Data Visualization with Python

By:

Hossein Haghbin

Department of Statistics,
Faculty of Intelligent Systems
Engineering and Data Science,
Persian Gulf University

October 4, 2025

Chapter 1

Table of contents

1. Introduction to Matplotlib

Introduction

Matplotlib is a library for producing publication-quality figures. mpl (for short) was designed from the beginning to serve two purposes:

1. allow for interactive, cross-platform control of figures and plots
2. make it easy to produce static raster or vector graphics files without the need for any GUIs.

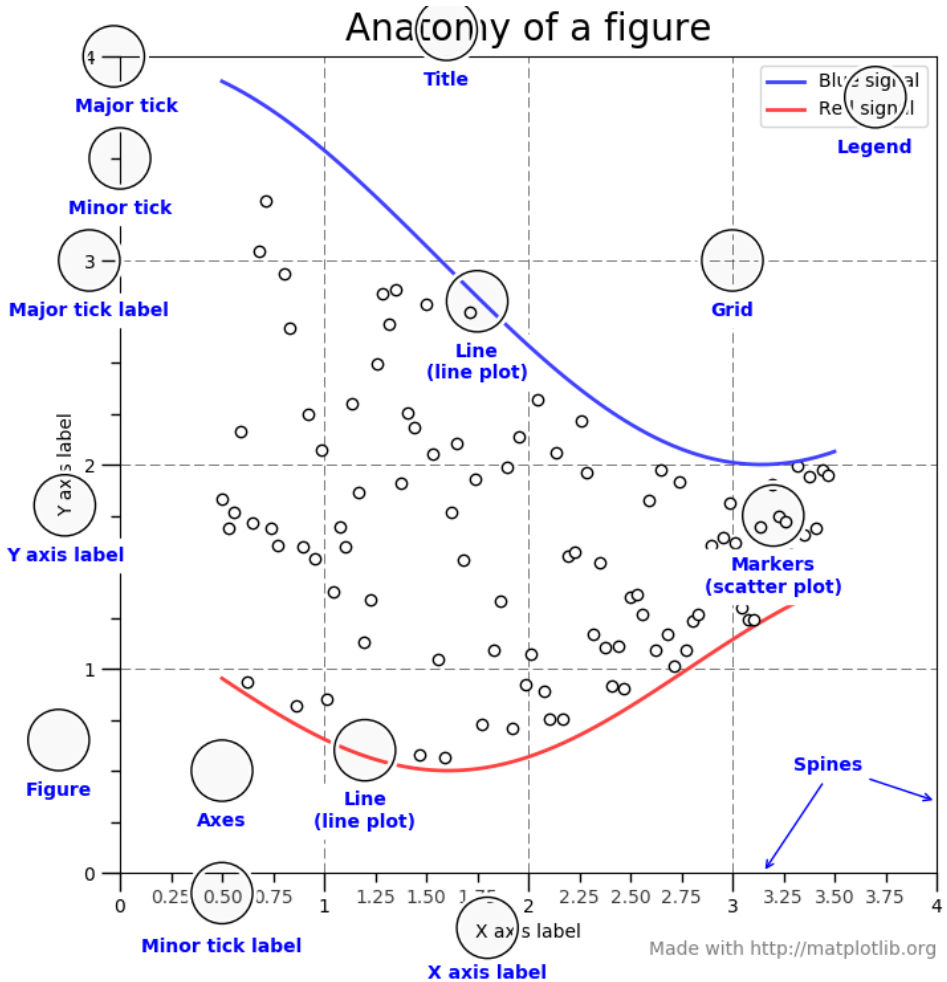
Furthermore, mpl – much like Python itself – gives the developer complete control over the appearance of their plots, while still being very usable through a powerful defaults system.

Gallery

Many users of Matplotlib are often faced with the question, "I want to make a figure that has X with Y in the same figure, but it needs to look like Z". Good luck getting an answer from a web search with that query! This is why the gallery is so useful, because it showcases the variety of ways one can make figures. Browse through the gallery, click on any figure that has pieces of what you want to see and the code that generated it. Soon enough, you will be like a chef, mixing and matching components to produce your masterpiece!

Parts of a Figure

Now, let's have a deeper look at the components of a Matplotlib figure.



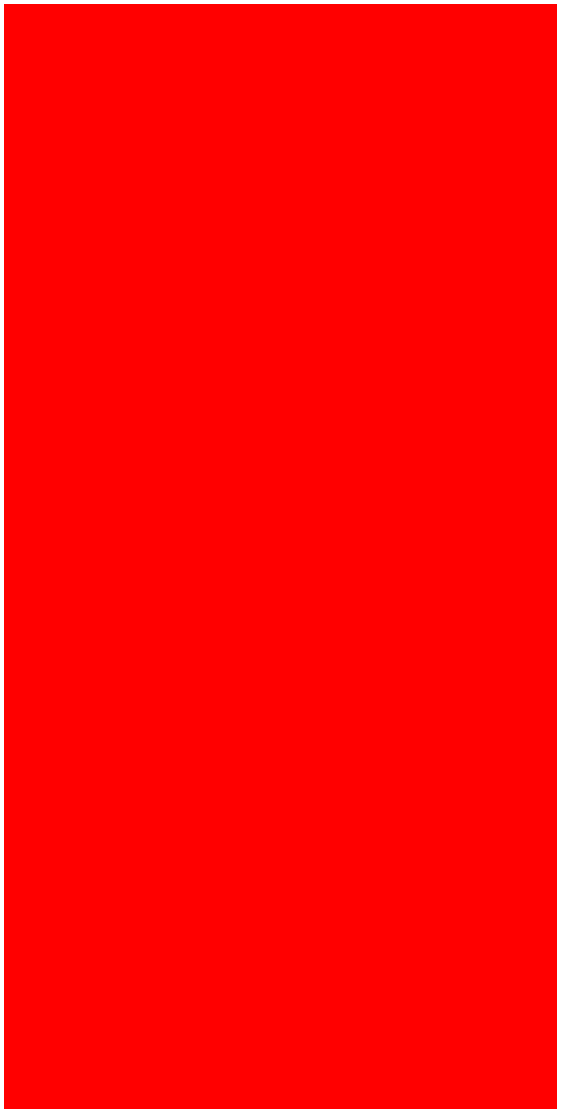
Figures

Matplotlib graphs your data on `Figures`, each of which can contain one or more `Axes`. The most simple way of creating a figure with an axes is using `pyplot.subplots`. We can then use `Axes.plot` to draw some data on the axes:

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure(facecolor=(1, 0, 0, .1))
# red background to see where the figure is
plt.show()
```

However, while we're on the topic, you can control the size of the figure through the figsize argument, which expects a tuple of (width, height) in inches. A really useful utility function is **figaspect**

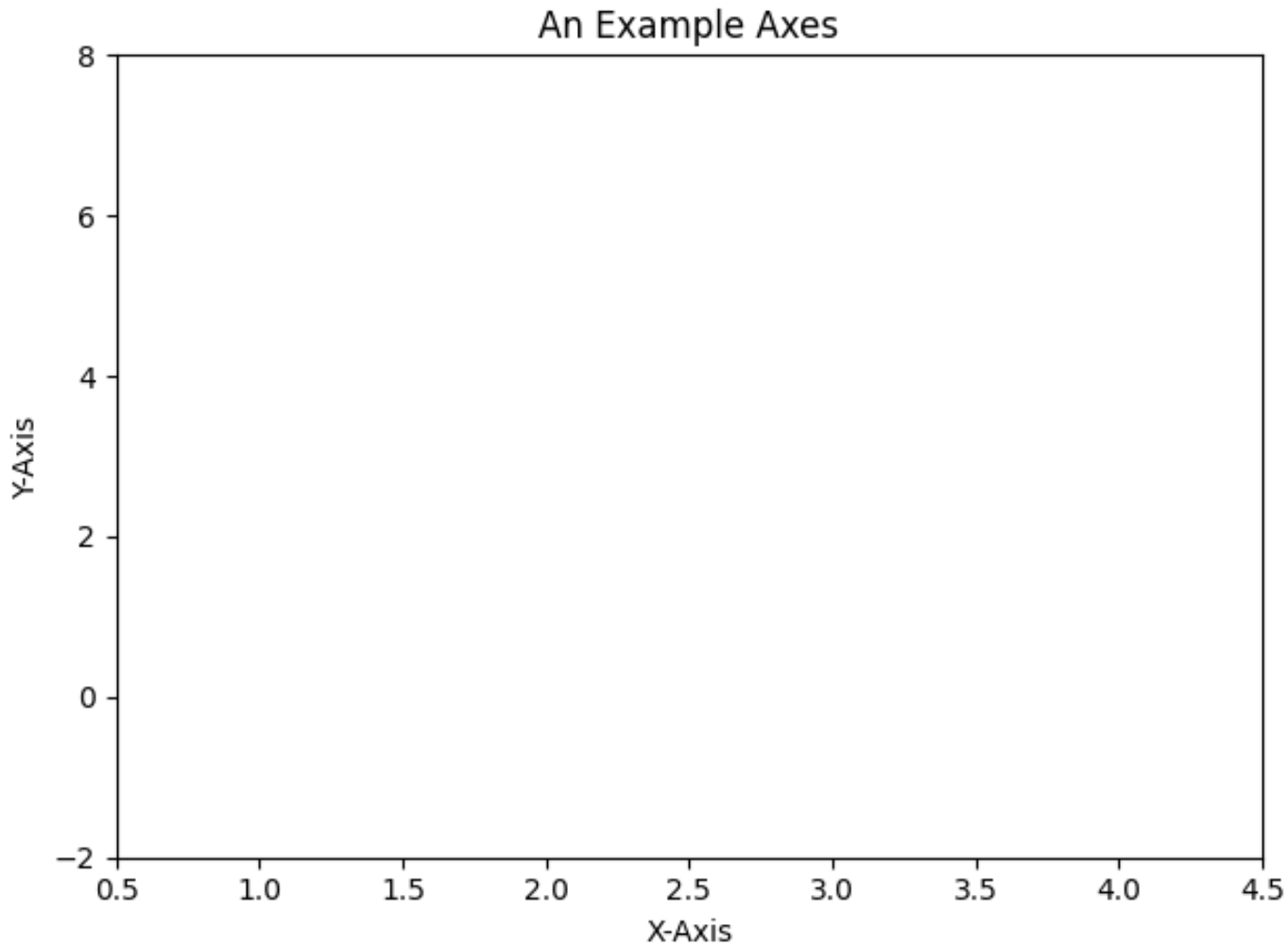
```
# Twice as tall as it is wide:
fig = plt.figure(figsize=plt.figaspect(2.0),
                                facecolor=(1, 0, 0, .1))
plt.show()
```



Axes

All plotting is done with respect to an Axes. An Axes is made up of Axis objects and many other things. An Axes object must belong to a Figure (and only one Figure). Most commands you will ever issue will be with respect to this Axes object.
Typically, you'll set up a Figure, and then add an Axes to it. You can use *fig.add_axes*, but in most cases, you'll find that adding a subplot will fit your needs perfectly.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set(xlim=[0.5, 4.5], ylim=[-2, 8],
       title="An Example Axes",
       ylabel="Y-Axis", xlabel="X-Axis")
plt.show()
```


An Example Axes
Y-Axis
X-Axis
8
6
4
2
0
−2
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

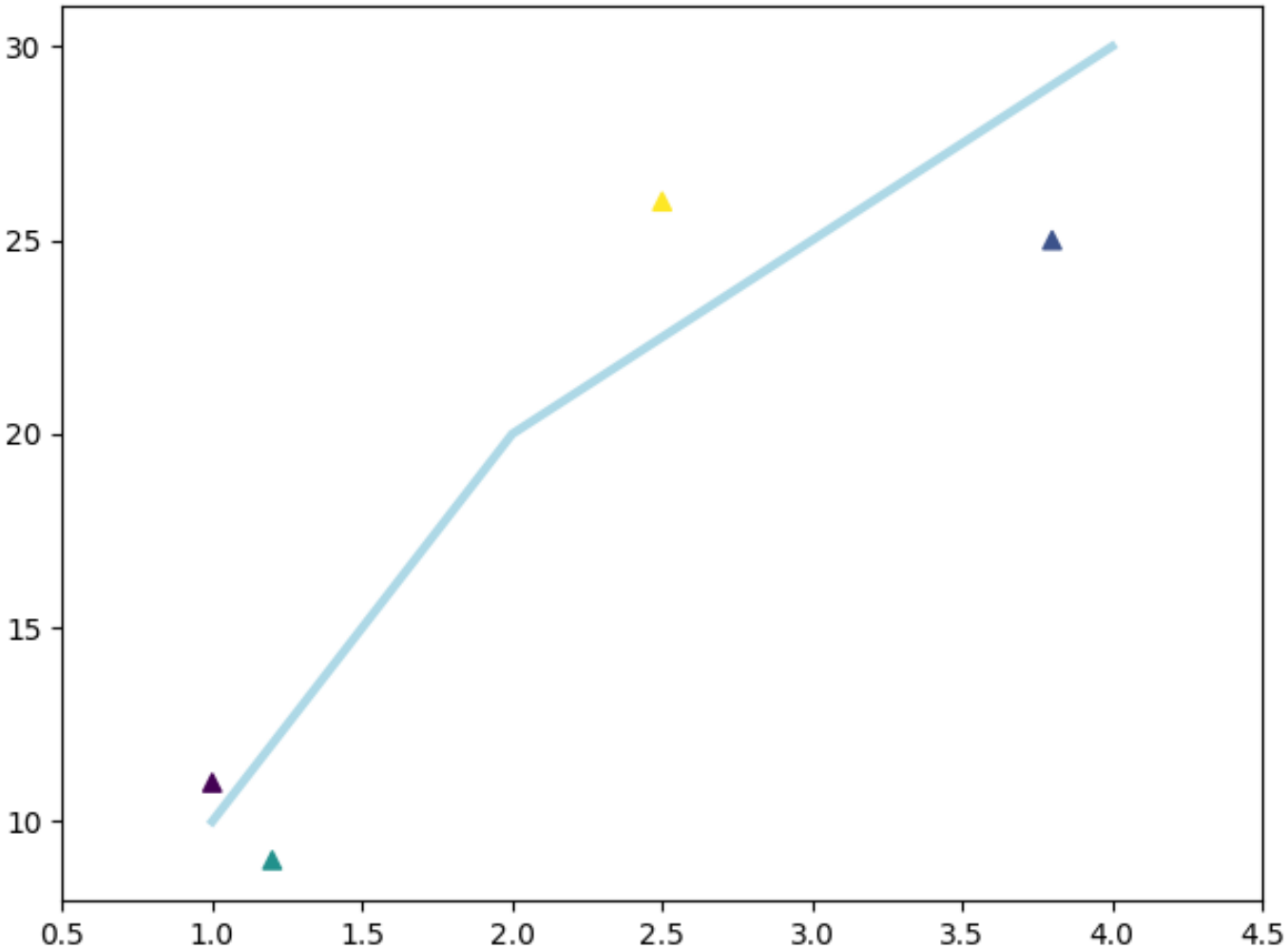
Notice the call to *set*. Matplotlib's objects typically have lots of "explicit setters" – in other words, functions that start with *set_ < something >* and control a particular option. For example, we could have written the third line above as:

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_xlim([0.5, 4.5])
ax.set_ylim([-2, 8])
ax.set_title("A Different Example Axes Title")
ax.set_ylabel("Y-Axis (changed)")
ax.set_xlabel("X-Axis (changed)")
plt.show()
```

Basic Plotting

Most plotting happens on an Axes. Therefore, if you're plotting something on an axes, then you'll use one of its methods. For now, let's focus on two methods: **plot** and **scatter**. plot draws points with lines connecting them. scatter draws unconnected points, optionally scaled or colored by additional variables. As a basic example:

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([1, 2, 3, 4], [10, 20, 25, 30],
        color="lightblue", linewidth=3)
ax.scatter([0.3, 3.8, 1.2, 2.5], [11, 25, 9, 26],
        c=[1, 2, 3, 5], marker="^")
ax.set_xlim(0.5, 4.5)
plt.show()
```


30
25
20
15
10
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

Types of inputs to plotting functions

All of plotting functions expect `numpy.array` as input. Classes that are 'array-like' such as pandas data objects and `numpy.matrix` may or may not work as intended. It is best to convert these to `numpy.array` objects prior to plotting.

```
# To convert a pandas.DataFrame:
a = pandas.DataFrame(np.random.rand(4, 5),
                     columns = list("abcde"))
a_asarray = a.values

# and to convert a numpy.matrix
b = np.matrix([[1, 2], [3, 4]])
b_asarray = np.asarray(b)
```

The object-oriented and the pyplot interface

As noted above, there are essentially two ways to use Matplotlib:

- Explicitly create figures and axes, and call methods on them (the ”object-oriented (OO) style”).
- Rely on pyplot to automatically create and manage the figures and axes, and use pyplot functions for plotting.

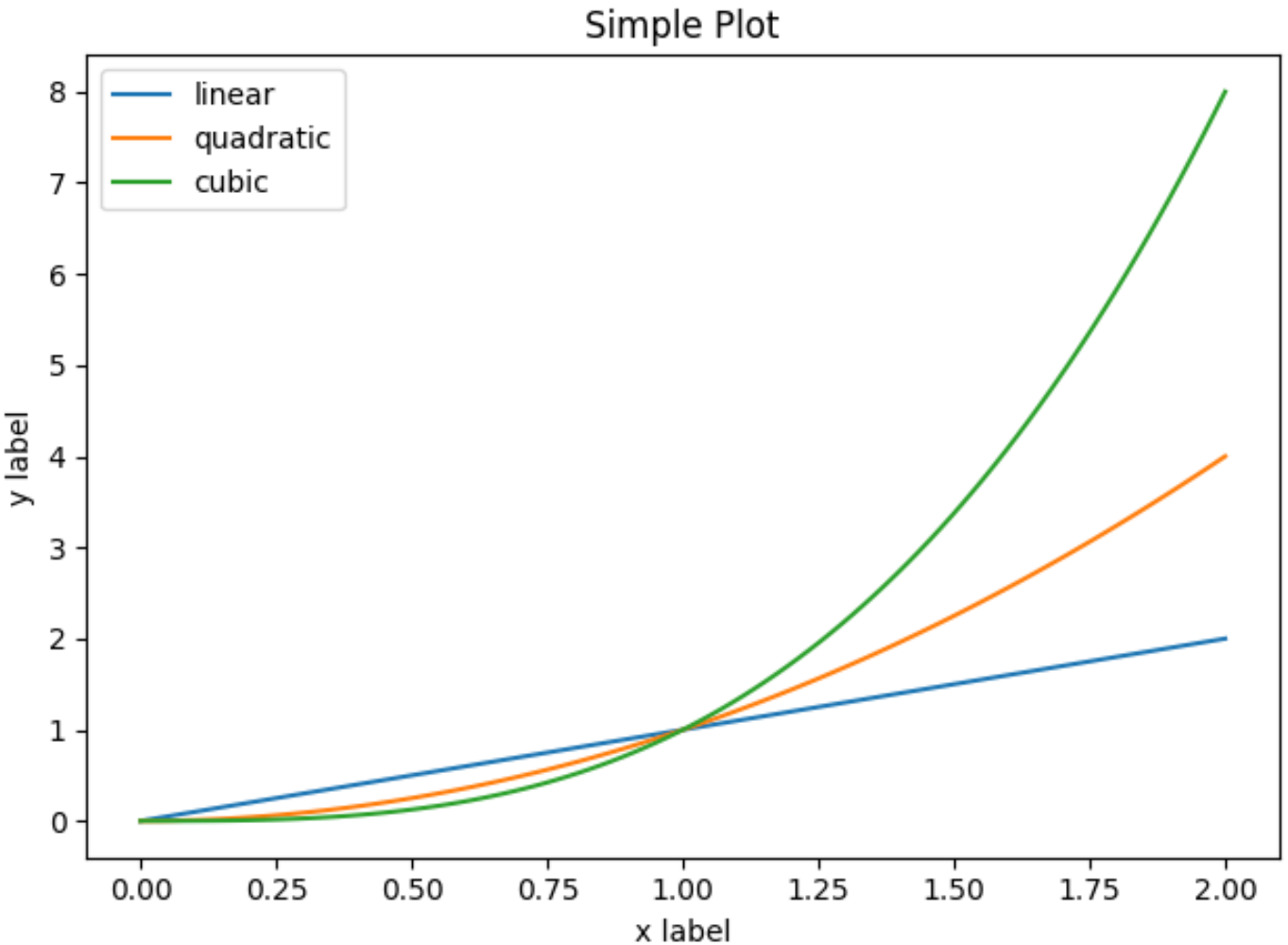
So one can do (OO-style)

OO-style

```
x = np.linspace(0, 2, 100)
fig, ax = plt.subplots()
ax.plot(x, x, label="linear")
ax.plot(x, x**2, label="quadratic")
ax.plot(x, x**3, label="cubic")
ax.set_xlabel("x label")
ax.set_ylabel("y label")
ax.set_title("Simple Plot")
ax.legend()
plt.show()
```

pyplot-style

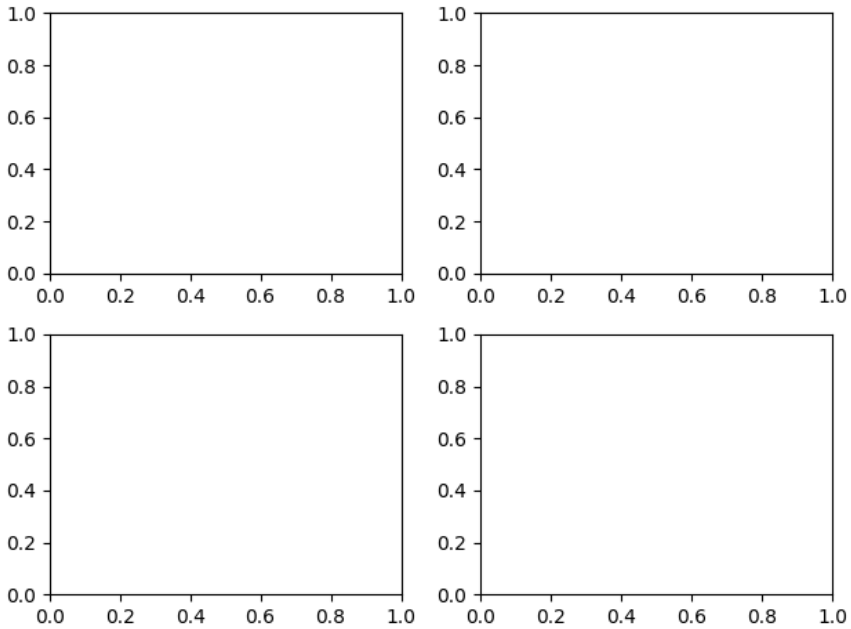
```
x = np.linspace(0, 2, 100)
plt.plot(x, x, label="linear")
plt.plot(x, x**2, label="quadratic")
plt.plot(x, x**3, label="cubic")
plt.xlabel("x label")
plt.ylabel("y label")
plt.title("Simple Plot")
plt.legend()
```


Simple Plot
linear
quadratic
cubic
y label
x label
0
1
2
3
4
5
6
7
8
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Multiple Axes

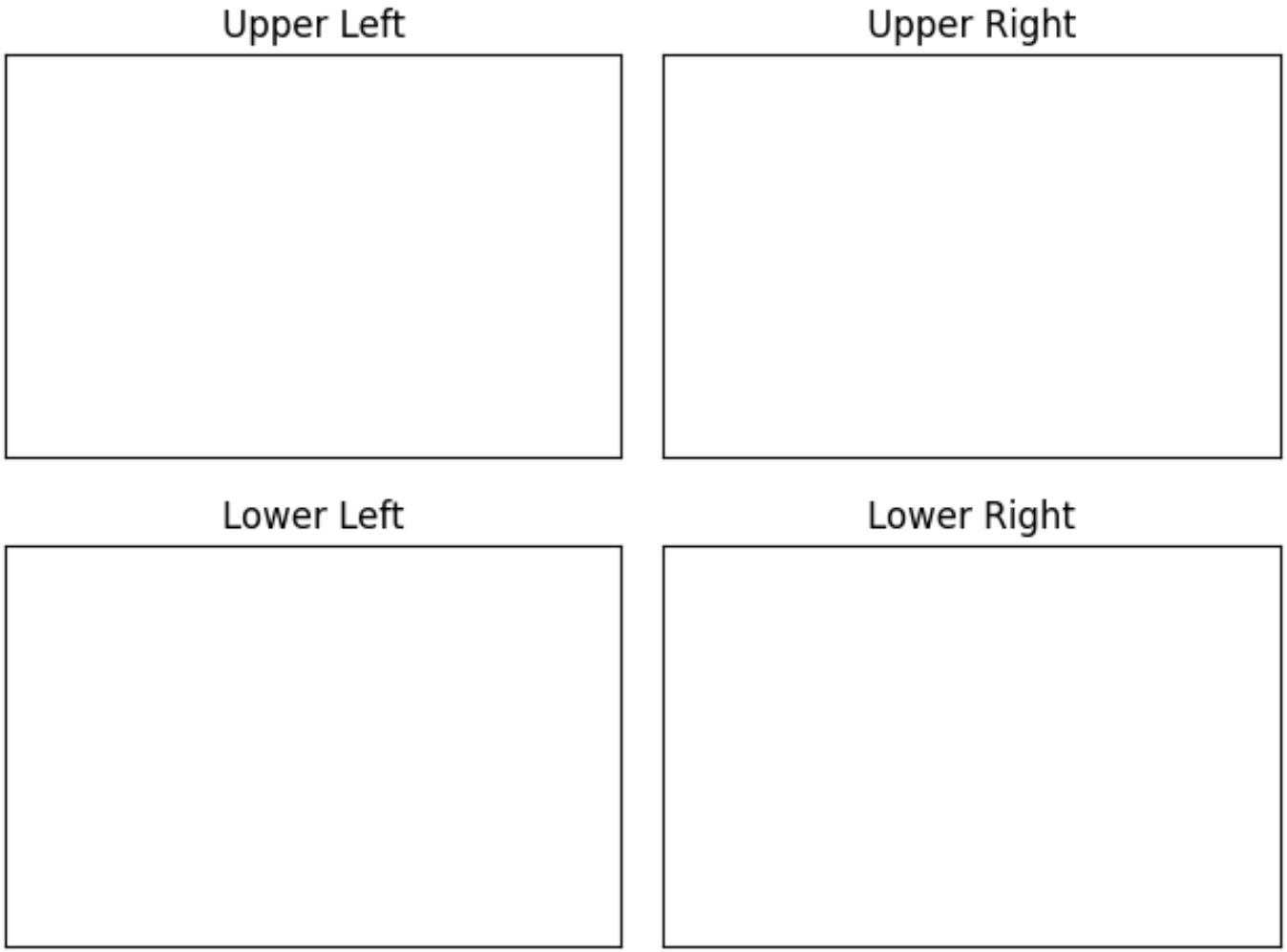
We've mentioned before that a figure can have more than one Axes on it. If you want your axes to be on a regular grid system, then it's easiest to use plt.subplots(...) to create a figure and add the axes to it automatically. For example:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
plt.show()
```



`plt.subplots(...)` created a new figure and added 4 subplots to it. The axes object that was returned is a 2D numpy object array. Each item in the array is one of the subplots. They're laid out as you see them on the figure. Therefore, when we want to work with one of these axes, we can index the axes array and use that item's methods. For example:

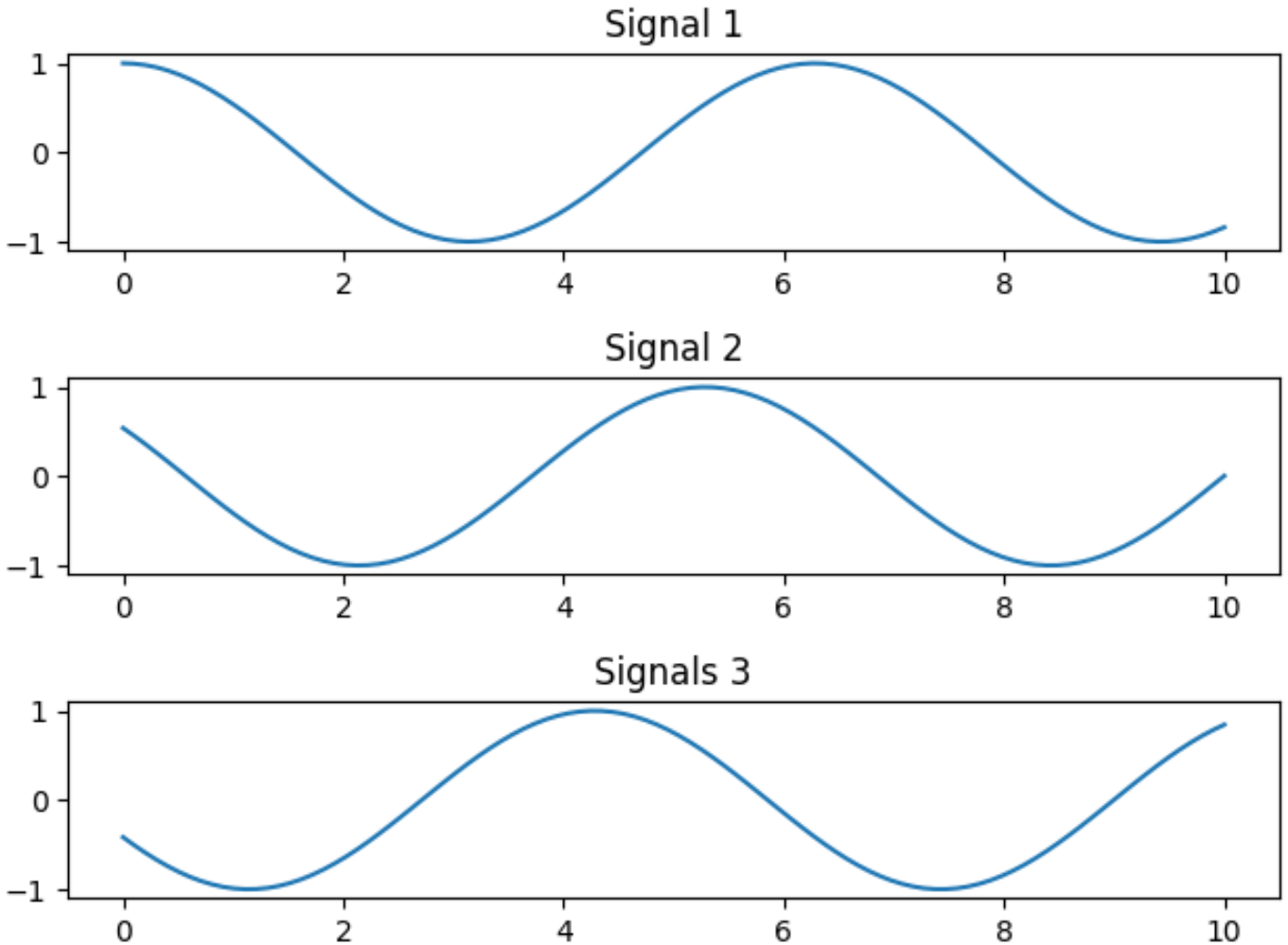
```
fig, axes = plt.subplots(nrows=2, ncols=2)
axes[0,0].set(title="Upper Left")
axes[0,1].set(title="Upper Right")
axes[1,0].set(title="Lower Left")
axes[1,1].set(title="Lower Right")

for ax in axes.flat:
    # Remove all xticks and yticks...
    ax.set(xticks=[], yticks=[])

plt.show()
```


Upper Left
Upper Right
Lower Left
Lower Right

A figure can contain any number of Axes, but will typically have at least one. The easiest way to create a new figure is with `pyplot`:

```
# an empty figure with no Axes
fig = plt.figure()
# a figure with a single Axes
fig, ax = plt.subplots()
# a figure with a 2x2 grid of Axes
fig, axs = plt.subplots(2, 2)
```

```
x = np.linspace(0, 10, 100)
y1, y2, y3 = np.cos(x), np.cos(x + 1), np.cos(x + 2)

fig, axes = plt.subplots(3,1)
axes[0].plot(x,y1)
axes[0].set(title="Signal 1")

axes[1].plot(x,y2)
axes[1].set(title="Signal 2")

axes[2].plot(x,y3)
axes[2].set(title="Signals 3")
plt.show()
```


Signal 1
1
0
−1
0
2
4
6
8
10
Signal 2
1
0
−1
0
2
4
6
8
10
Signals 3
1
0
−1
0
2
4
6
8
10

interactive mode

Use of an interactive backendpermits–but does not by itself require or ensure–plotting to the screen. Whether and when plotting to the screen occurs, and whether a script or shell session continues after a plot is drawn on the screen, depends on the functions and methods that are called, and on a state variable that determines whether matplotlib is in ”interactive mode”. Turning interactive mode on and off in the middle of a stream of plotting commands, whether in a script or in a shell, is rarely needed and potentially confusing, so in the following we will assume all plotting is done with interactive mode either on or off.

```
import matplotlib.pyplot as plt
plt.ion()
plt.plot([1.6, 2.7])
```

This will pop up a plot window. Your terminal prompt will remain active, so that you can type additional commands such as:

```
plt.title("interactive test")
plt.xlabel("index")
```

Start a fresh session as in the previous example, but now turn interactive mode off:

```
plt.ioff()
plt.plot([1.6, 2.7])
```

Nothing happened–or at least nothing has shown up on the screen. To make the plot appear, you need to do this:

```
plt.show()
```

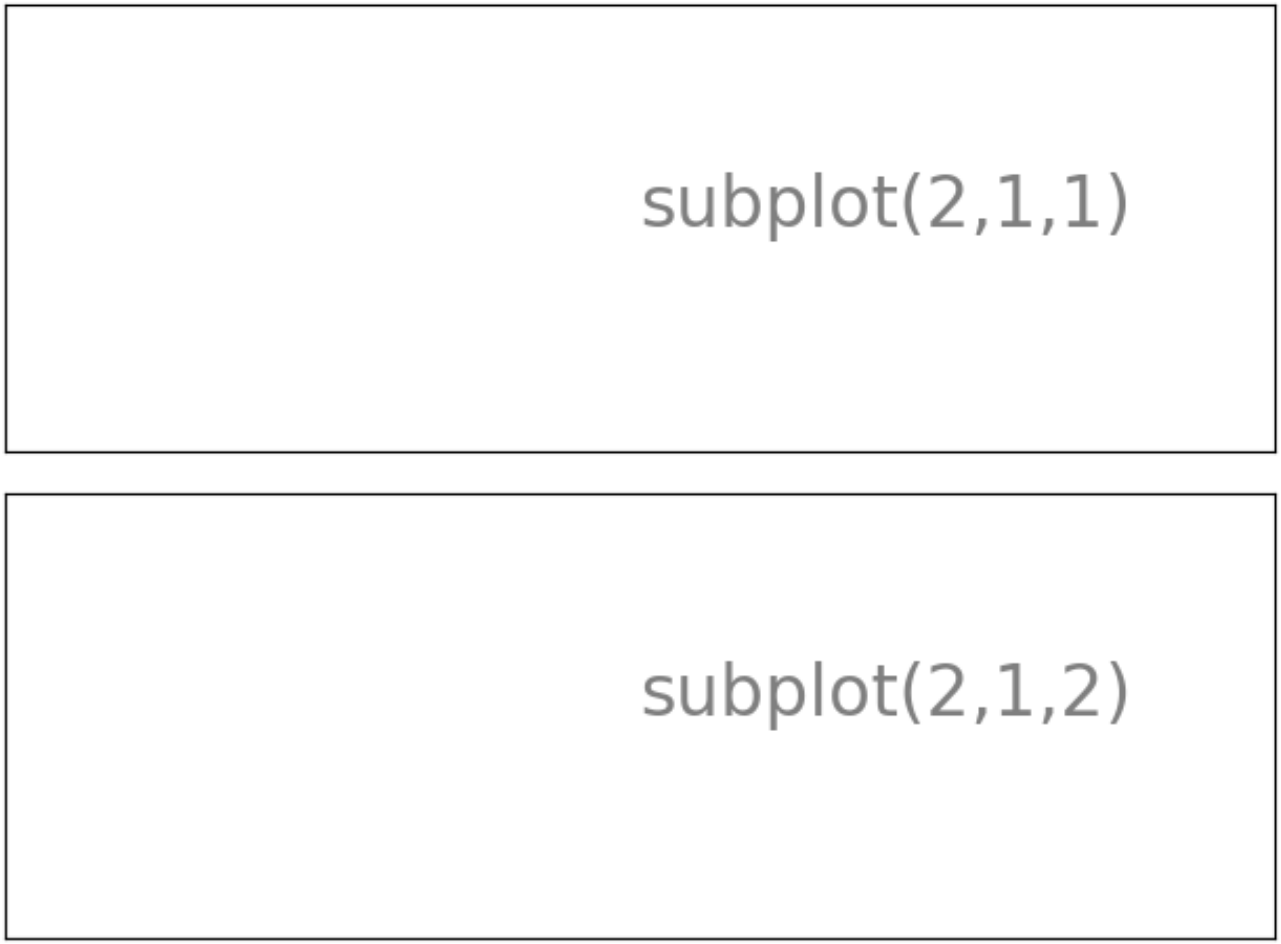
2. Examples of Matplotlib plots

Example 1:

```
plt.subplot(2,1,1)
plt.xticks([]), plt.yticks([])
plt.text(0.5,0.5,"subplot(2,1,1)",size=24,alpha=.5)
plt.subplot(2,1,2)
plt.xticks([]), plt.yticks([])
plt.text(0.5,0.5, "subplot(2,1,2)",size=24,alpha=.5)
# plt.savefig("Ex1.png", dpi=64)
plt.show()
```

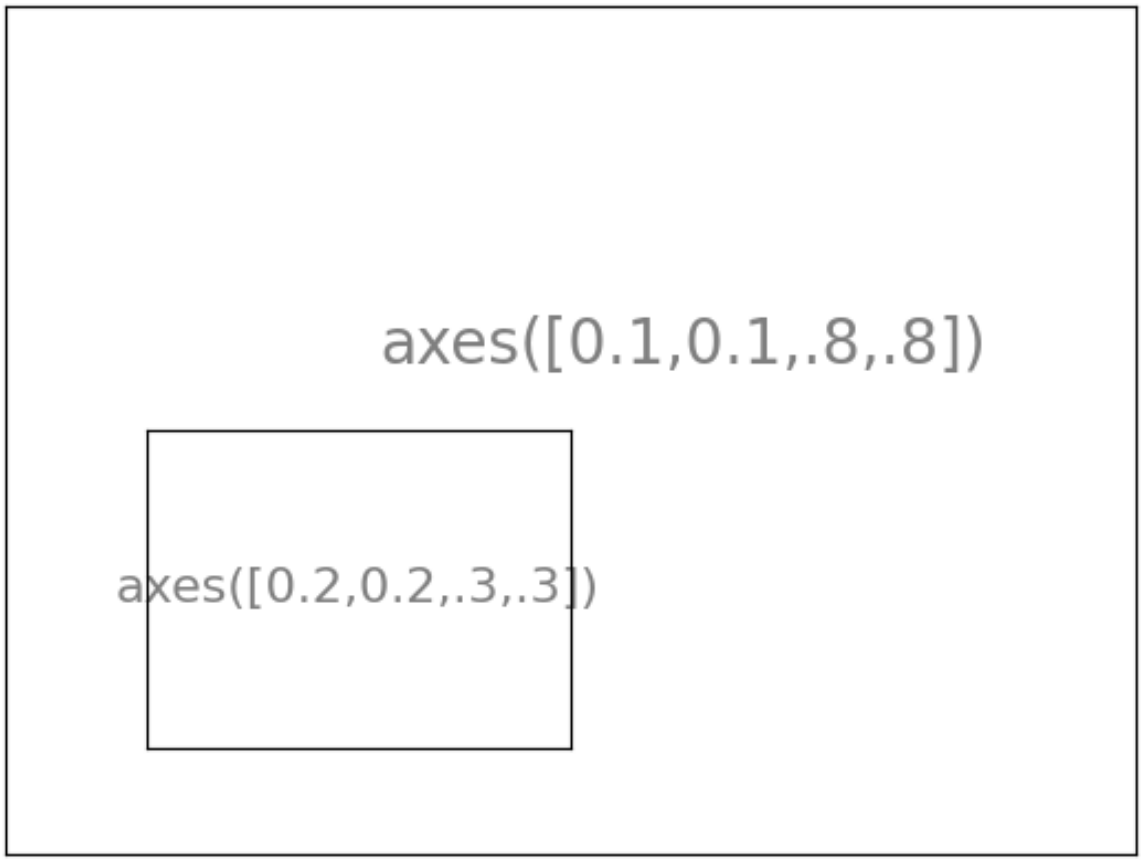
Example 1:...

Example 1:...



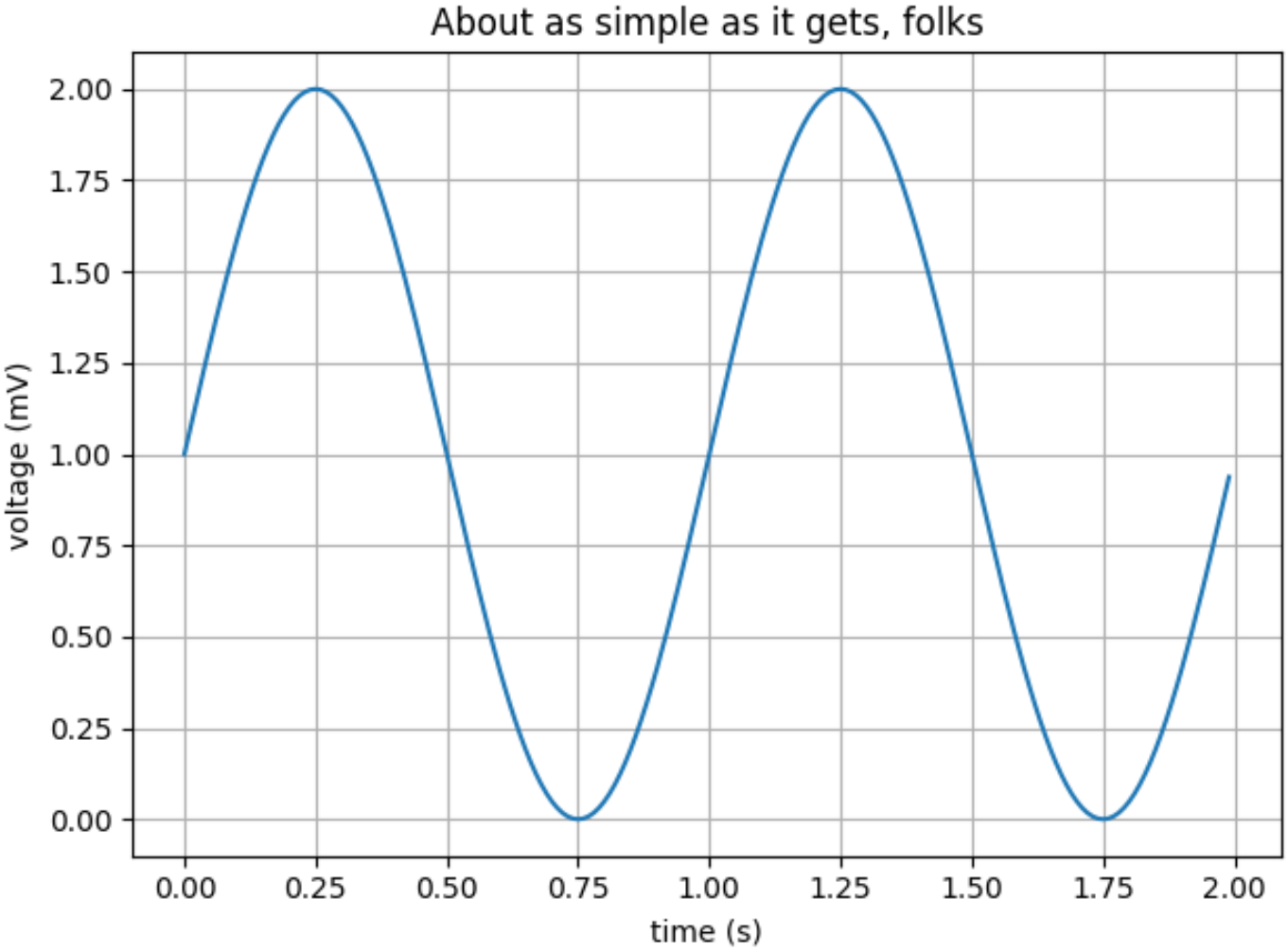
Example 2:

```
plt.axes([0.1,0.1,.8,.8])
plt.xticks([]), plt.yticks([])
plt.text(0.6,0.6, "axes([0.1,0.1,.8,.8])",
      size=20,alpha=.5)

plt.axes([0.2,0.2,.3,.3])
plt.xticks([]), plt.yticks([])
plt.text(0.5,0.5, "axes([0.2,0.2,.3,.3])",
      size=16,alpha=.5)

# plt.savefig("Ex2.png",dpi=64)
plt.show()
```


axes([0.1,0.1,.8,.8])
axes([0.2,0.2,.3,.3])

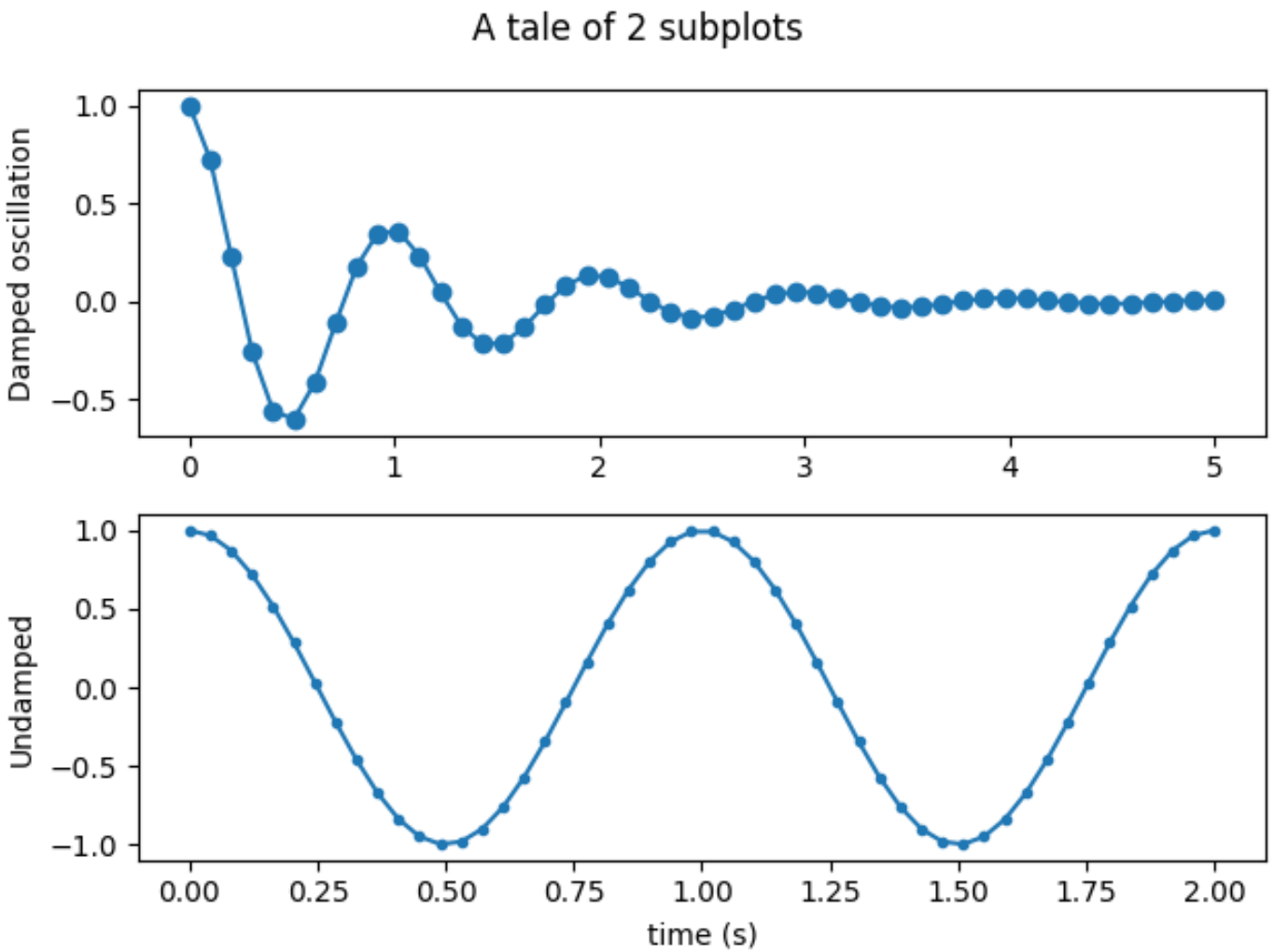
Example 3:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)
fig, ax = plt.subplots()
ax.plot(t, s)
ax.set(xlabel="time (s)", ylabel="voltage (mV)",
       title="About as simple as it gets, folks")
ax.grid()
fig.savefig("test.png")
plt.show()
```


About as simple as it gets, folks
voltage (mV)
time (s)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

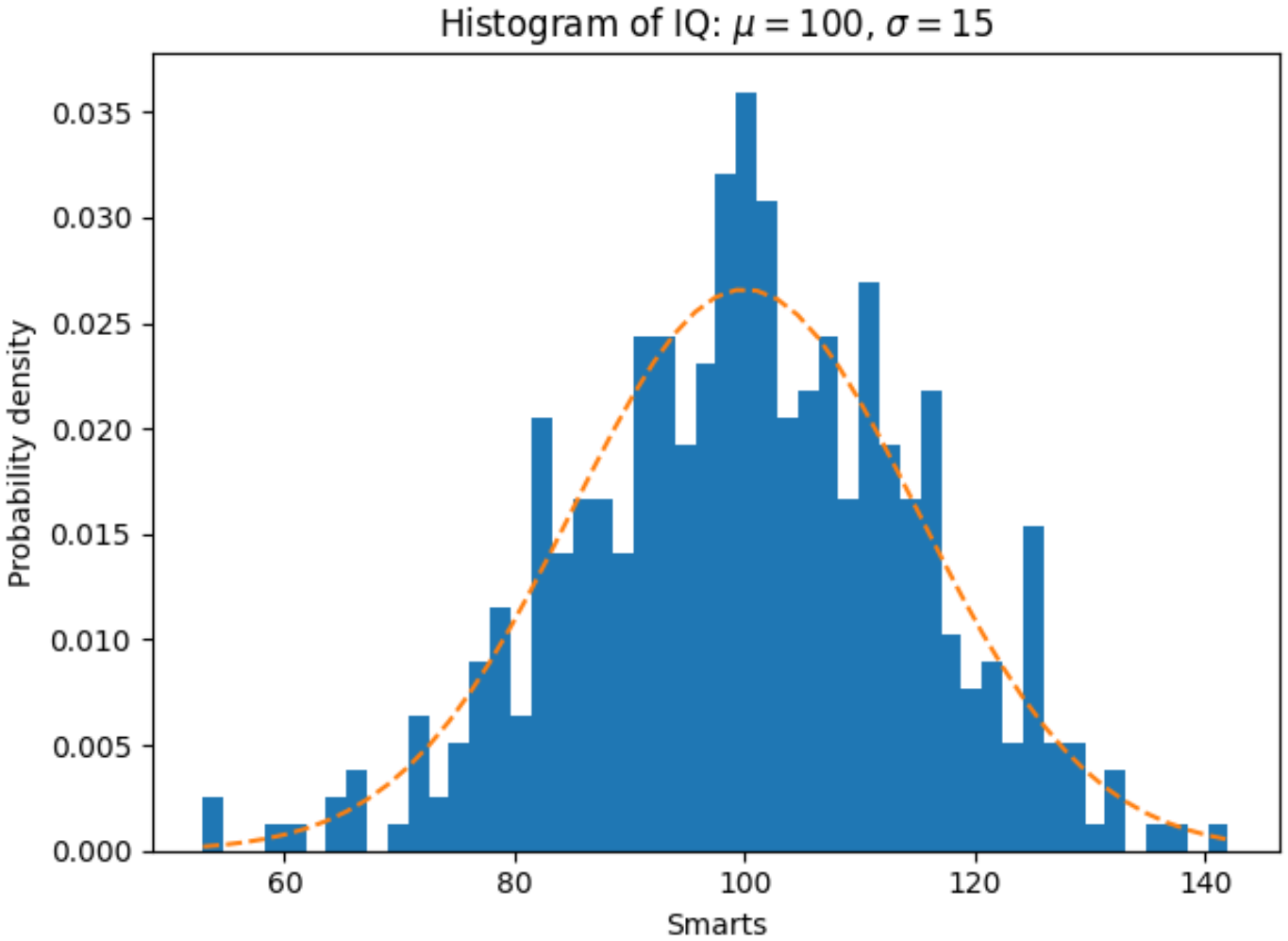
Example 4:

```
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)

y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle("A tale of 2 subplots")

ax1.plot(x1, y1, "o-")
ax1.set_ylabel("Damped oscillation")

ax2.plot(x2, y2, ".-")
ax2.set_xlabel("time (s)")
ax2.set_ylabel("Undamped")

plt.show()
```


A tale of 2 subplots
Damped oscillation
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
Undamped
1.0
0.5
0.0
−0.5
−1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
time (s)

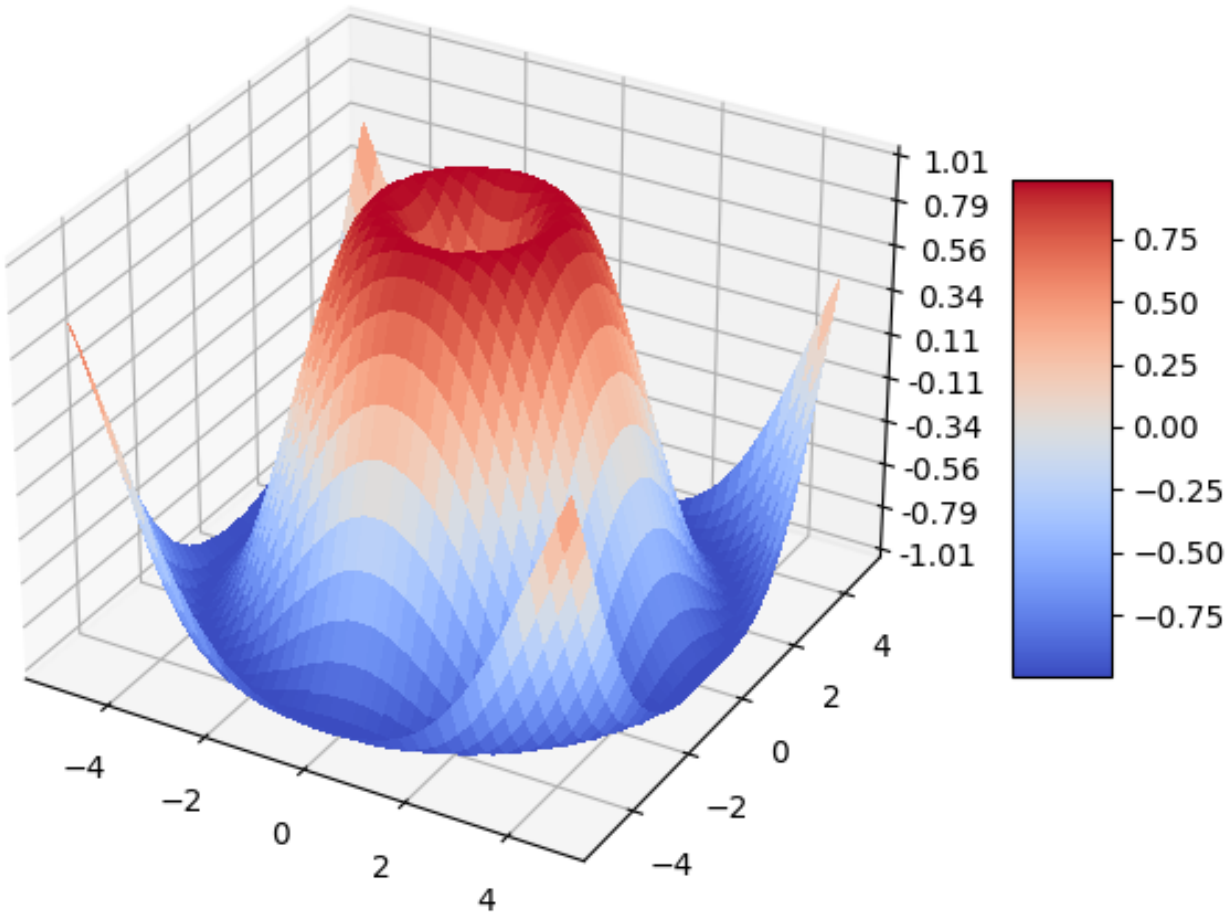
Example 5: Histogram

```
np.random.seed(19680801)
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(437)
num_bins = 50
fig, ax = plt.subplots()
n, bins, patches = ax.hist(x,num_bins,density=True)
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, "--")
ax.set_xlabel("Smarts")
ax.set_ylabel("Probability density")
ax.set_title(r"Histogram of IQ: $\mu=100$,
                                $\sigma=15$")
plt.show()
```


Histogram of IQ: $\mu = 100$, $\sigma = 15$
Probability density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
60
80
100
120
140
Smarts

Example 6:

```
from matplotlib import cm
from matplotlib.ticker import LinearLocator
fig, ax = plt.subplots(subplot_kw=
          {"projection": "3d"})
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm)
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```

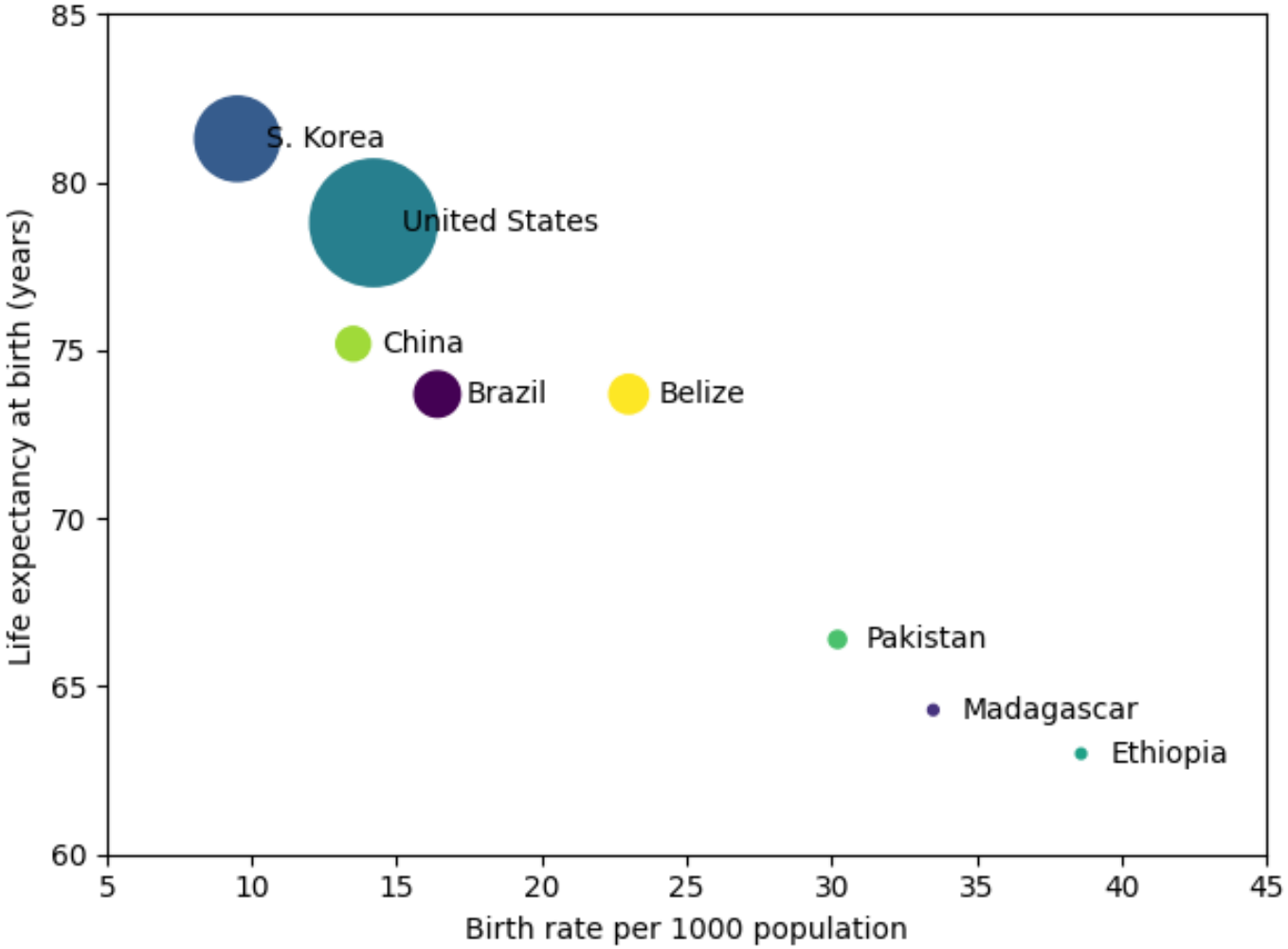

1.01
0.79
0.56
0.34
0.11
-0.11
-0.34
-0.56
-0.79
-1.01
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4
4
2
0
−2
−4

Example 7:A scatter plot of demographic data

```
countries = ["Brazil", "Madagascar", "S. Korea",
             "United States","Ethiopia","Pakistan",
             "China","Belize"]
# Birth rate per 1000 population
birth_rate = [16.4, 33.5, 9.5, 14.2, 38.6, 30.2,
              13.5, 23.0]
# Life expectancy at birth, years
life_expectancy = [73.7, 64.3, 81.3, 78.8, 63.0,
                   66.4, 75.2, 73.7]
# Per person income fixed to US Dollars in 2000
GDP = np.array([4800, 240, 16700, 37700, 230,
                670, 2640, 3490])
fig = plt.figure()
ax = fig.add_subplot(111)
```

Example 6:...

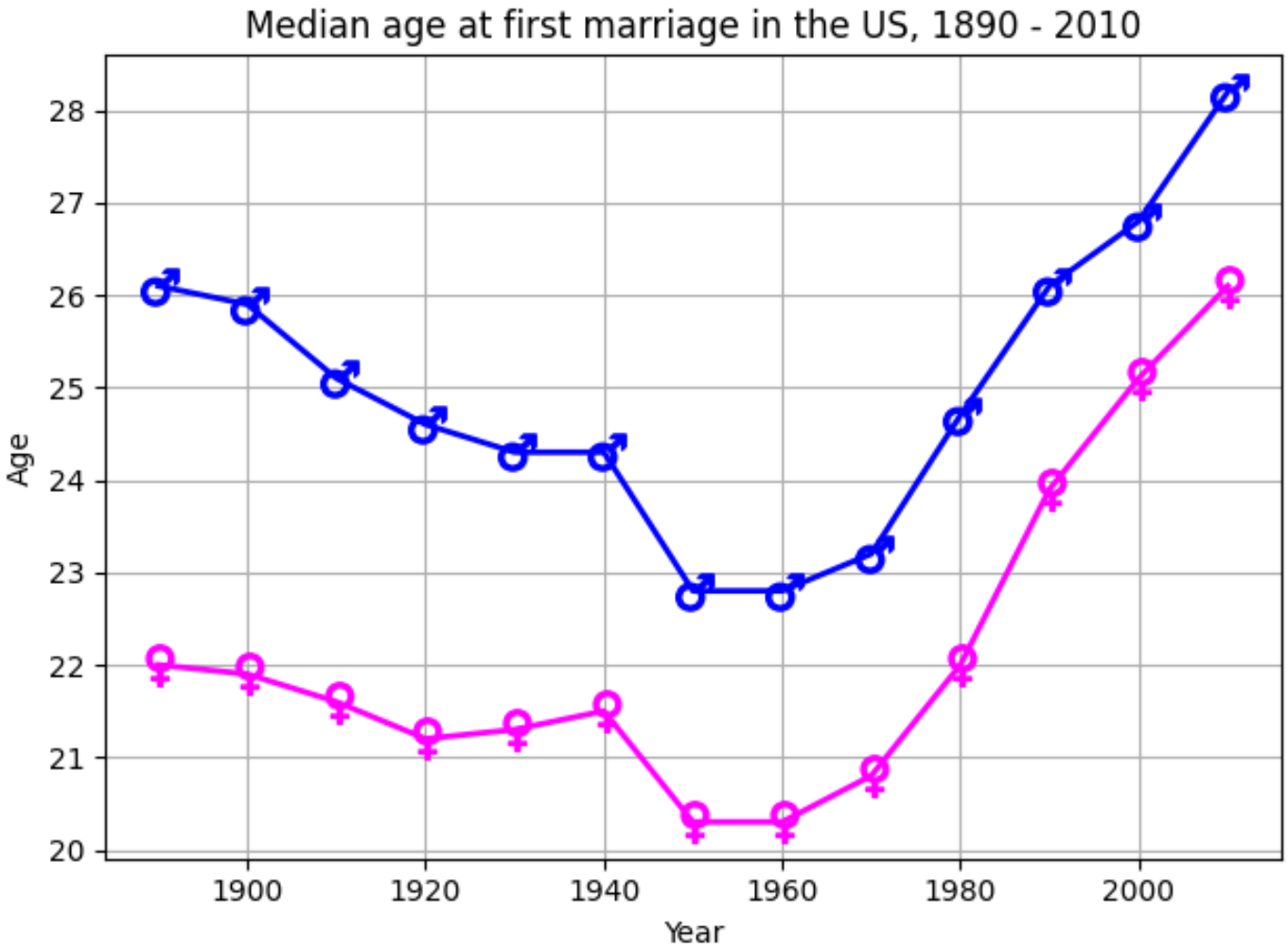
```
# Some random colours:
colours = range(len(countries))
ax.scatter(birth_rate, life_expectancy, c=colours,
                s=GDP/20)
ax.set_xlim(5, 45)
ax.set_ylim(60, 85)
ax.set_xlabel("Birth rate per 1000 population")
ax.set_ylabel("Life expectancy at birth (years)")

offset = 1
for x, y, s, country in zip(birth_rate,
                life_expectancy, GDP, countries):
    ax.text(x+offset, y, country, va="center")
plt.show()
```


S. Korea
United States
China
Brazil
Belize
Pakistan
Madagascar
Ethiopia
Life expectancy at birth (years)
Birth rate per 1000 population
85
80
75
70
65
60
5
10
15
20
25
30
35
40
45

Example 8:Median age at first marriage in the US over time

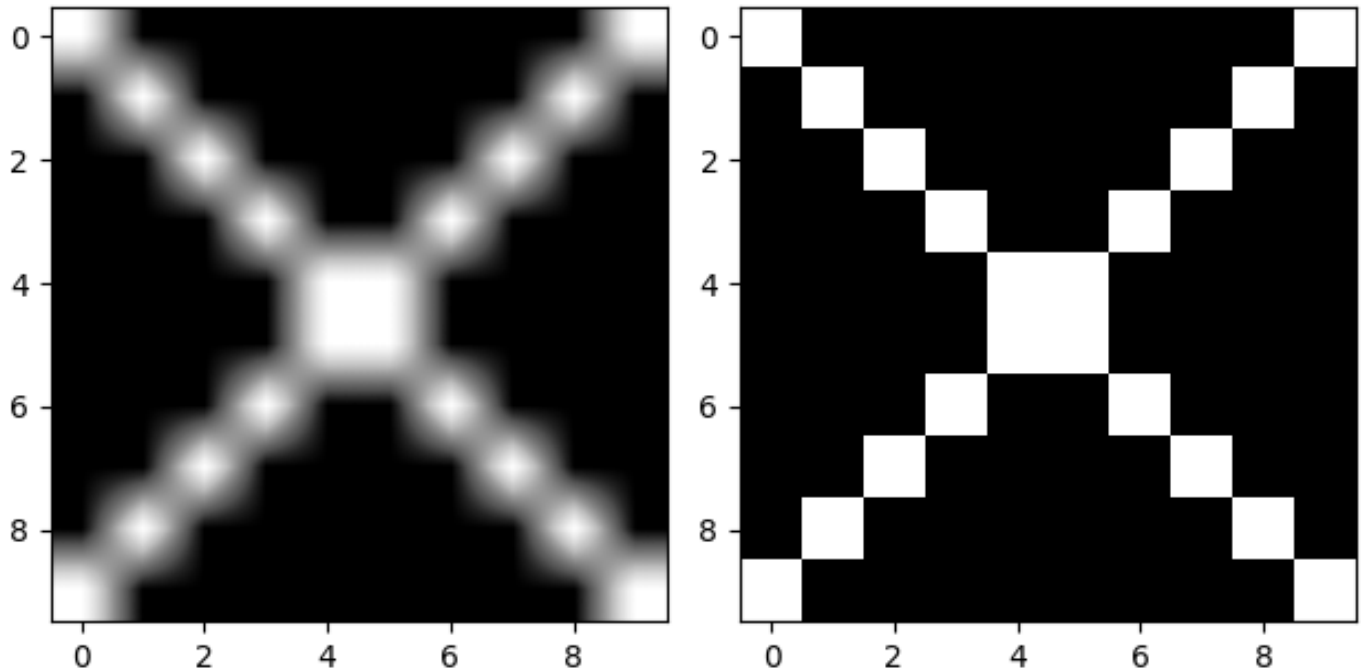
The data read in from the file `ex7-marriage-ages.txt`.

```
year, age_m, age_f = np.loadtxt(
      "data/ex7-marriage-ages.txt", unpack=True,
      skiprows=3)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(year, age_m, marker="$\u2642$",c="blue",
      markersize=14,lw=2,mfc="blue", mec="blue")
ax.plot(year, age_f, marker="$\u2640$", lw=2,
      markersize=14, c="magenta",mfc="magenta",
      mec="magenta")
ax.grid()
ax.set_xlabel("Year")
ax.set_ylabel("Age")
ax.set_title("Median age at first marriage in the US,
plt.show()
```


Median age at first marriage in the US, 1890 - 2010
Age
Year
20
21
22
23
24
25
26
27
28
1900
1920
1940
1960
1980
2000

Example 9:Visualizing a matrix with imshow

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
a = np.eye(10,10)
a += a[::-1,:]
fig = plt.figure()
ax1 = fig.add_subplot(121)
# Bilinear interpolation - this will look blurry
ax1.imshow(a, interpolation="bilinear",
      cmap=cm.Greys_r)
ax2 = fig.add_subplot(122)
# "nearest" interpolation - faithful but blocky
ax2.imshow(a, interpolation="nearest",
      cmap=cm.Greys_r)
plt.show()
```

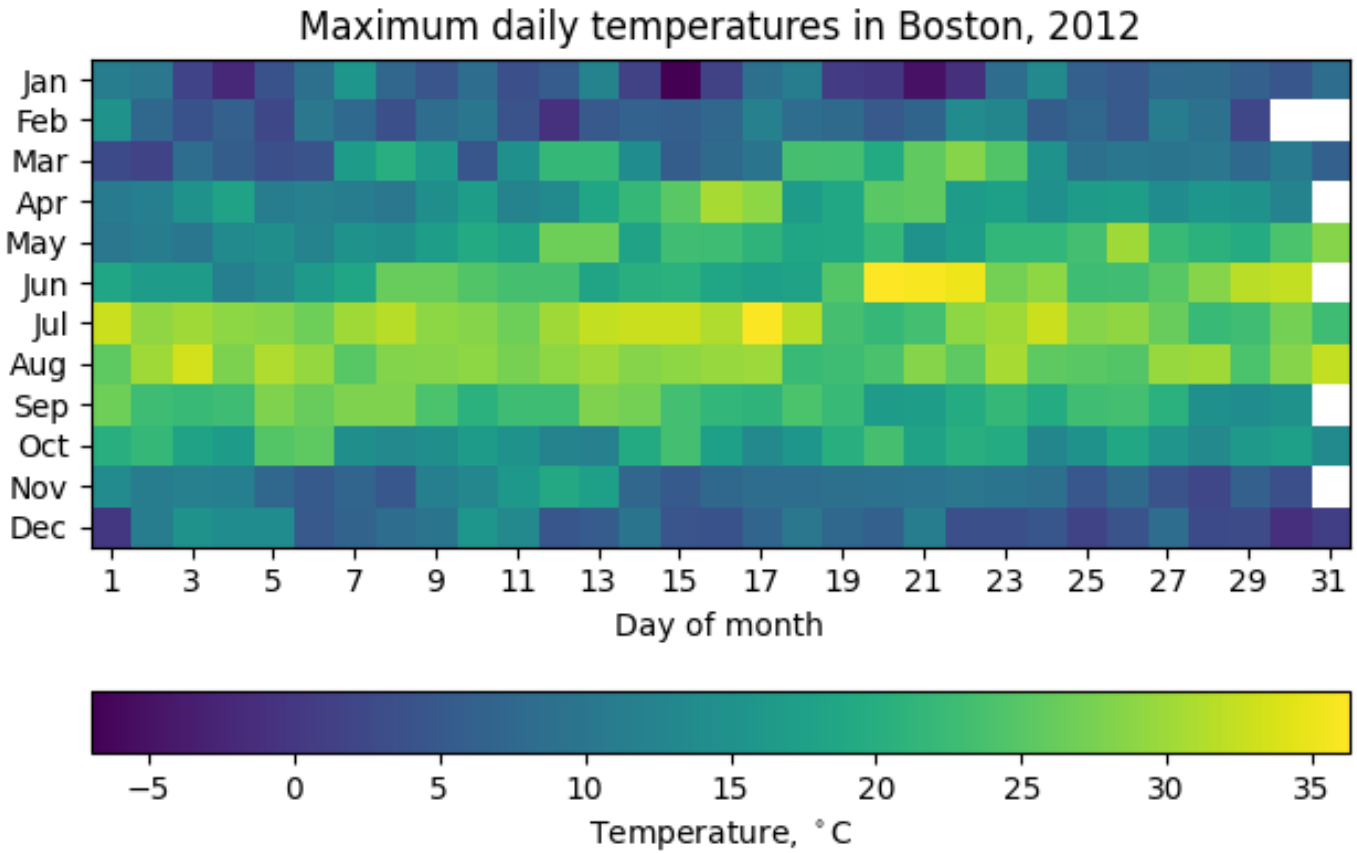

0
2
4
6
8
0
2
4
6
8
0
2
4
6
8
0
2
4
6
8

Example 10:A heatmap of Boston temperatures

```
import numpy as np
import matplotlib.pyplot as plt
dt = np.dtype([("month", np.int), ("day", np.int),
       ("T", np.float)])
data = np.genfromtxt("boston2012.dat", dtype=dt,
       usecols=(1,2,3),delimiter=(4,2,2,6))

heatmap = np.empty((12, 31))
heatmap[:] = np.nan
for month, day, T in data:
    heatmap[month-1, day-1] = T

fig = plt.figure()
ax = fig.add_subplot(111)
im = ax.imshow(heatmap, interpolation="nearest")
```

Example 9:...

```
ax.set_yticks(range(12))
ax.set_yticklabels(["Jan", "Feb", "Mar", "Apr",
  "May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"])
days = np.array(range(0, 31, 2))
ax.set_xticks(days)
ax.set_xticklabels(
  ["{:d}".format(day+1) for day in days])
ax.set_xlabel("Day of month")
ax.set_title("Maximum daily temp. in Boston")
# Add a colour bar along the bottom and label it
cbar=fig.colorbar(ax=ax,mappable=im,
        orientation="horizontal")
cbar.set_label("Temperature, $^\circ\mathrm{C}$")
plt.show()
```

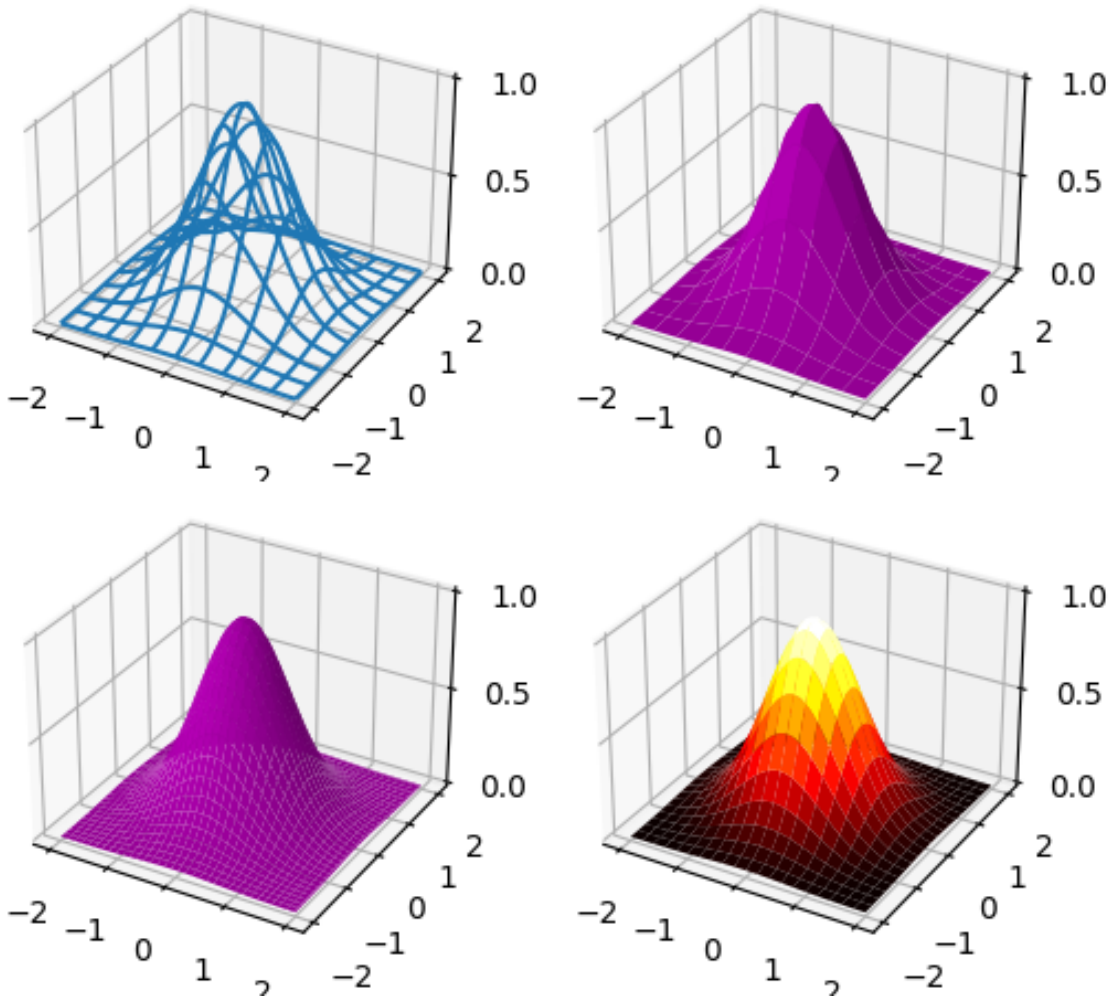

Maximum daily temperatures in Boston, 2012
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
Day of month
−5
0
5
10
15
20
25
30
35
Temperature, °C

Example 11: Simple surface plots

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm

L, n = 2, 400
x = np.linspace(-L, L, n)
y = x.copy()
X, Y = np.meshgrid(x, y)
Z = np.exp(-(X**2 + Y**2))

fig, ax = plt.subplots(nrows=2, ncols=2,
        subplot_kw={"projection": "3d"})
ax[0,0].plot_wireframe(X, Y, Z, rstride=40,
        cstride=40)
ax[0,1].plot_surface(X, Y, Z, rstride=40,
        cstride=40, color="m")
```

Example 10: ...

```
ax[1,0].plot_surface(X, Y, Z, rstride=12,
        cstride=12, color="m")
ax[1,1].plot_surface(X, Y, Z, rstride=20,
        cstride=20, cmap=cm.hot)
for axes in ax.flatten():
    axes.set_xticks([-2, -1, 0, 1, 2])
    axes.set_yticks([-2, -1, 0, 1, 2])
    axes.set_zticks([0, 0.5, 1])
fig.tight_layout()
plt.show()
```

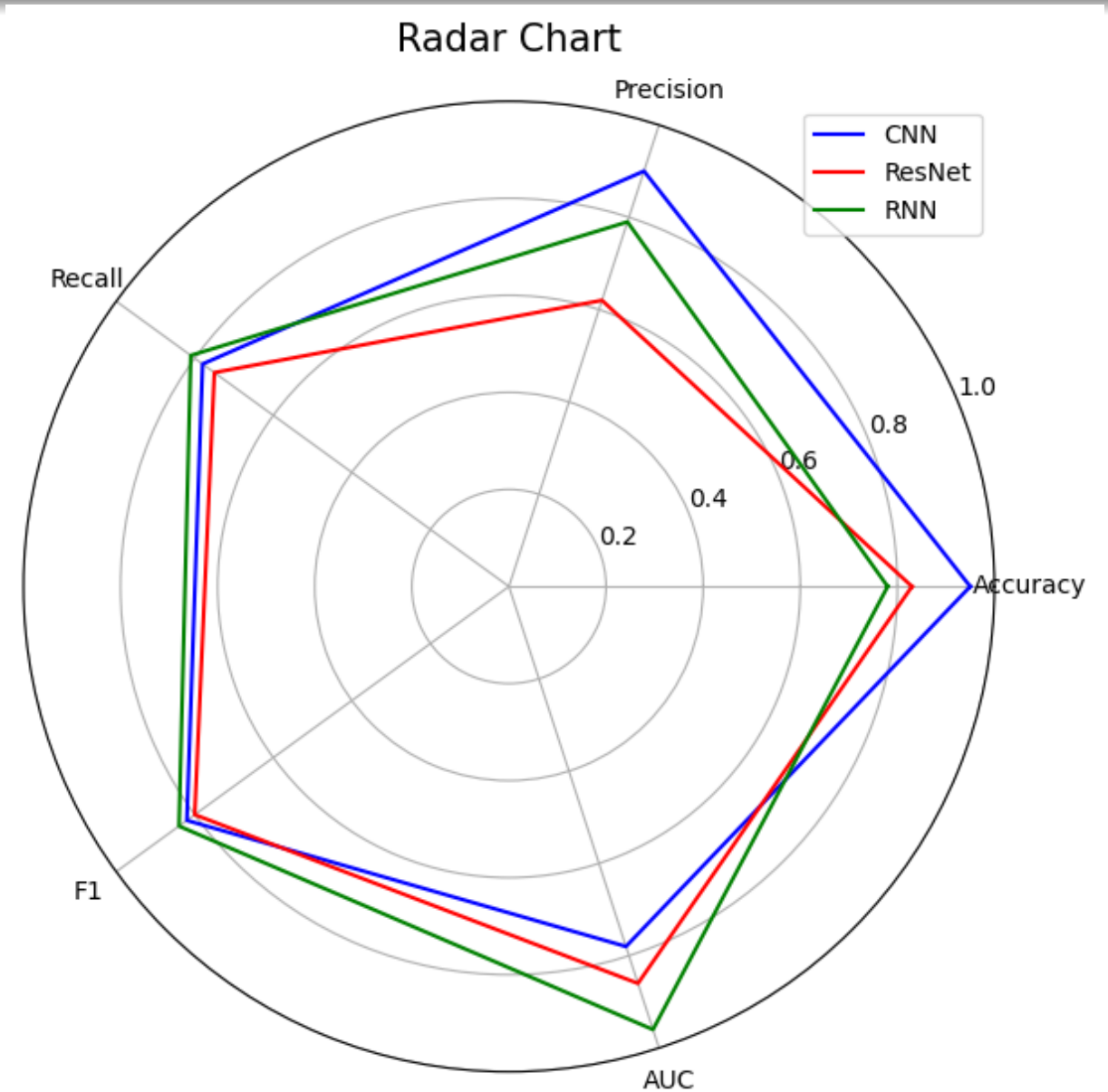

1.0
0.5
0.0
−2
−1
0
1
2
−2
−1
0
1
2

Example 12:

```
import matplotlib.pyplot as plt
import numpy as np
labels = ["Accuracy","Precision","Recall","F1","AUC"]
method1 = [0.95, 0.90, 0.78, 0.82, 0.78]
method2 = [0.83, 0.62, 0.75, 0.80, 0.86]
method3 = [0.78, 0.79, 0.81, 0.84, 0.96]
num_vars = len(labels)
angles = np.linspace(0, 2 * np.pi, num_vars,
  endpoint=False).tolist()
angles += angles[:1]
method1 += method1[:1]
method2 += method2[:1]
method3 += method3[:1]
fig, ax = plt.subplots(figsize=(7, 7),
  subplot_kw=dict(polar=True))
```

Example 12 ...

```
ax.plot(angles, method1, color="blue",label="CNN")
ax.plot(angles, method2, color="red",label="ResNet")
ax.plot(angles, method3, color="green",label="RNN")
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)
ax.set_ylim(0, 1)
plt.title("Radar Chart", size=15, pad=20)
plt.legend(loc="upper right")
plt.show()
```


Radar Chart
Precision
Recall
Accuracy
F1
AUC
CNN
ResNet
RNN
0.2
0.4
0.6
0.8
1.0

THE END