Plotting and Visualization

Part 1

- To disable auto-scrolling, execute this javascript in a notebook cell before other cells are executed:

In [2]:
```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

- The simplest way to follow the code examples is to use interactive plotting in the Jupyter notebook.
- To set this up, execute the following statement in a Jupyter notebook:

```
In [3]: %matplotlib notebook
```

A Brief matplotlib API Primer

Part 1

- With matplotlib, we use the following import convention:

```
In [4]: import matplotlib.pyplot as plt
```

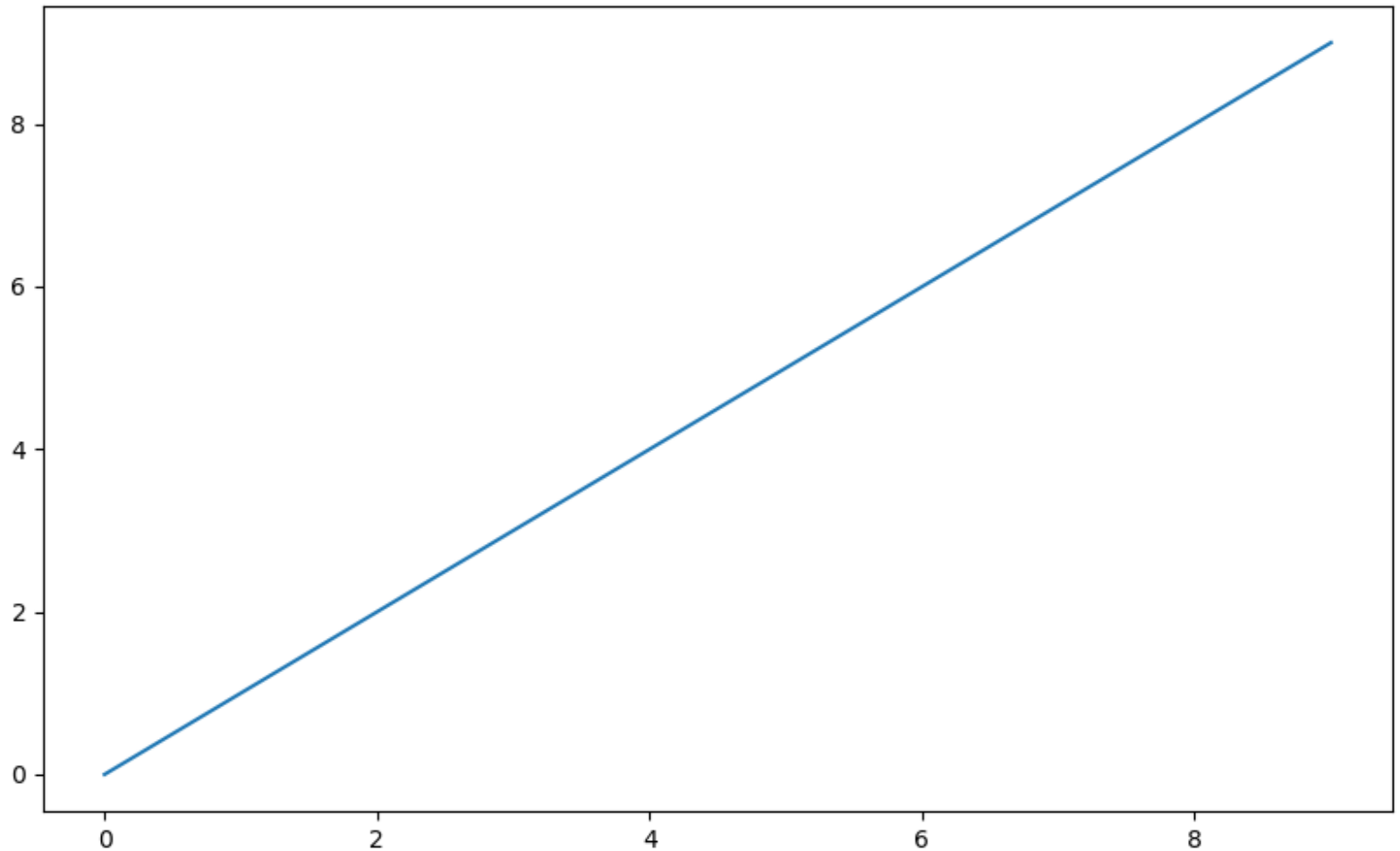
```
In [5]: import numpy as np

In [6]: data = np.arange(10)

In [7]: data
Out[7]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [8]:
```
plt.plot(data)
```

Figure 1



Figures and Subplots

- Plots in matplotlib reside within a `Figure` object.
- You can create a new figure with `plt.figure`.

In [9]:
```
fig = plt.figure()
```

- In IPython, an empty plot window will appear, but in Jupyter nothing will be shown until we use a few more commands.
- `plt.figure` has a number of options; notably, `figsize` will guarantee the figure has a certain size and aspect ratio if saved to disk.

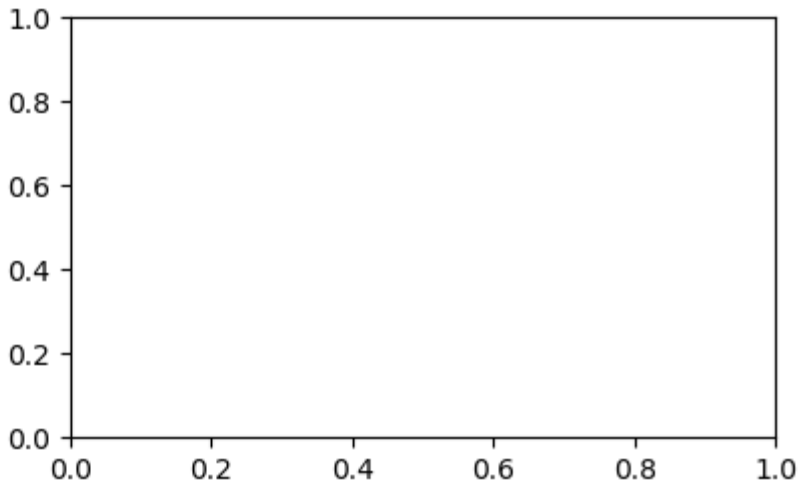
- You can’t make a plot with a blank figure.
- You have to create one or more subplots using `add_subplot`:

```
In [10]: ax1 = fig.add_subplot(2, 2, 1)
```

- This means that the figure should be 2 × 2 (so up to four plots in total), and we’re selecting the first of four subplots (numbered from 1).

In [9]:
```
fig = plt.figure()
```

Figure 2

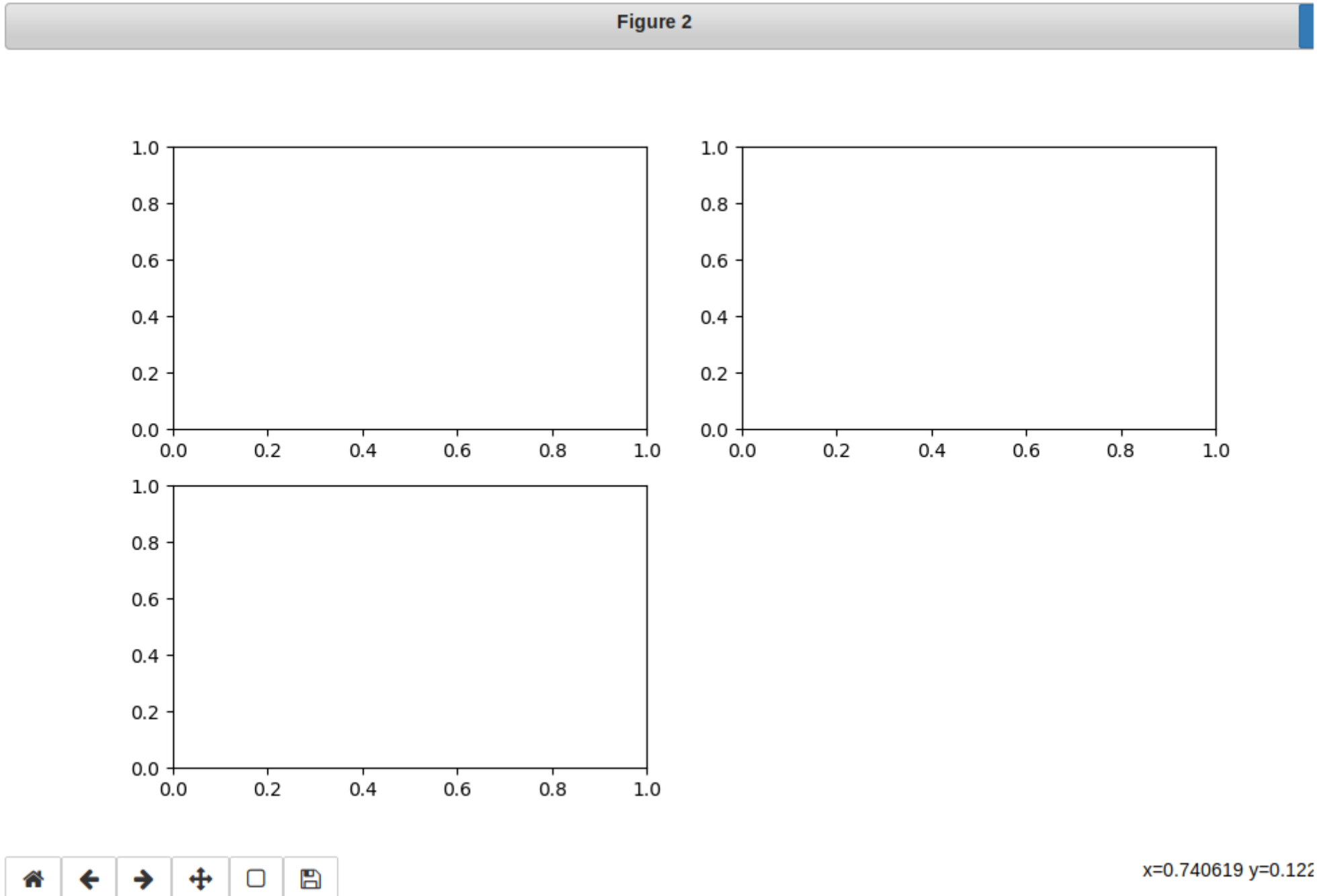


Download

- We can create two more two subplots.

```
In [11]: ax2 = fig.add_subplot(2, 2, 2)
         ax3 = fig.add_subplot(2, 2, 3)
```

```
In [9]: fig = plt.figure()
```

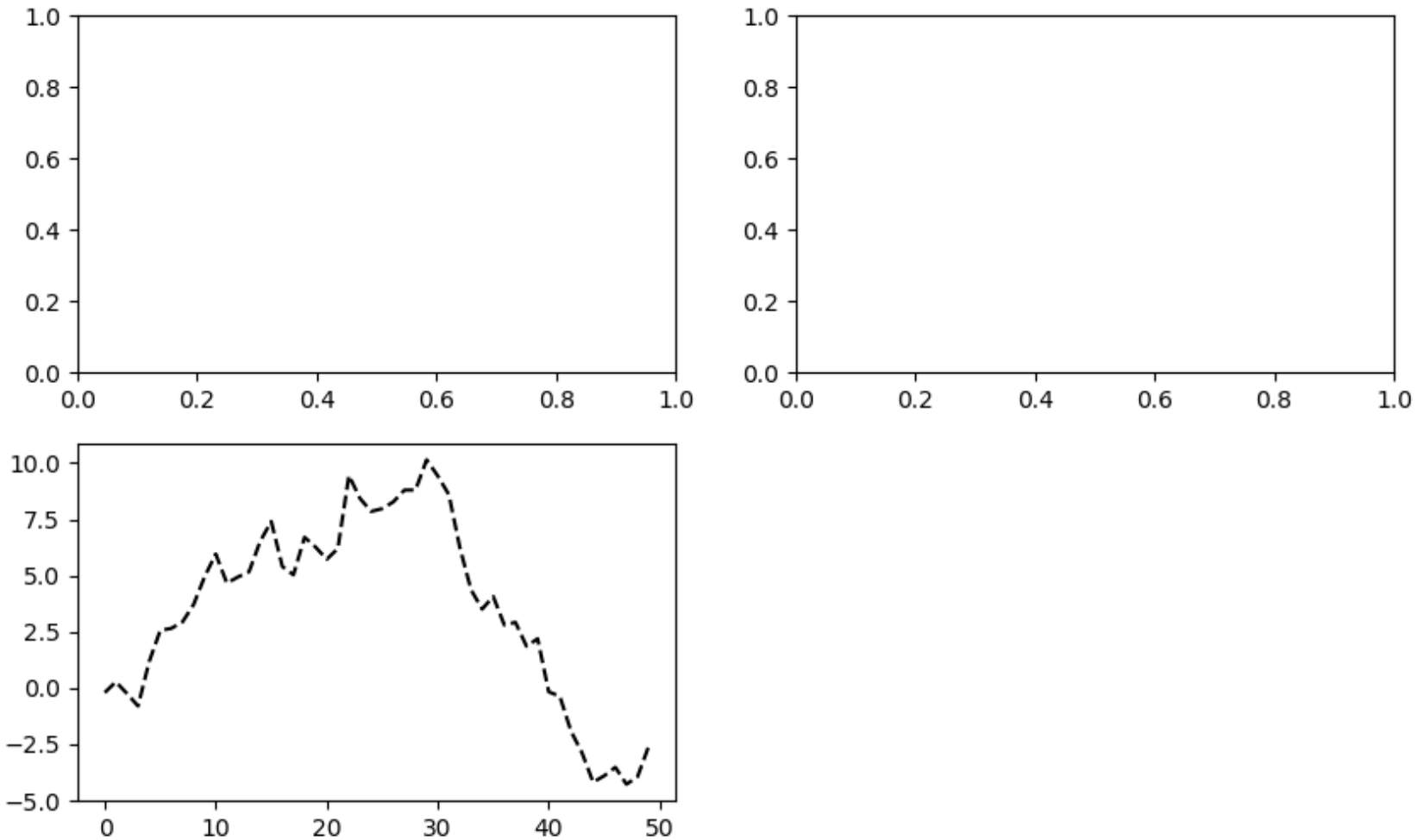


- When you issue a plotting command like `plt.plot([1.5, 3.5, -2, 1.6])`, matplotlib draws on the last figure and subplot used (creating one if necessary), thus hiding the figure and subplot creation.

```
In [12]: plt.plot(np.random.randn(50).cumsum(), 'k--')
Out[12]: [<matplotlib.lines.Line2D at 0x7f6b23d21550>]
```

In [9]:
```
fig = plt.figure()
```

Figure 2



x=29.513 y=-0.482

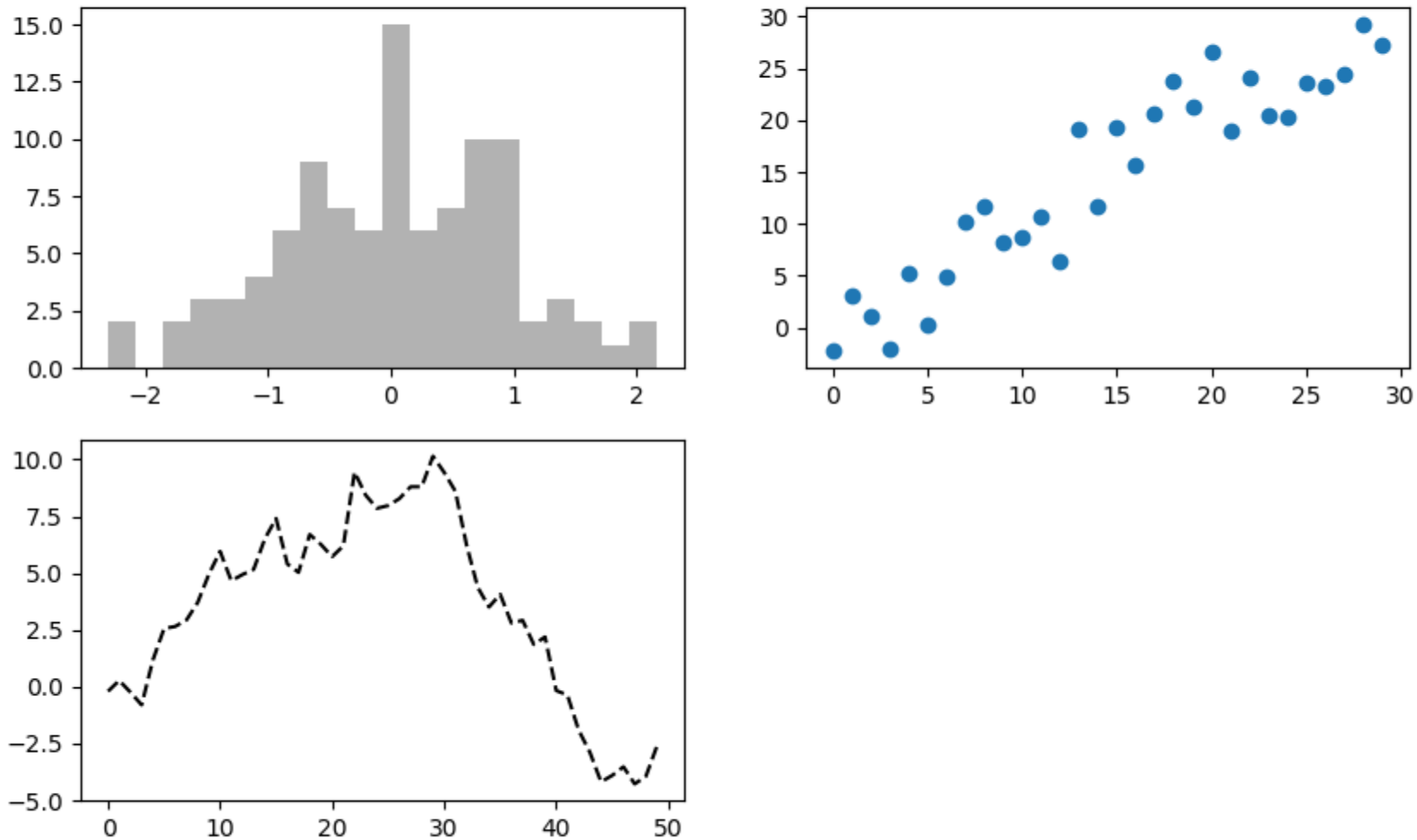
- The objects returned by `fig.add_subplot` here are `AxesSubplot` objects, on which you can directly plot on the other empty subplots by calling each one's instance method:

```
In [13]: _ = ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
         ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))

Out[13]: <matplotlib.collections.PathCollection at 0x7f6b23ca78d0>
```

In [9]:
```
fig = plt.figure()
```

Figure 2

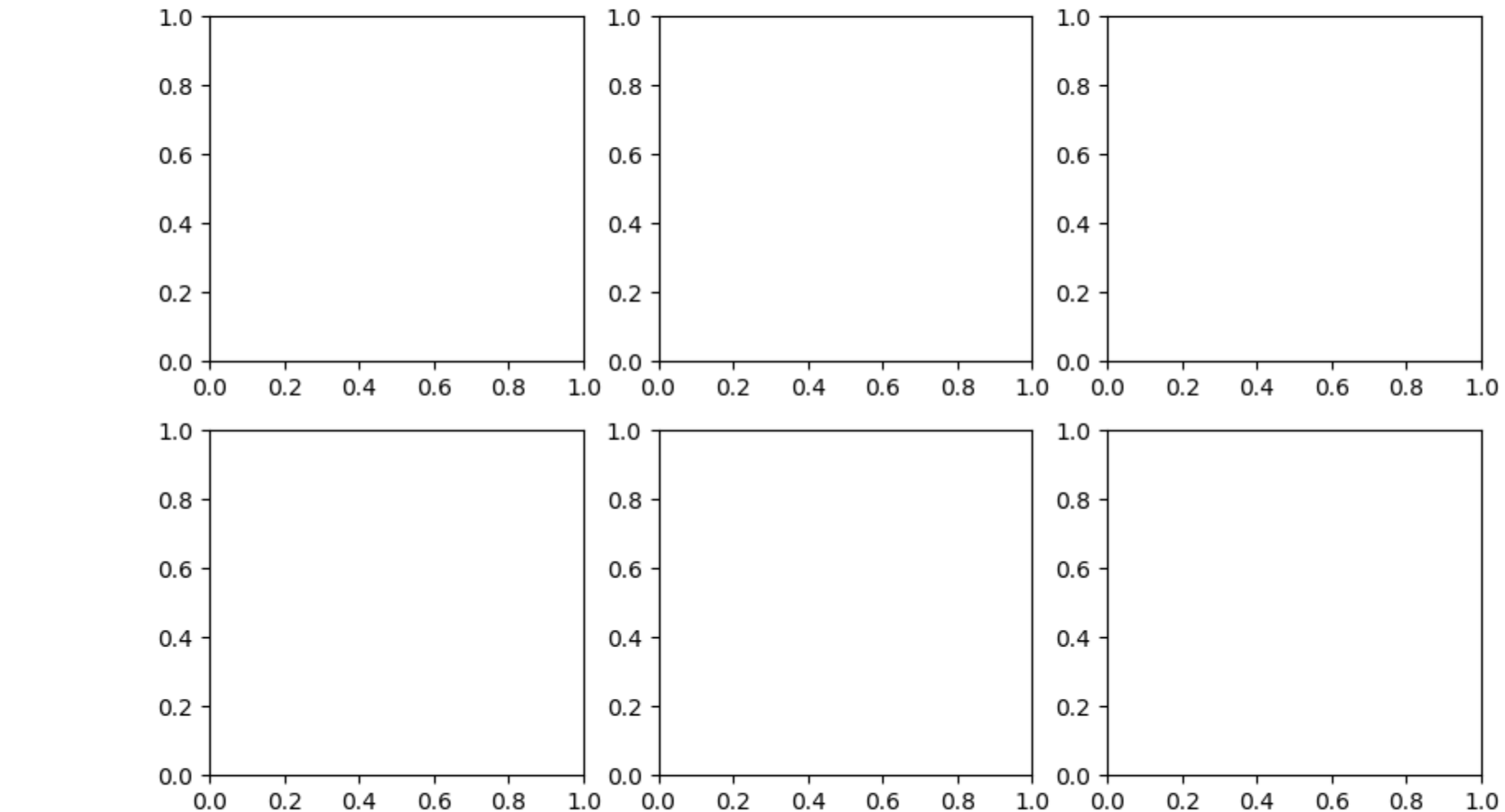


x=43.5896 y=-4.1(

- Creating a figure with a grid of subplots is a very common task, so matplotlib includes a convenience method, `plt.subplots`, that creates a new figure and returns a NumPy array containing the created subplot objects.

In [14]:
```
fig, axes = plt.subplots(2, 3)
```

Figure 3



In [15]:
```
axes
```

Out[15]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23d051d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23b6d390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23b955c0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23b3c7f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23d53470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6b23af89e8>]],
      dtype=object)
```