Advanced Python3: Object-oriented programming, databases and visualisation

Alexandra Diem
Simula Research Laboratory

simula consulting

simula

L03: Advanced Visualisation

simula consulting

simula

Visualisation packages in Python

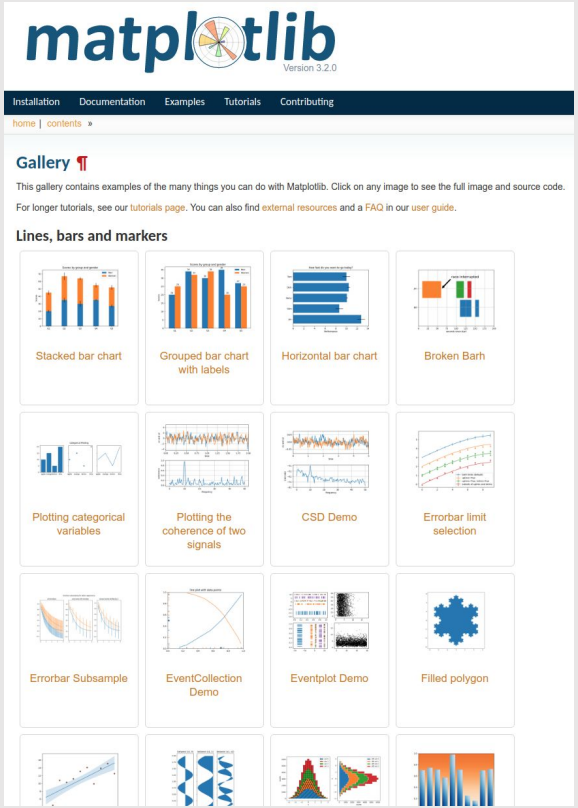
Matplotlib: Arguably the best known and most widely used visualisation package. It creates static figures and is best used for print graphs.

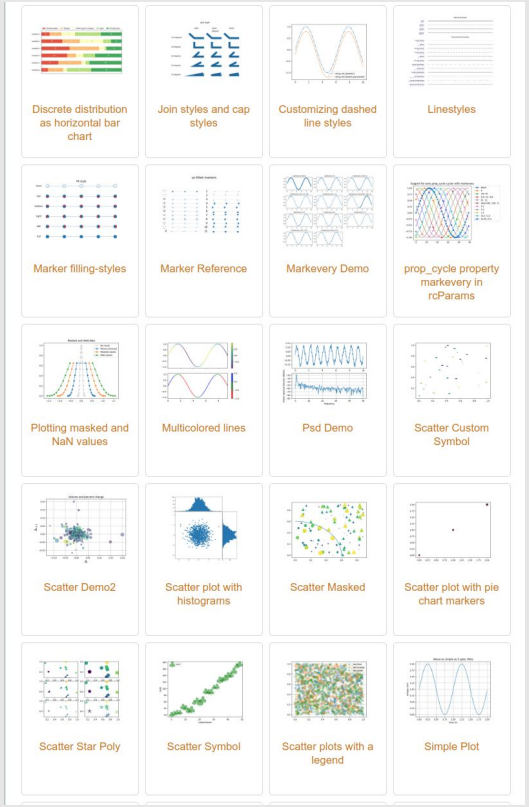
Pro:
- lightweight, but very versatile
- basic plots are extremely easy to implement
- simple customisation makes visually appealing, publication-ready figures
- Matplotlib Gallery (https://matplotlib.org/3.1.3/gallery/index.html) has example code for probably every kind of plot you'll ever need

Con:
- code to customise plots can get very complex quickly
- plots are static 2D raster graphics

> *"Matplotlib is mainly deployed for basic plotting."*
> *— Matplotlib*

Visualisation packages in Python





simula consulting

simula

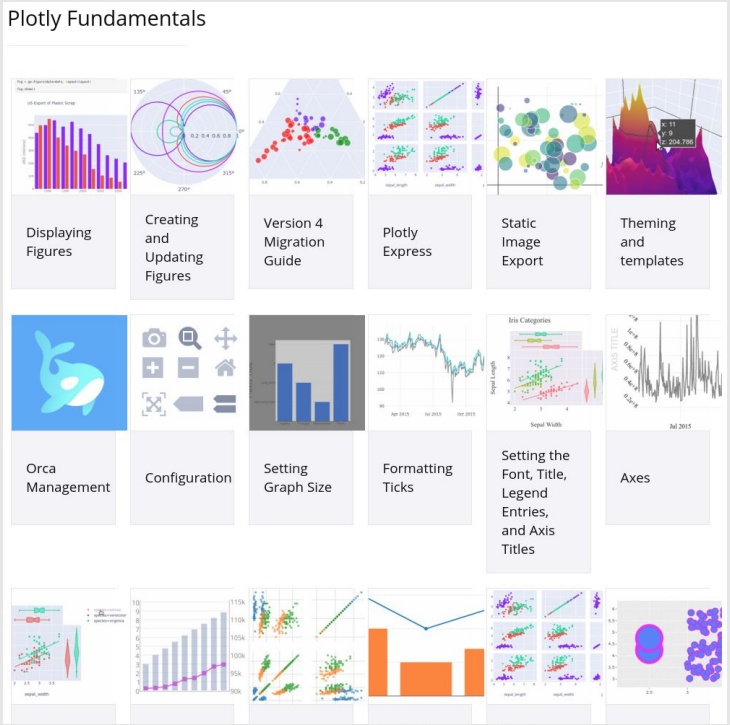
Visualisation packages in Python

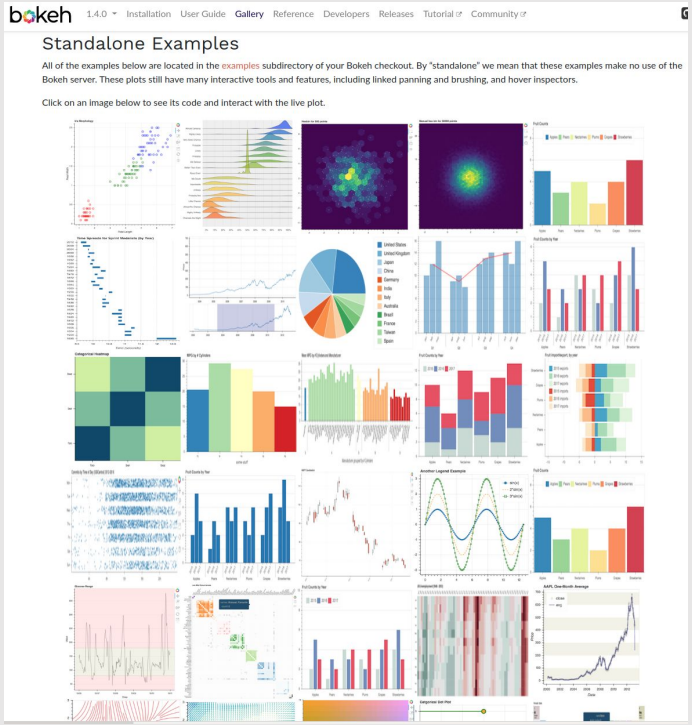
When analysing large datasets it becomes more desirable to be able to interact with your graphs directly, instead of having to plot a new raster graphic for every change. This is where Plotly and Bokeh come in.

Plotly: JavaScript-based library for interactive visualisation of datasets, integrates with pandas natively. While the Python interface to Plotly is open-source, their dashboard and browser-integration tools are not.

Bokeh: Browser-based library for interactive visualisation on dashboards. Less interactive than Plotly, but better at creating Matplotlib-style plots and completely open-source.

Visualisation packages in Python



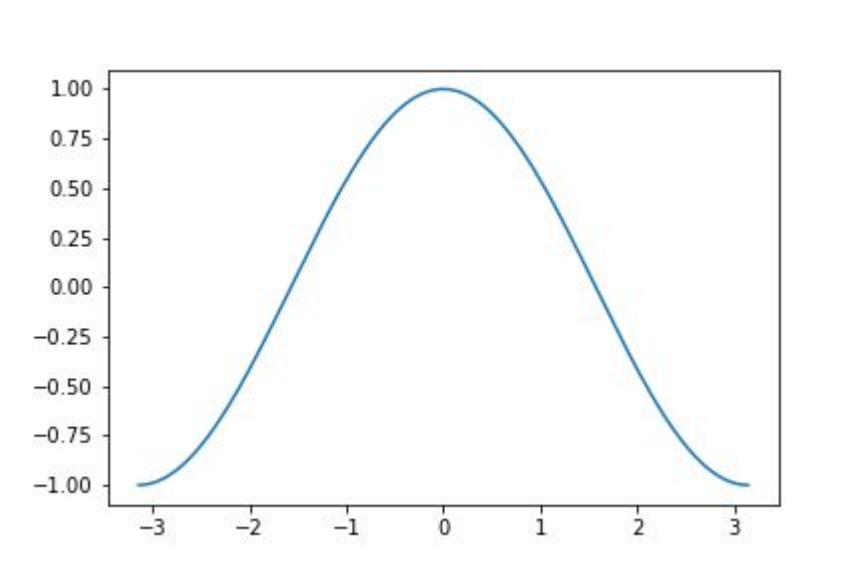


simula consulting

simula

Basic plotting with Matplotlib

```
# plot.py

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-np.pi, np.pi, 256)
y = np.cos(x)

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

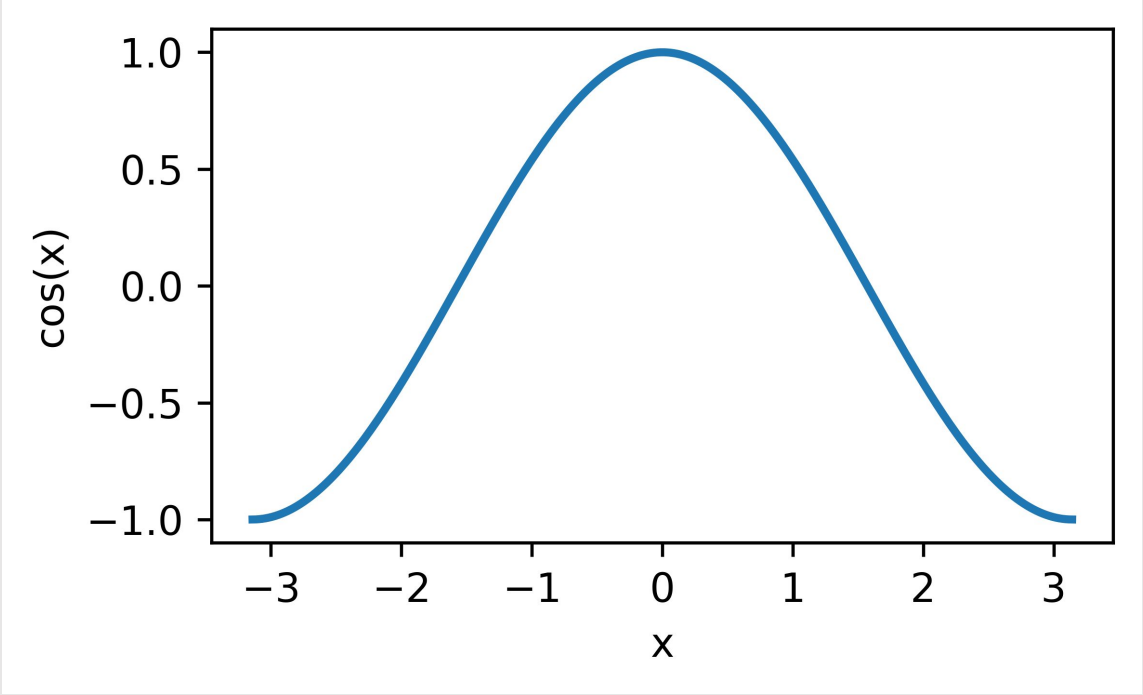


simula consulting

simula

Basic plotting with Matplotlib

```
# plot.py

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-np.pi, np.pi, 256)
y = np.cos(x)

golden = 1.618
h = 2; w = h*golden
fig = plt.figure(figsize=(w,h),
dpi=600)

plt.plot(x, y, lw=2)
plt.xlabel("x")
plt.ylabel("cos(x)")

fig.tight_layout()
plt.show()
```

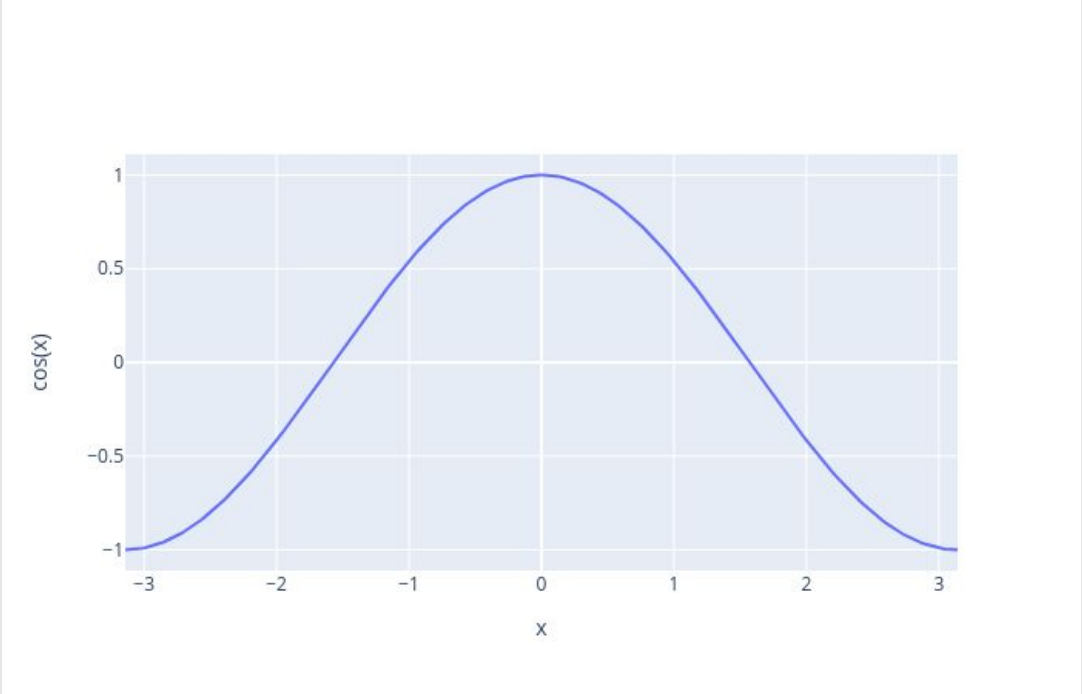


simula consulting

simula

Basic plotting with Plotly

```
# plot.py

import plotly.graph_objects as go
import numpy as np

x = np.linspace(-np.pi, np.pi, 256)
y = np.cos(x)

fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=y,
mode='lines'))

fig.update_layout(xaxis_title='x',
yaxis_title='cos(x)')
fig.show()
```

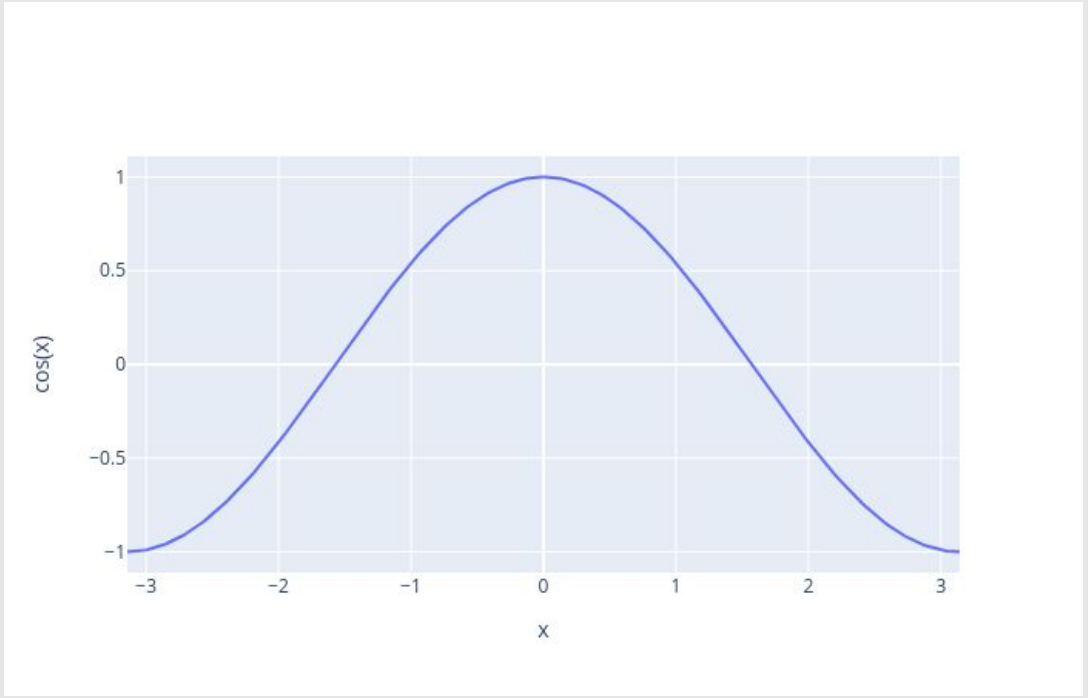


simula consulting

simula

Basic plotting with Plotly

```
# plot.py

...

fig.update_layout(xaxis=dict(
    title="x",
    linecolor='rgb(204, 204, 204)',
    linewidth=2,
    ticks='outside',
  ),
  yaxis=dict(
    title="cos(x)",
    linecolor='rgb(204, 204, 204)',
    linewidth=2,
    ticks='outside',
  ),
  autosize=False,
  plot_bgcolor='white',
)
```

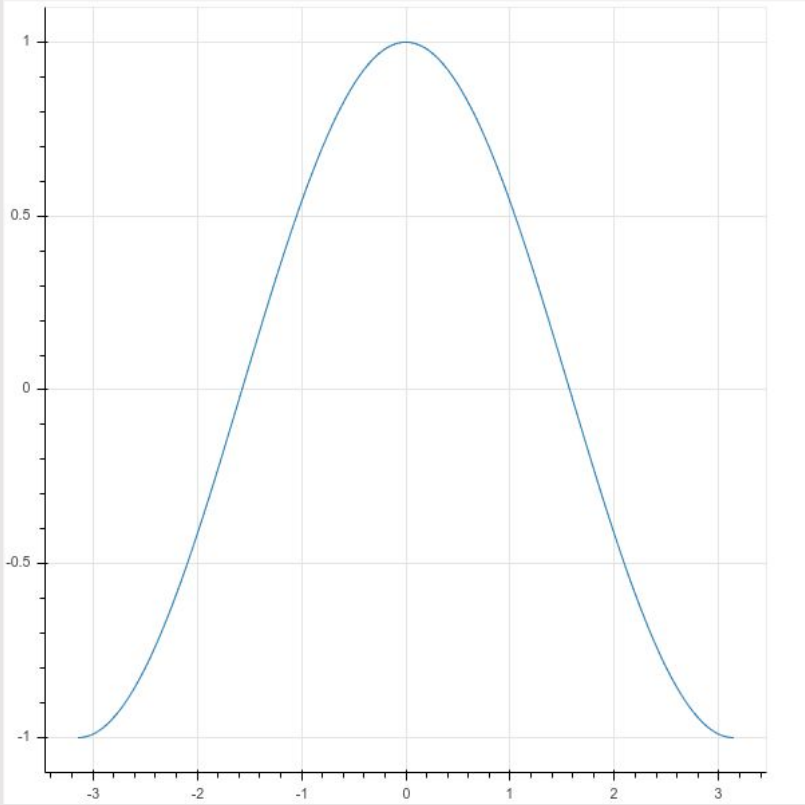


simula consulting

simula

Basic plotting with Bokeh

```
# plot.py

from bokeh.plotting import figure,
output_file, show

output_file("basic_bokeh.html")

p = figure()
p.line(x, y)

show(p)
```

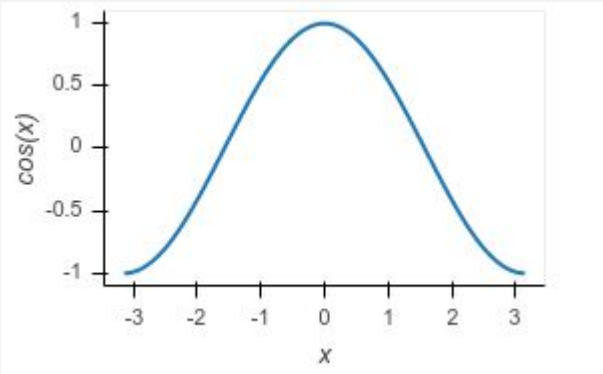


simula consulting

simula

Basic plotting with Bokeh

```
# plot.py

golden = 1.618
dpi = 75
h = int(2.5*dpi); w = int(h*golden)

p = figure(plot_width=w,
plot_height=h)
p.line(x, y, line_width=2)

p.xaxis.axis_label = "x"
p.yaxis.axis_label = "cos(x)”
p.xgrid.grid_line_color = None
p.ygrid.grid_line_color = None
p.xaxis.minor_tick_line_color = None
p.yaxis.minor_tick_line_color = None

show(p)
```



simula consulting

simula

Demo: Dataset visualisations in Plotly and Bokeh