TEXT ANNOTATION

- The most basic types of annotations we will use are axes labels and titles, here we will see some more visualization and annotation information's.
- Text annotation can be done manually with the plt.text/ax.text command, which will place text at a particular x/y value.
- The ax.text method takes an x position, a y position, a string, and then optional keywords specifying the color, size, style, alignment, and other properties of the text. Here we used ha='right' and ha='center', where ha is short for horizontal alignment.

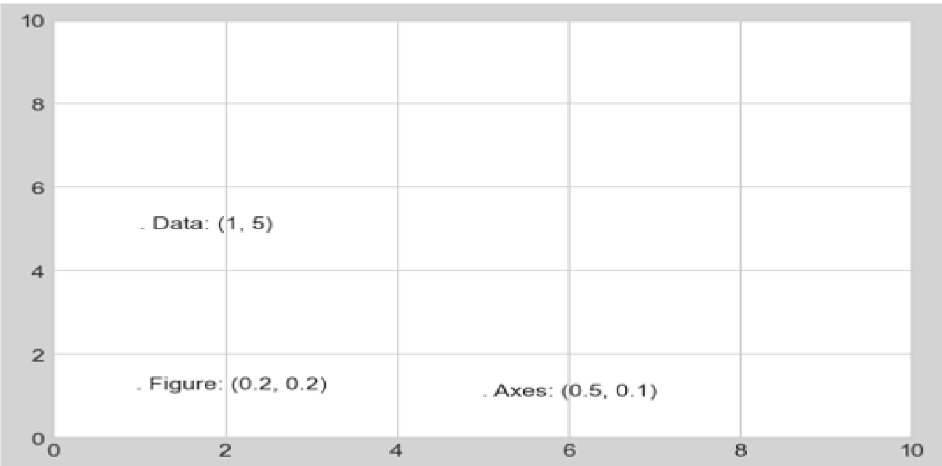
Transforms and Text Position

- We anchored our text annotations to data locations. Sometimes it's preferable to anchor the text to a position on the axes or figure, independent of the data. In Matplotlib, we do this by modifying the transform.
- Any graphics display framework needs some scheme for translating between coordinate systems.
- Mathematically, such coordinate transformations are relatively straightforward, and Matplotlib has a well- developed set of tools that it uses internally to perform them (the tools can be explored in the matplotlib.transforms submodule).
- There are three predefined transforms that can be useful in this situation.

  - ax.transData - Transform associated with data coordinates
  - ax.transAxes - Transform associated with the axes (in units of axes dimensions)
  - fig.transFigure - Transform associated with the figure (in units of figure dimensions)

Example

```
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.style.use('seaborn-whitegrid')
import numpy as np
import pandas as pd
fig, ax = plt.subplots(facecolor='lightgray')
ax.axis([0, 10, 0, 10])
# transform=ax.transData is the default, but we'll specify it anyway
ax.text(1, 5, ". Data: (1, 5)", transform=ax.transData)
ax.text(0.5, 0.1, ". Axes: (0.5, 0.1)", transform=ax.transAxes)
ax.text(0.2, 0.2, ". Figure: (0.2, 0.2)", transform=fig.transFigure);
```

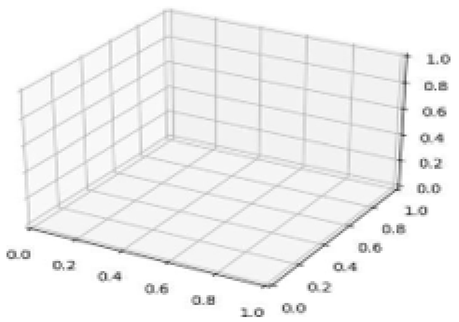


THREE DIMENSIONAL PLOTTING

We enable three-dimensional plots by importing the mplot3d toolkit, included with the main Matplotlib installation

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = plt.axes(projection='3d')
```

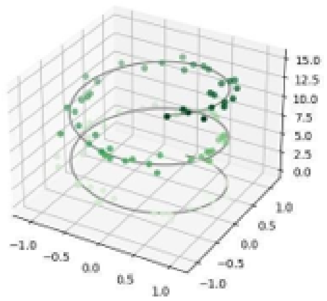
With this 3D axes enabled, we can now plot a variety of three-dimensional plot types.



Three-Dimensional Points and Lines

The most basic three-dimensional plot is a line or scatter plot created from sets of (x, y, z) triples. In analogy with the more common two-dimensional plots discussed earlier, we can create these using the ax.plot3D and ax.scatter3D functions.

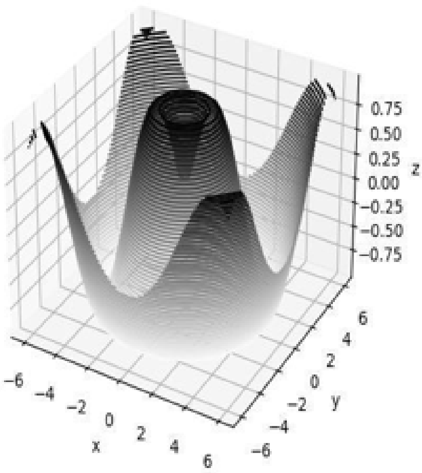
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
ax = plt.axes(projection='3d')
# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')
# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
```



Three-Dimensional Contour Plots

- mplot3d contains tools to create three-dimensional relief plots using the same inputs.
- Like two-dimensional ax.contour plots, ax.contour3D requires all the input data to be in the form of two- dimensional regular grids, with the Z data evaluated at each point.
- Here we'll show a three-dimensional contour diagram of a three dimensional sinusoidal function

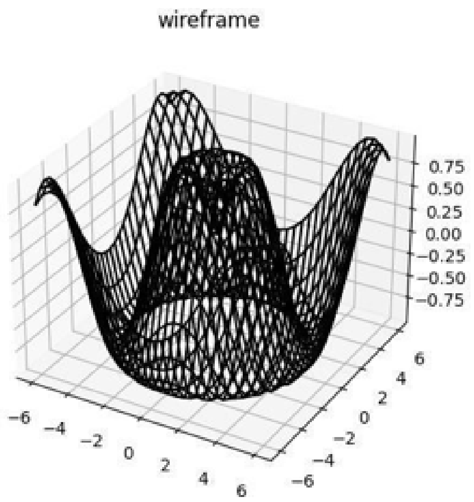
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
def f(x, y):
   return np.sin(np.sqrt(x ** 2 + y ** 2))
x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```



Wire frames and Surface Plots

- Two other types of three-dimensional plots that work on gridded data are wireframes and surface plots.
- These take a grid of values and project it onto the specified threedimensional surface, and can make the resulting three-dimensional forms quite easy to visualize.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, Z, color='black')
ax.set_title('wireframe');
plt.show()
```



- A surface plot is like a wireframe plot, but each face of the wireframe is a filled polygon.