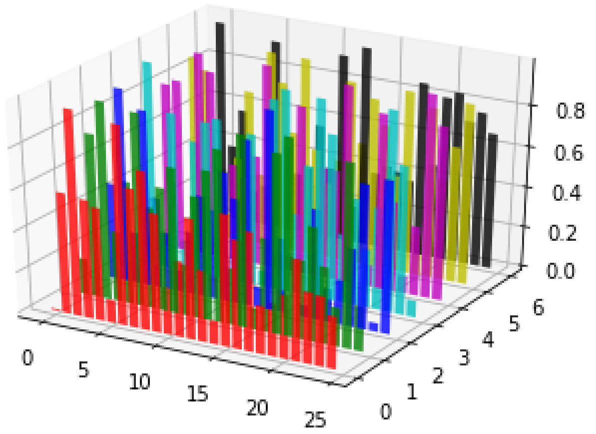
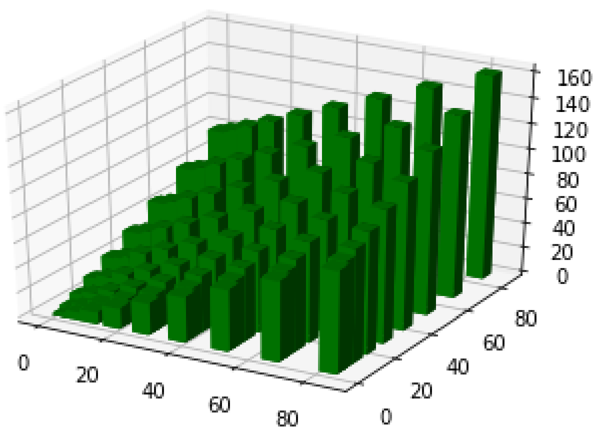
In [1]:

```
# You can show 2D bars in 3D axes. Let’s create a figure and axes,
# as shown here:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
# Let’s define colors for the bars.
colors = ['r', 'g', 'b', 'c', 'm', 'y','k']
yticks = [0, 1, 2, 3, 4, 5, 6]
# Now, let’s create bar graphs with the defined colors with the
# following loop:
for c, k in zip(colors, yticks):
    x = np.arange(25)
    y = np.random.rand(25)
    ax.bar(x, y, zs=k, zdir='y',color=c, alpha=0.8)
plt.show()
```



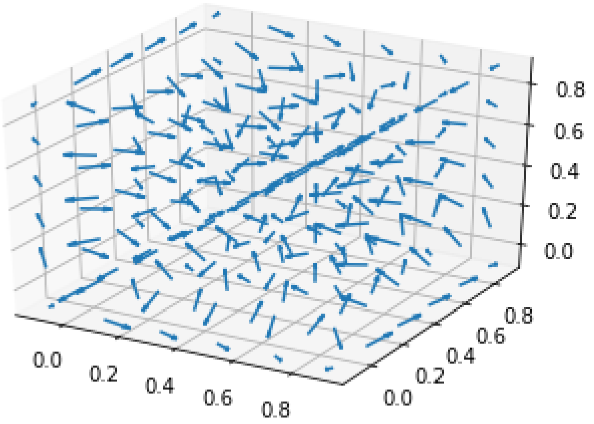
In [2]:

```
# You can also create a 3D bar graph with Matplotlib. Let’s create
# the data first, as shown here:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x = np.arange(10) * np.arange(10)
y = np.arange(10) * np.arange(10)
x, y = np.meshgrid(x, y)
x, y = x.ravel(), y.ravel()
top = x + y
bottom = np.zeros_like(top)
width = depth = 5
# You can then show this as 3D bars as follows:
ax.bar3d(x, y, bottom, width,depth, top,shade=True,color='g')
plt.show()
```



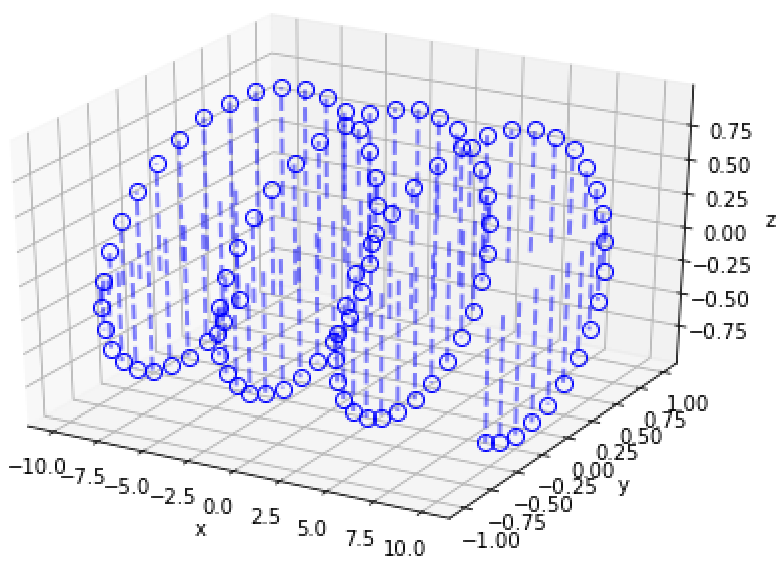
In [3]:

```
# Quiver and Stem Plots
# A quiver plot is used to represent directional entities
# (for example, vectors). Let’s define the data, as shown here:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
x = y = z = np.arange(-0.1, 1, 0.2)
X, Y, Z = np.meshgrid(x, y, z)
u = np.cos(np.pi * X) * np.sin(np.pi * Y) * np.sin(np.pi * Z)
v = -np.sin(np.pi * X) * np.cos(np.pi * Y) * np.sin(np.pi * Z)
w = np.sin(np.pi * X) * np.sin(np.pi * Y) * np.cos(np.pi * Z)
# Finally, you can visualize the data as follows:
ax.quiver(X, Y, Z, u, v, w,length=0.1,normalize=True)
plt.show()
```

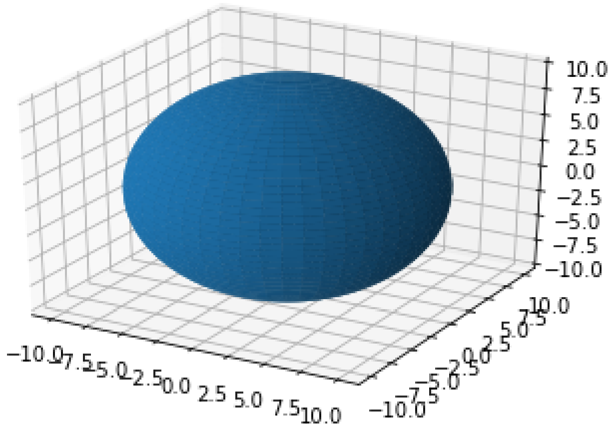


In [9]:

```
# You can also create stem plots where perpendicular lines are drawn
# in the visualization. Let’s use trigonometric functions to define the
# data, as shown here:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import Axes3D
from numpy import linspace, sin, cos
from pylab import figure, show
# generating some data
x = linspace(-10,10,100);
y = sin(x);
z = cos(x);
fig = figure()
ax = Axes3D(fig)
# plotting the stems
for i in range(len(x)):
    ax.plot([x[i], x[i]], [y[i], y[i]], [0, z[i]],
          '--', linewidth=2, color='b', alpha=.5)
# plotting a circle on the top of each stem
ax.plot(x, y, z, 'o', markersize=8,
        markerfacecolor='none', color='b',label='ib')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
show()
```



In [5]:
```
# 3D Volumes
# You can show 3D volumetric data as enclosed surfaces.
# Let's create such data as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
x = 10 * np.outer(np.cos(u), np.sin(v))
y = 10 * np.outer(np.sin(u), np.sin(v))
z = 10 * np.outer(np.ones(np.size(u)), np.cos(v))
# You can show this data as a sphere as follows:
ax.plot_surface(x, y, z)
plt.show()
```



In [6]:
```
# You can also use the function voxels() to visualize a volume as follows:

%matplotlib qt
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
ma = np.random.randint(1, 3, size=(3, 3, 3))
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.voxels(ma, edgecolor='k')
plt.show()
```