A Brief Introduction to matplotlib for Data Visualization

In [1]:
```
#Install matplotlib in python
# python3 -m pip install matplotlib
```

Data import and modules import

In [6]:
```
#We have to import pyplot to have an matlab like graphical environment and mli
nes to draw lines on a plot
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
from pandas.plotting import register_matplotlib_converters
```

Plotting Histograms

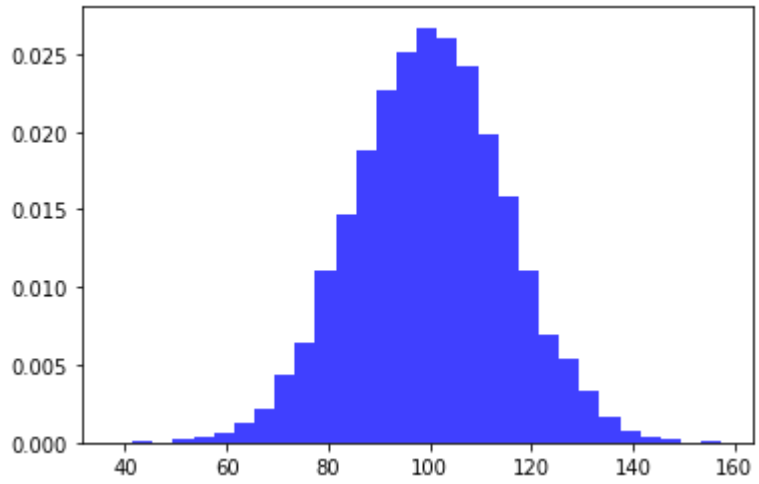
To plot a histogram, we follow a similar process and use the hist() function from pyplot. We will generate 10000 random data points, x, with a mean of 100 and standard deviation of 15.

The hist function takes the data, x, number of bins, and other arguments such as density, which normalizes the data to a probability density, or alpha, which sets the transparency of the histogram.

We will also use the library mlab to add a line representing a normal density function with the same mean and standard deviation:

In [7]:
```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
%matplotlib inline

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 30, density=1, facecolor='blue', alpha=0.75)

# add a 'best fit' line
y = mlab.normpdf( bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=4)

plt.xlabel('IQ')
plt.ylabel('Probability')
plt.title(r'$\mathrm{Histogram\ of\ IQ:}\ \mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)

plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-a11c21f4c42d> in <module>
     11
     12 # add a 'best fit' line
---> 13 y = mlab.normpdf( bins, mu, sigma)
     14 l = plt.plot(bins, y, 'r--', linewidth=4)
     15

AttributeError: module 'matplotlib.mlab' has no attribute 'normpdf'
```

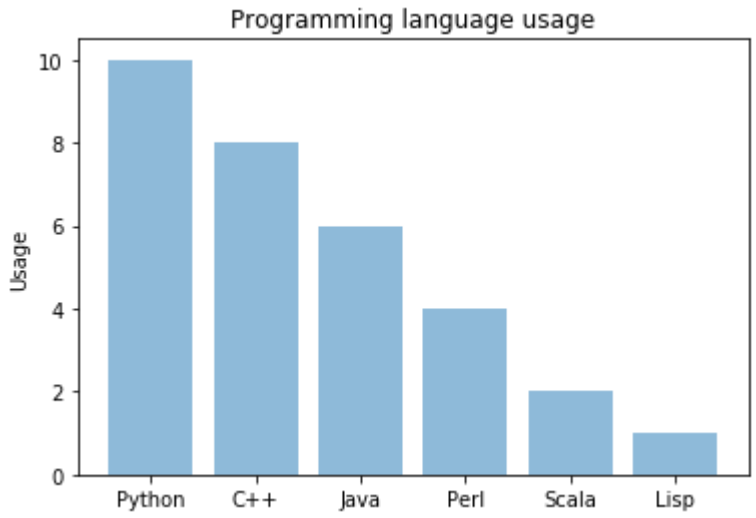


Bar Charts

While histograms helped us with visual densities, bar charts help us view counts of data. To plot a bar chart with matplotlib, we use the bar() function. This takes the counts and data labels as x and y, along with other arguments.

As an example, we could look at a sample of the number of programmers that use different languages:

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.title('Programming language usage')

plt.show()
```



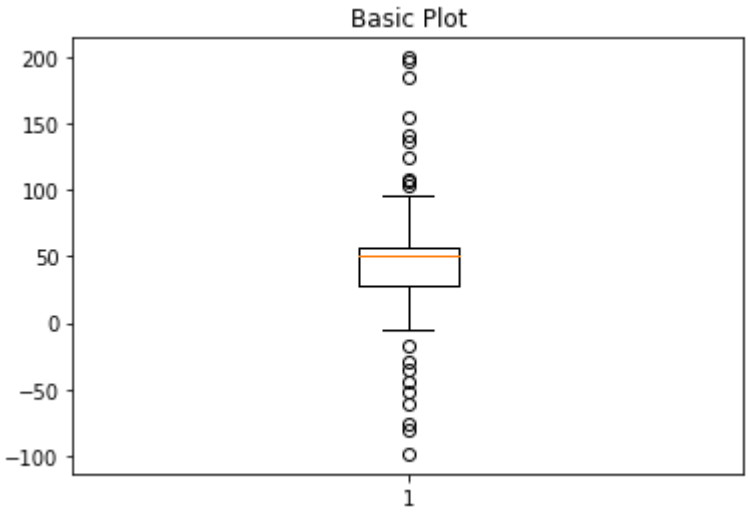
Boxplot

In [9]:

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

# Fixing random state for reproducibility
np.random.seed(2)

# fake up some data
spread = np.random.rand(50) * 100
print(spread)
center = np.ones(25) * 50
print(center)
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))

fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)


#Q1 = 25% PERCENTILE
#Q2 = MEDIAn
#Q3 = 75% PERCENTILE

# #IQR  - (Q3-Q1)
# LOWER WHISKER LIMIT : Q1 - [(Q3-Q1) * 1.5]
# HIGHER WHISKER LIMIT : Q3 + [(Q3-Q1) *1.5]
```

```
[43.59949021  2.59262318 54.96624779 43.53223926 42.03678021 33.0334821
 20.4648634  61.92709664 29.96546737 26.68272751 62.11338328 52.91420943
 13.45799453 51.35781213 18.44398656 78.53351478 85.39752926 49.42368374
 84.65614854  7.9645477  50.52460901  6.52865044 42.81223276  9.65309157
 12.71599717 59.6745309  22.60120006 10.69456843 22.03062071 34.9826285
 46.77874846 20.17432263 64.04067252 48.30698356 50.523672   38.68926511
 79.36374544 58.00041789 16.22985985 70.07523466 96.45510801 50.00083612
 88.95200639 34.16136527 56.71441276 42.75459633 43.6747263  77.6559185
 53.56041735 95.37422269]
[50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50. 50.
 50. 50. 50. 50. 50. 50. 50.]
```

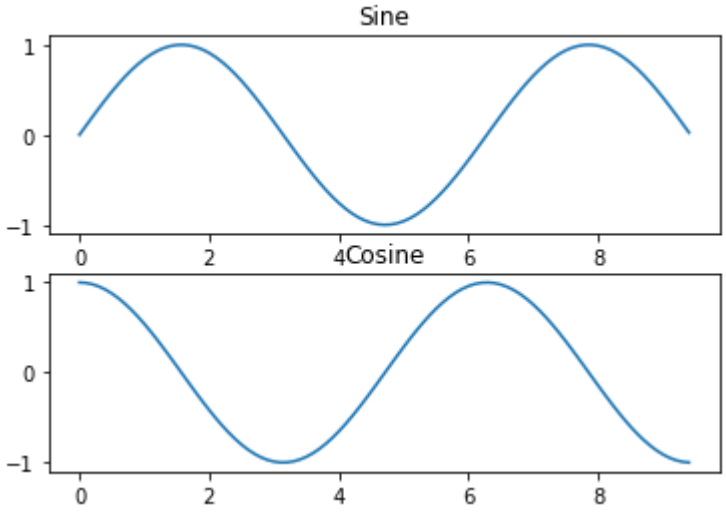
Out[9]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1be0bee7f08>,
  <matplotlib.lines.Line2D at 0x1be0bee7908>],
 'caps': [<matplotlib.lines.Line2D at 0x1be0beeba48>,
  <matplotlib.lines.Line2D at 0x1be0beebf88>],
 'boxes': [<matplotlib.lines.Line2D at 0x1be0bee7648>],
 'medians': [<matplotlib.lines.Line2D at 0x1be0beebbc8>],
 'fliers': [<matplotlib.lines.Line2D at 0x1be0bef3a88>],
 'means': []}
```



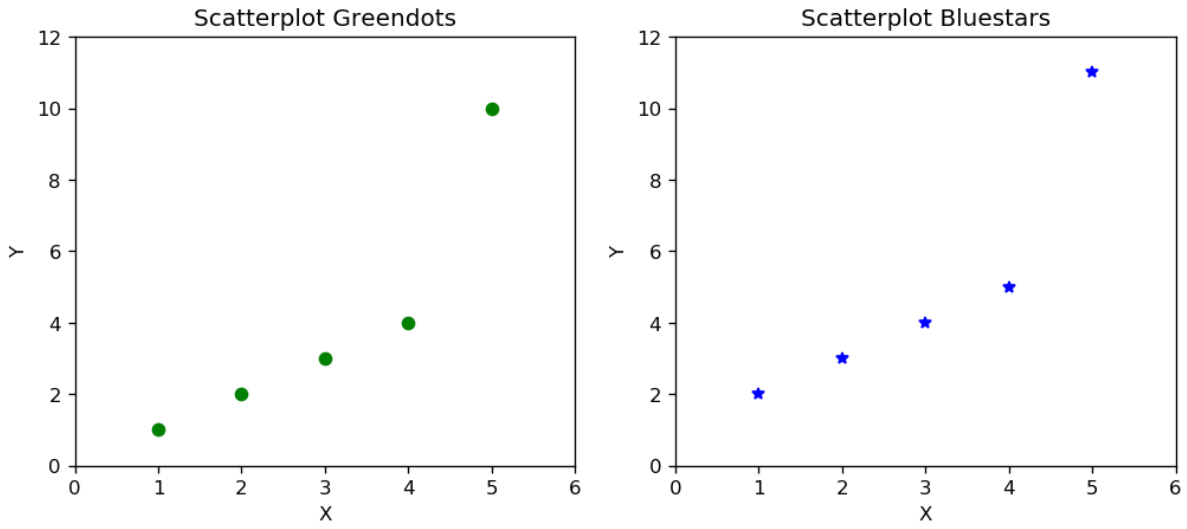
Subplots

The subplot() function allows you to plot different things in the same figure. In the following script, sine and cosine values are plotted.

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Set up a subplot grid that has height 2 and width 1,
# and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```



In [11]:
```
plt.figure(figsize=(10,4), dpi=120) # 10 is width, 4 is height

# Left hand side plot
plt.subplot(1,2,1)  # (nRows, nColumns, axes number to plot)
plt.plot([1,2,3,4,5], [1,2,3,4,10], 'go')  # green dots
plt.title('Scatterplot Greendots')
plt.xlabel('X'); plt.ylabel('Y')
plt.xlim(0, 6); plt.ylim(0, 12)

# Right hand side plot
plt.subplot(1,2,2)
plt.plot([1,2,3,4,5], [2,3,4,5,11], 'b*')  # blue stars
plt.title('Scatterplot Bluestars')
plt.xlabel('X'); plt.ylabel('Y')
plt.xlim(0, 6); plt.ylim(0, 12)
plt.show()
```



matplotlib.pyplot.subplot2grid(shape, loc, rowspan=1, colspan=1, fig=None, **kwargs)[source]

shape : sequence of 2 ints Shape of grid in which to place axis. First entry is number of rows, second entry is number of columns.

loc : sequence of 2 ints Location to place axis within grid. First entry is row number, second entry is column number.

rowspan : int Number of rows for the axis to span to the right.

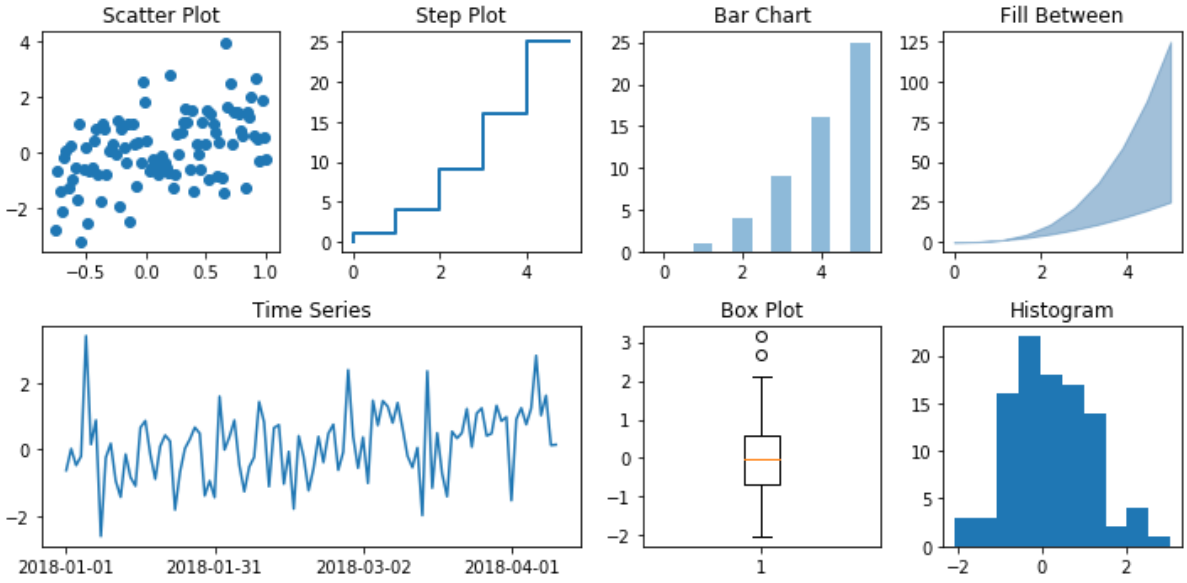
colspan : int Number of columns for the axis to span downwards.

fig : Figure, optional Figure to place axis in. Defaults to current figure.

**kwargs Additional keyword arguments are handed to add_subplot.

In [12]:

```
import pandas as pd

# Setup the subplot2grid Layout
fig = plt.figure(figsize=(10, 5))
ax1 = plt.subplot2grid((2,4), (0,0))
ax2 = plt.subplot2grid((2,4), (0,1))
ax3 = plt.subplot2grid((2,4), (0,2))
ax4 = plt.subplot2grid((2,4), (0,3))
ax5 = plt.subplot2grid((2,4), (1,0), colspan=2)
ax6 = plt.subplot2grid((2,4), (1,2))
ax7 = plt.subplot2grid((2,4), (1,3))

# Input Arrays
n = np.array([0,1,2,3,4,5])
x = np.linspace(0,5,10)
xx = np.linspace(-0.75, 1., 100)

# Scatterplot
ax1.scatter(xx, xx + np.random.randn(len(xx)))
ax1.set_title("Scatter Plot")

# Step Chart
ax2.step(n, n**2, lw=2)
ax2.set_title("Step Plot")

# Bar Chart
ax3.bar(n, n**2, align="center", width=0.5, alpha=0.5)
ax3.set_title("Bar Chart")

# Fill Between
ax4.fill_between(x, x**2, x**3, color="steelblue", alpha=0.5);
ax4.set_title("Fill Between");

# Time Series
dates = pd.date_range('2018-01-01', periods = len(xx))
ax5.plot(dates, xx + np.random.randn(len(xx)))
ax5.set_xticks(dates[::30])
ax5.set_xticklabels(dates.strftime('%Y-%m-%d')[::30])
ax5.set_title("Time Series")

# Box Plot
ax6.boxplot(np.random.randn(len(xx)))
ax6.set_title("Box Plot")

# Histogram
ax7.hist(xx + np.random.randn(len(xx)))
ax7.set_title("Histogram")

fig.tight_layout()
```



In [5]:
```
# Credits to
# Matplotlib tutorial
# Nicolas P. Rougier
# https://github.com/rougier/matplotlib-tutorial
```