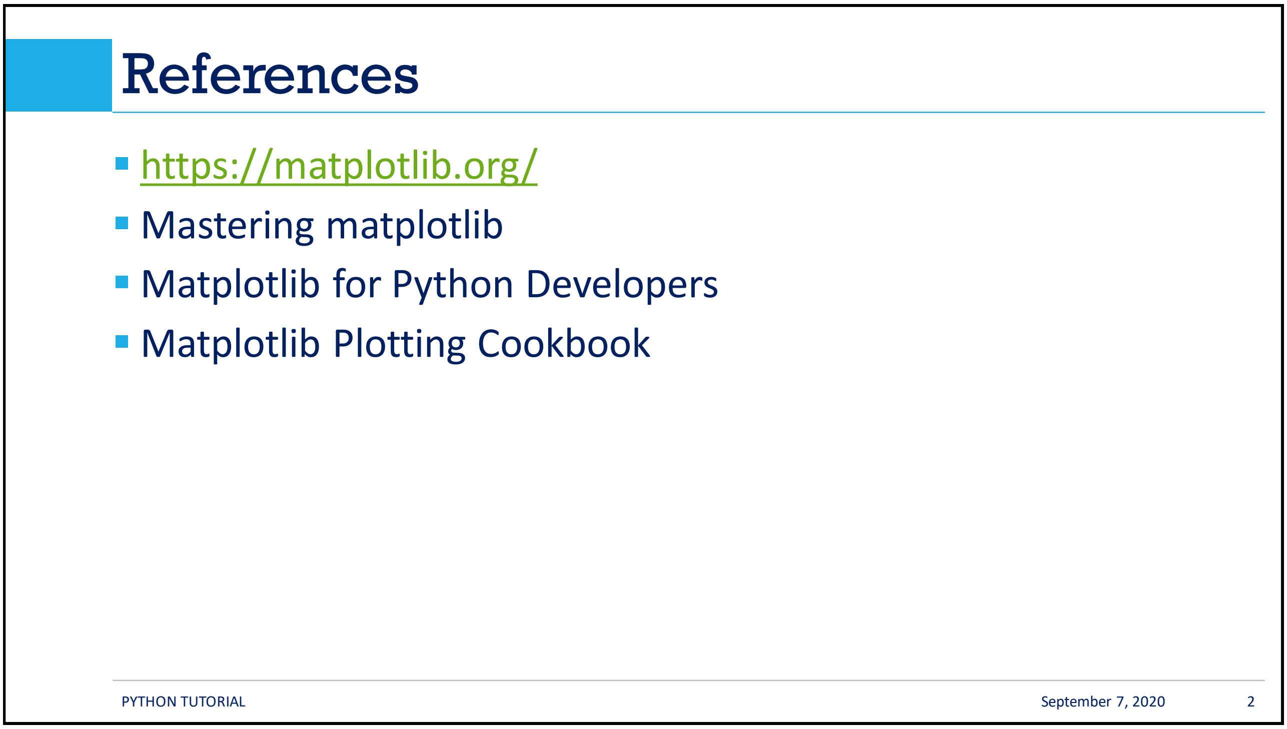
Matplotlib



References

- https://matplotlib.org/
- Mastering matplotlib
- Matplotlib for Python Developers
- Matplotlib Plotting Cookbook

Introduction

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Install packages:

```
pip install matplotlib
pip install numpy
```

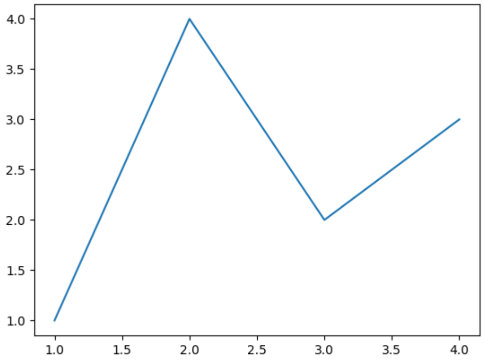
- Import packages:

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

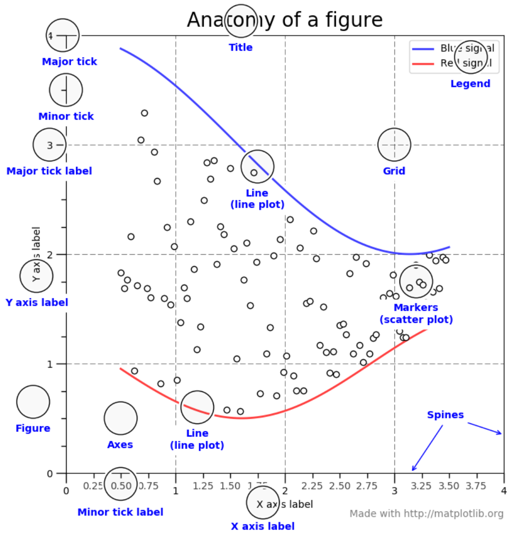
A Simple Example

- Matplotlib graphs your data on Figures, each of which can contain one or more Axes.

```
# Create a figure containing a single axes
fig, ax = plt.subplots()
# Plot some data on the axes
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
# Display the figure
plt.show()
```



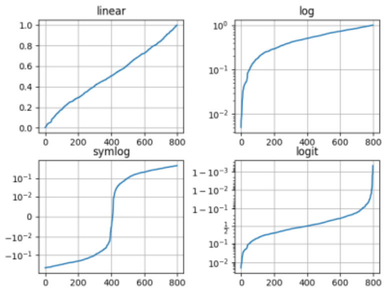
Parts of a Figure



Creating Figures and Axes

- A figure can contain any number of Axes, but will typically have at least one.

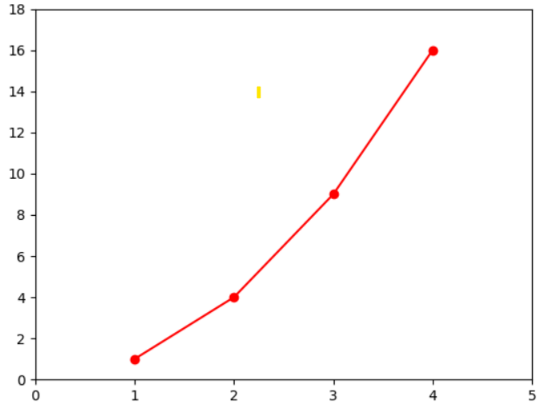
```
fig = plt.figure()  # an empty figure with no Axes
fig, ax = plt.subplots()  # a figure with a single Axes
fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
```



Line Chart

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], ‘ro-’)
plt.axis([0, 6, 0, 20])
plt.show()
```

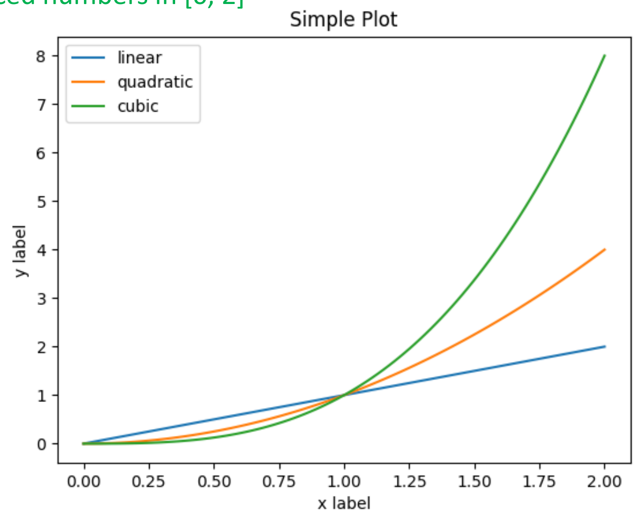
- Where:
  - ‘ro-’ = '[color][marker][line]’
  - plt.axis([xmin, xmax, ymin, ymax])



https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot

Line Chart

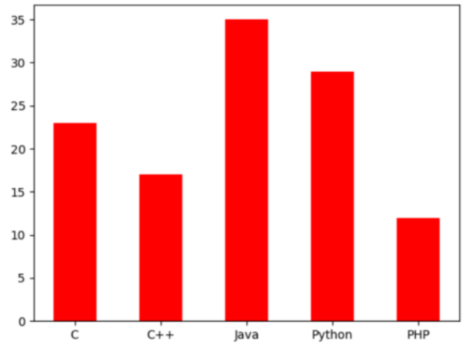
```
# Create an array including 100 equally spaced numbers in [0, 2]
x = np.linspace(0, 2, 100)
# Create a figure and an axes
fig, ax = plt.subplots()
ax.plot(x, x, label='linear')  # y = x
ax.plot(x, x**2, label='quadratic')  # y = x^2
ax.plot(x, x**3, label='cubic')  # y = x^3
# Add labels to the axes
ax.set_xlabel('x label’)
ax.set_ylabel('y label’)
# Add a title to the axes
ax.set_title("Simple Plot")
ax.legend()  # Add a legend
plt.show()
```

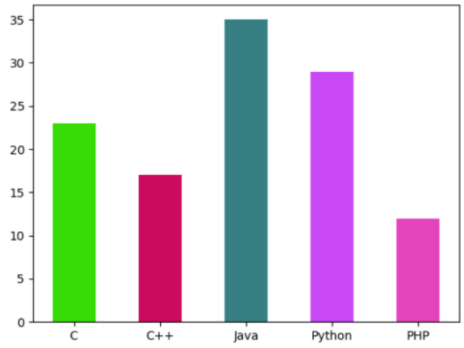


Bar Chart

```
langs = ['C', 'C++', 'Java', 'Python', 'PHP’]
students = [23,17,35,29,12]
plt.bar(langs, students, width=0.5, color = 'red’)
plt.show()
```

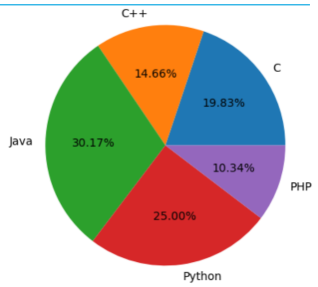
```
color=['r','g','b','k','y']
color=['red','green','blue','black','yellow']
color = np.random.rand(5,3)
```



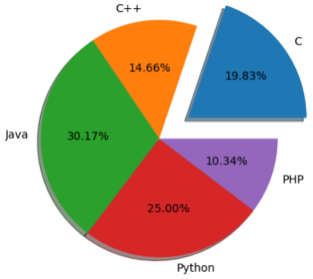


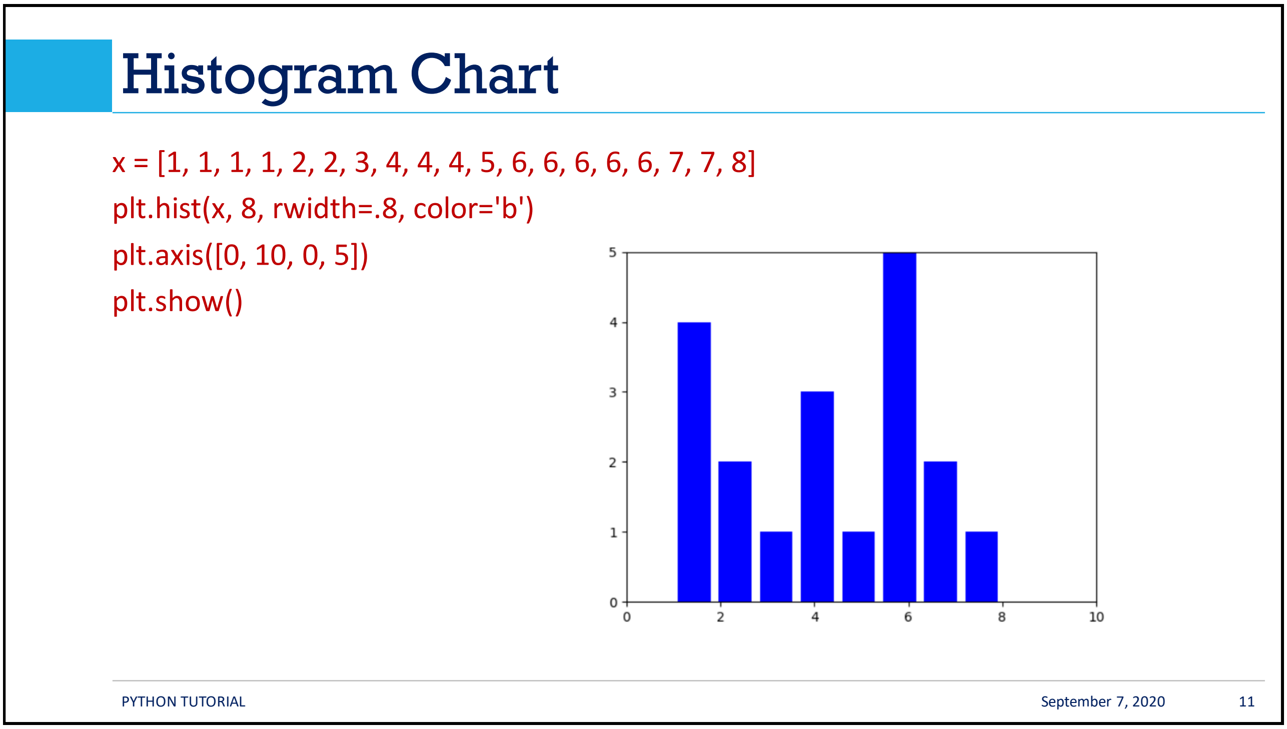
Pie Chart

```
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
plt.pie(students, labels = langs, autopct='%1.2f%%')
plt.show()
```



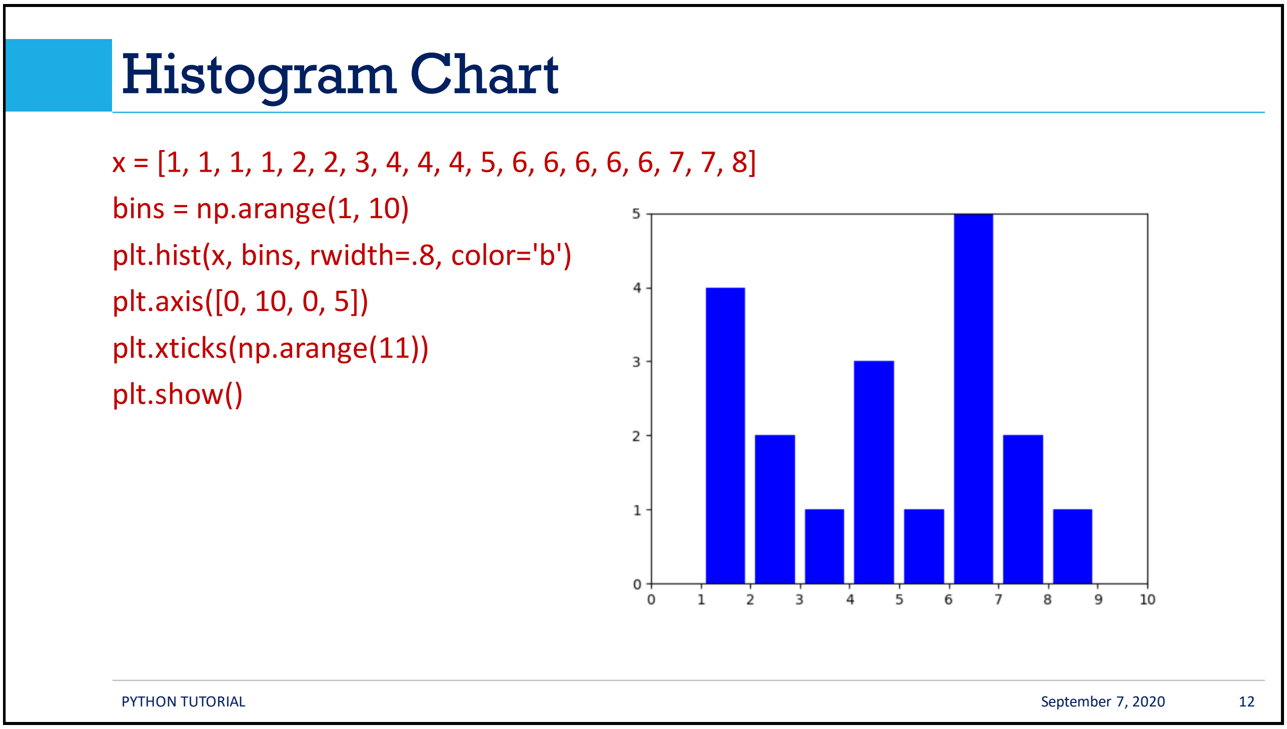
```
plt.pie(students, labels = langs, autopct='%1.2f%%’,
          shadow=True,
          explode=(0.3, 0., 0., 0., 0.))
```





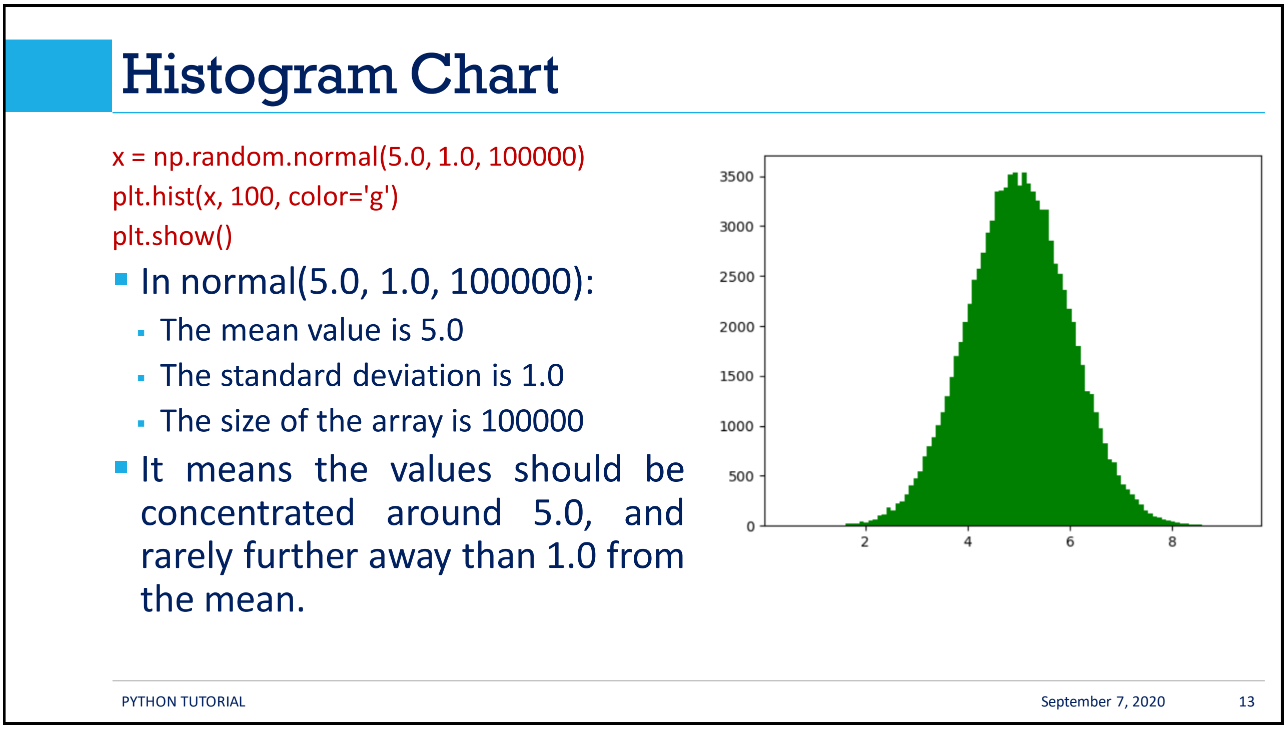
Histogram Chart

```
x = [1, 1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 6, 6, 6, 6, 6, 7, 7, 8]
plt.hist(x, 8, rwidth=.8, color='b')
plt.axis([0, 10, 0, 5])
plt.show()
```



Histogram Chart

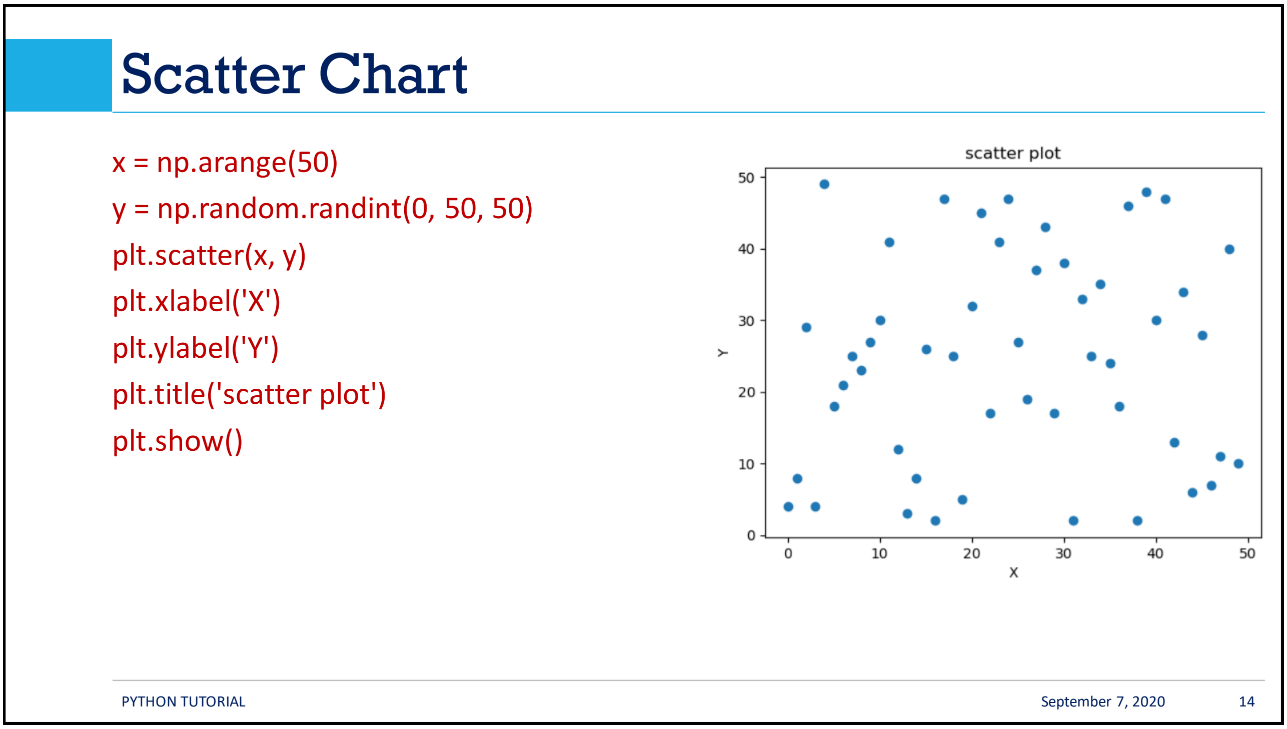
```
x = [1, 1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 6, 6, 6, 6, 6, 7, 7, 8]
bins = np.arange(1, 10)
plt.hist(x, bins, rwidth=.8, color='b')
plt.axis([0, 10, 0, 5])
plt.xticks(np.arange(11))
plt.show()
```



Histogram Chart

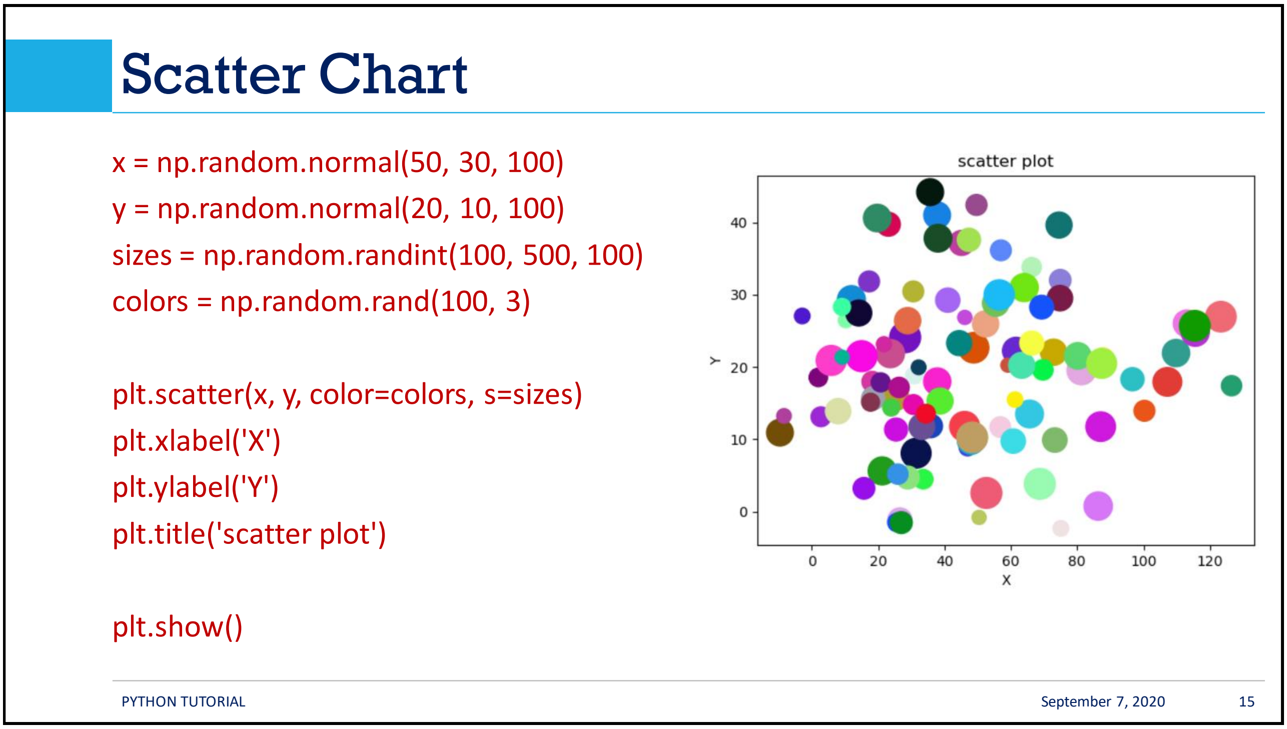
```
x = np.random.normal(5.0, 1.0, 100000)
plt.hist(x, 100, color='g')
plt.show()
```

- In normal(5.0, 1.0, 100000):
  - The mean value is 5.0
  - The standard deviation is 1.0
  - The size of the array is 100000
- It means the values should be concentrated around 5.0, and rarely further away than 1.0 from the mean.



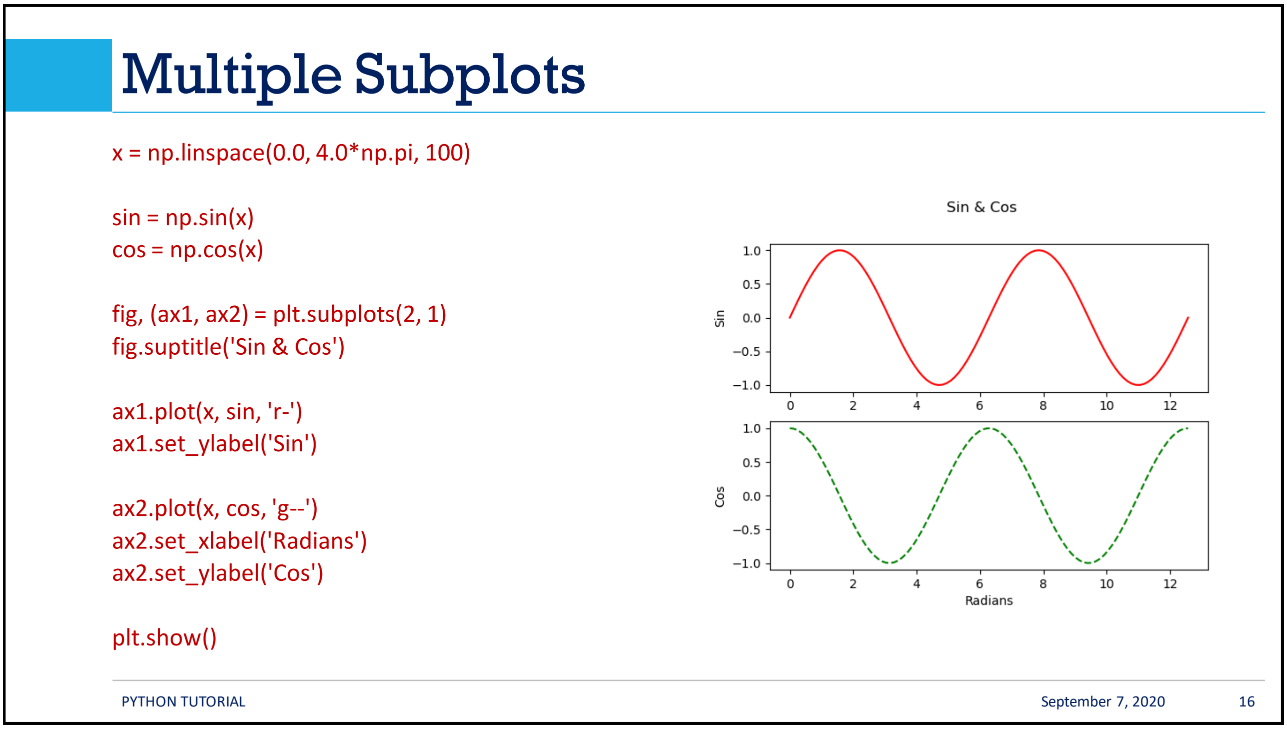
Scatter Chart

```
x = np.arange(50)
y = np.random.randint(0, 50, 50)
plt.scatter(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('scatter plot')
plt.show()
```



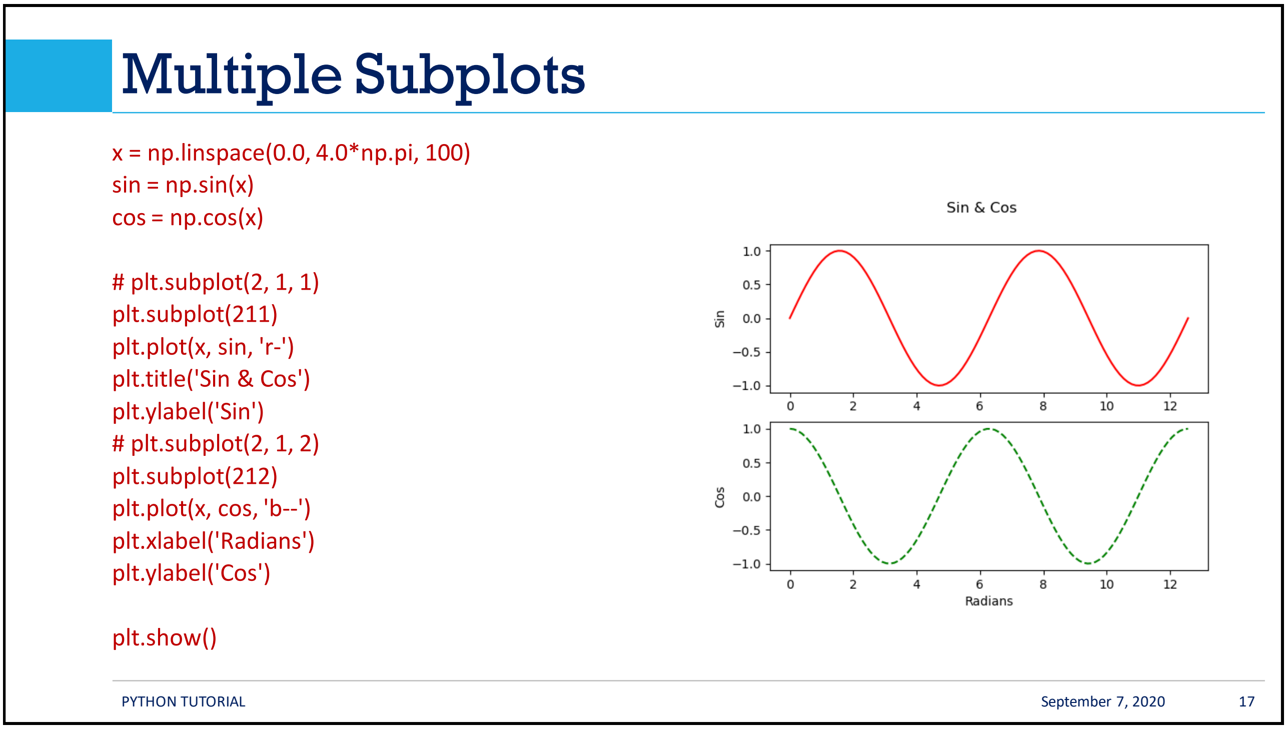
Scatter Chart

```
x = np.random.normal(50, 30, 100)
y = np.random.normal(20, 10, 100)
sizes = np.random.randint(100, 500, 100)
colors = np.random.rand(100, 3)

plt.scatter(x, y, color=colors, s=sizes)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('scatter plot')

plt.show()
```



Multiple Subplots

```
x = np.linspace(0.0, 4.0*np.pi, 100)

sin = np.sin(x)
cos = np.cos(x)

fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle('Sin & Cos')

ax1.plot(x, sin, 'r-')
ax1.set_ylabel('Sin')

ax2.plot(x, cos, 'g--')
ax2.set_xlabel('Radians')
ax2.set_ylabel('Cos')

plt.show()
```



Multiple Subplots

```
x = np.linspace(0.0, 4.0*np.pi, 100)
sin = np.sin(x)
cos = np.cos(x)

# plt.subplot(2, 1, 1)
plt.subplot(211)
plt.plot(x, sin, 'r-')
plt.title('Sin & Cos')
plt.ylabel('Sin')
# plt.subplot(2, 1, 2)
plt.subplot(212)
plt.plot(x, cos, 'b--')
plt.xlabel('Radians')
plt.ylabel('Cos')

plt.show()
```

Thank you!

QUESTIONS?