Author : Sanjoy Biswas

Topic : MatplotLib Pracetice : Python Practice

Email : sanjoy.eee32@gmail.com

What Is Python Matplotlib?

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Is Matplotlib Included in Python?

Matplotlib is not a part of the Standard Libraries which is installed by default when Python, there are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.

Basemap: It is a map plotting toolkit with various map projections, coastlines and political boundaries.

Cartopy: It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.

Excel tools: Matplotlib provides utilities for exchanging data with Microsoft Excel.

Mplot3d: It is used for 3-D plots.

Natgrid: It is an interface to the natgrid library for irregular gridding of the spaced data.

You may go through this recording of Python Matplotlib where our instructor has explained how to download Matplotlib in Python and the topics in a detailed manner with examples that will help you to understand this concept better.

Import Libraries

In [1]:

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

Functional Method

In [2]:

```
x = np.linspace(0, 5, 11)
y = x**2
```

In [3]:

```
x
```

Out[3]:

```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

In [4]:

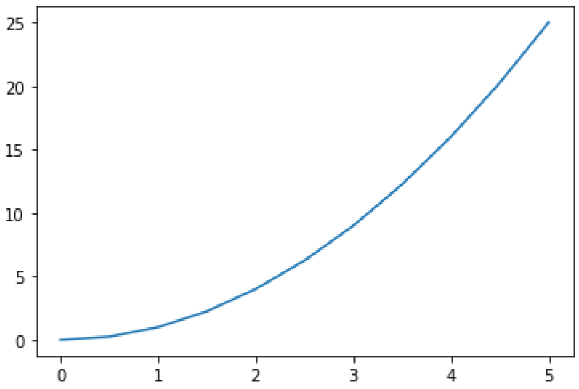
```
y
```

Out[4]:

```
array([ 0.  ,  0.25,  1.  ,  2.25,  4.  ,  6.25,  9.  , 12.25, 16.  ,
       20.25, 25.  ])
```

In [6]:

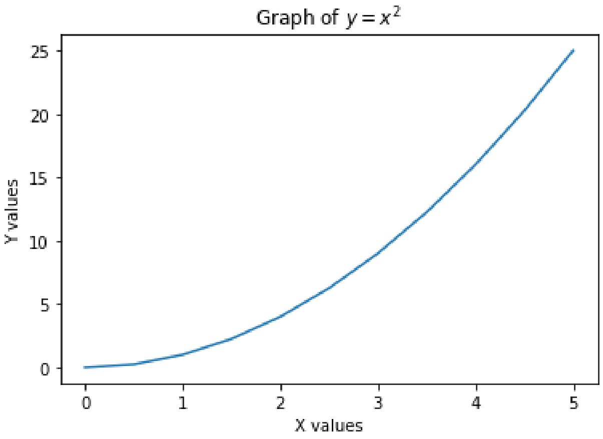
```
plt.plot(x, y)
plt.show()
```



Title and Labels

In [9]:

```
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('Graph of $y = x^2$')

plt.plot(x, y)
plt.show()
```



In [13]:

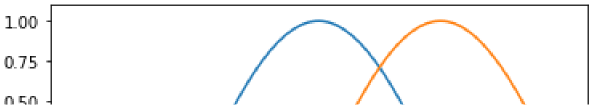
```
x1 = np.linspace(-np.pi, np.pi, 256)
x2 = np.linspace(-np.pi, np.pi, 256)

y1 = np.cos(x1)
y2 = np.sin(x2)
```

In [14]:

```
plt.plot(x1, y1, x2, y2)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x2bc46623848>,
 <matplotlib.lines.Line2D at 0x2bc4662c808>]
```



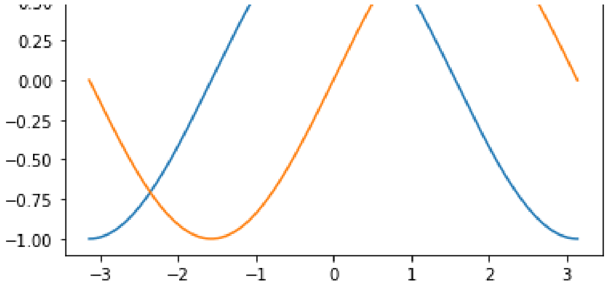
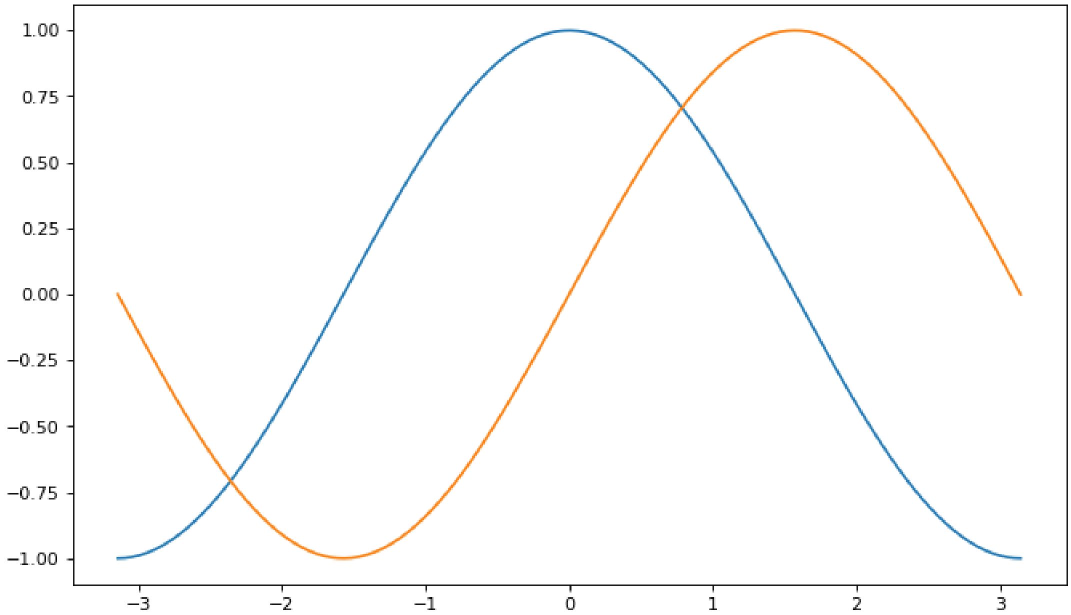


Figure Size and dpi

In [18]:

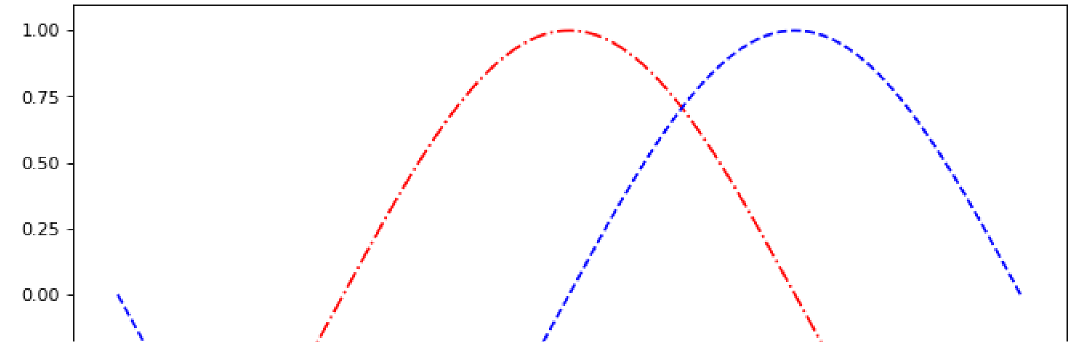
```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1)
plt.plot(x2, y2)

plt.show()
```

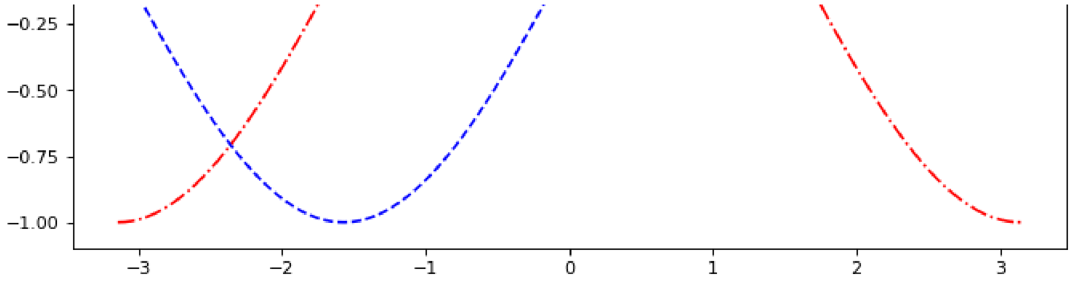


Styling Figure

In [22]:

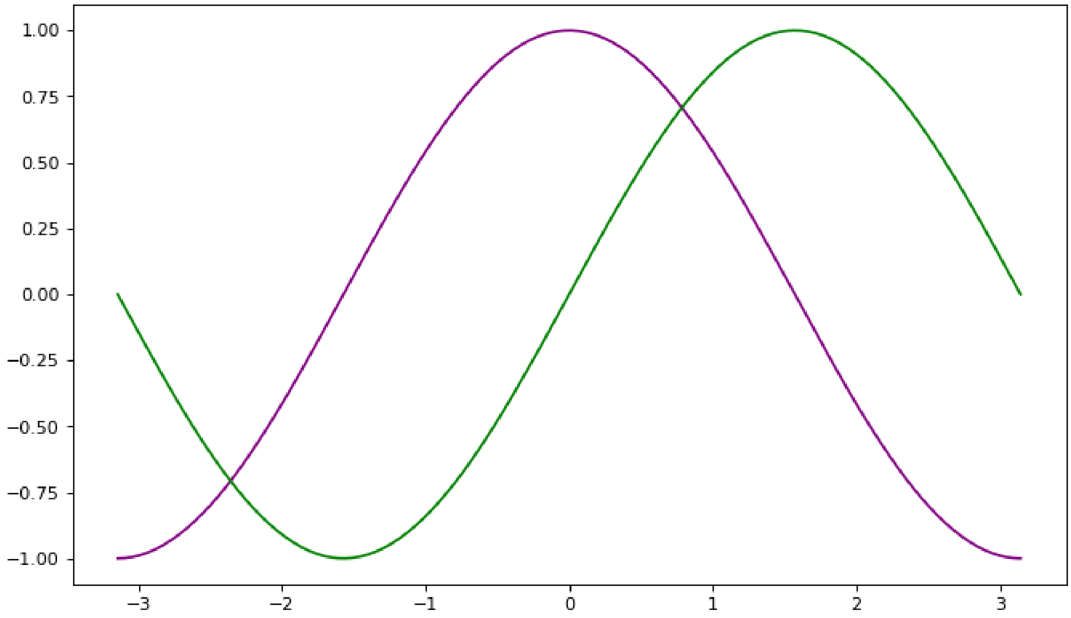
```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, 'r-.')
plt.plot(x2, y2, 'b--')

plt.show()
```





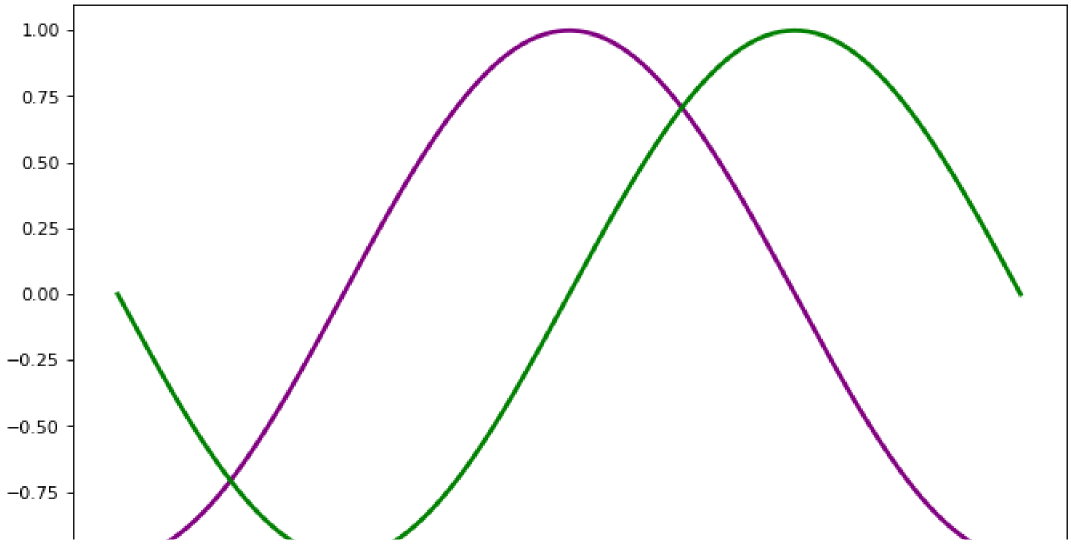
In [25]:

```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple')
plt.plot(x2, y2, color = 'green')

plt.show()
```



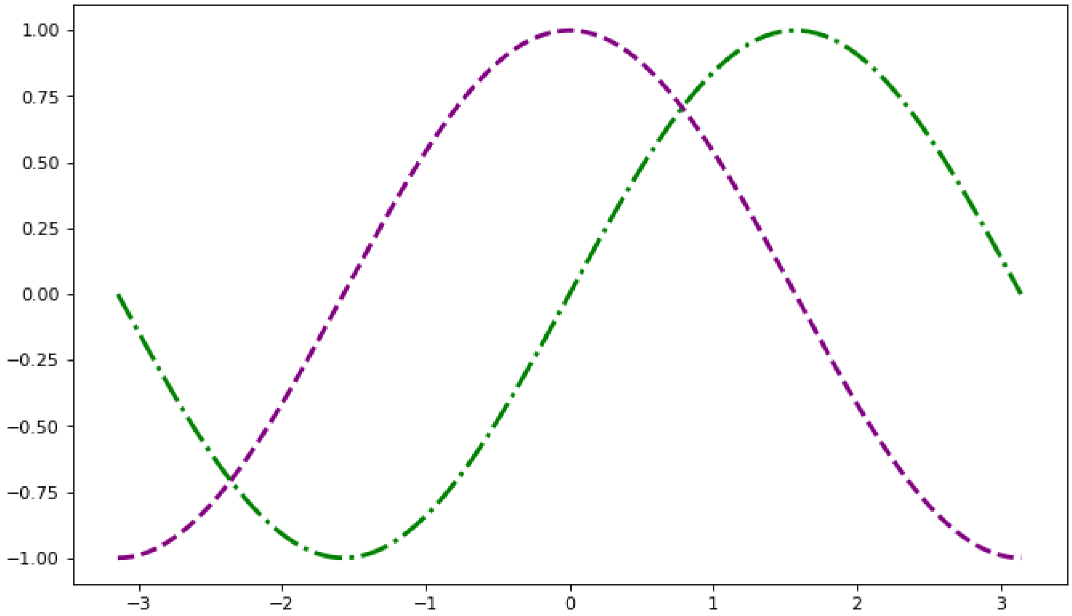
In [26]:

```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple', lw = 2.5)
plt.plot(x2, y2, color = 'green', lw = 2.5)

plt.show()
```



In [27]:

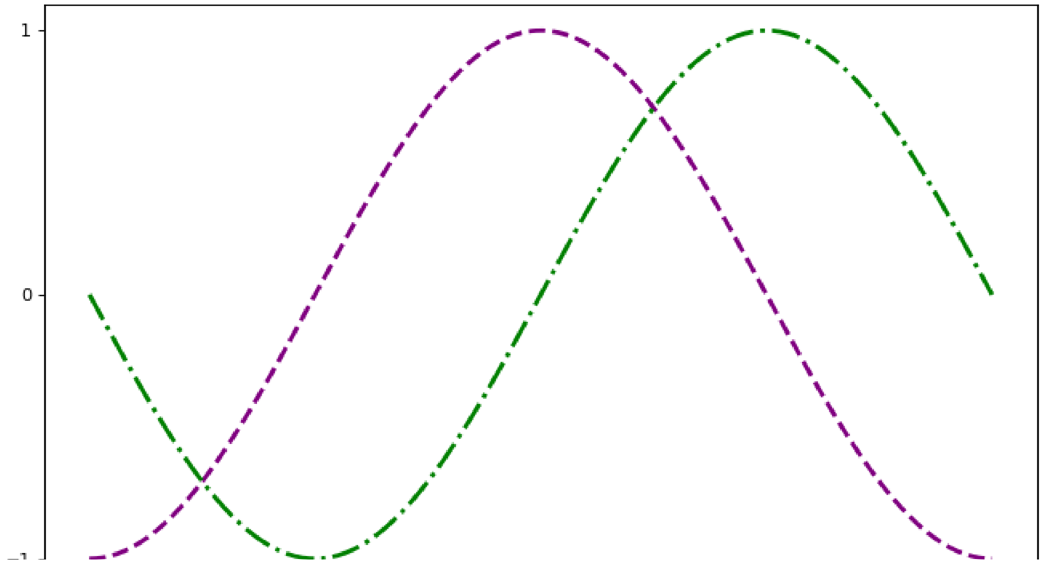
```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple', lw = 2.5, ls = '--')
plt.plot(x2, y2, color = 'green', lw = 2.5, ls = '-.')

plt.show()
```

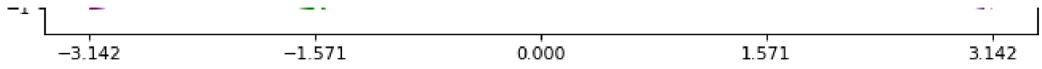


xticks and yticks

In [28]:

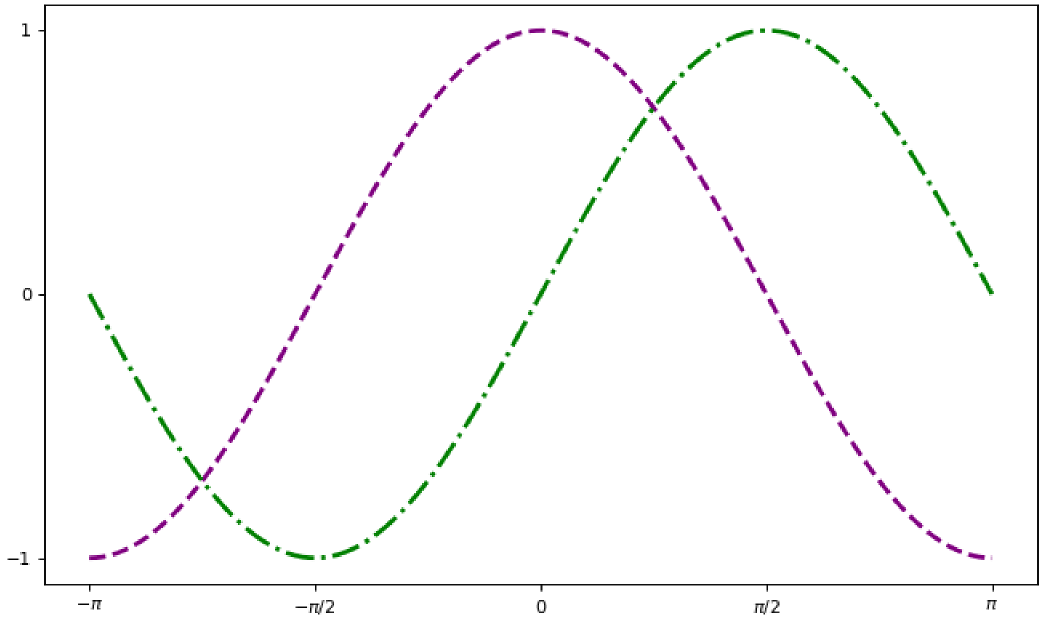
```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple', lw = 2.5, ls = '--')
plt.plot(x2, y2, color = 'green', lw = 2.5, ls = '-.')

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, 1])

plt.show()
```





In [29]:

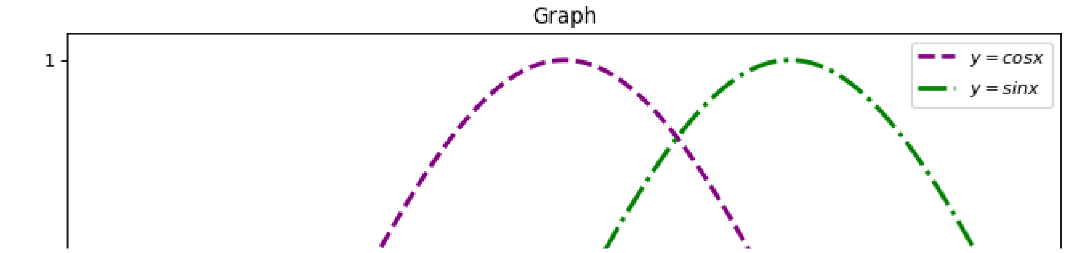
```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple', lw = 2.5, ls = '--')
plt.plot(x2, y2, color = 'green', lw = 2.5, ls = '-.')

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi], [r'$-\pi$', r'$-\pi/2$',
r'$0$',r'$\pi/2$',r'$\pi$'])
plt.yticks([-1, 0, 1])

plt.show()
```

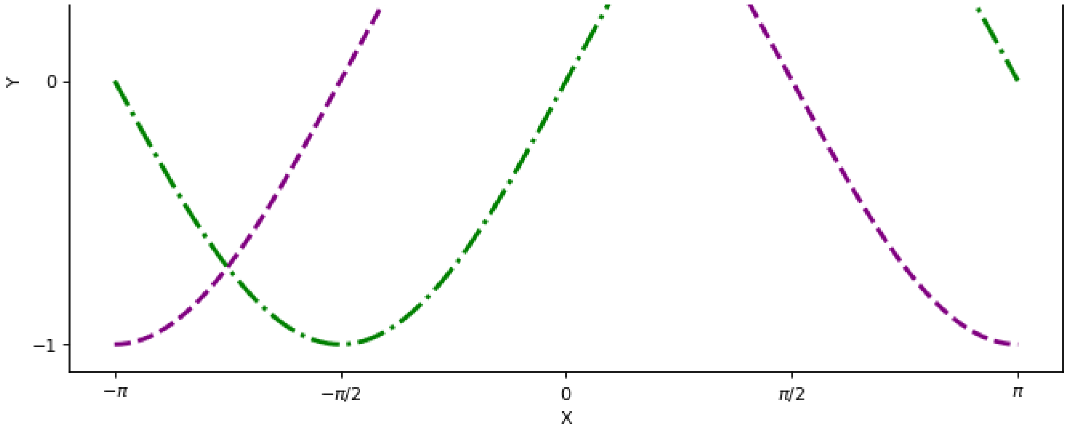


Legend

In [37]:

```
plt.figure(figsize = (10, 6), dpi = 80)

plt.plot(x1, y1, color = 'purple', lw = 2.5, ls = '--', label = '$y = cosx$')
plt.plot(x2, y2, color = 'green', lw = 2.5, ls = '-.', label = '$y = sinx$')

plt.xlabel('X')
plt.ylabel('Y')

plt.title('Graph')

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi], [r'$-\pi$', r'$-\pi/2$',
r'$0$',r'$\pi/2$',r'$\pi$'])
plt.yticks([-1, 0, 1])

plt.legend(loc = 0)

plt.show()
```





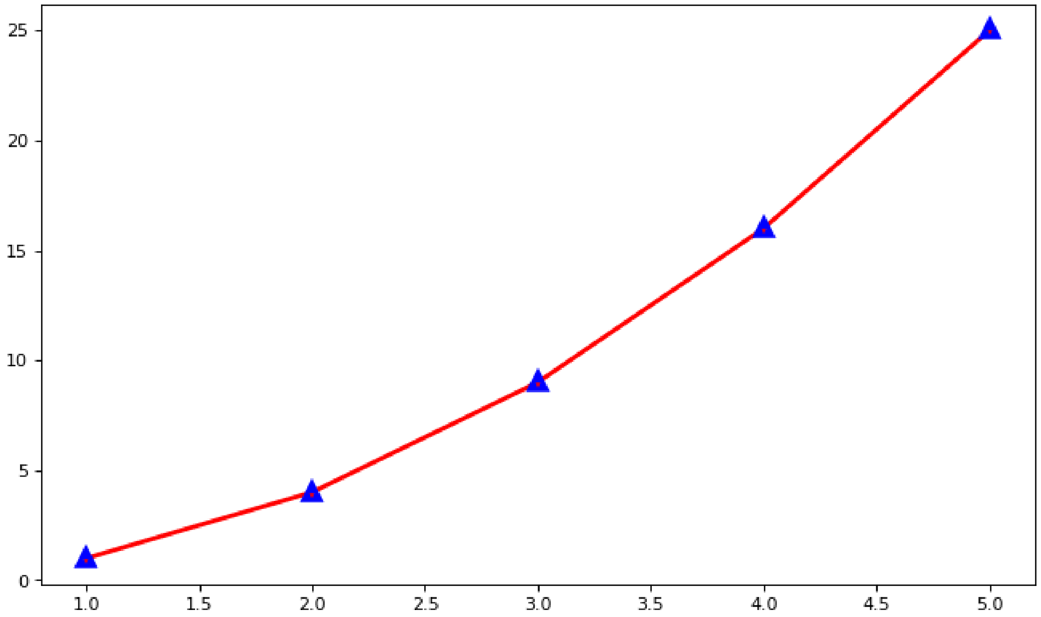
In [33]:

```
plt.legend??
```

Marker

In [45]:

```
plt.figure(figsize = (10,6), dpi = 80)

plt.plot([1,2,3,4,5], [1,4,9,16, 25], color = 'red', lw = 2.5, marker = '^', mec = 'blue', mew = 5)

plt.show()
```



In [46]:

```
plt.plot??
```

Subplot

In [48]:

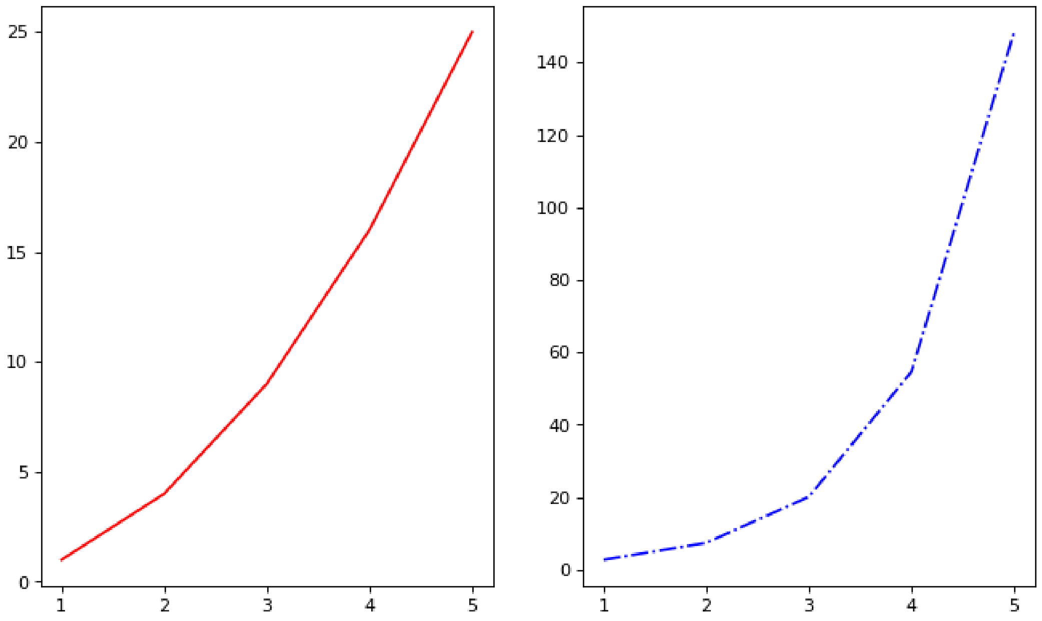
```
x = np.array([1,2,3,4,5])

y1 = x**2
y2 = np.exp(x)
```

In [54]:

```
plt.figure(figsize = (10, 6), dpi = 80)

plt.subplot(1, 2, 1)
plt.plot(x, y1, 'r')

plt.subplot(1,2, 2)
plt.plot(x, y2, 'b-.')
```

Out[54]:

```
[<matplotlib.lines.Line2D at 0x2bc46794748>]
```



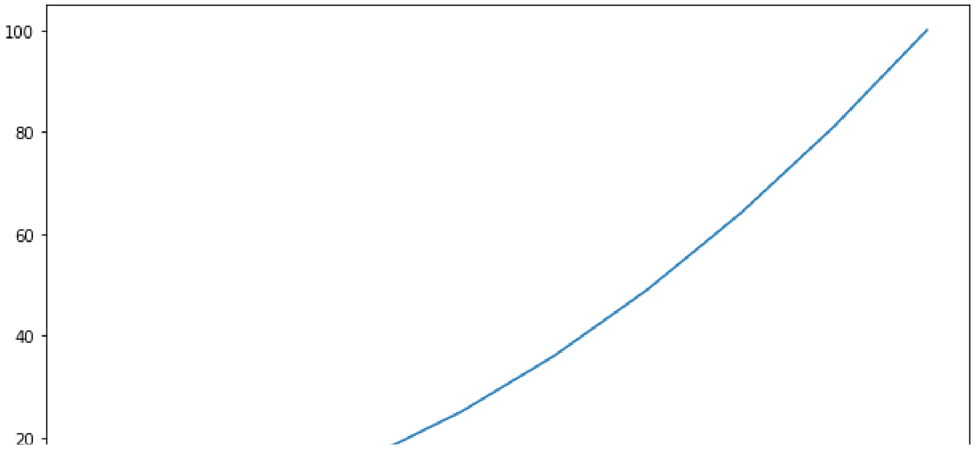
Object Oriented Methods

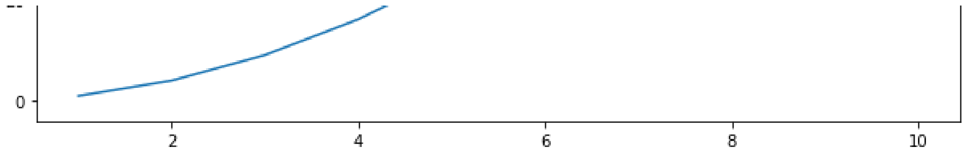
In [55]:

```
x = np.array([1,2,3,4,5,6,7,8,9,10])
y = x**2
```

In [60]:

```
fig = plt.figure(figsize = (10,6))

axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])

axes.plot(x, y)

plt.show()
```





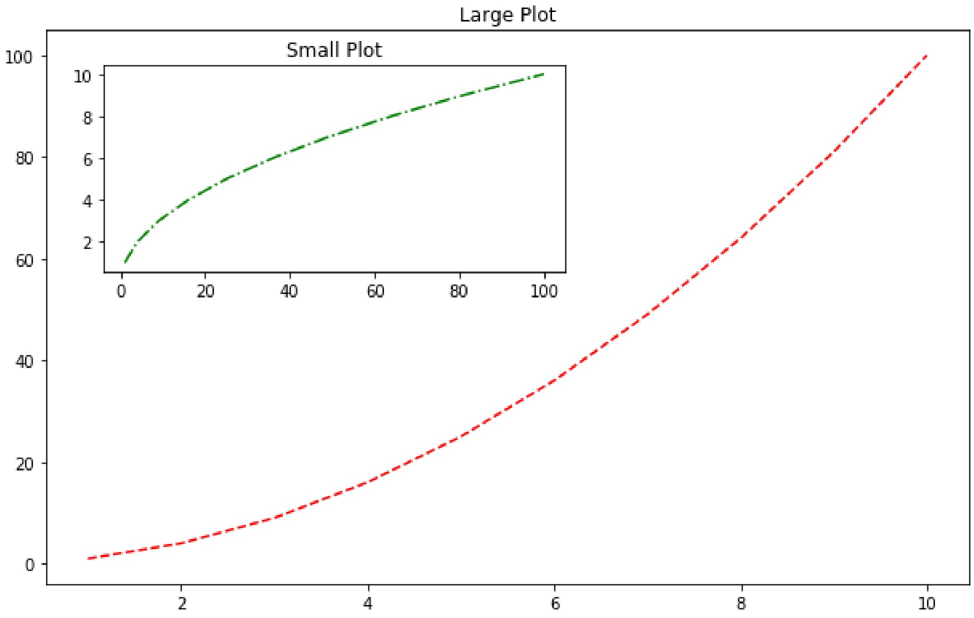
Axes in Figure

In [69]:

```
fig = plt.figure(figsize = (10,6))

axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes2 = fig.add_axes([0.15, 0.55, 0.4, 0.3])

axes1.plot(x,y, 'r--')
axes1.set_title('Large Plot')

axes2.plot(y,x, 'g-.')
axes2.set_title('Small Plot')
```

Out[69]:

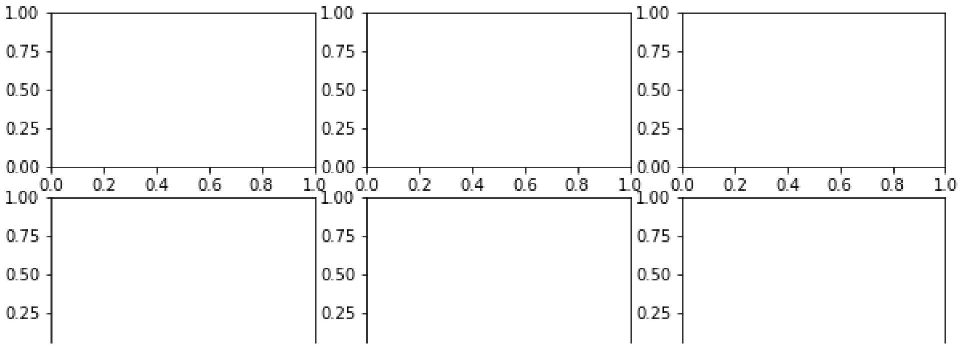
```
Text(0.5, 1.0, 'Small Plot')
```

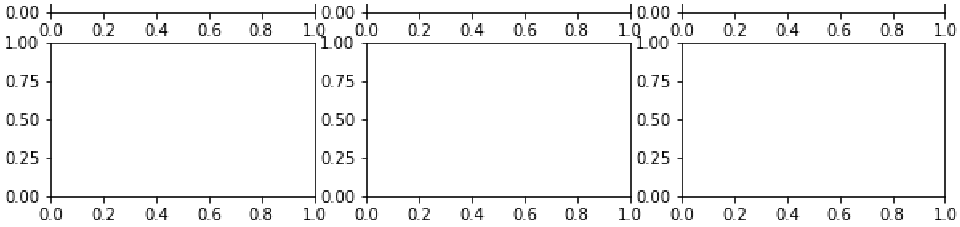


Subplots

In [71]:

```
fig = plt.figure(figsize = (10, 6))

axes = fig.subplots(3, 3)
```





In [72]:

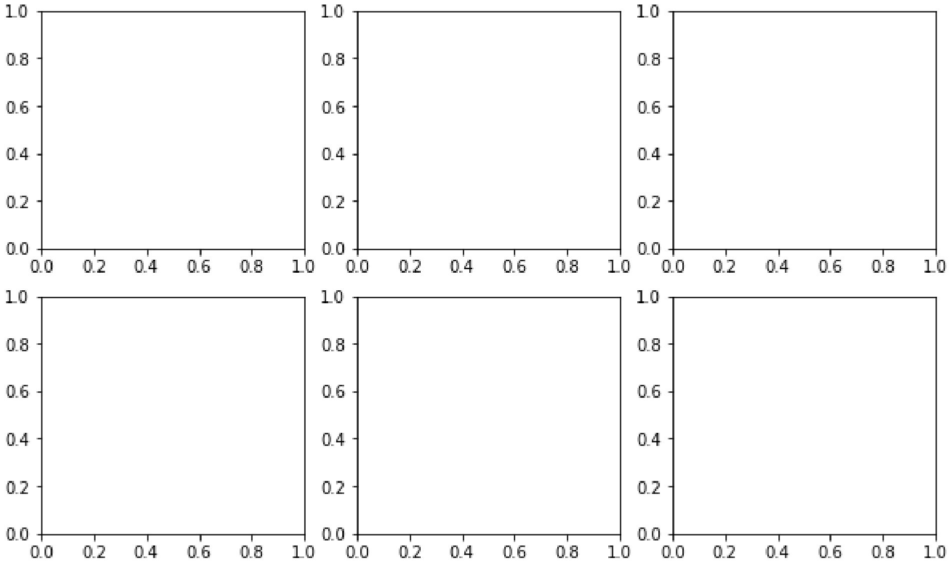
```
type(axes)
```

Out[72]:

```
numpy.ndarray
```

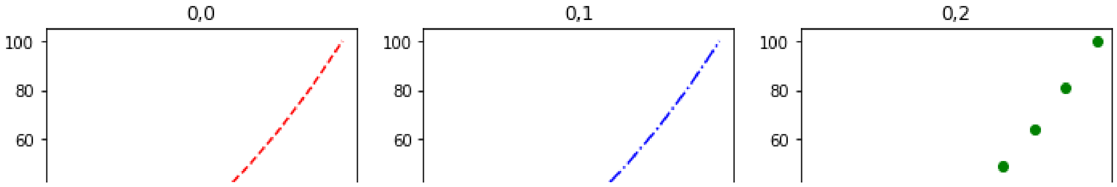
In [73]:

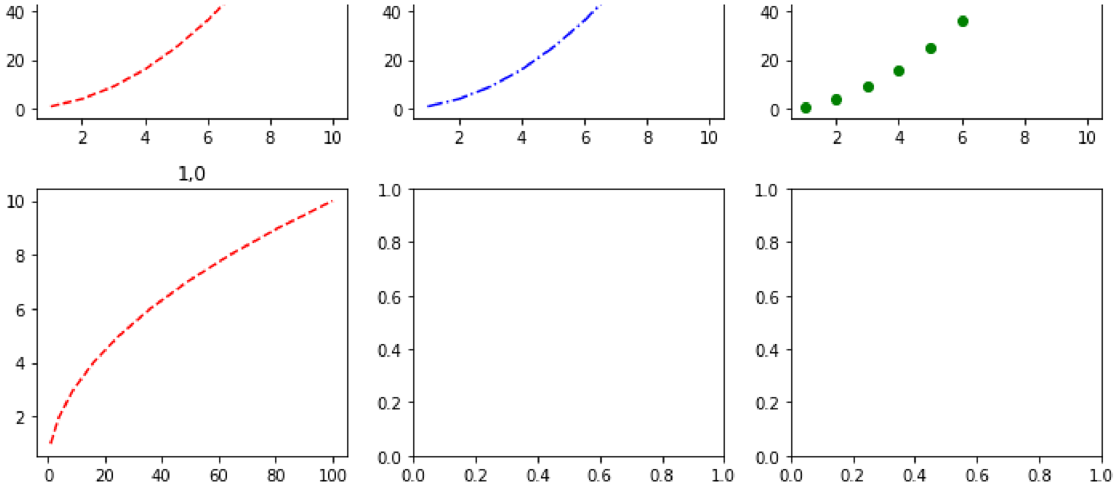
```
fig = plt.figure(figsize = (10, 6))

axes = fig.subplots(2, 3)
```



In [79]:

```
fig = plt.figure(figsize = (10, 6))

axes = fig.subplots(2, 3)

axes[0][0].plot(x,y,'r--')
axes[0][0].set_title('0,0')

axes[0][1].plot(x,y,'b-.')
axes[0][1].set_title('0,1')

axes[0][2].plot(x,y,'go')
axes[0][2].set_title('0,2')

axes[1,0].plot(y,x,'r--')
axes[1,0].set_title('1,0')

fig.tight_layout()
```





2D Plot

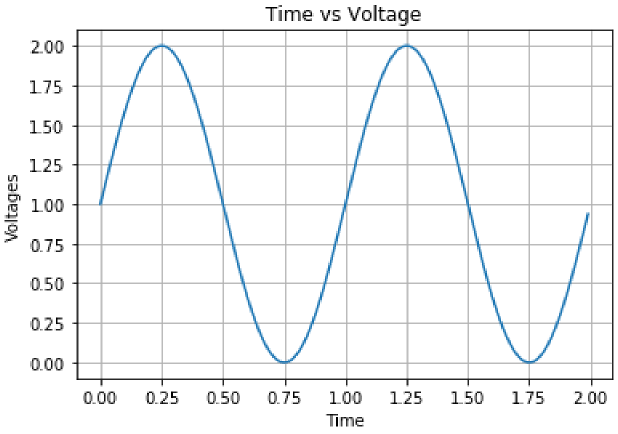
Line Plot

In [3]:

```
t = np.arange(0, 2,0.01)
s = 1 + np.sin(2*np.pi*t)
```

In [6]:

```
fig, ax = plt.subplots()

ax.plot(t, s)
ax.set(xlabel = 'Time', ylabel = 'Voltages', title = 'Time vs Voltage')

ax.grid()
plt.show()
```



Scatter Plot

In [2]:

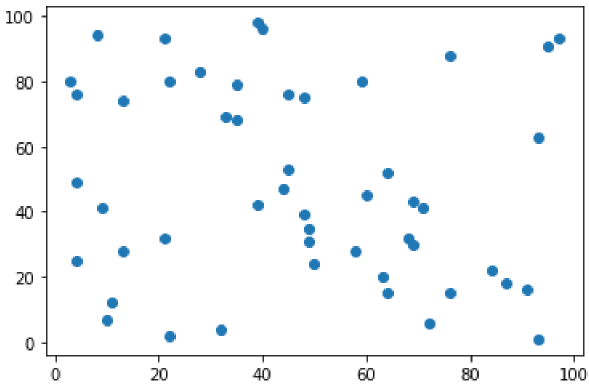
```
x = np.random.randint(100, size = 50)

y = np.random.randint(100, size = 50)
```

In [3]:

```
plt.scatter(x, y, marker = 'o')
```

Out[3]:

```
<matplotlib.collections.PathCollection at 0x25d1aed3ec8>
```



3D Plot

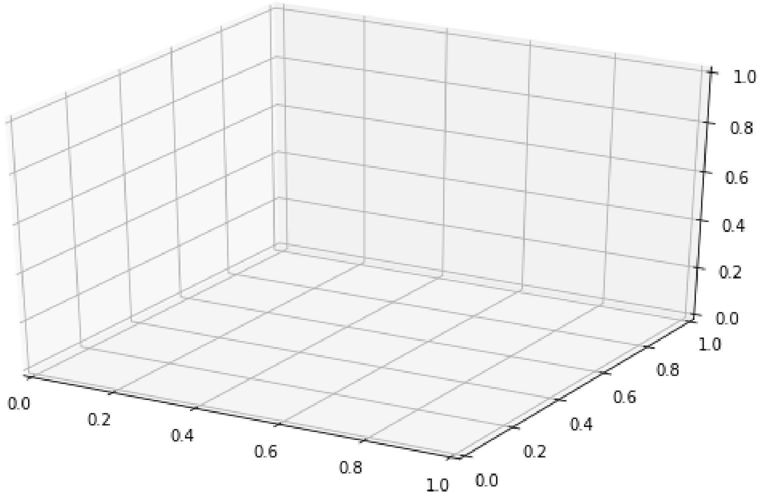
Line Plot

In [4]:

```
from mpl_toolkits import mplot3d
```

In [5]:

```
fig = plt.figure(figsize= (10,6))

ax = plt.axes(projection = '3d')
```



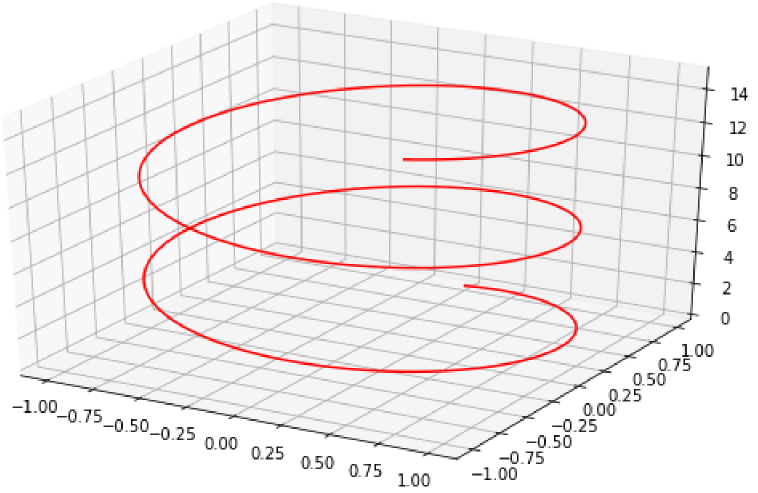
In [8]:

```
fig = plt.figure(figsize= (10,6))

ax = plt.axes(projection = '3d')

zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)

ax.plot3D(xline, yline, zline, 'red')
```

Out[8]:

```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x25d1e3bd1c8>]
```

```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x25d1e5bd1c8>]
```



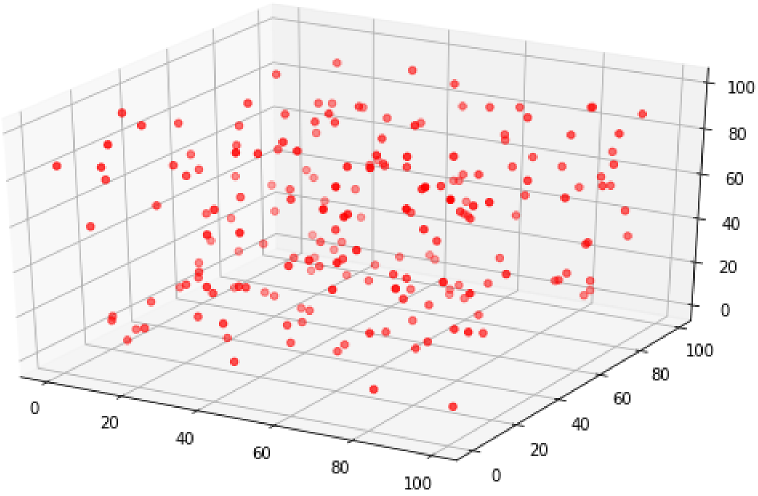
3D Scatter Plot

In [12]:

```
fig = plt.figure(figsize = (10, 6))

ax = plt.axes(projection = '3d')

x = np.random.randint(100, size = 200)
y = np.random.randint(100, size = 200)
z = np.random.randint(100, size = 200)

ax.scatter3D(x, y, z, c = 'red')
```

Out[12]:

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x25d1eb25d48>
```



3D Surface Plot

In [13]:

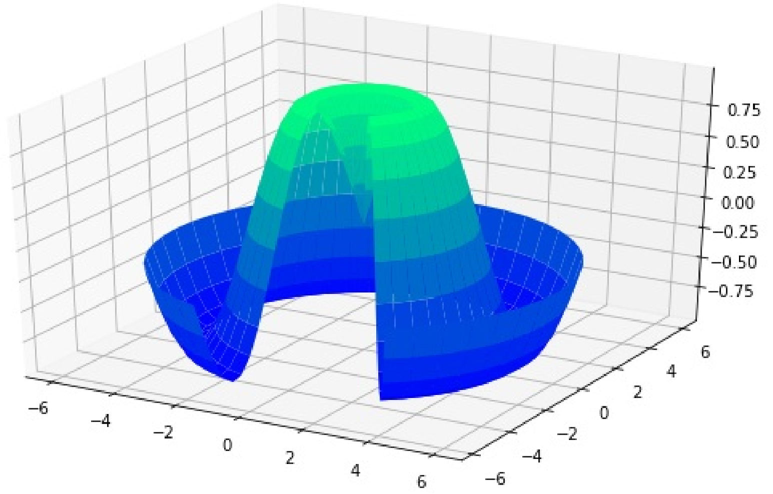
```
def f(x, y):
    return np.sin(np.sqrt(x**2+y**2))
```

In [15]:

```
r = np.linspace(0, 6, 20)
theta = np.linspace(-0.9*np.pi, 0.8*np.pi, 40)

r,theta = np.meshgrid(r, theta)

X = r*np.sin(theta)
Y = r*np.cos(theta)
Z = f(X,Y)
```

In [19]:

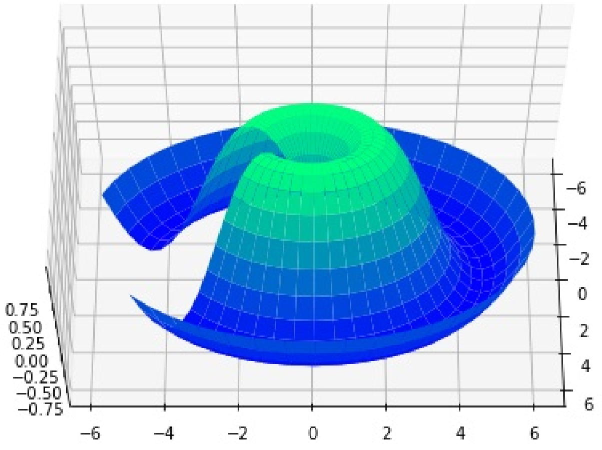
```
fig = plt.figure(figsize = (10, 6))

ax = plt.axes(projection = '3d')

ax.plot_surface(X, Y, Z, cmap = 'winter')

plt.show()
```



In [22]:

```
ax.view_init(60, 0)
fig
```

Out[22]:



Saving Figure

In [24]:

```
fig.savefig('test.pdf', dpi = 200)
```