Matplotlib

Matplotlib

- Python 2D plotting library
- Line plots, histograms, scatter plots etc
- Various rendering backends
  - Display, Raster/Vector images, PDF

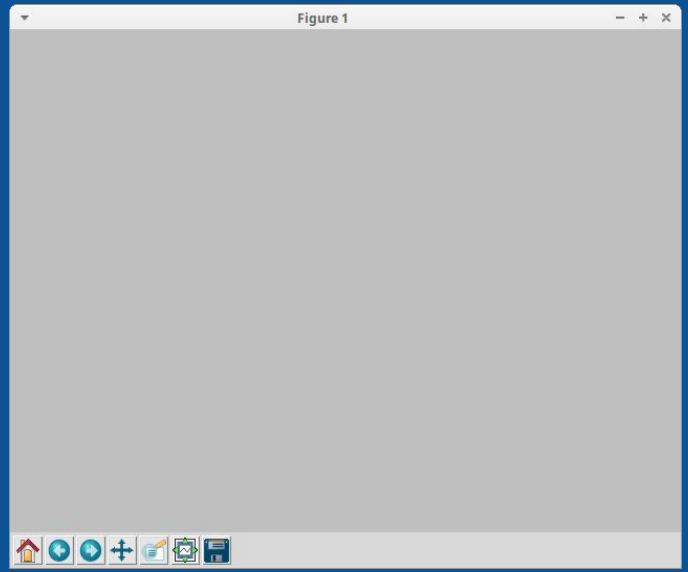
Matplotlib Interfaces

- `pylab`
  - MATLAB like interface
- Artists Interface
  - Object oriented interface
  - More fine-grained control than pylab
- Combination of both

Typical Sequence of Operations

- Create a figure instance
  - Window that will contain subplots

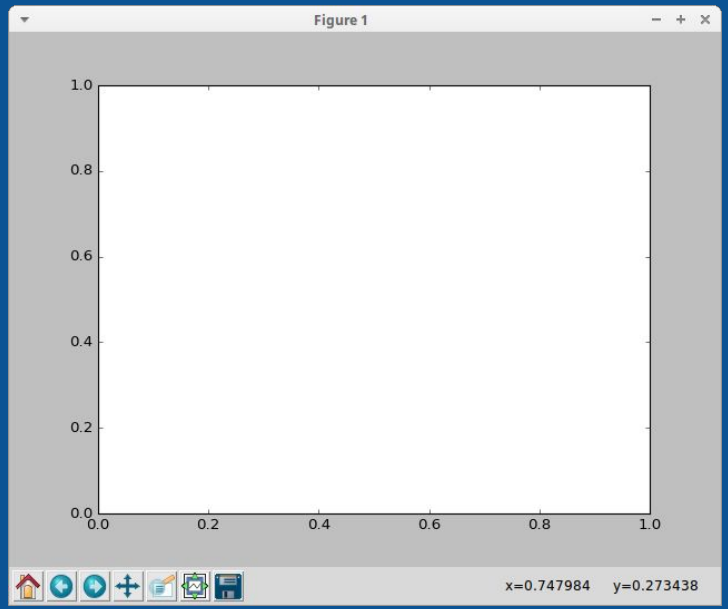
```
import matplotlib.pyplot as plt

fig = plt.figure()
plt.show()
```



```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
plt.show()
```

Typical Sequence of Operations

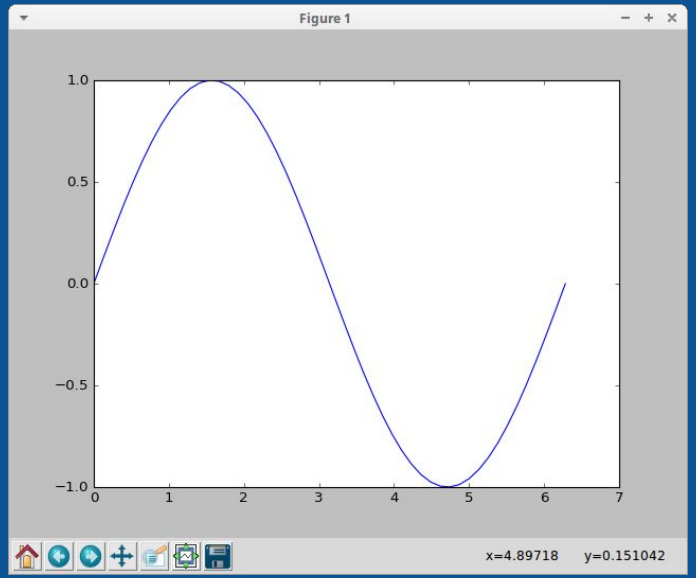
- One or more `Axes` objects are associated with the figure object
- Using `add_subplot()` convenience methods



Typical Sequence of Operations

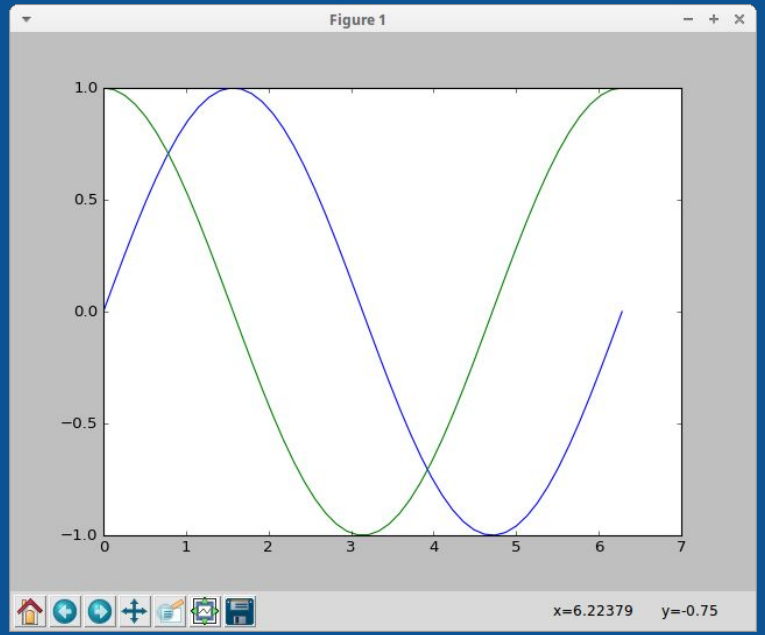
- Each graphical element in the plot (line, text, polygon, etc.) is represented by a class derived from the `Artist` class
- And is associated with an `Axes` object
- Axes object has helper methods to create these objects and associate them with the axes

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
ax.plot(x, y)

plt.show()
```



```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y1 = np.cos(x)
ax.plot(x, y)
ax.plot(x, y1)
print(ax.lines)
plt.show()
```

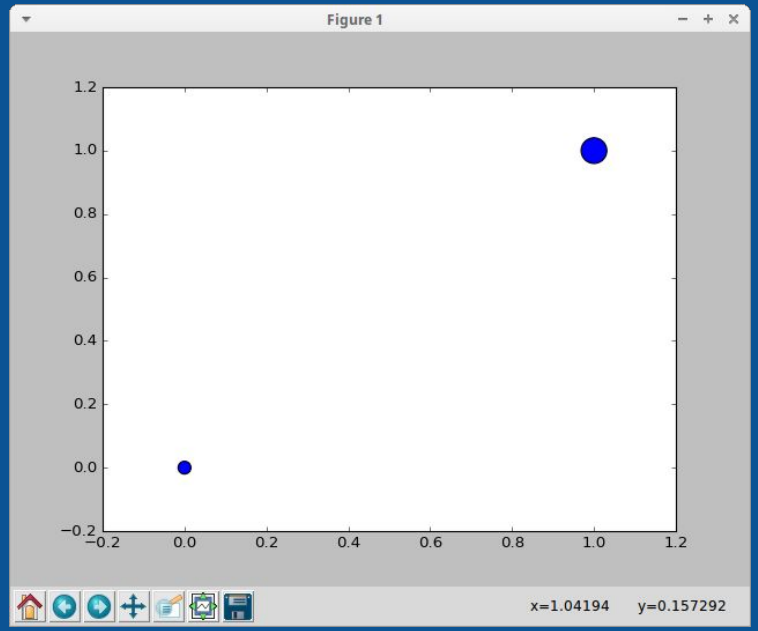
```
$ python3 artist4.py
[<matplotlib.lines.Line2D object at
0x7fd35c77ba90>, <matplotlib.lines.Line2D
object at 0x7fd3597dcf28>]
```



```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

ax.scatter([0,1], [0,1], [100, 400])
plt.show()
```

Scatter Plot

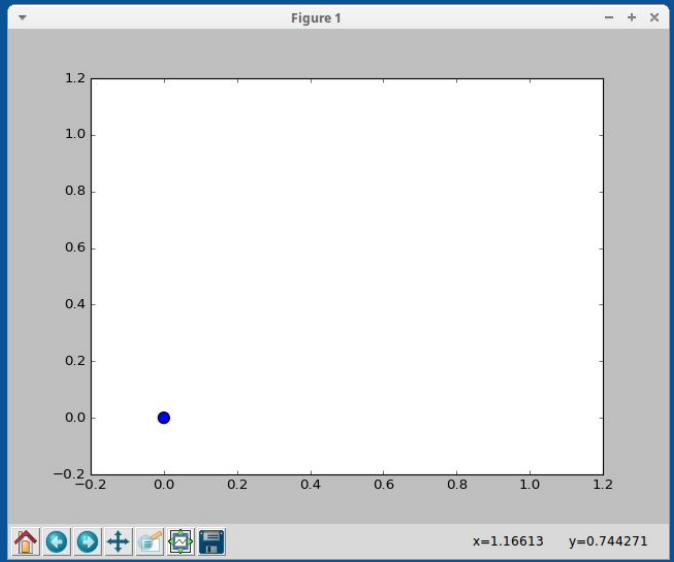
- scatter(x, y, size…)



```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

s = ax.scatter([0,1], [0,1], [100, 400])
s.set_offsets([[0,0], [1,-1]])

plt.show()
```

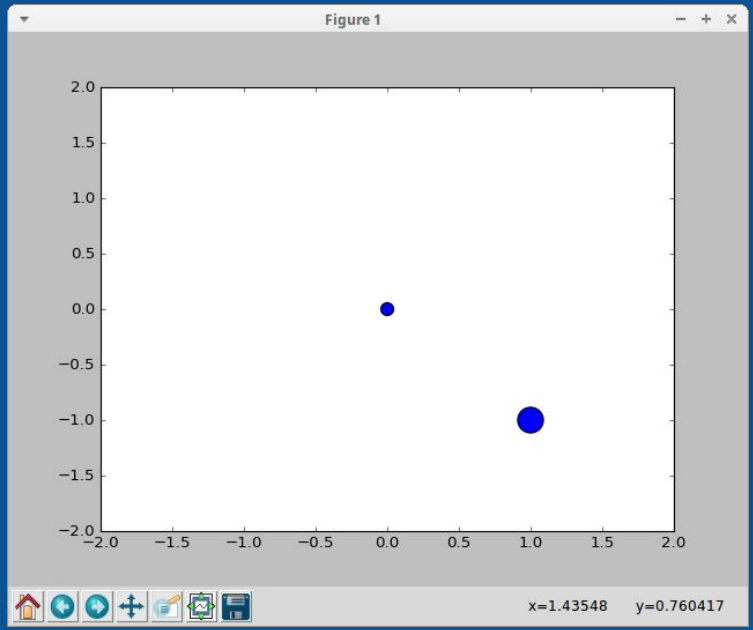
Modify an graphical object

- using methods of the scatter object
- Coordinates specified differently



Modify Axes properties

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(1,1,1)

s = ax.scatter([0,1], [0,1], [100, 400])
s.set_offsets([[0,0], [1,-1]])
ax.set_xlim(-2,2)
ax.set_ylim(-2,2)

plt.show()
```



Animating with Matplotlib

- `matplotlib.animation` module
- `FuncAnimation()` class
  - `__init__(self, fig, func, frames=None, init_func=None, fargs=None, save_count=None, **kwargs)`

```
import scipy as sp
import matplotlib.pyplot as plt
import matplotlib.animation as animation

def sincos(deg):
    fact = sp.pi/180
    x, y = sp.sin(deg*fact), sp.cos(deg*fact)
    dots.set_offsets(sp.array([[0,0], [x,y]], dtype=float))


fig = plt.figure()
ax = fig.add_subplot(111, xlim=(-1,1), ylim=(-1,1))
ax.set_xticks([])
ax.set_xticklabels([])
ax.set_xticks([])
ax.set_xticklabels([])
dots = ax.scatter([], [], [100,100])
ani = animation.FuncAnimation(fig, sincos, frames=sp.arange(0,360), interval=10, repeat=True)
plt.show()
```

Misc Hints

- Scale/Size
- Use classes
- Use `dir()` and `help()` to find what methods are available for different Artist objects and what they do
- Extensive documentation and examples on the matplotlib web site