

Matplotlib draws charts. Most sections start with `import matplotlib.pyplot as plt` and `fig, ax = plt.subplots()`. They use `x = [1, 2, 3, 4]` (four days) and `y = [20, 23, 21, 25]` (the temperature on each day). On your own computer, install it with `pip install matplotlib`.

FIGURE AND AXES

The figure (fig) is the whole picture. An axes (ax) is one chart inside it, with its own x axis and y axis. You draw on ax.

```
import matplotlib.pyplot as plt
plt is the usual short name for pyplot
```

```
fig, ax = plt.subplots()
one figure with one chart in it
```

```
fig.get_size_inches().tolist()
→ [6.4, 4.8]
the default size: 6.4 by 4.8 inches
```

```
fig, ax = plt.subplots(figsize=(8, 4))
a figure 8 inches wide and 4 inches high
```

KINDS OF CHART

```
ax.plot(x, y)
len(ax.lines)
→ 1
a line chart: x values first, then y values
```

```
ax.scatter(heights, weights)
a scatter chart: one dot for each (height, weight) pair
```

```
ax.bar(["Mon", "Tue", "Wed"], [3, 5, 2])
a bar chart; ax.barh() draws the bars sideways
```

```
counts, _, _ = ax.hist(ages, bins=3)
counts.tolist()
→ [2.0, 3.0, 2.0]
a histogram: how many ages fall in each of 3 ranges
```

```
ax.pie([30, 70], labels=["tea",
"coffee"])
a pie chart: the parts of a whole
```

LABELS AND LEGEND

```
ax.set_title("Temperature")
ax.get_title()
→ 'Temperature'
```

```
ax.set_xlabel("Day")
ax.set_ylabel("°C")
labels the axes: always say what each axis shows
```

```
ax.plot(x, y, label="Oslo")
ax.plot(x, [9, 8, 11, 10], label="Rome")
first give each line a label
```

```
leg = ax.legend()
[t.get_text() for t in leg.get_texts()]
→ ['Oslo', 'Rome']
then legend() shows a key with the line names
```

```
ax.text(2, 23, "warmest")
writes text at the point x=2, y=23
```

COLOURS AND STYLES

```
line, = ax.plot(x, y, color="red",
linestyle="--", marker="o")
line.get_linestyle()
→ '--'
a red dashed line with a dot at each point
```

```
line, = ax.plot(x, y, "g:s")
line.get_marker()
→ 's'
short form: g is green, : is dotted, s is square markers
```

```
ax.plot(x, y, linewidth=3, alpha=0.5)
a thicker line that is half see-through
```

```
ax.bar(x, y, color="#D9412B")
a colour given as a hex code
```

```
plt.style.use("ggplot")
applies a ready-made style to every chart made after this line
```

SEVERAL CHARTS IN ONE FIGURE

```
fig, axs = plt.subplots(1, 2)
axs.shape
→ (2,)
1 row of 2 charts, side by side
```

```
axs[0].plot(x, y)
axs[1].bar(["Ada", "Hedy"], [90, 96])
draw on each chart by its position
```

```
fig, axs = plt.subplots(2, 2)
axs.shape
→ (2, 2)
2 by 2 grid: axs[row, col], so axs[1, 0] is bottom left
```

```
fig.tight_layout()
moves the charts apart so their labels do not overlap
```

AXIS LIMITS AND TICKS

Ticks are the small marks along an axis, with their labels.

```
ax.set_xlim(0, 5)
the x axis goes from 0 to 5
```

```
ax.set_ylim(bottom=0)
the y axis starts at 0
```

```
ax.set_xticks([1, 2, 3, 4])
ax.get_xticks().tolist()
→ [1, 2, 3, 4]
the positions of the tick marks on the x axis
```

```
ax.set_xticks([1, 2, 3, 4], ["Mon",
"Tue", "Wed", "Thu"])
tick marks with your own labels
```

```
ax.grid(True)
draws light grid lines across the chart
```

EXTRA MARKS

```
ax.axhline(22, color="grey")
a horizontal line at y=22, for example to show an average
```

```
bars = ax.bar(["Mon", "Tue"], [3, 5])
labels = ax.bar_label(bars)
[t.get_text() for t in labels]
→ ['3', '5']
writes each bar's value on top of the bar
```

```
ax.annotate("warmest", xy=(4, 25),
xytext=(2.5, 24.5),
arrowprops={"arrowstyle": "->"})
a label with an arrow that points at (4, 25)
```

```
ax.fill_between(x, y, alpha=0.3)
shades the area under the line
```

```
ax2 = ax.twinx()
len(fig.axes)
→ 2
a second y axis on the right, for a different unit
```

CHARTS FROM PANDAS

```
import pandas as pd
df = pd.DataFrame({
    "day": ["Mon", "Tue", "Wed", "Thu"],
    "temp": [20, 23, 21, 25]})
a small table with a day column and a temp column
```

```
df.plot(x="day", y="temp", ax=ax)
len(ax.lines)
→ 1
```

pandas draws on your ax and adds the legend for you

```
df["temp"].plot(kind="hist", ax=ax)
kind can be line, bar, barh, hist, box, pie or scatter
```

SAVE AND SHOW

```
fig.savefig("chart.png", dpi=200,
bbox_inches="tight")
dpi is dots per inch: higher is sharper.
bbox_inches="tight" trims the empty border
```

```
fig.savefig("chart.pdf")
the file type comes from the end of the file name (.pdf)
```

```
import io
buf = io.BytesIO()
fig.savefig(buf, format="png")
saves the image into memory instead of a file, for
example to send it from a web app
```

```
plt.show()
opens a window on your computer. On
pythonmastery.io, charts appear below the output
without plt.show()
```

```
plt.close(fig)
closes the figure to free memory when you make many
charts
```

WATCH OUT

```
plt.title("A")
ax.set_title("B")
plt.title() changes the current chart (the last one made),
which may not be the one you mean. Use ax.set_title()
```

```
plt.show()
plt.savefig("chart.png")
saves a blank image: show() ends the figure. Save first,
then show
```

```
ax.plot(x, y)
in a .py file, no window opens until you call plt.show() at
the end
```

```
fig.tight_layout()
if labels overlap, call this before you save
```

```
fig, axs = plt.subplots(1, 2)
axs.plot(x, y)
→ raises AttributeError
axs is an array of charts: draw on one of them, such as
axs[0].plot(x, y)
```

```
ax.plot([1, 2, 3], [5, 6])
→ raises ValueError
x and y need the same number of values
```

```
ax.set_title = "Sales"
ax.set_title("Sales")
→ raises TypeError
set_title is a method: call it; do not assign a string to it
```