More Visualizations

Previously, we saw an overview of how pandas `plot` method worked and how to use the basic API of matplotlib. We'll provide more details in this lesson.

Hands on!

In [1]:

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

%matplotlib inline
```

Global API

Matplotlib's default pyplot API has a global, MATLAB-style interface, as we've already seen:

In [2]:

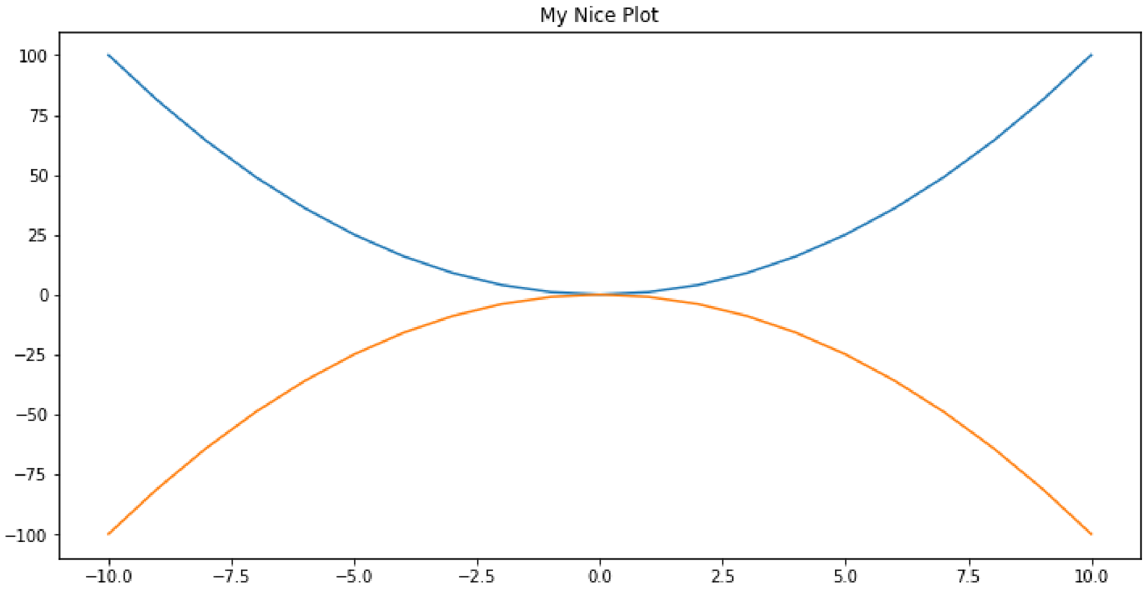
```
x = np.arange(-10, 11)
```

In [3]:

```
plt.figure(figsize=(12, 6))

plt.title('My Nice Plot')

plt.plot(x, x ** 2)
plt.plot(x, -1 * (x ** 2))
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x7fa54bea0ee0>]
```

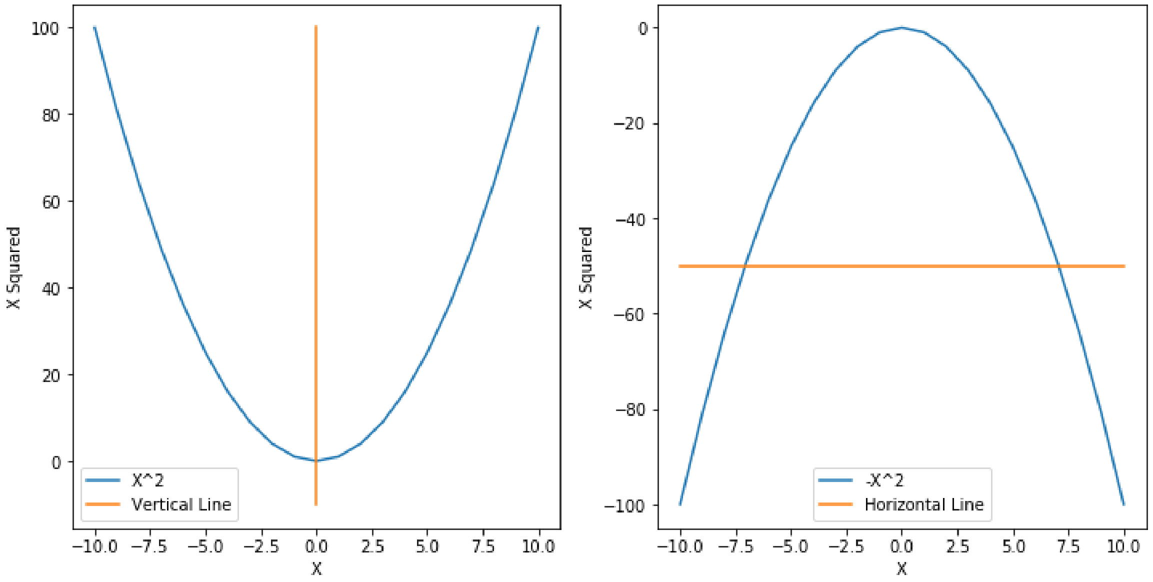


In [4]:

```
plt.figure(figsize=(12, 6))
plt.title('My Nice Plot')

plt.subplot(1, 2, 1)  # rows, columns, panel selected
plt.plot(x, x ** 2)
plt.plot([0, 0, 0], [-10, 0, 100])
plt.legend(['X^2', 'Vertical Line'])
plt.xlabel('X')
plt.ylabel('X Squared')

plt.subplot(1, 2, 2)
plt.plot(x, -1 * (x ** 2))
plt.plot([-10, 0, 10], [-50, -50, -50])
plt.legend(['-X^2', 'Horizontal Line'])

plt.xlabel('X')
plt.ylabel('X Squared')
```

Out[4]:

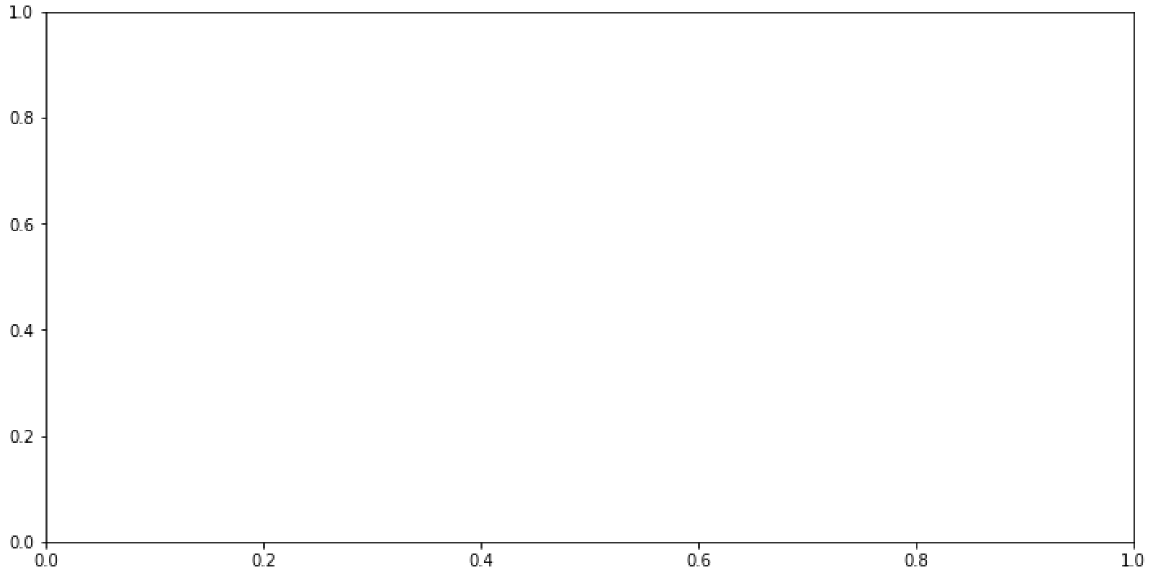
```
Text(0, 0.5, 'X Squared')
```



OOP Interface

In [5]:

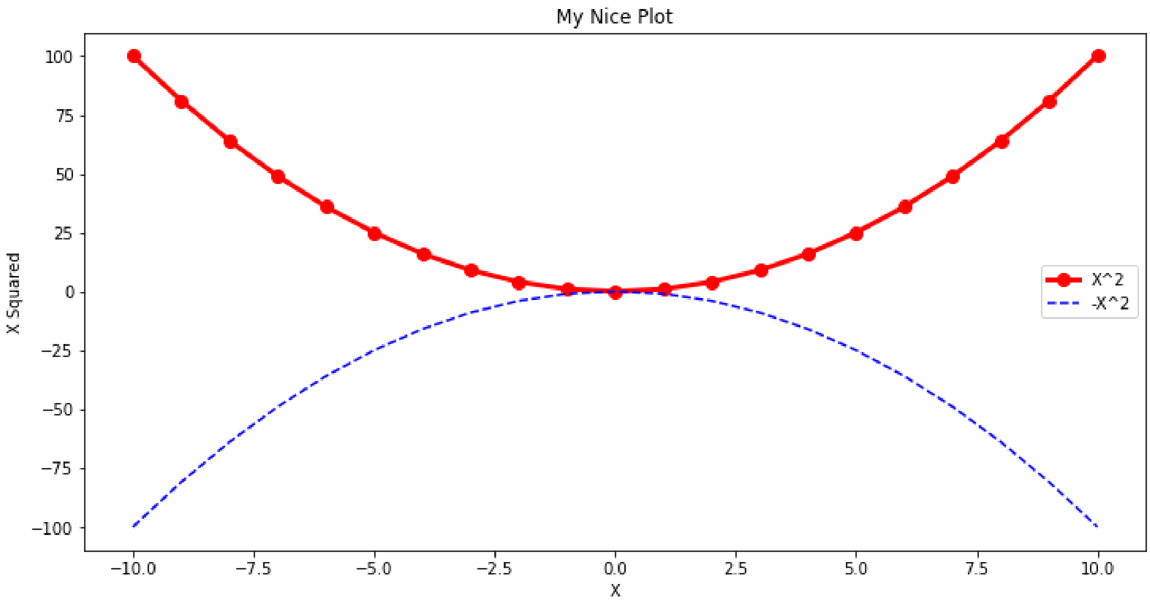
```
fig, axes = plt.subplots(figsize=(12, 6))
```



In [6]:

```
axes.plot(
    x, (x ** 2), color='red', linewidth=3,
    marker='o', markersize=8, label='X^2')

axes.plot(x, -1 * (x ** 2), 'b--', label='-X^2')

axes.set_xlabel('X')
axes.set_ylabel('X Squared')

axes.set_title("My Nice Plot")

axes.legend()

fig
```

Out[6]:

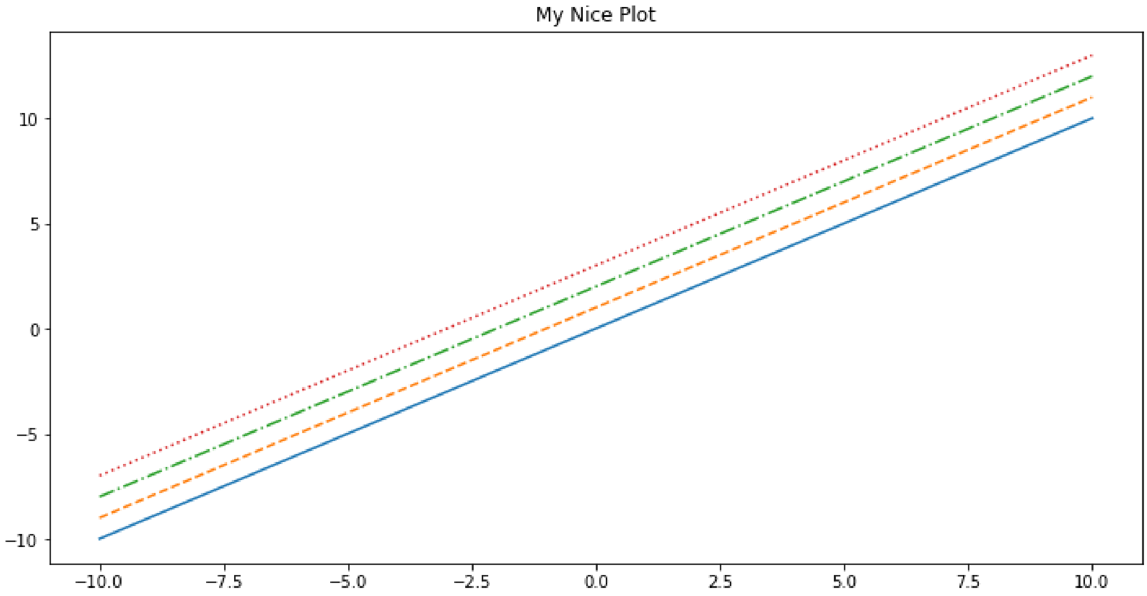


In [7]:

```
fig, axes = plt.subplots(figsize=(12, 6))

axes.plot(x, x + 0, linestyle='solid')
axes.plot(x, x + 1, linestyle='dashed')
axes.plot(x, x + 2, linestyle='dashdot')
axes.plot(x, x + 3, linestyle='dotted');

axes.set_title("My Nice Plot")
```

Out[7]:

```
Text(0.5, 1.0, 'My Nice Plot')
```

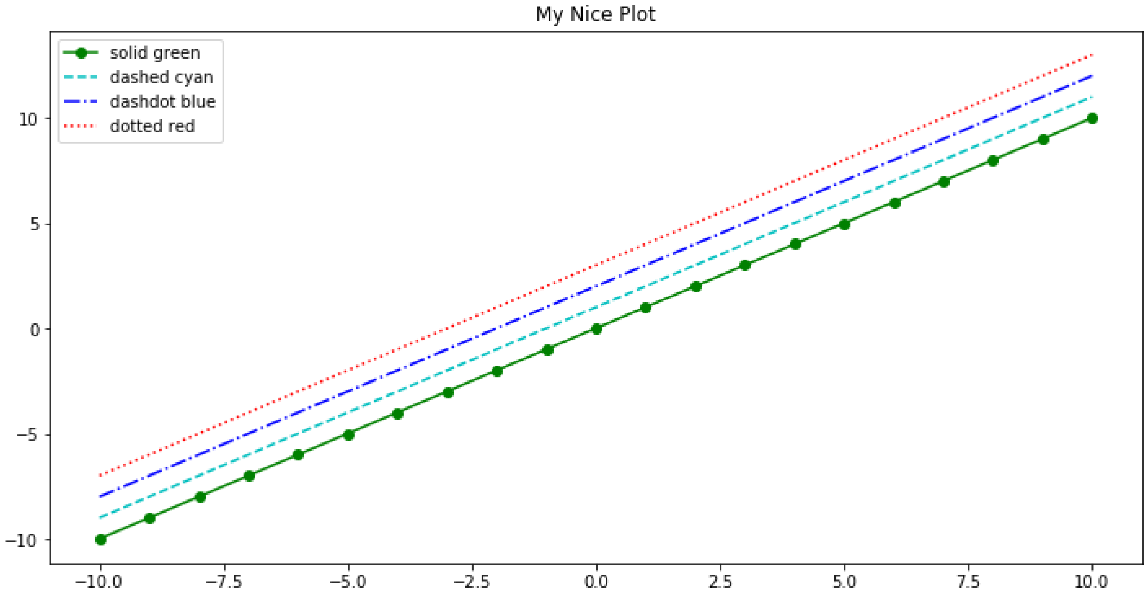


In [8]:

```
fig, axes = plt.subplots(figsize=(12, 6))

axes.plot(x, x + 0, '-og', label="solid green")
axes.plot(x, x + 1, '--c', label="dashed cyan")
axes.plot(x, x + 2, '-.b', label="dashdot blue")
axes.plot(x, x + 3, ':r', label="dotted red")

axes.set_title("My Nice Plot")

axes.legend()
```

Out[8]:

```
<matplotlib.legend.Legend at 0x7fa54bc5f130>
```



There are a lot of line and marker types.

In [ ]:

```
print('Markers: {}'.format([m for m in plt.Line2D.markers]))
```

In [ ]:

```
linestyles = ['_', '-', '--', ':']

print('Line styles: {}'.format(linestyles))
```

Figures and subfigures

When we call the `subplots()` function we get a tuple containing a `Figure` and a `axes` element.

In [ ]:

```
plot_objects = plt.subplots()

fig, ax = plot_objects

ax.plot([1,2,3], [1,2,3])

plot_objects
```

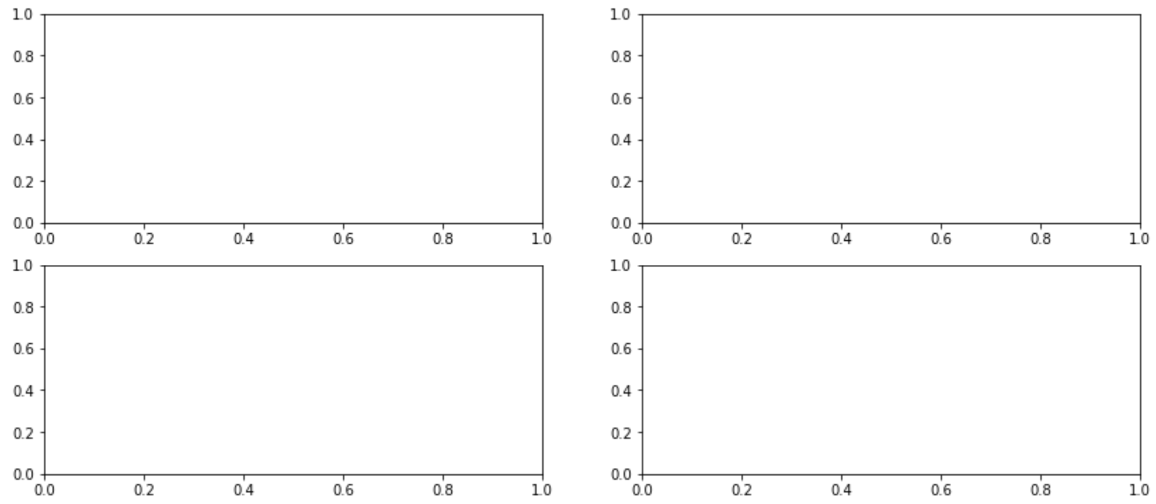
We can also define how many elements we want inside our figure. To do that we can set the `nrows` and `ncols` params.

In [11]:

```
plot_objects = plt.subplots(nrows=2, ncols=2, figsize=(14, 6))

fig, ((ax1, ax2), (ax3, ax4)) = plot_objects

plot_objects
```

Out[11]:

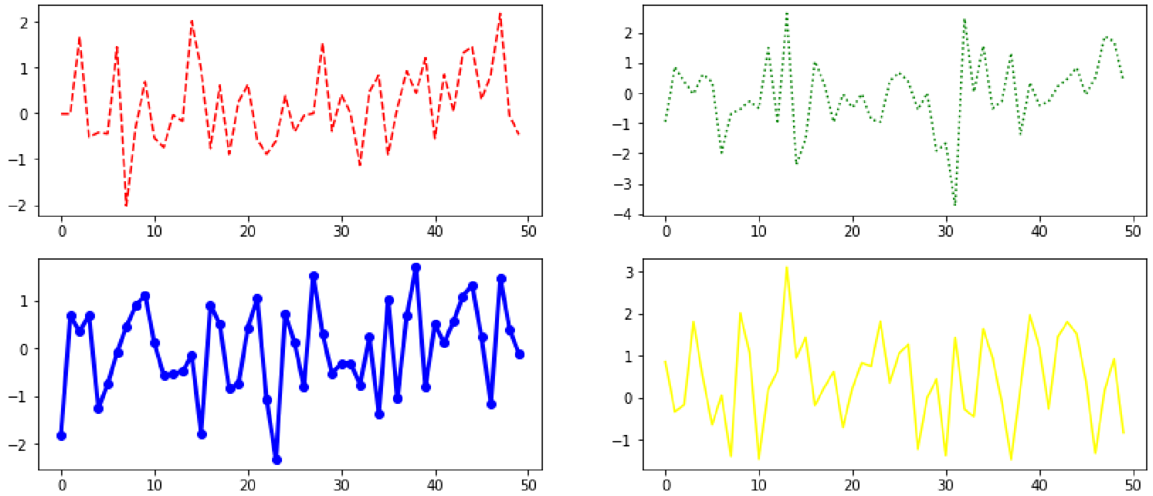
```
(<Figure size 1008x432 with 4 Axes>,
 array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fa549f8
91c0>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x7fa549fa
3760>],
        [<matplotlib.axes._subplots.AxesSubplot object at 0x7fa54a14
cdc0>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x7fa54a10
1640>]],
       dtype=object))
```



In [12]:

```
ax4.plot(np.random.randn(50), c='yellow')
ax1.plot(np.random.randn(50), c='red', linestyle='--')
ax2.plot(np.random.randn(50), c='green', linestyle=':')
ax3.plot(np.random.randn(50), c='blue', marker='o', linewidth=3.0)


fig
```

Out[12]:

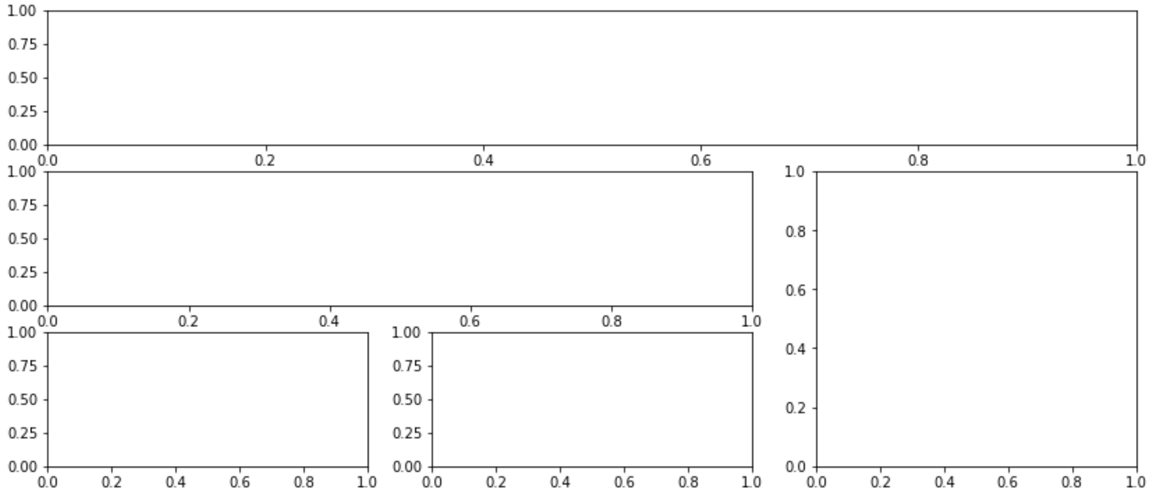


The `subplot2grid` command

There is another way to make subplots using a grid-like format:

In [13]:

```
plt.figure(figsize=(14, 6))

ax1 = plt.subplot2grid((3,3), (0,0), colspan=3)
ax2 = plt.subplot2grid((3,3), (1,0), colspan=2)
ax3 = plt.subplot2grid((3,3), (1,2), rowspan=2)
ax4 = plt.subplot2grid((3,3), (2,0))
ax5 = plt.subplot2grid((3,3), (2,1))
```



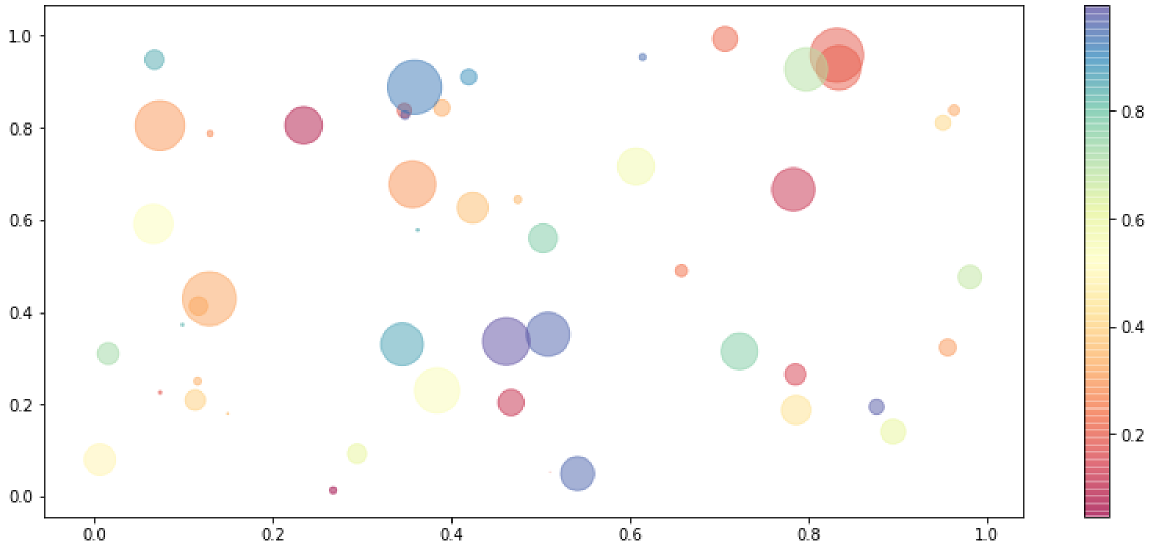
Scatter Plot

In [14]:

```
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (20 * np.random.rand(N))**2  # 0 to 15 point radii
```

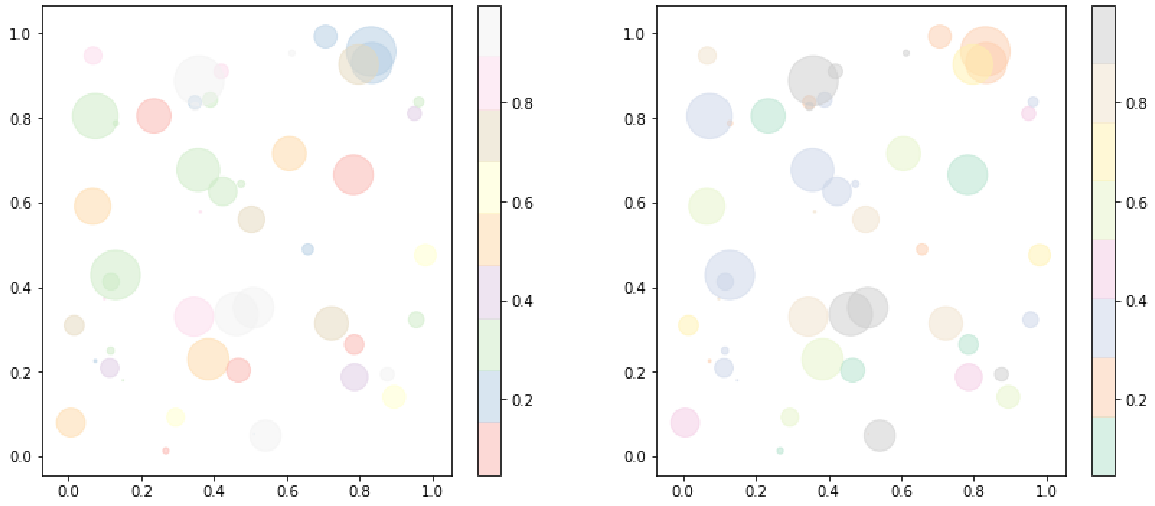
In [15]:

```
plt.figure(figsize=(14, 6))

plt.scatter(x, y, s=area, c=colors, alpha=0.5, cmap='Spectral')
plt.colorbar()

plt.show()
```



In [16]:

```
fig = plt.figure(figsize=(14, 6))

ax1 = fig.add_subplot(1,2,1)
plt.scatter(x, y, s=area, c=colors, alpha=0.5, cmap='Pastel1')
plt.colorbar()

ax2 = fig.add_subplot(1,2,2)
plt.scatter(x, y, s=area, c=colors, alpha=0.5, cmap='Pastel2')
plt.colorbar()

plt.show()
```



Here is the full `cmap` options available: [https://matplotlib.org/users/colormaps.html (https://matplotlib.org/users/colormaps.html)](https://matplotlib.org/users/colormaps.html)

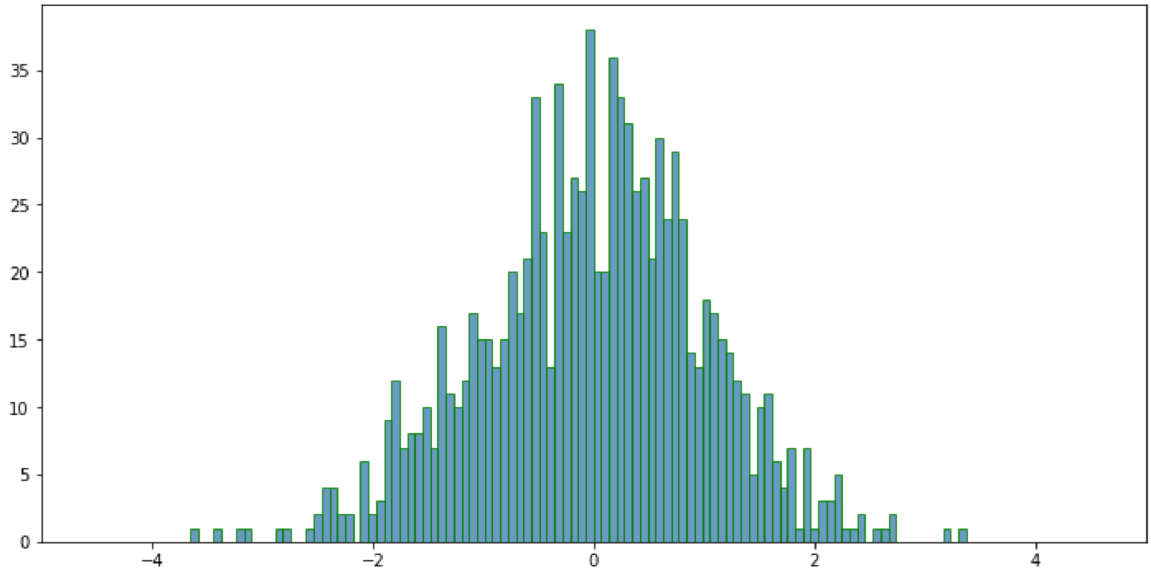
Histograms

In [17]:

```
values = np.random.randn(1000)
```

In [18]:

```
plt.subplots(figsize=(12, 6))

plt.hist(values, bins=100, alpha=0.8,
         histtype='bar', color='steelblue',
         edgecolor='green')
plt.xlim(xmin=-5, xmax=5)

plt.show()
```



In [19]:

```
fig.savefig('hist.png')
```

KDE (kernel density estimation)

In [20]:

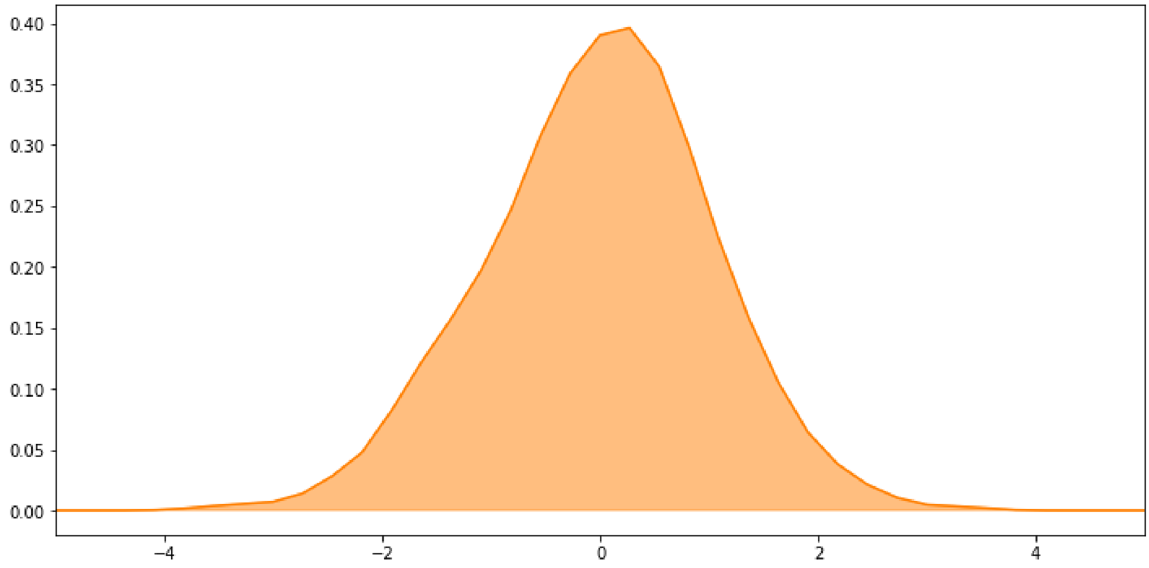
```
from scipy import stats

density = stats.kde.gaussian_kde(values)
density
```

Out[20]:

```
<scipy.stats.kde.gaussian_kde at 0x7fa546cec790>
```

In [21]:

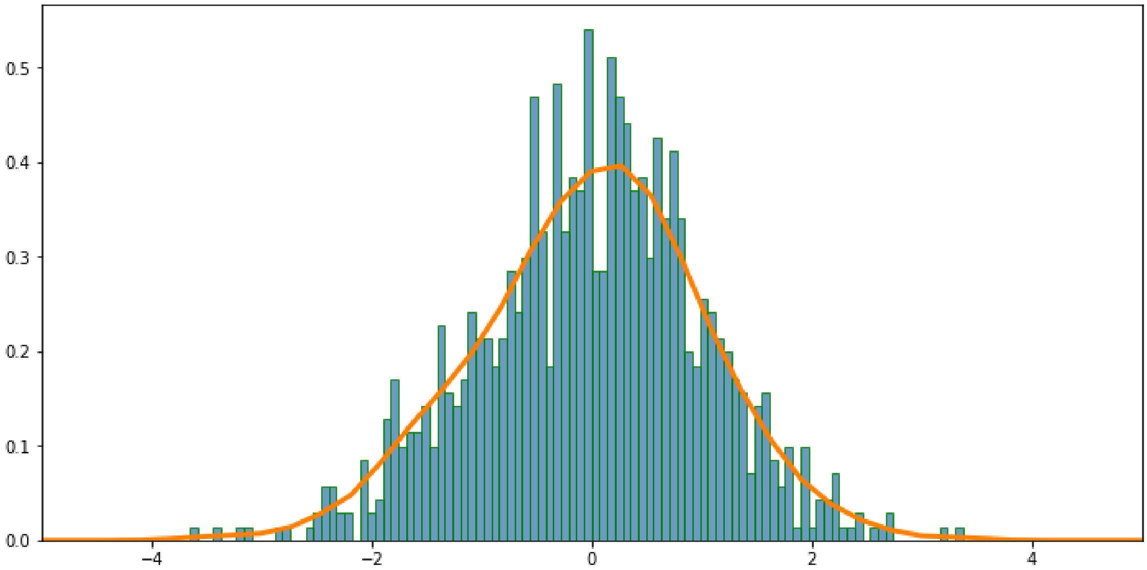
```
plt.subplots(figsize=(12, 6))

values2 = np.linspace(min(values)-10, max(values)+10, 100)

plt.plot(values2, density(values2), color='#FF7F00')
plt.fill_between(values2, 0, density(values2), alpha=0.5, color='#FF7F00')
plt.xlim(xmin=-5, xmax=5)

plt.show()
```



Combine plots

In [22]:

```
plt.subplots(figsize=(12, 6))

plt.hist(values, bins=100, alpha=0.8, density=1,
          histtype='bar', color='steelblue',
          edgecolor='green')

plt.plot(values2, density(values2), color='#FF7F00', linewidth=3.0)
plt.xlim(xmin=-5, xmax=5)

plt.show()
```



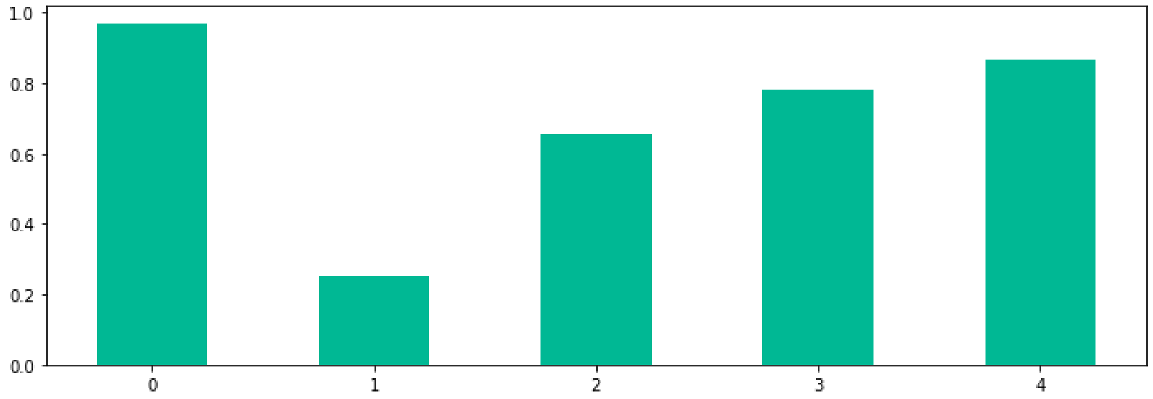
Bar plots

In [23]:

```
Y = np.random.rand(1, 5)[0]
Y2 = np.random.rand(1, 5)[0]
```

In [24]:

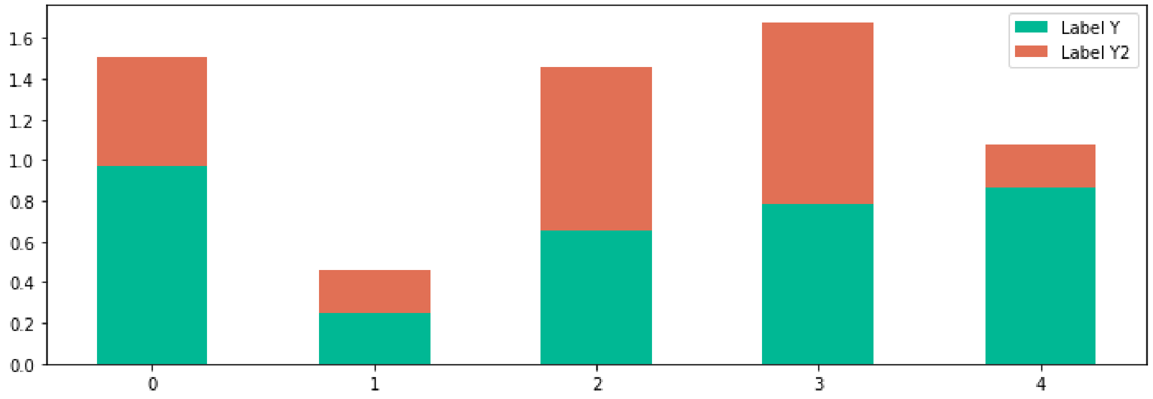
```
plt.figure(figsize=(12, 4))

barWidth = 0.5
plt.bar(np.arange(len(Y)), Y, width=barWidth, color='#00b894')

plt.show()
```



Also can be stacked bars, and add a legend to the plot:

In [25]:

```
plt.figure(figsize=(12, 4))

barWidth = 0.5
plt.bar(np.arange(len(Y)), Y, width=barWidth, color='#00b894', label='Label Y')
plt.bar(np.arange(len(Y2)), Y2, width=barWidth, color='#e17055', bottom=Y, label
='Label Y2')

plt.legend()
plt.show()
```



Boxplots and outlier detection

In [26]:

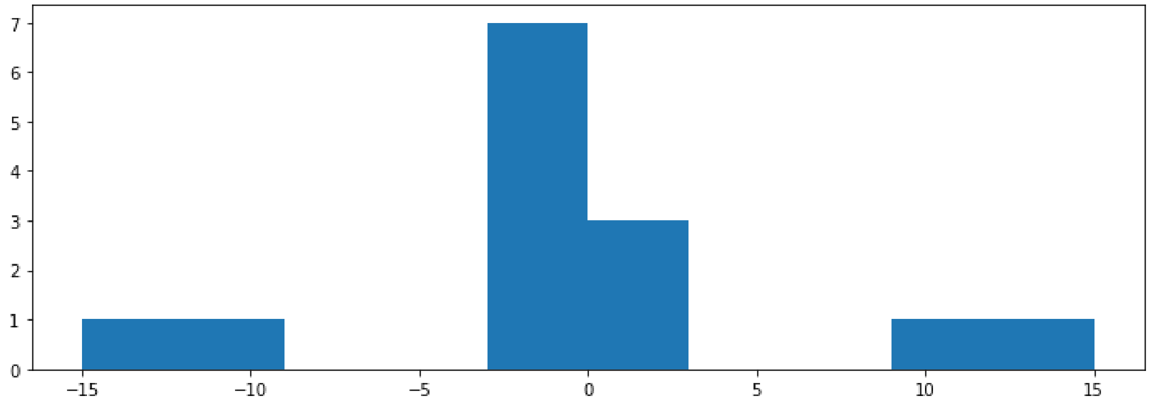
```
values = np.concatenate([np.random.randn(10), np.array([10, 15, -10, -15])])
```

In [27]:

```
plt.figure(figsize=(12, 4))

plt.hist(values)
```

Out[27]:

```
(array([1., 1., 0., 0., 7., 3., 0., 0., 1., 1.]),
 array([-15., -12.,  -9.,  -6.,  -3.,   0.,   3.,   6.,   9.,  12.,
15.]),
 <a list of 10 Patch objects>)
```

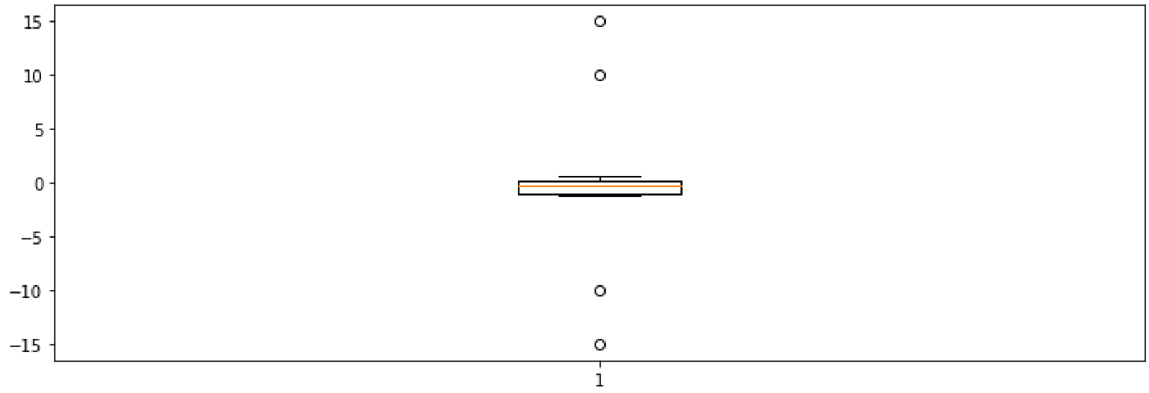


In [28]:

```
plt.figure(figsize=(12, 4))

plt.boxplot(values)
```

Out[28]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fa53d27ba90>,
  <matplotlib.lines.Line2D at 0x7fa53d289160>],
 'caps': [<matplotlib.lines.Line2D at 0x7fa53d289490>,
  <matplotlib.lines.Line2D at 0x7fa53d2897c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fa53d27ba60>],
 'medians': [<matplotlib.lines.Line2D at 0x7fa53d289af0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fa53d27ba00>],
 'means': []}
```


15
10
5
0
−5
−10
−15
1