numpyClass_Matplotlib

November 1, 2019

1 Matplotlib

Matplotlib is one of the librries commonly used for data visualization in Python. It is reccomended to visit the official Matplotlib web page: http://matplotlib.org/

1.1 Importing

Import the `matplotlib.pyplot` module under the name `plt` (the tidy way):

```
In [1]: import matplotlib.pyplot as plt
```

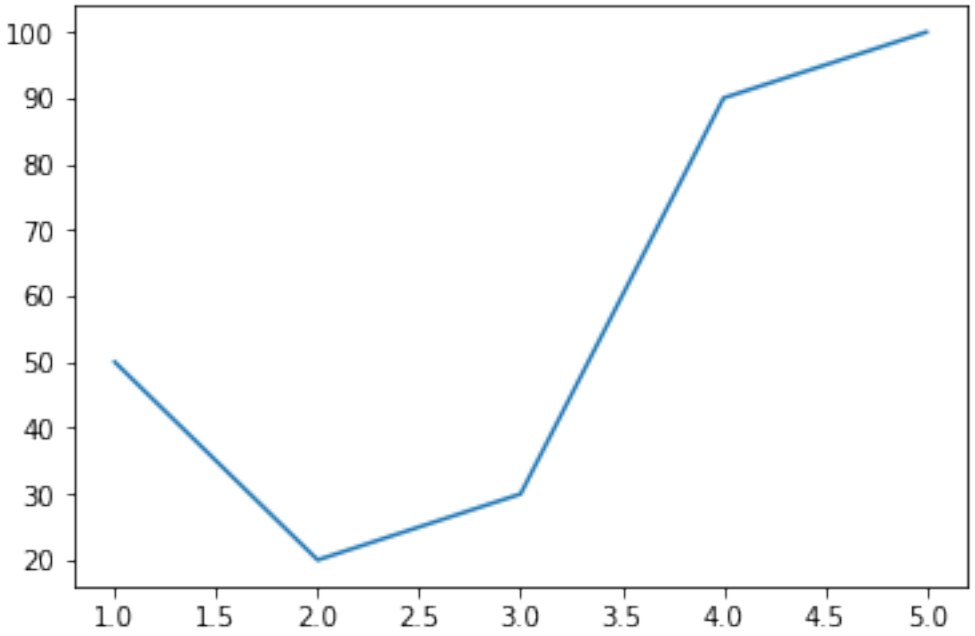
Write this line in Jupyter notebook:

```
In [2]: %matplotlib inline
```

This will produce all the plots genereted in the code, inside the notebook. This is *not* required in Spyder, etc.

2 Example 1

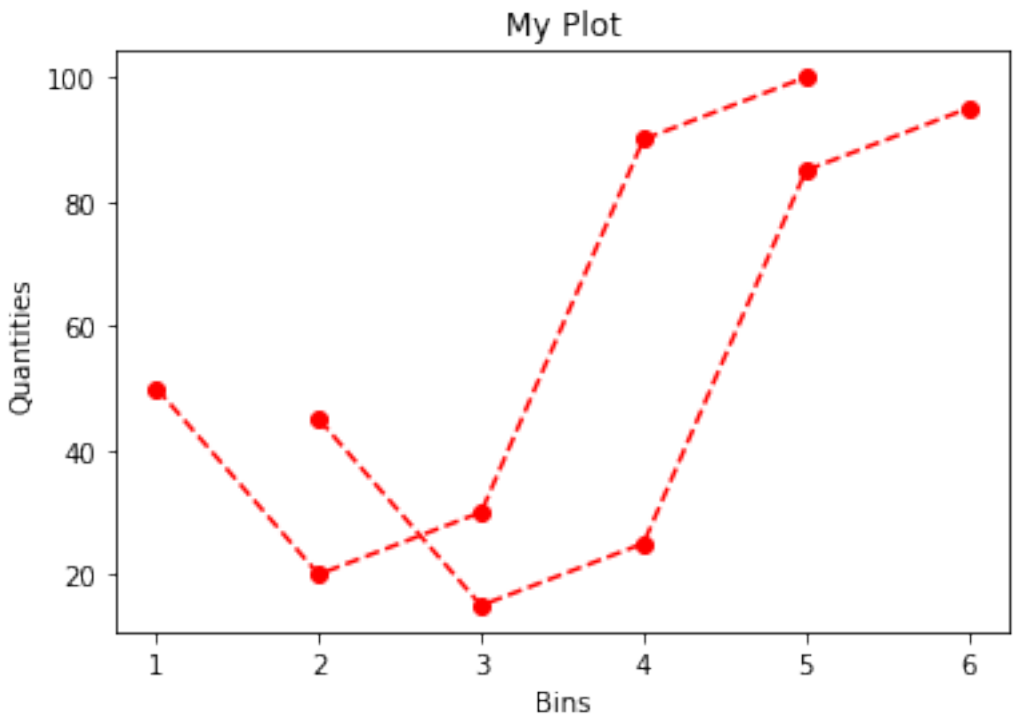
```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
        #Used only in Jupyter Notebook to get inline plots:
        %matplotlib inline
        x = np.array([1, 2, 3, 4, 5])
        y = np.array([50, 20, 30, 90, 100])
        plt.plot (x,y)
        plt.show()
        print(x)
```




```
[1 2 3 4 5]
```

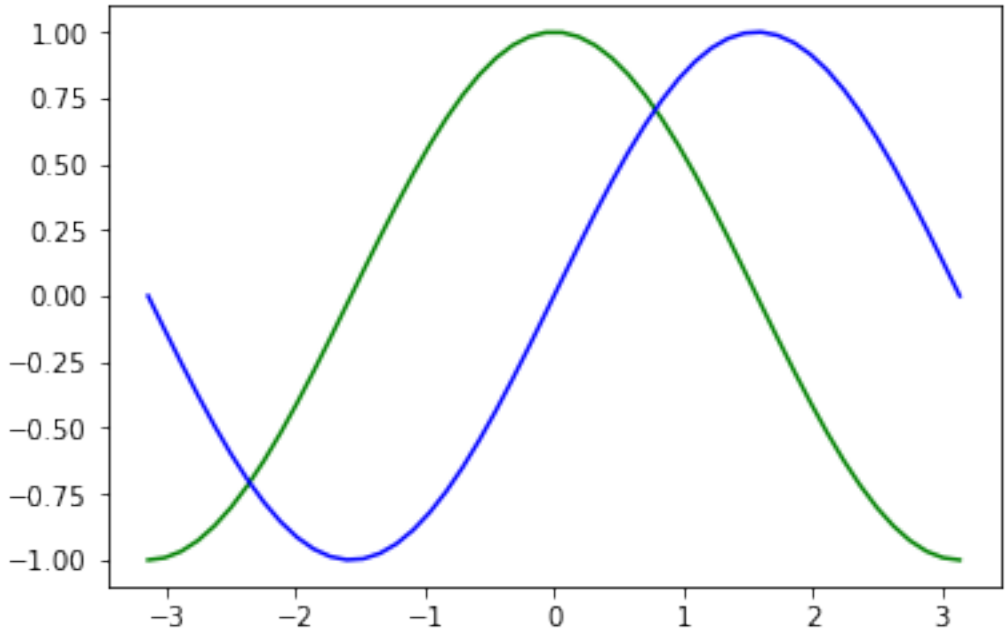
2.1 Line Style

```
In [4]: x=np.array([1,2,3,4,5])
        y=np.array([50,20,30,90,100])
        plt.plot(x, y, color="red", ls='--', marker='o')
        plt.plot(x+1, y-5, color="red", ls='--', marker='o')
        plt.xlabel('Bins')
        plt.ylabel('Quantities')
        plt.title('My Plot')
        plt.show()
```



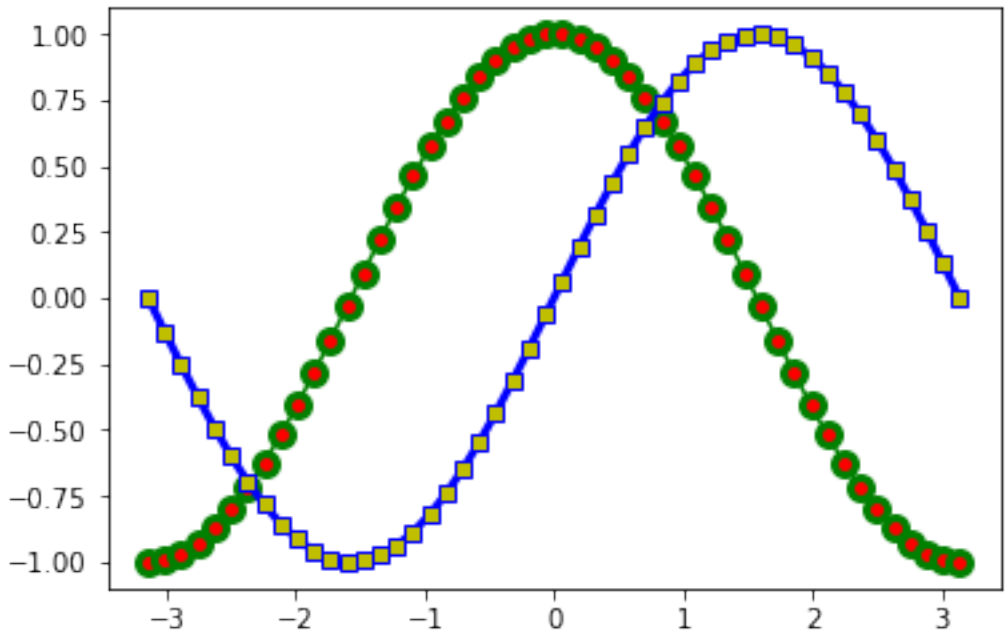
2.2 Sin and Cos

```
In [5]: x = np.linspace(-np.pi, np.pi)
        y, z = np.cos(x), np.sin(x)

        plt.plot(x, y, color = 'green')
        plt.plot(x, z, color = 'blue')
        plt.show()
```



2.2.1 Sin and Cos with Line Styles

```
In [6]: x = np.linspace(-np.pi, np.pi)
        y, z = np.cos(x), np.sin(x)
        plt.plot(x, y, color = 'green', ls='-',
                 marker='o',markersize=8,
                 markeredgecolor="green", markeredgewidth=3,
                 markerfacecolor="red")
        plt.plot(x, z, color = 'b', ls='-',lw=3,marker='s',
                 markerfacecolor="y")

        plt.show()
```

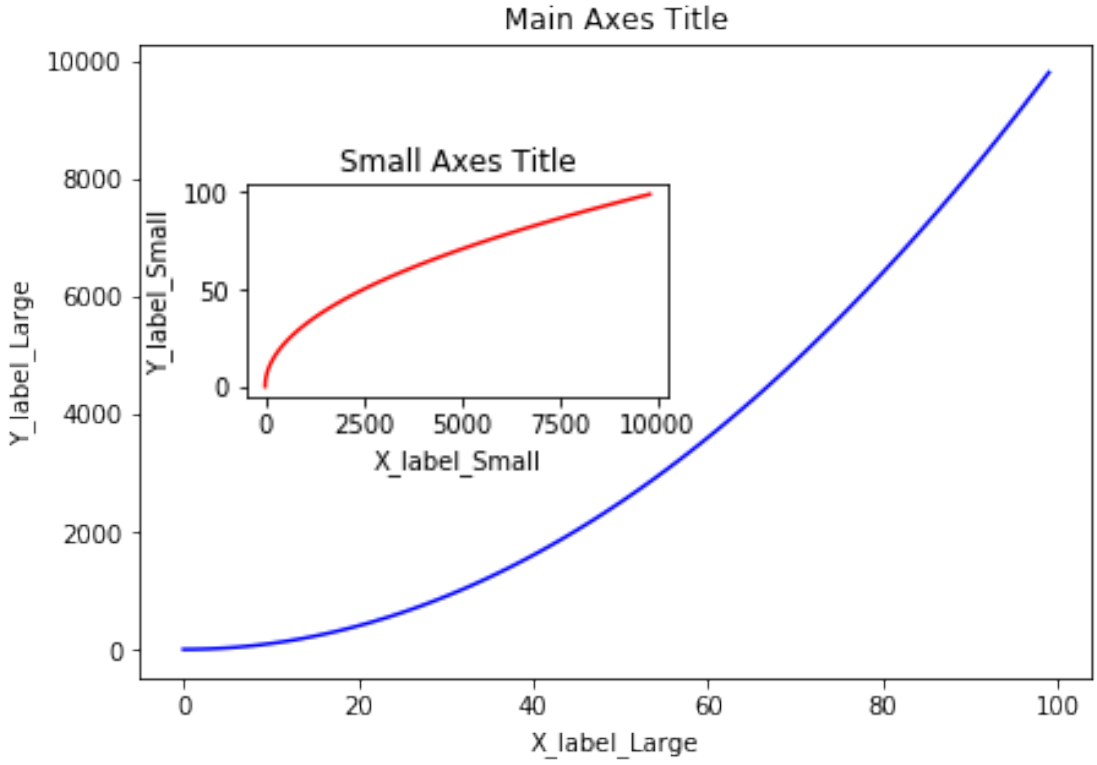


2.3 Multiple Axes

Code is a little more complicated, but the advantage is that we now have full control of where the plot axes are placed, and we can easily add more than one axis to the figure:

```
In [7]: x=np.arange(100)
        y=x**2
        # Creates blank canvas
        fig = plt.figure()
        print("type of fig:" , type(fig))

        outer = fig.add_axes([0.1, 0.1, 0.9, 0.9]) # main axes
        inner = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # "inner" axes

        # Larger Figure Axes 1
        outer.plot(x, y, 'b')
        print("type(outer)=", type(outer))
        outer.set_xlabel('X_label_Large')
        outer.set_ylabel('Y_label_Large')
        outer.set_title('Main Axes Title')

        # Insert Figure Axes 2
        inner.plot(y, x, 'r')
        inner.set_xlabel('X_label_Small')
        inner.set_ylabel('Y_label_Small')
        inner.set_title('Small Axes Title');
```

```
type of fig: <class 'matplotlib.figure.Figure'>
type(outer)= <class 'matplotlib.axes._axes.Axes'>
```



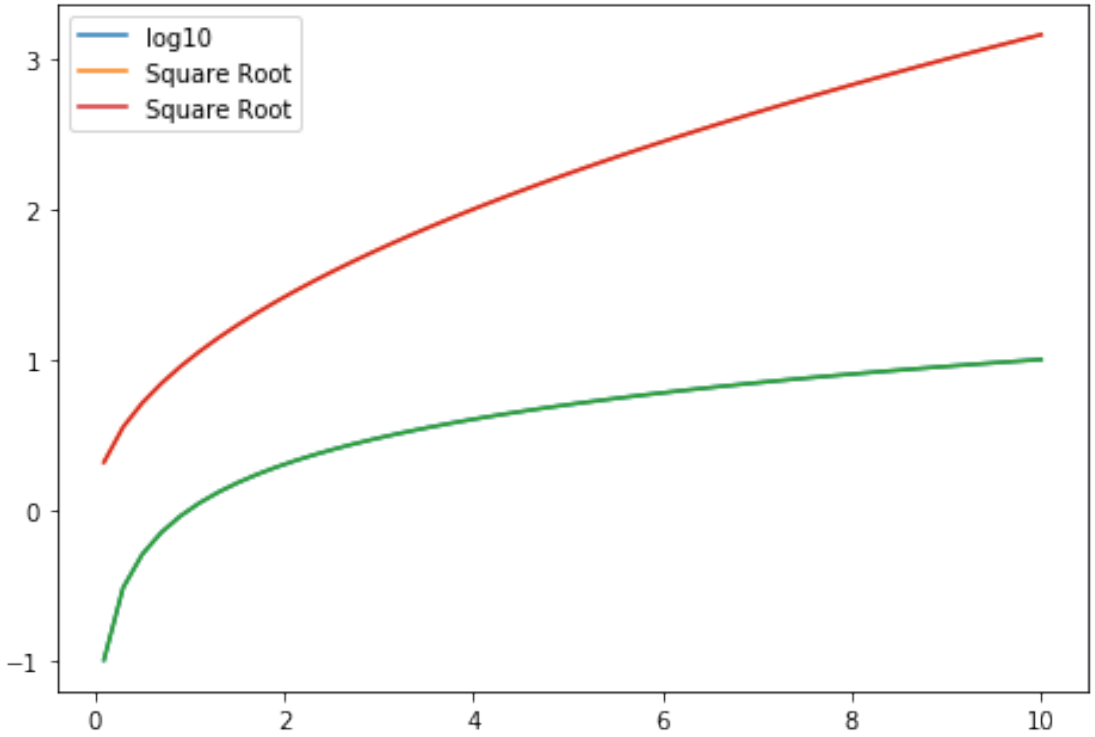
2.4 Legends and Labels

2.4.1 Legends

```
In [8]: fig = plt.figure()
        x=np.linspace(0.1,10.0,num=50)
        ax = fig.add_axes([0,0,1,1])
        import math
        ax.plot(x, np.log10(x), label="log10")
        ax.plot(x, x**0.5, label="Square Root")
        ax.legend

        """
        fig2.plot(x, np.log10(x), label="log10") # NO PLOT FOR FIGURE
        fig2.plot(x, x**0.5, label="Square Root")
        fig2.legend
        """
```

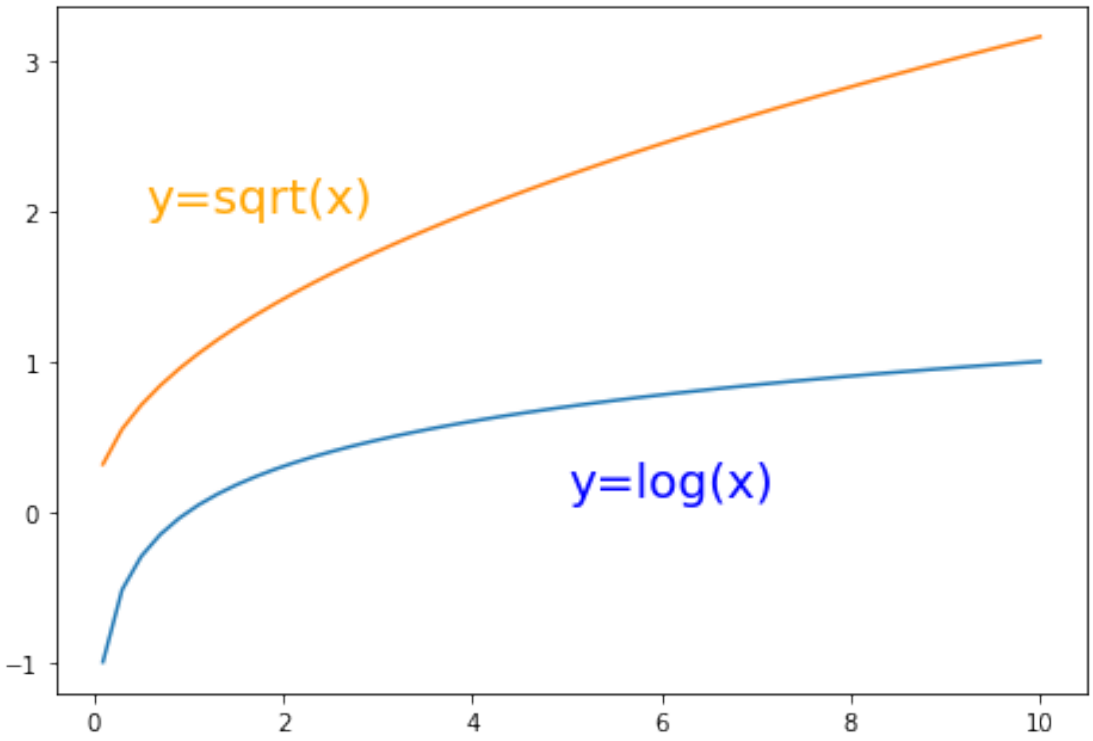
```
        plt.plot(x, np.log10(x))
        plt.plot(x, x**0.5, label="Square Root")
        #Without axes - only 1 legend...
        plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x20c52abe518>



In [9]:
```
fig = plt.figure()
x=np.linspace(0.1,10.0,num=50)
ax = fig.add_axes([0,0,1,1])
import math
ax.plot(x, np.log10(x), label="log10")
ax.plot(x, x**0.5, label="Square Root")
#ax.text(0.55, 2, r"$y=sqrt(x)$", fontsize=20, color="orange")
#ax.text(5, 0.1, r"$y=log(x)$", fontsize=20, color="blue");
ax.text(0.55, 2, "y=sqrt(x)", fontsize=20, color="orange")
ax.text(5, 0.1, "y=log(x)", fontsize=20, color="blue")
```

Out[9]: Text(5, 0.1, 'y=log(x)')



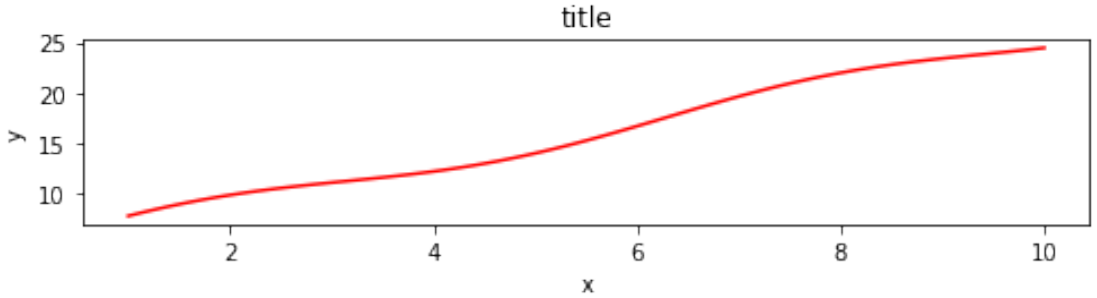
2.5 Figure size & Saving Figures

```
In [41]: """
         fig, axes = plt.subplots(figsize=(5,3))
         x=np.linspace(1.0,10.0,num=100)
         y=2*x+np.sin(x)+5
         axes.plot(x, y, 'r')
         axes.set_xlabel('x')
         axes.set_ylabel('y')
         axes.set_title('title');
         """

         fig = plt.figure()
         plt.plot(figsize=(15,3))
         ax = fig.add_axes([0,0,10,10])
         x=np.linspace(0.0,10.0,num=100)
         y=2*x+np.sin(x)+5
         ax.plot(x, x+5, 'g')
         #ax.plot(x, y**2, 'r')
         #plt.show()
         """
         plt.set_xlabel('x')
         plt.set_ylabel('y')
```

```
        plt.set_title('title');
        """
```

Out[41]: "\nplt.set_xlabel('x')\nplt.set_ylabel('y')\nplt.set_title('title');\n"

In [16]:
```
fig, axes = plt.subplots(figsize=(8,1.5))
x=np.linspace(1.0,10.0,num=100)
y=2*x+np.sin(x)+5
axes.plot(x, y, 'r')
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_title('title');
```

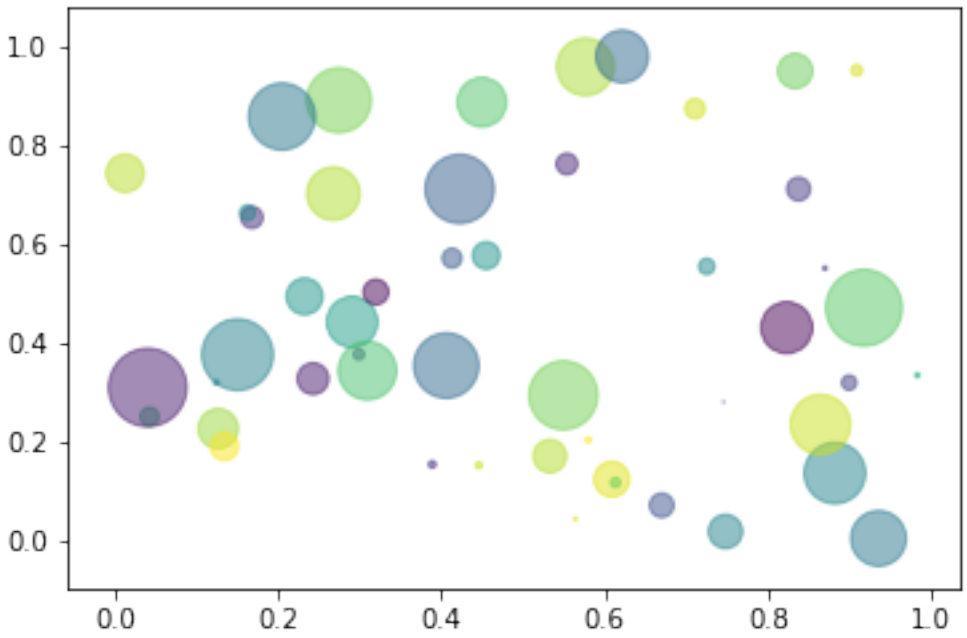


2.6 Saving figures

```
In [17]: fig.savefig("filename1.png")

In [18]: fig.savefig("filename2.pdf", dpi=150)
```

3 Some Nice (and Useful!) Plots

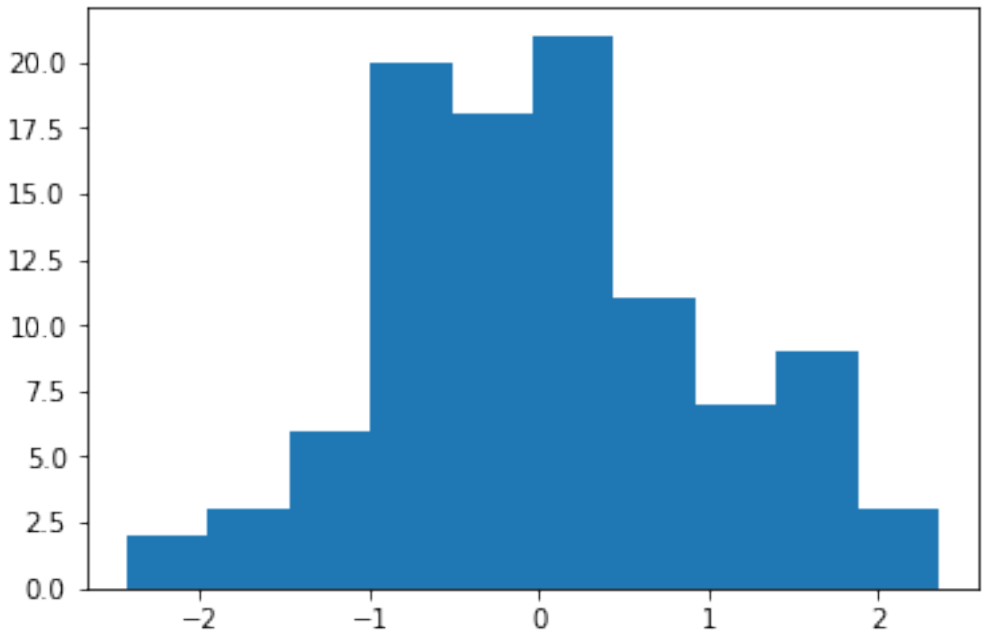
```
In [22]: import random

         N = 50
         x = np.random.rand(N)
         y = np.random.rand(N)
         colors = np.random.rand(N)
         area = (30 * np.random.rand(N))**2

         plt.scatter(x, y, s=area, c=colors, alpha=0.5)
         plt.show()
```



3.1 Histogram

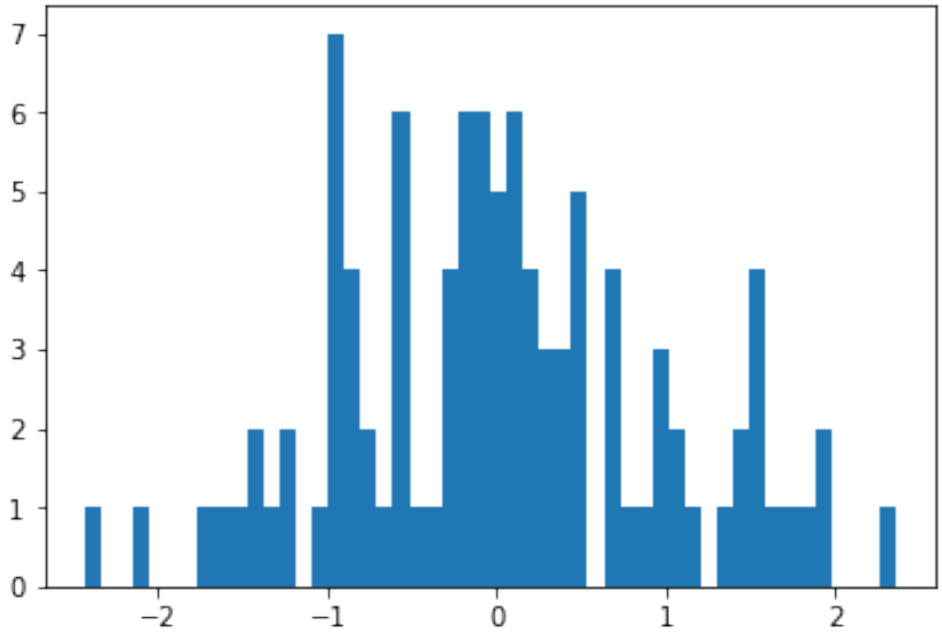
```
In [25]: import random
         vals = np.random.randn(100)
         plt.hist(vals, bins=10)
```

```
Out[25]: (array([ 2.,  3.,  6., 20., 18., 21., 11.,  7.,  9.,  3.]),
          array([-2.43603992, -1.95654028, -1.47704065, -0.99754101, -0.51804137,
                 -0.03854174,  0.4409579 ,  0.92045754,  1.39995718,  1.87945681,
                  2.35895645]),
          <a list of 10 Patch objects>)
```




```
In [26]: plt.hist(vals, bins=50)
```

```
Out[26]: (array([1., 0., 0., 1., 0., 0., 0., 1., 1., 1., 2., 1., 2., 0., 1., 7., 4.,
                 2., 1., 6., 1., 1., 4., 6., 6., 5., 6., 4., 3., 3., 5., 0., 4., 1.,
                 1., 3., 2., 1., 0., 1., 2., 4., 1., 1., 1., 2., 0., 0., 0., 1.]),
          array([-2.43603992, -2.34013999, -2.24424007, -2.14834014, -2.05244021,
                 -1.95654028, -1.86064036, -1.76474043, -1.6688405 , -1.57294057,
                 -1.47704065, -1.38114072, -1.28524079, -1.18934086, -1.09344094,
                 -0.99754101, -0.90164108, -0.80574115, -0.70984123, -0.6139413 ,
                 -0.51804137, -0.42214144, -0.32624152, -0.23034159, -0.13444166,
                 -0.03854174,  0.05735819,  0.15325812,  0.24915805,  0.34505797,
                  0.4409579 ,  0.53685783,  0.63275776,  0.72865768,  0.82455761,
                  0.92045754,  1.01635747,  1.11225739,  1.20815732,  1.30405725,
                  1.39995718,  1.4958571 ,  1.59175703,  1.68765696,  1.78355689,
                  1.87945681,  1.97535674,  2.07125667,  2.1671566 ,  2.26305652,
                  2.35895645]),
          <a list of 50 Patch objects>)
```



3.2 3D Plots

```
In [155]: from mpl_toolkits.mplot3d.axes3d import Axes3D
```

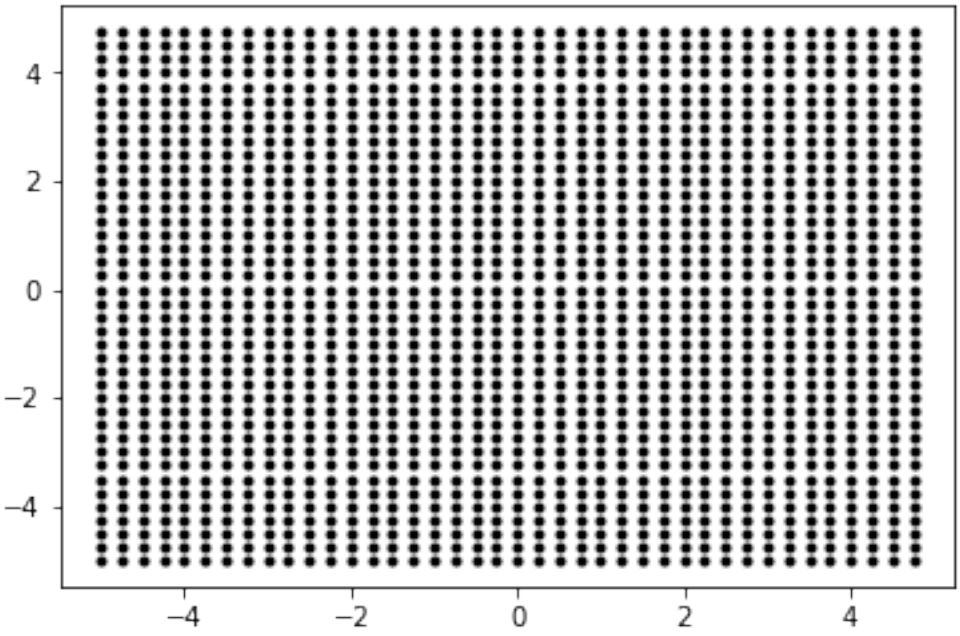
3.3 Meshgrid

```
In [148]: # Make data.
          XX = np.arange(-5, 5, 0.25)
          YY = np.arange(-5, 5, 0.25)
          X, Y = np.meshgrid(XX, YY)
          R = np.sqrt(X**2 + Y**2)
          Z=np.sin(R)
          Z2=np.sqrt(R)
          X

Out[148]: array([[-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75],
                 [-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75],
                 [-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75],
                 ...,
                 [-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75],
                 [-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75],
                 [-5.  , -4.75, -4.5 , ...,  4.25,  4.5 ,  4.75]])

In [137]: Y

Out[137]: array([[-5.  , -5.  , -5.  , ..., -5.  , -5.  , -5.  ],
                 [-4.75, -4.75, -4.75, ..., -4.75, -4.75, -4.75],
```

```
                [-4.5 , -4.5 , -4.5 , ..., -4.5 , -4.5 , -4.5 ],
                ...,
                [ 4.25,  4.25,  4.25, ...,  4.25,  4.25,  4.25],
                [ 4.5 ,  4.5 ,  4.5 , ...,  4.5 ,  4.5 ,  4.5 ],
                [ 4.75,  4.75,  4.75, ...,  4.75,  4.75,  4.75]])
```

```
In [139]: plt.plot(X,Y, marker='.', color='k', linestyle='none')
```

```
Out[139]: [<matplotlib.lines.Line2D at 0x16aa2606e48>,
           <matplotlib.lines.Line2D at 0x16aa2606f98>,
           <matplotlib.lines.Line2D at 0x16aa26067f0>,
           <matplotlib.lines.Line2D at 0x16aa26062e8>,
           <matplotlib.lines.Line2D at 0x16aa2606470>,
           <matplotlib.lines.Line2D at 0x16aa164bda0>,
           <matplotlib.lines.Line2D at 0x16aa2215ef0>,
           <matplotlib.lines.Line2D at 0x16aa22153c8>,
           <matplotlib.lines.Line2D at 0x16aa2215e80>,
           <matplotlib.lines.Line2D at 0x16aa2215eb8>,
           <matplotlib.lines.Line2D at 0x16aa2215550>,
           <matplotlib.lines.Line2D at 0x16aa22154e0>,
           <matplotlib.lines.Line2D at 0x16aa2215048>,
           <matplotlib.lines.Line2D at 0x16aa22158d0>,
           <matplotlib.lines.Line2D at 0x16aa2215ba8>,
           <matplotlib.lines.Line2D at 0x16aa2215c18>,
           <matplotlib.lines.Line2D at 0x16aa2205cf8>,
           <matplotlib.lines.Line2D at 0x16aa2205668>,
           <matplotlib.lines.Line2D at 0x16aa2205710>,
           <matplotlib.lines.Line2D at 0x16aa2205518>,
           <matplotlib.lines.Line2D at 0x16aa2205be0>,
           <matplotlib.lines.Line2D at 0x16aa22051d0>,
           <matplotlib.lines.Line2D at 0x16aa22055f8>,
           <matplotlib.lines.Line2D at 0x16aa2205d30>,
           <matplotlib.lines.Line2D at 0x16aa2205780>,
           <matplotlib.lines.Line2D at 0x16aa22054e0>,
           <matplotlib.lines.Line2D at 0x16aa2205278>,
           <matplotlib.lines.Line2D at 0x16aa2205128>,
           <matplotlib.lines.Line2D at 0x16aa20b4748>,
           <matplotlib.lines.Line2D at 0x16aa20b40f0>,
           <matplotlib.lines.Line2D at 0x16aa20b4b70>,
           <matplotlib.lines.Line2D at 0x16aa20b4a58>,
           <matplotlib.lines.Line2D at 0x16aa20b4828>,
           <matplotlib.lines.Line2D at 0x16aa20b4630>,
           <matplotlib.lines.Line2D at 0x16aa20b44a8>,
           <matplotlib.lines.Line2D at 0x16aa20b4f60>,
           <matplotlib.lines.Line2D at 0x16aa20b4e10>,
           <matplotlib.lines.Line2D at 0x16aa20b4080>,
           <matplotlib.lines.Line2D at 0x16aa20b4d30>,
           <matplotlib.lines.Line2D at 0x16aa1561be0>]
```



In [159]:
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


fig = plt.figure()
ax = fig.gca(projection='3d')

# Make data.
XX = np.arange(-5, 5, 0.25)
YY = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(XX, YY)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
```

```
# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```