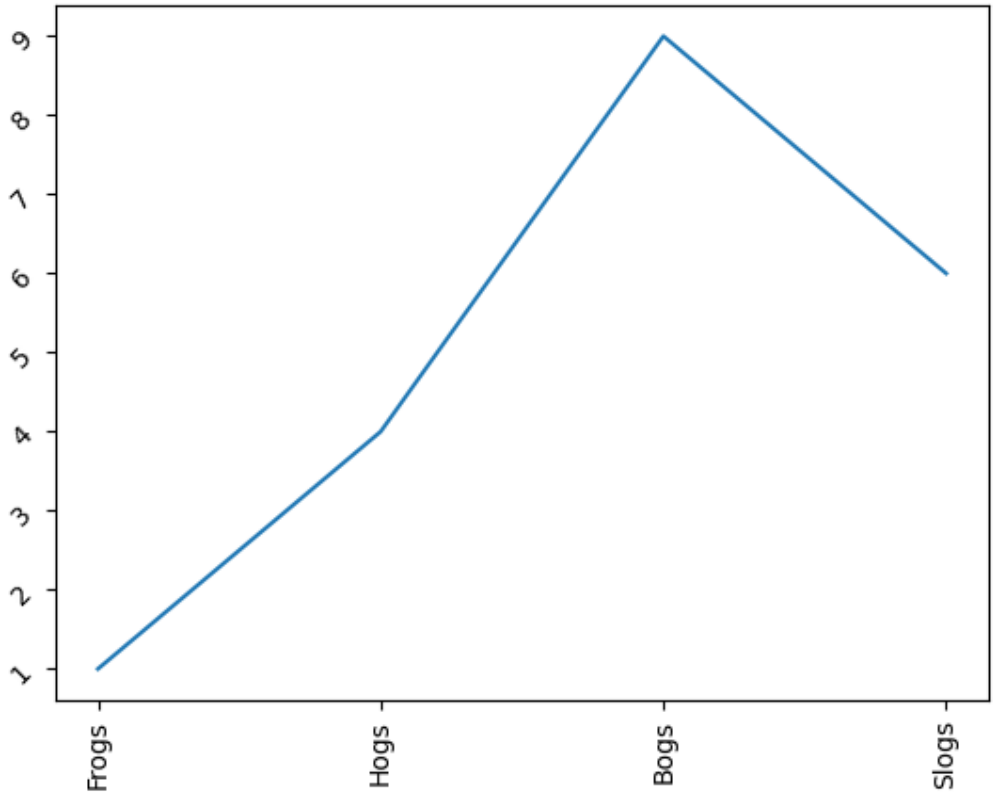
```
(0, 0.0022)
```








5
4
3
2
1
0
Full
token emb
char emb
char LSTM
token LSTM
feed forward ANN



Theory Behind Figure Size & Aspect Ratio:

- In visualisation, how a plot appears (its size and shape) affects readability and interpretation. A cramped plot may have overlapping labels, too many ticks, or axes difficult to read.
- Setting a proper figure size (e.g., figsize=(10,6)) ensures the chart is appropriately proportioned for the medium (screen, slide, print).
- Aspect ratio controls the relative scaling of axes; in some plots (e.g., maps or scatter where both axes have equal importance) you may want equal aspect ratio to avoid distortion.

Theory Behind Customising Axes:

- **Axis limits (xlim, ylim)**: Setting min/max values manually can focus on relevant data ranges or prevent misleading scaling (e.g., avoid starting y-axis at a non-zero value unless justified).
- **Ticks and Tick Labels**: You may need to choose tick interval, tick rotation (e.g., months on x-axis), label formatting (%, currency, thousands). Good tick configuration improves clarity.
- **Label Orientation & Readability**: If tick labels overlap, rotating them (e.g., 45°) improves readability.

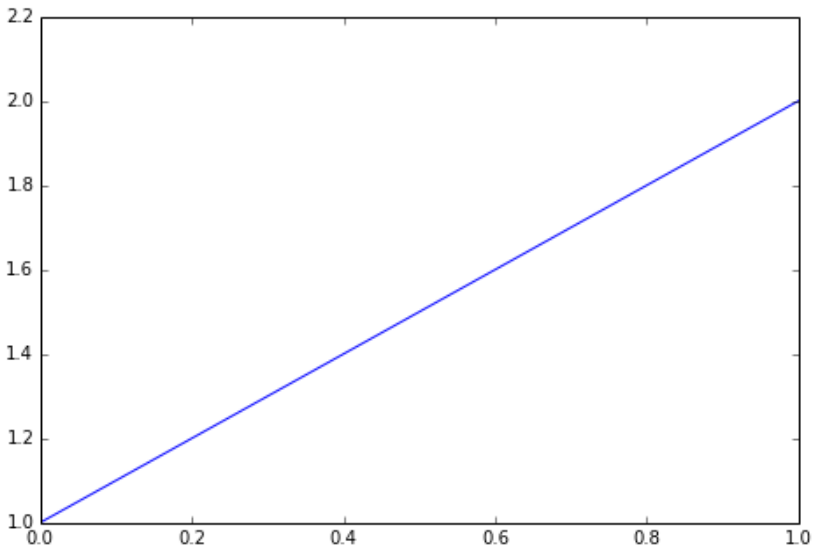
- These formatting tasks fall under good visualisation design theory: visuals should clearly convey data without distortion or clutter.

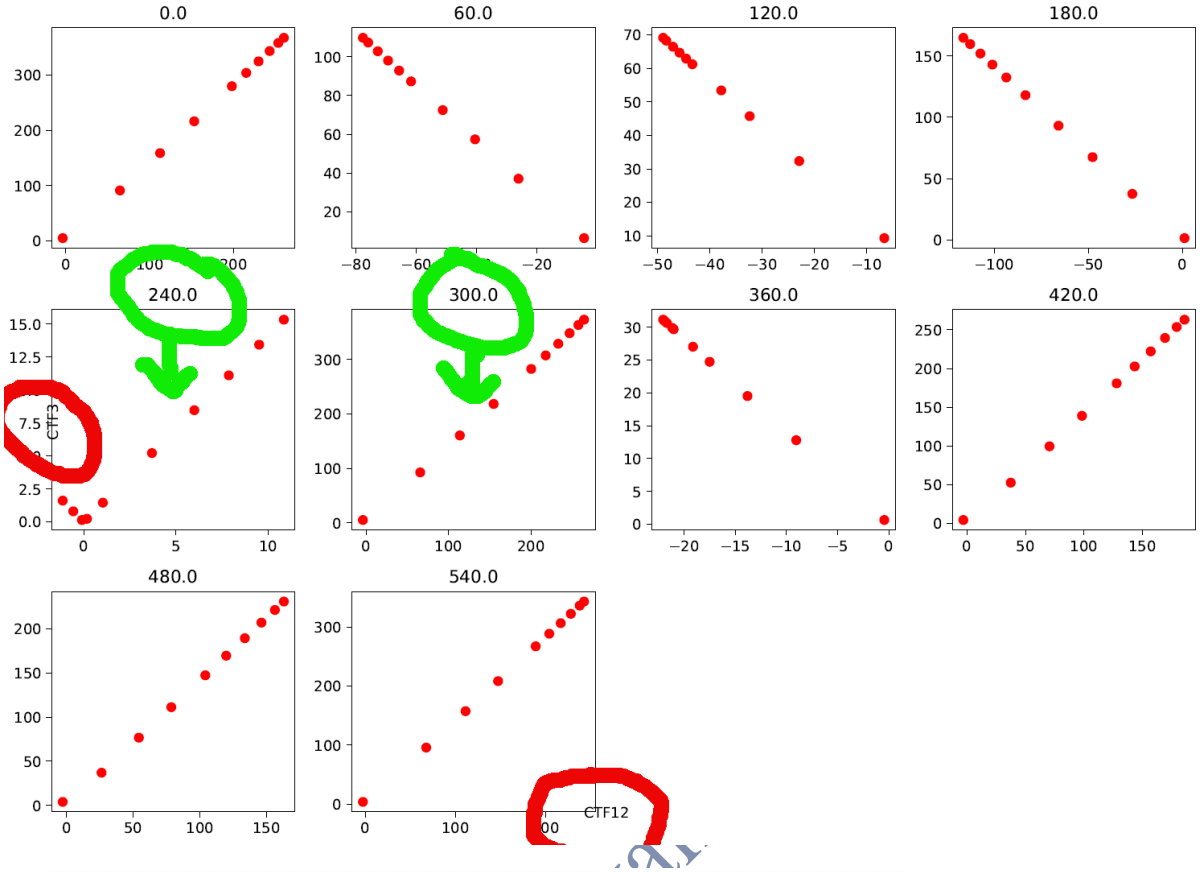
Theory Highlights for Exam:

- Be ready to explain why you might set figure size or axis limits.
- Mention risks: e.g., if you set axis limit poorly, you might hide important variation or exaggerate differences.
- Understand that axes customisation is not cosmetic only — it supports accurate communication of data.
- You may be asked: *"Explain the importance of figsize in Matplotlib and how it affects readability."* or *"Why might you customise tick labels on the axes?"*

5.4 Exporting & Saving Visualisations; Creating Interactive Visualisations

```
In [192]: import pandas as pd
          import matplotlib.pyplot as plt

          fig = plt.figure()
          ax = fig.add_axes([1,1,1,1])
          ax.plot([1,2])

          fig.savefig('test.png')
```






```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-69-3043dba66f44> in <module>()
     23 plt.plot(Min_Vals, 'b-')
     24 fig.set_size_inches(30.,18.)
---> 25 plt.savefig(dpi=300)
     26 plt.show()
     27

/opt/conda/lib/python3.5/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
```

```
# Escalando
plt.yticks([400,450,500,550])
plt.grid(True)
plt.legend()
plt.xlabel('Corriente en Linea, $A$')
plt.ylabel('Voltaje en los terminales, $V$')
plt.savefig('terminales_4.jpg', dpi=300)
plt.close()

plt.plot(v_t_retraso,a)
plt.xlabel('Corriente en Linea, $A$')
plt.ylabel('Voltaje en los terminales, $V$')
plt.savefig('terminales_5.jpg', dpi=300)
plt.close()
```

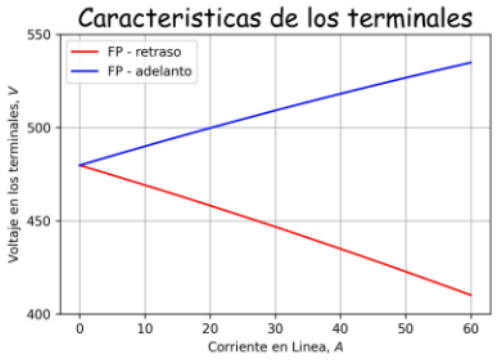


Figura 1

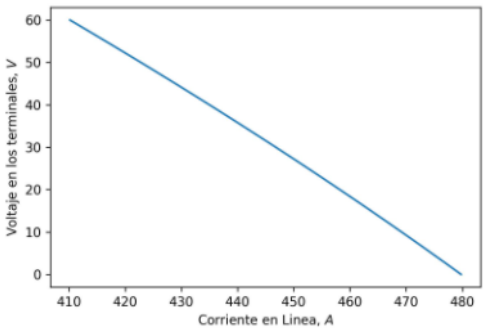
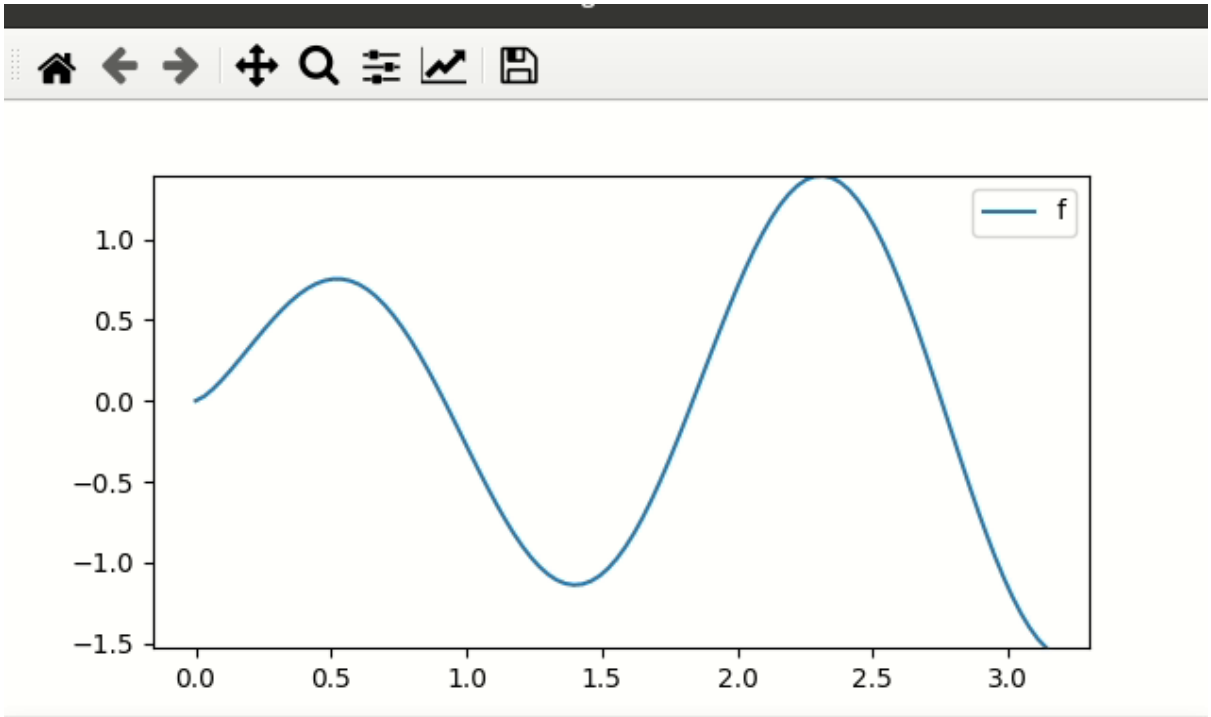


Figura 2


f
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

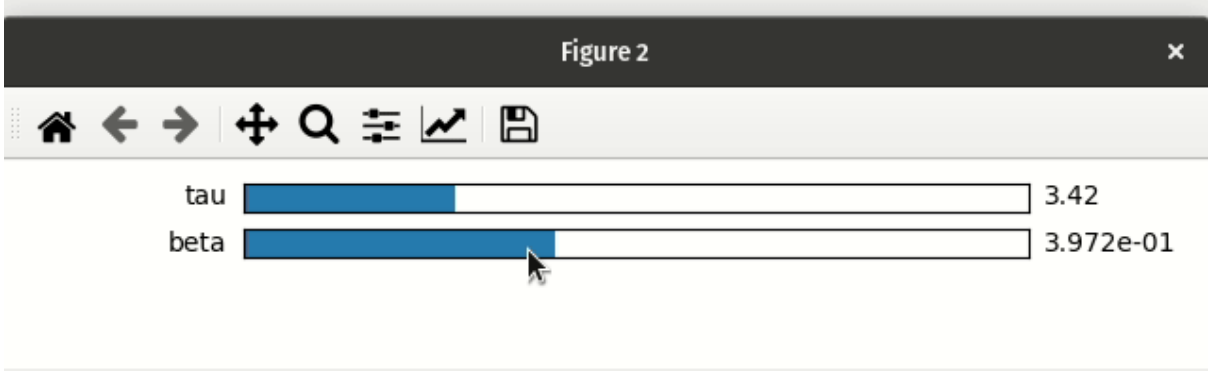
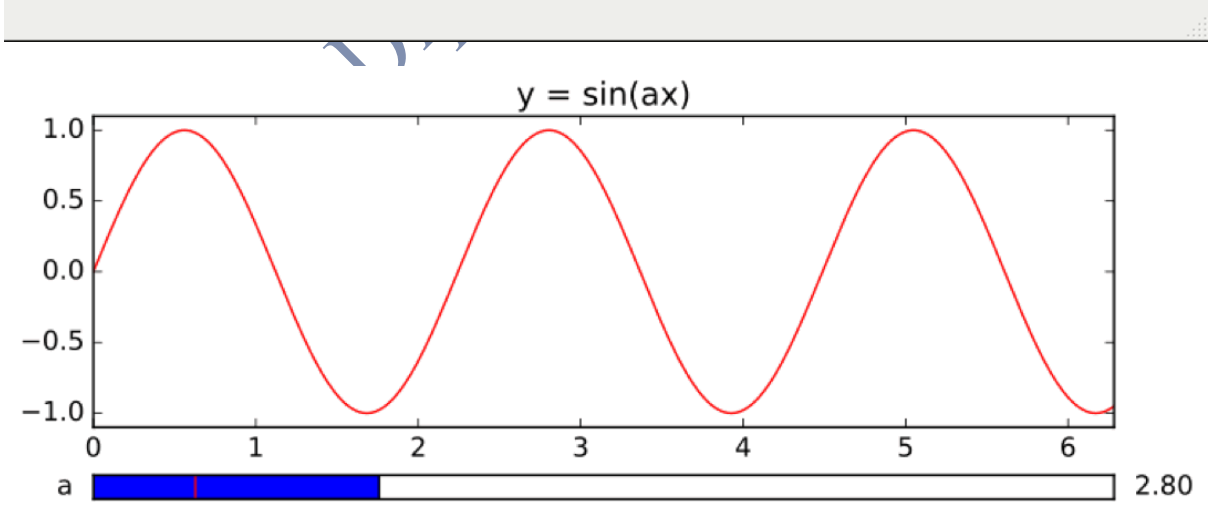
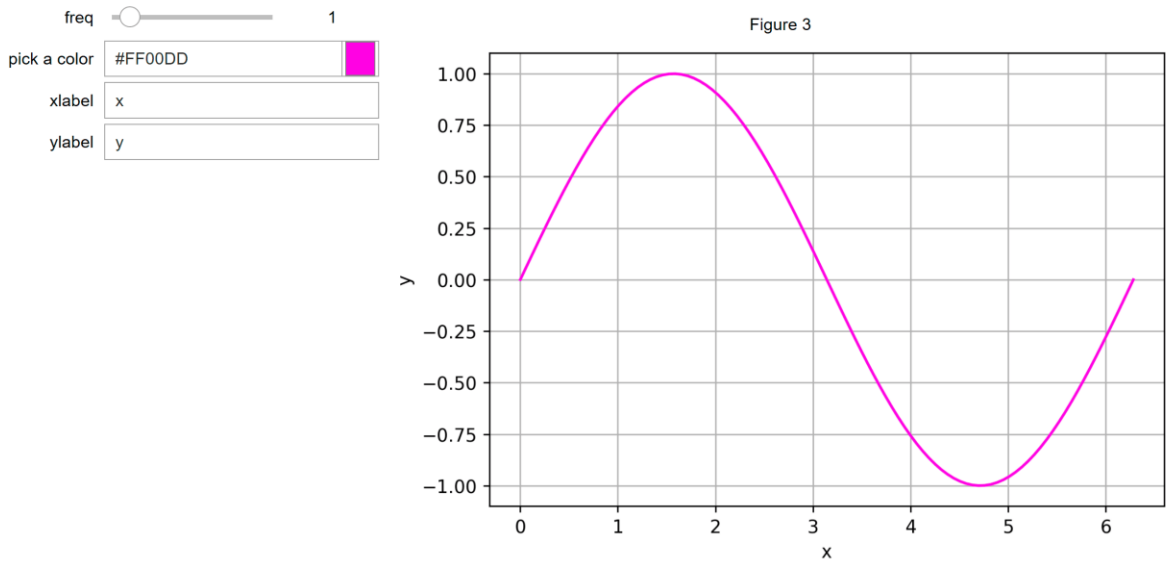
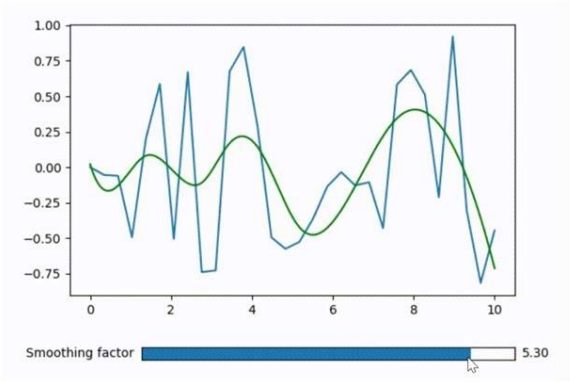


Figure 2
tau
3.42
beta
3.972e-01


y = sin(ax)
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6
a
2.80



finxter



Theory of Exporting / Saving Visualisations:

- Once a plot is created, you often need to save it for use in reports, presentations or web. Matplotlib provides savefig() for this.
- You must choose appropriate **format** (e.g., PNG for web, PDF/SVG for vector print) and **resolution** (dpi parameter) for quality output.
- Example: plt.savefig('chart.png', dpi=300, format='png', bbox_inches='tight').
- Exporting is part of the full visualisation workflow: Create → customise → export. In a theory exam, you may need to explain these steps.

Theory of Interactive Visualisations:

- Beyond static plots, Matplotlib supports interactive elements via matplotlib.widgets, such as sliders, buttons. These allow the user to manipulate a parameter and see changes live.
- Interactive visualisations enhance exploration: the user can adjust time window, filter data categories, explore "what-if" scenarios.
- In exam questions you may be asked: *"What benefits do interactive visualisations give compared to static ones?"* or *"Describe how a slider widget can be used in Matplotlib."*

Theory Highlights for Exam:

- Know the difference between **raster** and **vector** image formats (PNG vs SVG/PDF) and when to use each.
- Be able to explain dpi and how resolution affects quality.
- Explain what an interactive widget is and an example use-case (e.g., a slider controlling data series).
- You may not need full code, but steps and reasoning: select plot → customise → call savefig(); to add interactivity, import widget, define callback, connect to plot.
- Emphasise visualisation workflow: data → plot → customise → share/export → explore (interactive).

Summary for Theory Exam

- Make sure you **define** all major terms: Matplotlib, Figure, Axes, Artist, line plot, scatter, bar chart, histogram, figure size, axis limits, export formats, interactive widgets.
- Provide **explanation** of *why* each concept matters in visualization (not just *how*).
- Use the web figures to illustrate concepts (e.g., anatomy of a figure, types of plots, axis ticks).
- Be ready for question formats:
    - "Explain …" (definition + reasoning)
    - "What is the importance of …?"
    - "Compare …" (e.g., histogram vs bar chart)

    - "Describe the steps to …" (e.g., save a figure, set figure size, add interactivity)
- Use clear headings and bullet points in your answers to align with exam style (and improve readability).

Diploma Wallah

Made with 💗 by Sangam