

11 – Plotting

Bálint Aradi

Scientific Programming in Python (2025)

<https://atticlectures.net/scipro/python-2025/>

Matplotlib interfaces

- Matlab-like simplified interface with instance variables (recommended)
- Matlab-like simplified interface with global variables
- Fully object oriented interface (when embedding)

Matplotlib render engines

- Embedding plots into the IPython/Jupyter notebook

```
%matplotlib inline
```

In JupyterLab this is already the default

- Showing plots in separate windows (when using from script or from IPython-console)
- Creating graphical files (pdf, jpg, etc.)

Self-containing plotting example

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
xx = np.linspace(0.0, 4.0 * np.pi, 200, endpoint=True)      Generating x/y values
```

```
y1 = np.cos(xx)
```

```
y2 = np.sin(xx)
```

```
fig, ax = plt.subplots()
```

```
ax.plot(xx, y1, color='red', linewidth=1.0, linestyle="--", label='cos(x)')
```

```
ax.plot(xx, y2, color='blue', linewidth=1.0, linestyle="-", label='sin(x)')
```

```
ax.legend() ← Create legend box
```

```
plt.show() ← Render plot/figure (optional in JupyterLab)
```

Create Figure and Axes objects (multiple subplots possible)

Plot curves through given x/y values

- If you do not use **plt.show()** in Jupyter, append semicolon (“;”) to last line of the cell to suppress additional non-graphical output

- If you use a GUI-backend, you can also use **fig.show()** to render a figure

Self-containing plotting example

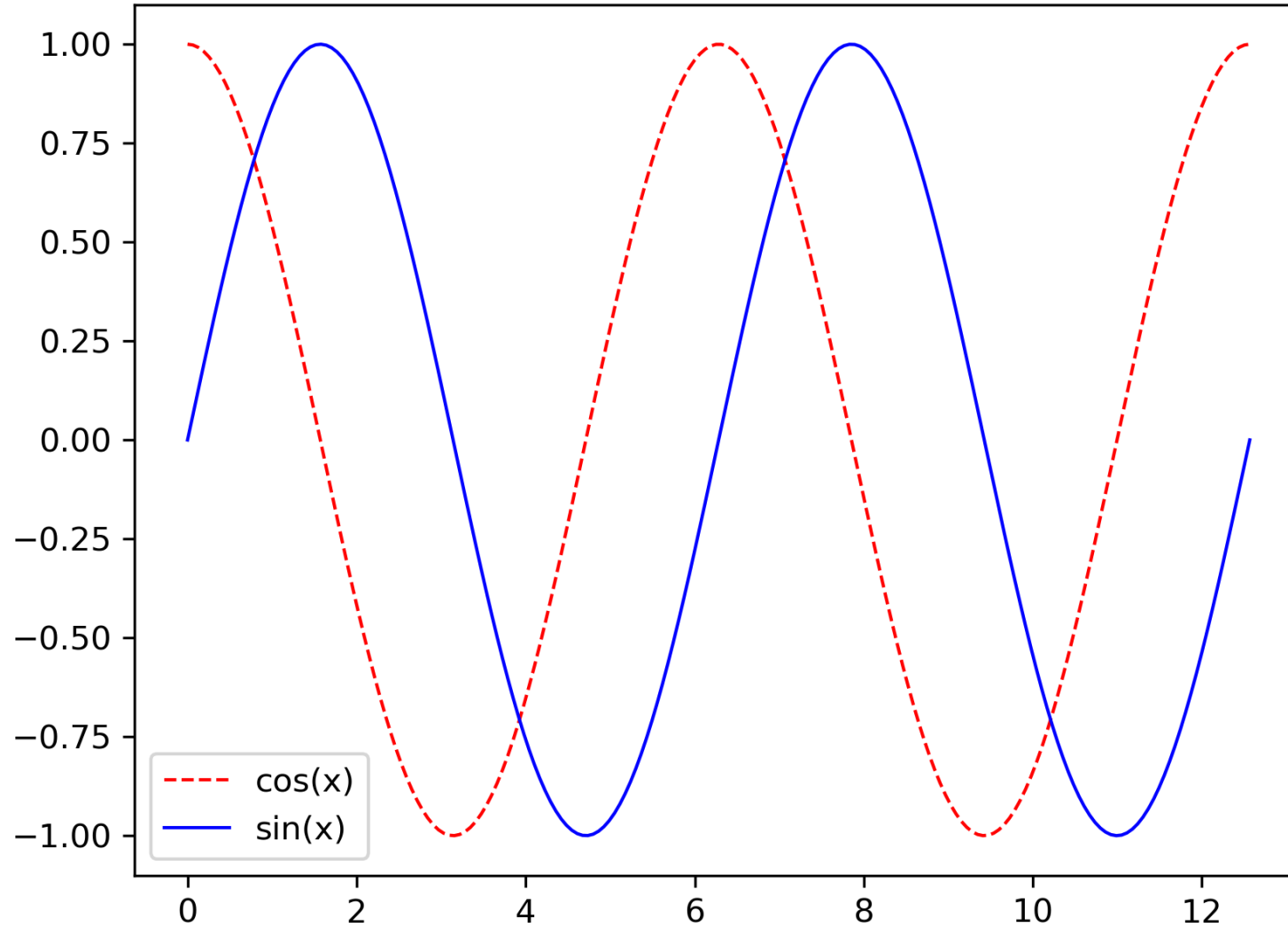


Figure and Axes objects

Figure

- A Figure object instance represents the figure
- Figure objects enables to manipulate the global figure parameters or to execute global actions

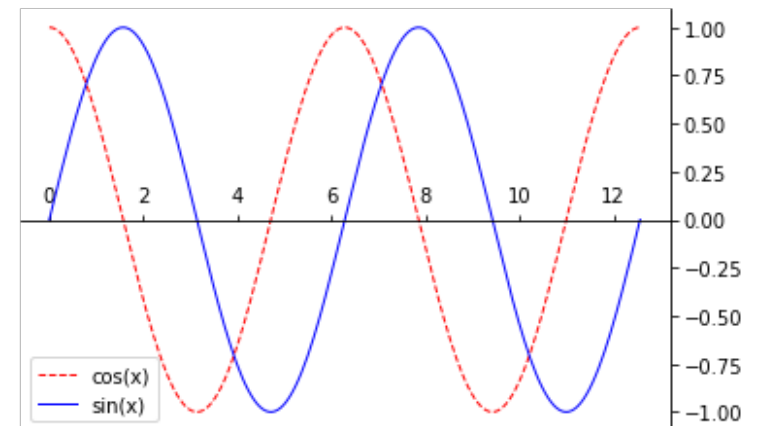
```
fig.set_size_inches(10, 8)  
fig.set_dpi(300)
```

```
fig.savefig('plot.pdf')
```

Axes

- An Axes-object instance represents one plot within the figure
- Axes-object enables very detailed tuning of the resulting plot

```
ax.xaxis.set_ticks_position('top')  
ax.yaxis.set_ticks_position('right')  
ax.spines['top'].set_position(('data', 0))  
ax.spines['bottom'].set_color('none')  
ax.spines['left'].set_color('none')
```



Multiple subplots

- The **subplots()** command can create multiple subfigures
- It returns individual Axes objects (one for each subfigure)

```
fig, (ax1, ax2) = plt.subplots(2, 1)
```

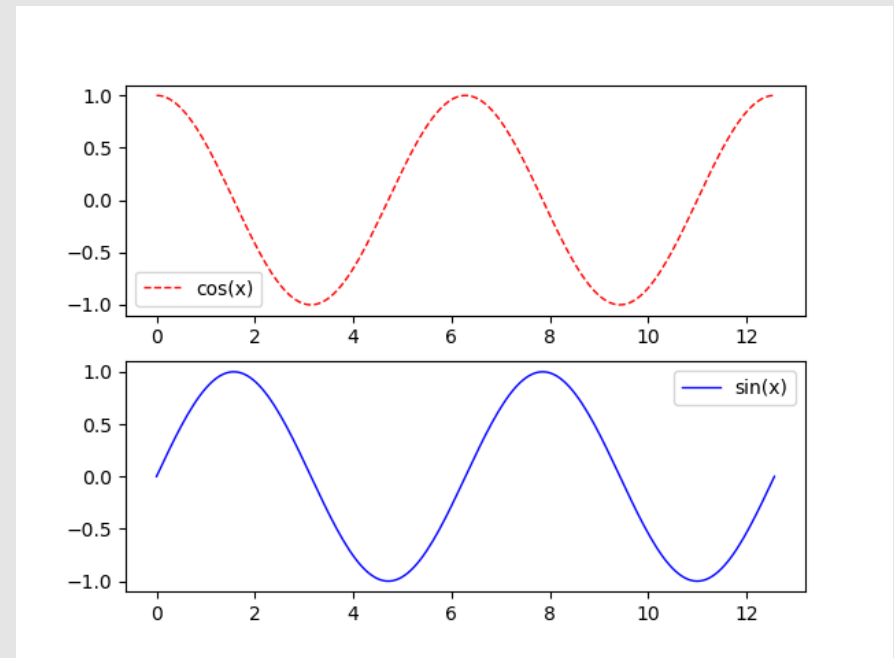
Two rows, one column (2 figures)

```
ax1.plot(xx, y1, color='red', linewidth=1.0, linestyle="--", label='cos(x)')  
ax1.legend()
```

```
ax2.plot(  
    xx, y2, color='blue', linewidth=1.0,  
    linestyle="--", label='sin(x)')
```

```
)  
ax2.legend()
```

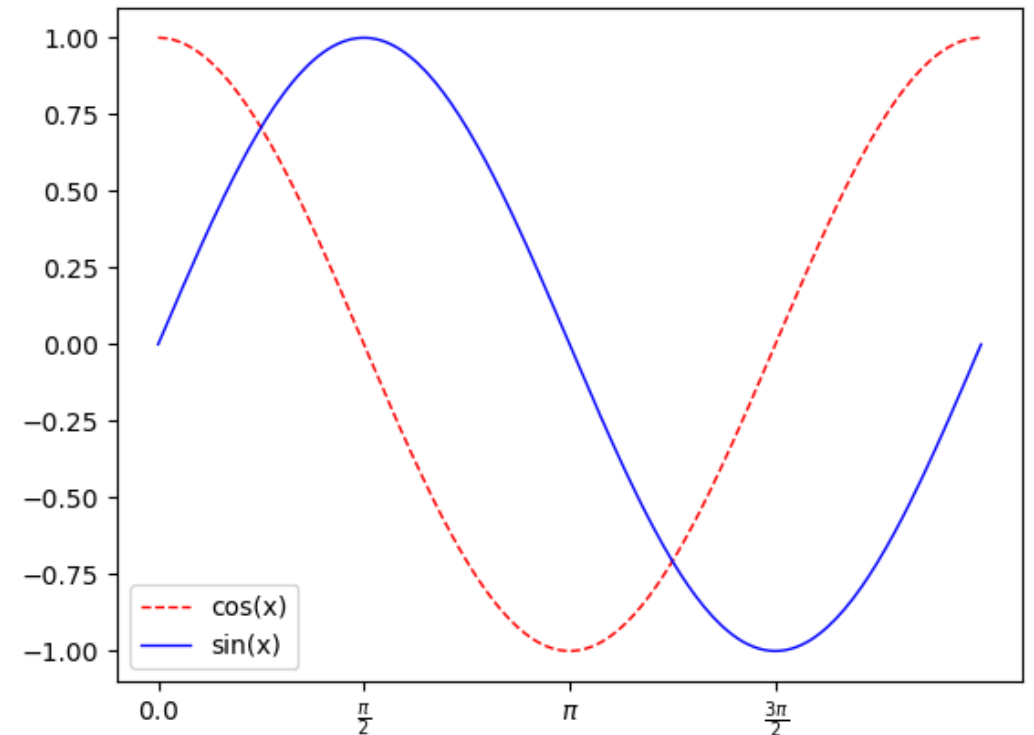
```
plt.show()
```



Rendering TeX within plots

```
ax.set_xticks(  
    [0.0, np.pi / 2, np.pi, 3 * np.pi / 2],  
    [r'$0.0$', r'$\frac{\pi}{2}$', r'$\pi$', r'$\frac{3\pi}{2}$']  
)
```

- Matplotlib can render TeX sequences in plots
- TeX-sequences should be **delimited by \$**
- It is advisable to put TeX-sequences into **raw-strings (r'something')**
- In raw-strings, **backslashes are interpreted literally** and not as special Python commands (e.g. `\n` as “\” “n” and not as newline)
- Useful when passing backslash commands to various engines (TeX-sequences in Matplotlib, regular expressions, ...)



Further useful Axes methods

<code>ax.set_xlim(), ax.set_ylim()</code>	Setting/Querying x/y limits
<code>ax.set_xticks(), ax.set_yticks()</code>	Setting customized ticks (and tick labels)
<code>ax.annotate()</code>	Write text into the plot
<code>ax.plot()</code>	Curve plot
<code>ax.scatter()</code>	Scatter plot
<code>ax.bar()</code>	Bar plot
<code>ax.contour()</code>	Contour plot
<code>ax.imshow()</code>	Bitmap image
<code>ax.pie()</code>	Pie charts
<code>ax.quiver()</code>	Quiver plots
:	

- Various excellent tutorials on Matplotlib available
- See for example [Matplotlib Quick Start Guide](#) or [Matplotlib: Plotting](#) (in [Scipy Lecture Notes](#))
- Some tutorials (e.g. Scipy-lectures) use the global interface access (easy to convert)