Matplotlib Practice Problems$Easy ß Hard)$

Generated for practice

December 1, 2025

Contents

Instructions

- Each exercise describes a plotting task using `matplotlib` (and optionally `numpy`/`pandas`).
- You can write a short script or notebook cell to solve each exercise. Example skeletons are intentionally omitted so you can practice.
- Where numeric data is required, sample arrays are provided; you may also generate your own data where appropriate.

1 Easy (1–10)

1

Plot the list `a = [1,2,3,4,5]` against `b = [1,4,9,16,25]` as a line plot. Label axes and add a title.

2

Create a bar chart showing the counts: `counts = [5, 3, 9, 2]` with labels `['A','B','C','D']`. Add value labels on top of each bar.

3

Make a scatter plot using `x = np.linspace(0,10,50)` and `y = np.sin(x)`. Use circle markers and a dashed line connecting points.

4

Plot a pie chart for market share: `shares=[40,25,20,15]` with labels `['P','Q','R','S']`. Show percentages and explode the largest slice slightly.

5

Create a histogram of 1000 random numbers sampled from a normal distribution (mean=0, std=1). Use 30 bins and add a grid.

6

Draw a simple stem plot for `x = range(1,11)` and `y = x**2`. Add axis labels and adjust marker style.

7

Save the figure from exercise 1 as `lineplot.png` with DPI=150.

8

Create a horizontal bar chart (`barh`) for languages and their scores: `langs=['Py','JS','C++'], scores=[30,20,10]`. Sort bars by score descending.

9

Make a plot that contains text annotation: plot `y = x**2` for x in 0..5 and annotate the point x=3 with the text "(3,9)" and an arrow.

10

Change a plot style using `plt.style.use('ggplot')` and reproduce exercise 3 under that style. Note the visual differences.

2 Intermediate (11–25)

11

Create a grouped bar chart comparing two years for 6 categories. Use different colors and include a legend and value labels.

12

Generate a line plot with two y-axes (left and right) using `twinx()`. Example: temperature vs. day (left), precipitation vs. day (right).

13

Use subplots (2 rows x 2 cols) to show: 1) line plot, 2) scatter, 3) histogram, 4) pie chart. Share x-axis labeling where appropriate.

14

Plot a heatmap of the 10x10 matrix `A` where `A[i,j] = i*j`. Add a colorbar and set an appropriate colormap and aspect ratio.

15

Create a violin plot or boxplot for three different samples (e.g., drawn from different normal distributions). Label each violin/box.

16

Given the dataframe (create with pandas) of monthly sales (12 months) for 3 products, plot a stacked area chart showing cumulative sales over time.

17

Make a scatter plot where the point sizes reflect a third variable and the color reflects a fourth variable. Include a colorbar and legend for sizes.

18

Create custom ticks: use x values 0..100 but show tick labels every 10 units in a rotated format. Format tick labels to include a percentage sign.

19

Using `matplotlib.animation.FuncAnimation`, create a simple animation showing a point moving along `y=sin(x)`. Save as an MP4 or GIF.

20

Plot a polar chart: draw `r = 1 + 0.5*cos(5*theta)` for theta in 0..2pi.

21

Create a multi-panel figure using `GridSpec` where one tall column contains a heatmap and a narrow column contains a colorbar and a small histogram.

22

Annotate multiple points on a scatter plot with different styles and demonstrate how to use `bbox` and opacity to make annotations readable.

23

Plot a contour plot of function (f(x,y)=$\sin(x)\cos(y)$)$overagrid.Addcontourlabels$(`clabel`)$andcolorshading$(

24

Create a logarithmic plot: plot `y = 10**(0.1*x)` and use a log scale for y-axis. Show minor and major ticks appropriately.

25

Demonstrate how to use `plt.style.available`: produce the same plot under 3 different styles side-by-side to compare aesthetics.

3 Hard (26–35)

26

Create an interactive-like plot in a notebook: add widgets (e.g., with `ipywidgets`) to control a parameter (frequency) of a plotted sine wave and update the plot inline.

27

Implement a small dashboard layout: left column with a list of checkboxes to toggle lines, right area shows an updating plot. (You may use Jupyter + ipywidgets).

28

Given a large time series (100k points), plot a downsampled representation efficiently (e.g., use decimation or aggregation) such that rendering stays responsive.

29

Create a custom colormap by blending three colors and apply it to an image/heatmap. Show a colorbar with custom ticks and labels.

30

Produce a publication-quality figure: tight typography, embedded fonts, LaTeX-rendered math in labels, and save as a high-resolution PDF suitable for inclusion in papers.

31

Write a function that accepts a pandas DataFrame and automatically creates an EDA dashboard: histograms for numeric columns, a correlation heatmap, and top-10 value bar charts.

32

Create a 3D surface plot of $(\mathrm{z} = \sin(\sqrt{x^2 + y^2})/(\sqrt{x^2 + y^2} + 1e{-}6))with lighting{-}likeshading(use$

33

Implement a custom annotation system that places labels for peaks in a line series without overlapping (you can use heuristic repulsion or adjust offsets programmatically).

34

Write code to export a Matplotlib figure to both SVG and PDF, and programmatically modify the SVG (e.g., change a text label color by editing XML) before saving.

35

Advanced composition: combine Matplotlib with Pillow to annotate a plotted image with textual badges and save the final raster output. Show how DPI affects the quality of annotations.

4 Optional Extras (36–40)

36

Create a ternary plot (3-component composition) using Matplotlib primitives (no external ternary library). Show how to convert (a,b,c) to 2D coordinates.

37

Recreate a complex multi-layered visualization (e.g., population pyramid) and make it horizontally mirrored with annotations and percentage labels.

38

Visualize missing data in a DataFrame as a heatmap (missing vs present). Provide an ordering to rows/columns to highlight patterns (clustering optional).

39

Benchmark rendering time for different numbers of scatter points (10k, 100k, 1M) and summarize strategies to speed up plotting (Rasterization, alpha blending, downsampling).

40

Create a reproducible plotting template: a Python module that exposes functions to create standardized figure margins, fonts, colors, and automatically applies to all figures in a project.

Notes for instructors / self-check:

- Each exercise may include small sub-tasks (e.g., annotate, save figure, change style) — encourage adding 1–2 extras for deeper practice.