Simple matplotlib Libraries

from matplotlib import pyplot as plt plt.plot([1,2,3],[4,5,1]) plt.show()

In [1]:

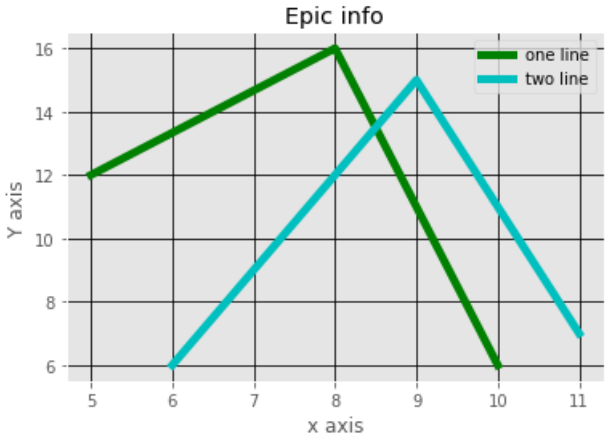
```
from matplotlib import pyplot as plt

x= [5,8,10]
y= [12,16,6]

plt.plot(x,y)
plt.title('info',color='b')
plt.ylabel('y axis',color='w')
plt.xlabel('x axis',color='w')
plt.show()
```

Out[1]:

```
<Figure size 640x480 with 1 Axes>
```

In [10]:

```
from matplotlib import pyplot as plt
from matplotlib import style
style.use('ggplot')

x= [5,8,10]
y= [12,16,6]

x2= [6,9,11]
y2= [6,15,7]

plt.plot(x,y,'g',label="one line",linewidth= 5)
plt.plot(x2,y2,'c',label="two line",linewidth= 5)

plt.title('Epic info')
plt.ylabel('Y axis')
plt.xlabel('x axis')

plt.legend()
plt.grid(True,color='k')

plt.show()
```

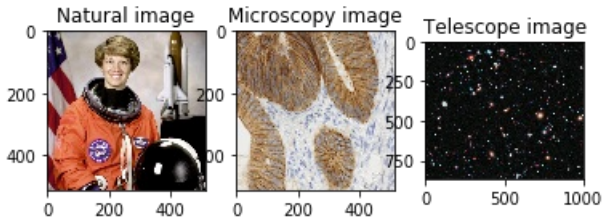


In [3]:

```
%matplotlib notebook
```

In [2]:

```
import matplotlib.pyplot as plt
from skimage import data

astronaut = data.astronaut()
ihc = data.immunohistochemistry()
hubble = data.hubble_deep_field()

#Initialize the subplot panels side by side

fig, ax = plt.subplots(nrows=1, ncols=3)

#show an image in each subplot
ax[0].imshow(astronaut)
ax[0].set_title('Natural image')
ax[1].imshow(ihc)
ax[1].set_title('Microscopy image')
ax[2].imshow(hubble)
ax[2].set_title('Telescope image');
```



In [10]:

```
%matplotlib notebook
```

In [11]:

```
import matplotlib.pyplot as plt
```

In [12]:

```
import tempfile

#Create a temporary directory

d = tempfile.mkdtemp()
```

In [13]:

```
import os

os.path.basename('http://google.com/attention.zip')
```

Out[13]:

```
'attention.zip'
```

In [ ]:

```
from urllib.request import urlretrieve

#Define url

url = 'http://www.fil.ion.ucl.ac.uk/spm/download/data/attention/attention.zip'

#Retrieve the data
fn, info = urlretrieve(url, os.path.join(d, 'attention.zip'))
```

In [3]:

```
import nibabel
```

In [9]:

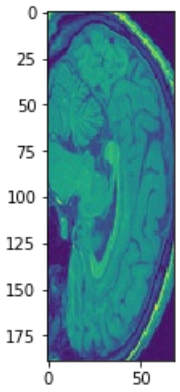
```
from skimage import io

struct_arr=io.imread("https://s3.amazonaws.com/assets.datacamp.com/blog_assets/attention-mri.tif")
```

In [13]:

```
plt.imshow(struct_arr[75])
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x5d1b10>
```

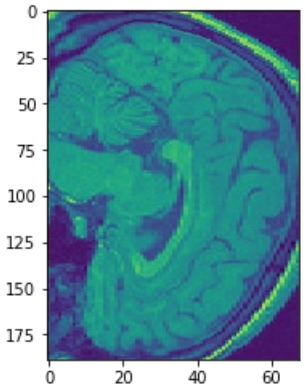


In [ ]:

```
plt.imshow(struct_arr[75], aspect=0.5)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x15b475f0>
```

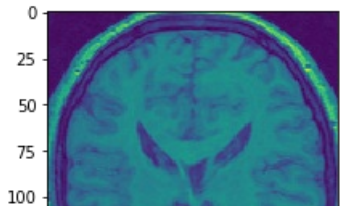


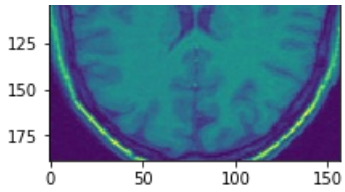
In [14]:

```
struct_arr2 = struct_arr.T

plt.imshow(struct_arr2[34])
```

Out[14]:

```
<matplotlib.image.AxesImage at 0xc39510>
```



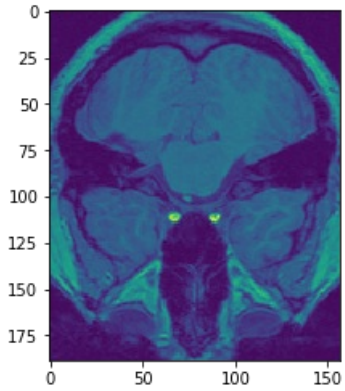


In [15]:

```
plt.imshow(struct_arr2[5])
```

Out[15]:

```
<matplotlib.image.AxesImage at 0x1515b5f0>
```



In [8]:

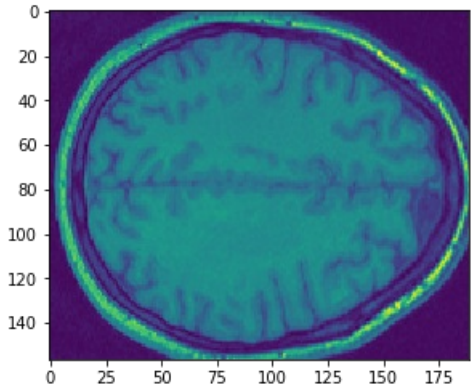
```
def previous_slice():
    pass
def next_slice():
    pass
def process_key(event):
    if event.key == 'j':
        previous_slice()
    elif event.key == 'k':
        next_slice()
```

In [9]:

```
fig, ax = plt.subplots()
ax.imshow(struct_arr[..., 43])
fig.canvas.mpl_connect('key_press_event', process_key)
```

Out[9]:

```
4
```

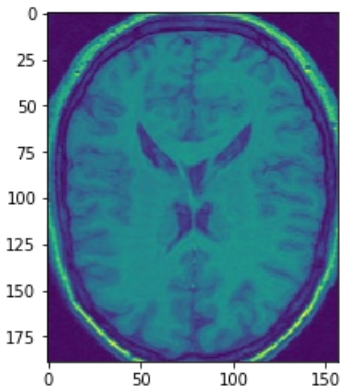


In [17]:

```
def multi_slice_viewer(volume):
    fig, ax = plt.subplots()
    ax.volume = volume
    ax.index = volume.shape[0] //2
    ax.imshow(volume[ax.index])
    fig.canvas.mpl_connect('key_press_event', process_key)

def process_key(event):
    fig = event.canvas.figure
    ax = fig.axes[0]
    if event.key == 'j':
        previous_slice(ax)
    elif event.key == 'k':
        next_slice(ax)
    fig.canvas.draw()

def previous_slice(ax):
    """Go the previous slice"""

    volume = ax.volume
    ax.index = (ax.index - 1 ) % volume.shape[0] #wrap around using%
    ax.images[0].set_array(volume[ax.index])

def next_slice(ax):
    """Go to the next slice"""
    volume = ax.volume
    ax.index = (ax.index + 1) % volume.shape[0]
    ax.images[0].set_array(volume[ax.index])
```

In [18]:

```
multi_slice_viewer(struct_arr2)
```



In [7]:

```
#plt.rcParams["keymap.<command>"] = ['<key1>', '<key2>']
```

In [19]:

```
def remove_keymap_conflicts(new_keys_set):
    for prop in plt.rcParams:
        if prop.startswith('keymap.'):
            keys = plt.rcParams[prop]
            remove_list = set(keys) & new_keys_set
            for key in remove_list:
                keys.remove(key)
```

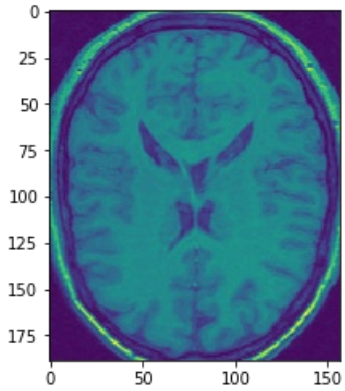
In [20]:

```
def multi_slice_viewer(volume):
    remove_keymap_conflicts({'j', 'k'})
    fig, ax = plt.subplots()
    ax.volume = volume
    ax.index = volume.shape[0] // 2
    ax.imshow(volume[ax.index])
    fig.canvas.mpl_connect('key_press_event', process_key)
```

```
def process_key(event):
    fig = event.canvas.figure
    ax = fig.axes[0]
    if event.key == 'j':
        previous_slice(ax)
    elif event.key == 'k':
        next_slice()
    fig.canvas.draw()

def previous_slice(ax):
    volume = ax.volume
    ax.index = (ax.index - 1 ) % volume.shape[0] #wrap around using%
    ax.images[0].set_array(volume[ax.index])



def next_slice(ax):
    """Go to the next slice"""
    volume = ax.volume
    ax.index = (ax.index + 1) % volume.shape[0]
    ax.images[0].set_array(volume[ax.index])
```

In [21]:

```
multi_slice_viewer(struct_arr2)
```



In [12]:

```
#bar Grapg
import matplotlib.pyplot as plt
plt.bar([1,3,5,7,9],[5,2,7,8,2],label='Example one')
plt.bar([2,4,6,8,10],[8,6,2,5,6],label='Example two',color='g')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title('My plot yo!')
plt.show()
```

In [1]:

```
#histogram
import matplotlib.pyplot as plt
population_ages= [22,55,62,45,21,22,34,56,87,98,43,55,11,65,44,22,111,344,776,443,122,344,111,130,1
15,112,80]
bins= [0,10,20,30,40,50,60,70,80,90,100,110,120,130]
plt.hist(population_ages,bins,histtype='bar',rwidth=0.8)

plt.xlabel('x')
plt.ylabel('y')
plt.title('histogram')
plt.legend()
plt.show()
```

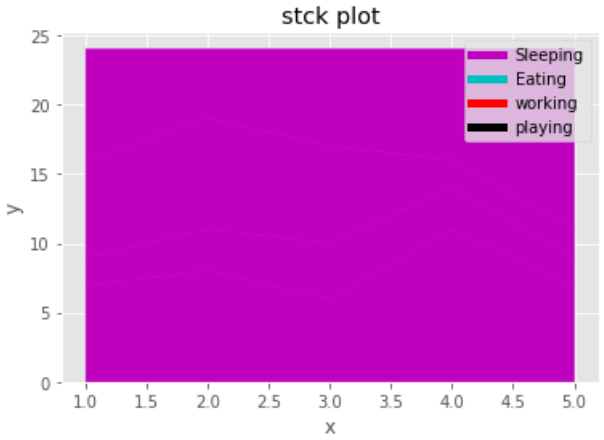
```
No handles with labels found to put in legend.
```

Out[1]:

```
<Figure size 640x480 with 1 Axes>
```

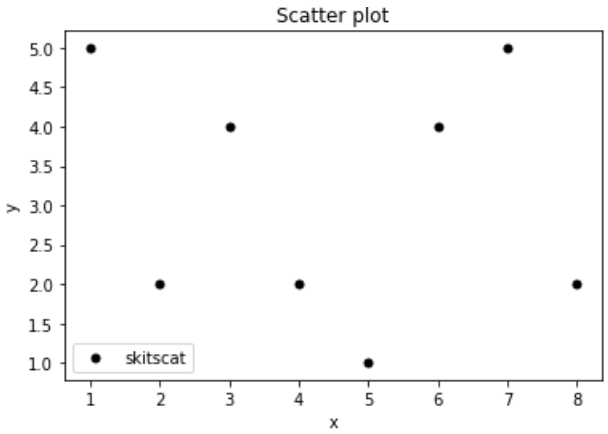
In [19]:

```
import matplotlib.pyplot as plt
days=[1,2,3,4,5]
sleeping=[7,8,6,11,7]
eating=[2,3,4,3,2]
working=[7,8,7,2,2]
playing=[8,5,7,8,13]
plt.plot([],[],color='m',label='Sleeping',linewidth=5)
plt.plot([],[],color='c',label='Eating',linewidth=5)
plt.plot([],[],color='r',label='working',linewidth=5)
plt.plot([],[],color='k',label='playing',linewidth=5)
plt.stackplot(days, sleeping, eating, working, playing, color=['m','c','r','k'])
plt.xlabel('x')
plt.ylabel('y')
plt.title('stck plot')
plt.legend()
plt.show()
```



In [5]:

```
#Scatter plot
import matplotlib.pyplot as plt
x= [1,2,3,4,5,6,7,8]
y= [5,2,4,2,1,4,5,2]
plt.scatter(x,y,label='skitscat',color='k',s=25,marker="o")
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter plot')
plt.legend()
plt.show()
```



In [2]:

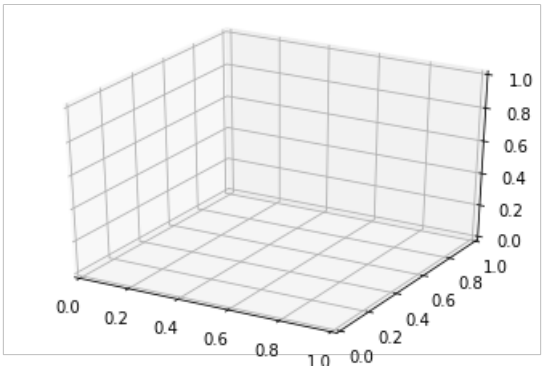
```
from mpl_toolkits import mplot3d
```

In [3]:

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

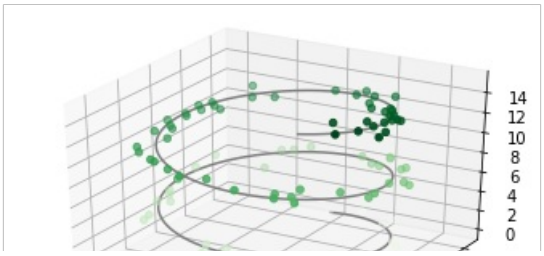
In [4]:

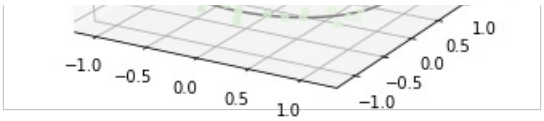
```
fig = plt.figure()
ax = plt.axes(projection='3d')
```



In [6]:

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

# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')

# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```



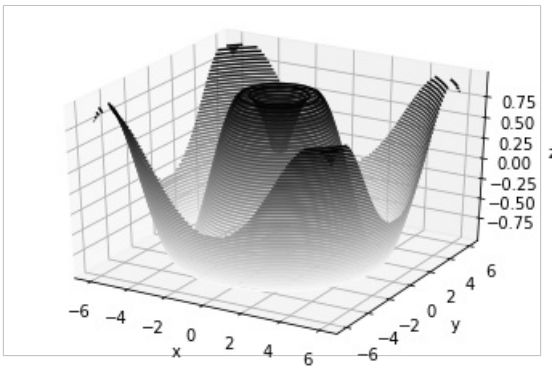


In [7]:

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

x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)
```

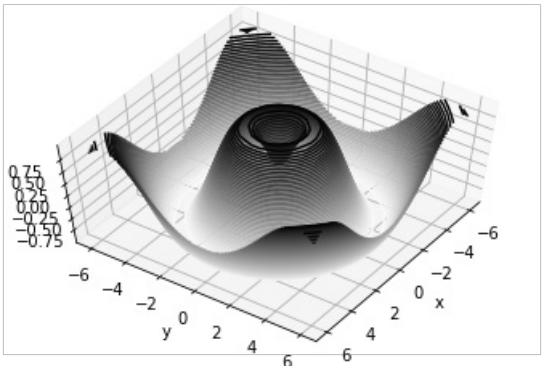
In [8]:

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z');
```



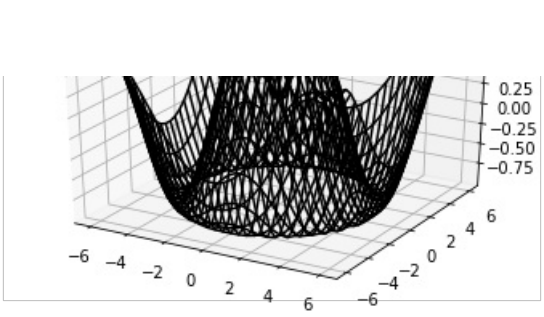
In [9]:

```
ax.view_init(60, 35)
fig
```



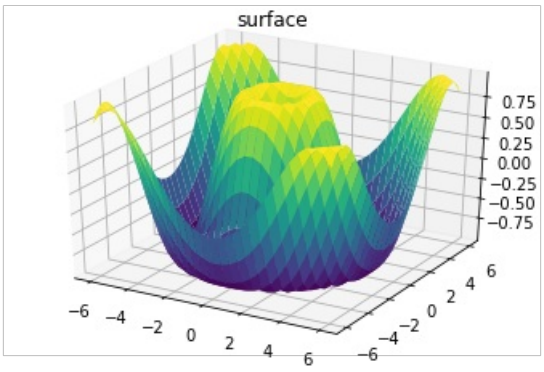
In [10]:

```
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, Z, color='black')
ax.set_title('wireframe');
```



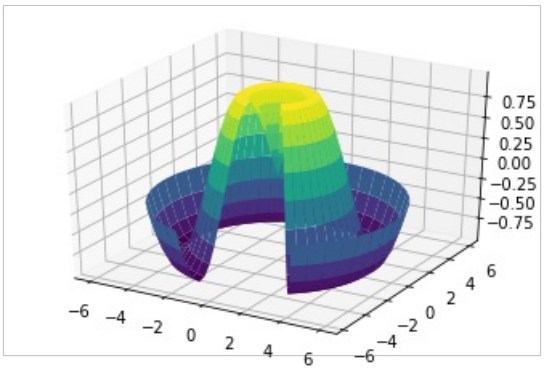
In [11]:

```
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.set_title('surface');
```



In [12]:

```
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)

ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none');
```



In [15]:

```

```

In [18]:

```
theta = 2 * np.pi * np.random.random(1000)
r = 6 * np.random.random(1000)
x = np.ravel(r * np.sin(theta))
y = np.ravel(r * np.cos(theta))
z = f(x, y)
```

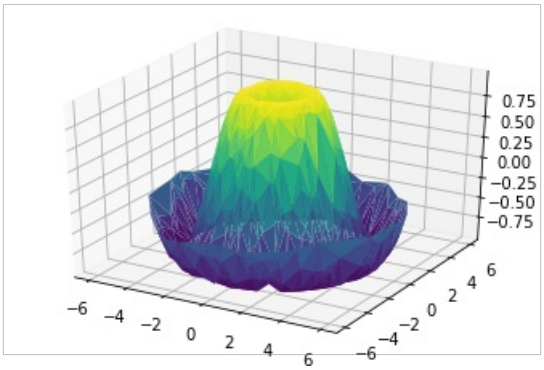
In [1]:

```
ax = plt.axes(projection='3d')
ax.scatter(x, y, z, c=z, cmap='viridis', linewidth=0.5);
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-6186dc2db8b1> in <module>
----> 1 ax = plt.axes(projection='3d')
      2 ax.scatter(x, y, z, c=z, cmap='viridis', linewidth=0.5);

NameError: name 'plt' is not defined
```

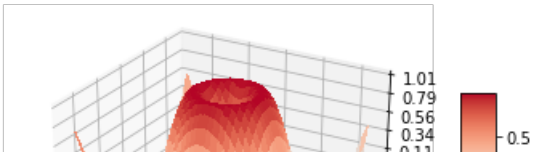
In [20]:

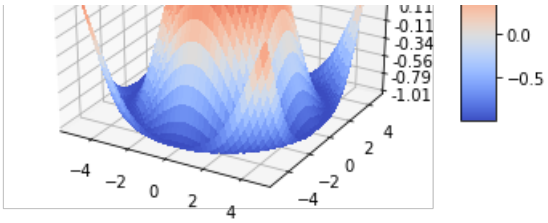
```
ax = plt.axes(projection='3d')
ax.plot_trisurf(x, y, z,
                cmap='viridis', edgecolor='none');
```



In [21]:

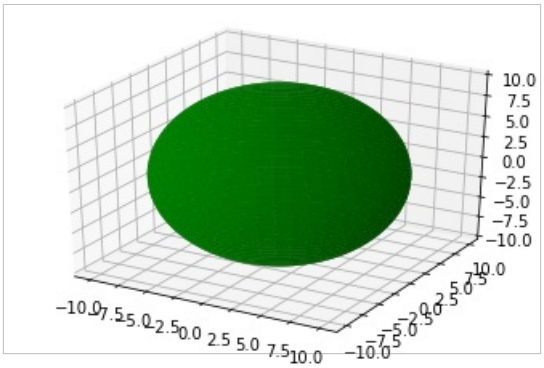
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


fig = plt.figure()
ax = fig.gca(projection='3d')

# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```





In [11]:

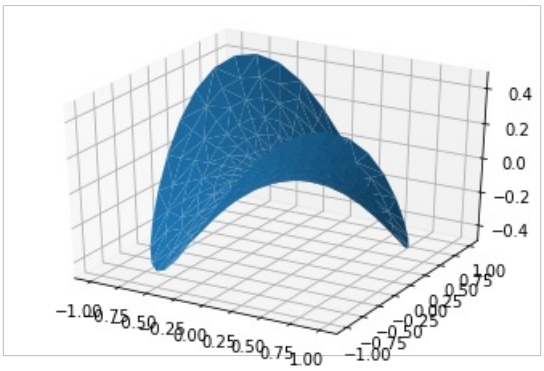
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Make data
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))

# Plot the surface
ax.plot_surface(x, y, z, color='g')

plt.show()
```



In [23]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np


n_radii = 8
n_angles = 36

# Make radii and angles spaces (radius r=0 omitted to eliminate duplication).
radii = np.linspace(0.125, 1.0, n_radii)
angles = np.linspace(0, 2*np.pi, n_angles, endpoint=False)

# Repeat all angles for each radius.
angles = np.repeat(angles[..., np.newaxis], n_radii, axis=1)

# Convert polar (radii, angles) coords to cartesian (x, y) coords.
# (0, 0) is manually added at this stage,  so there will be no duplicate
# points in the (x, y) plane.
x = np.append(0, (radii*np.cos(angles)).flatten())
y = np.append(0, (radii*np.sin(angles)).flatten())

# Compute z to make the pringle surface.
z = np.sin(-x*y)

fig = plt.figure()
ax = fig.gca(projection='3d')
```

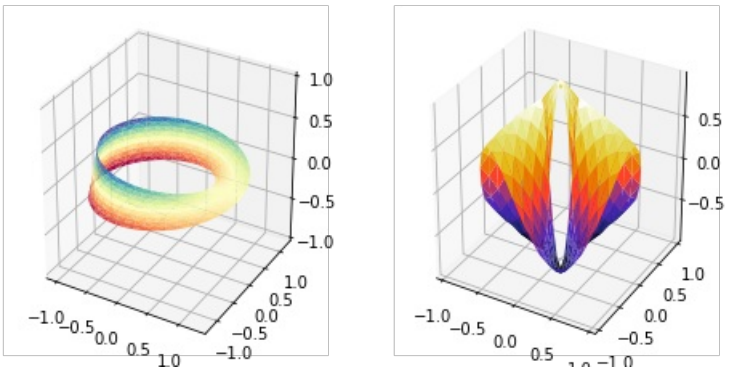
```
ax.plot_trisurf(x, y, z, linewidth=0.2, antialiased=True)

plt.show()
```



In [3]:

```
'''
============================
More triangular 3D surfaces
============================

Two additional examples of plotting surfaces with triangular mesh.

The first demonstrates use of plot_trisurf's triangles argument, and the
second sets a Triangulation object's mask and passes the object directly
to plot_trisurf.
'''

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.tri as mtri


fig = plt.figure(figsize=plt.figaspect(0.5))

#=============
# First plot
#=============

# Make a mesh in the space of parameterisation variables u and v
u = np.linspace(0, 2.0 * np.pi, endpoint=True, num=50)
v = np.linspace(-0.5, 0.5, endpoint=True, num=10)
u, v = np.meshgrid(u, v)
u, v = u.flatten(), v.flatten()

# This is the Mobius mapping, taking a u, v pair and returning an x, y, z
# triple
x = (1 + 0.5 * v * np.cos(u / 2.0)) * np.cos(u)
y = (1 + 0.5 * v * np.cos(u / 2.0)) * np.sin(u)
z = 0.5 * v * np.sin(u / 2.0)

# Triangulate parameter space to determine the triangles
tri = mtri.Triangulation(u, v)

# Plot the surface.  The triangles in parameter space determine which x, y, z
# points are connected by an edge.
ax = fig.add_subplot(1, 2, 1, projection='3d')
ax.plot_trisurf(x, y, z, triangles=tri.triangles, cmap=plt.cm.Spectral)
ax.set_zlim(-1, 1)


#=============
# Second plot
#=============

# Make parameter spaces radii and angles.
n_angles = 36
n_radii = 8
min_radius = 0.25
radii = np.linspace(min_radius, 0.95, n_radii)
```

```
angles = np.linspace(0, 2*np.pi, n_angles, endpoint=False)
angles = np.repeat(angles[..., np.newaxis], n_radii, axis=1)
angles[:, 1::2] += np.pi/n_angles

# Map radius, angle pairs to x, y, z points.
x = (radii*np.cos(angles)).flatten()
y = (radii*np.sin(angles)).flatten()
z = (np.cos(radii)*np.cos(angles*3.0)).flatten()

# Create the Triangulation; no triangles so Delaunay triangulation created.
triang = mtri.Triangulation(x, y)

# Mask off unwanted triangles.
xmid = x[triang.triangles].mean(axis=1)
ymid = y[triang.triangles].mean(axis=1)
mask = np.where(xmid**2 + ymid**2 < min_radius**2, 1, 0)
triang.set_mask(mask)

# Plot the surface.
ax = fig.add_subplot(1, 2, 2, projection='3d')
ax.plot_trisurf(triang, z, cmap=plt.cm.CMRmap)


plt.show()
```



In [14]:

```
x= np.linspace(0,10,10)
print(x)
```

```
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444  5.55555556
  6.66666667  7.77777778  8.88888889 10.        ]
```

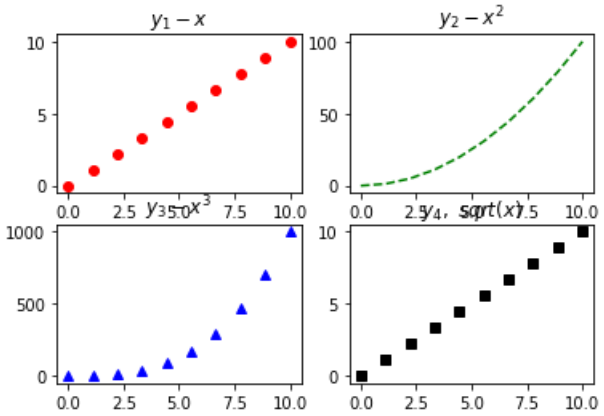
In [25]:

```
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
x=np.linspace(0,10,10)
y1= x
y2= x**2
y3= x**3
y4= np.sqrt(x)
plt.figure()
plt.subplot(2,2,1)
plt.plot(x,y1,'ro')
plt.title('$y_1-x$')

plt.subplot(2,2,2)
plt.plot(x,y2,'g--')
plt.title('$y_2-x^2$')

plt.subplot(2,2,3)
plt.plot(x,y3,'b^')
plt.title('$y_3-x^3$')

plt.subplot(2,2,4)
```

```
plt.plot(x,y1,'ks')
plt.title('$y_4, \ sqrt(x)$')
```

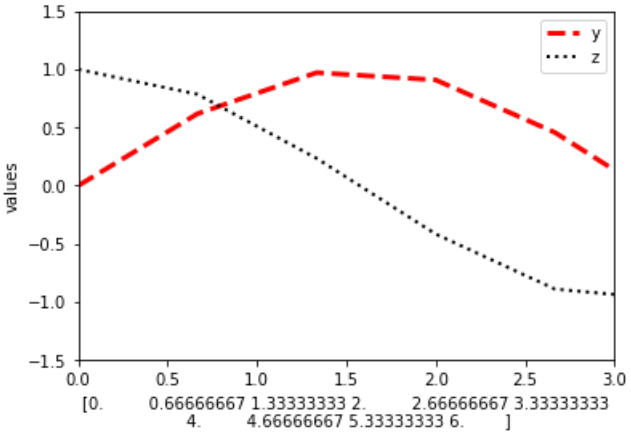
Out[25]:

```
Text(0.5, 1.0, '$y_4, \\ sqrt(x)$')
```



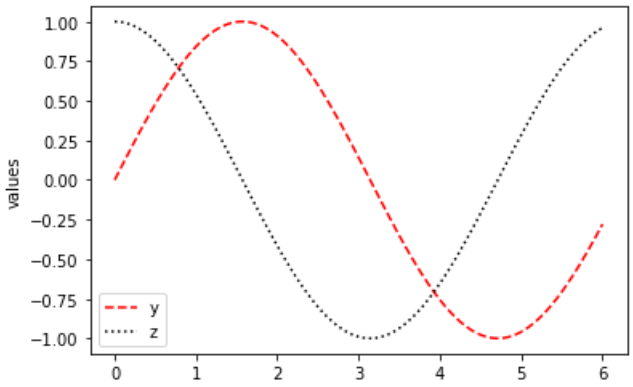
In [29]:

```
import  numpy as np
x= np.linspace(0,6,10)
y= np.sin(x)
z= np.cos(x)
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(x,y,'r--',linewidth=3)
plt.plot(x,z,'k:',linewidth=2)
plt.legend(['y','z'])
plt.xlabel(x)
plt.ylabel('values')
plt.xlim(0,3)
plt.ylim([-1.5,1.5])
plt.savefig('myFigure.png')
plt.savefig('myFigure.eps')
```



In [40]:

```
import numpy as np
import matplotlib.pyplot as plt
x= np.linspace(0,6,100)
y= np.sin(x)
z= np.cos(x)

plt.plot(x, y,'r--')
plt.plot(x, z,'k:')
plt.legend(['y','z'])
plt.xlabel(x)
plt.ylabel('values')
plt.show()
```

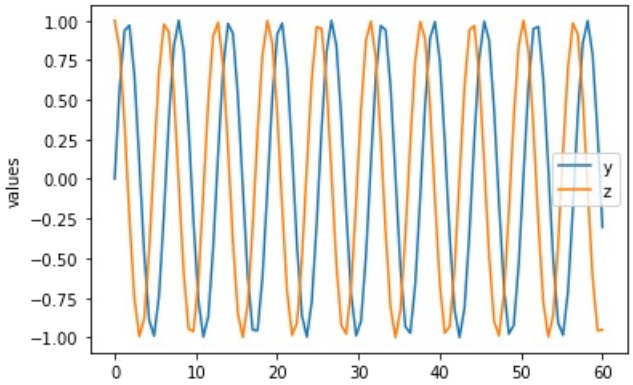



```
[0.         0.06060606 0.12121212 0.18181818 0.24242424 0.3030303
 0.36363636 0.42424242 0.48484848 0.54545455 0.60606061 0.66666667
 0.72727273 0.78787879 0.84848485 0.90909091 0.96969697 1.03030303
 1.09090909 1.15151515 1.21212121 1.27272727 1.33333333 1.39393939
 1.45454545 1.51515152 1.57575758 1.63636364 1.6969697  1.75757576
 1.81818182 1.87878788 1.93939394 2.         2.06060606 2.12121212
 2.18181818 2.24242424 2.3030303  2.36363636 2.42424242 2.48484848
 2.54545455 2.60606061 2.66666667 2.72727273 2.78787879 2.84848485
 2.90909091 2.96969697 3.03030303 3.09090909 3.15151515 3.21212121
 3.27272727 3.33333333 3.39393939 3.45454545 3.51515152 3.57575758
 3.63636364 3.6969697  3.75757576 3.81818182 3.87878788 3.93939394
 4.         4.06060606 4.12121212 4.18181818 4.24242424 4.3030303
 4.36363636 4.42424242 4.48484848 4.54545455 4.60606061 4.66666667
 4.72727273 4.78787879 4.84848485 4.90909091 4.96969697 5.03030303
 5.09090909 5.15151515 5.21212121 5.27272727 5.33333333 5.39393939
 5.45454545 5.51515152 5.57575758 5.63636364 5.6969697  5.75757576
 5.81818182 5.87878788 5.93939394 6.        ]
```

In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
x= np.linspace(0,60,100)
y= np.sin(x)
z= np.cos(x)

plt.plot(x, y)
plt.plot(x, z)
plt.legend(['y', 'z'])
plt.xlabel(x)
plt.ylabel('values')
plt.show()
```

```
C:\Users\ganesh chaurasiya\PycharmProjects\First_Project\venv\lib\site-
packages\matplotlib\text.py:1191: FutureWarning: elementwise comparison failed; returning scalar i
nstead, but in the future will perform elementwise comparison
  if s != self._text:
```




```
[ 0.         0.60606061  1.21212121  1.81818182  2.42424242  3.03030303
  3.63636364  4.24242424  4.84848485  5.45454545  6.06060606  6.66666667
  7.27272727  7.87878788  8.48484848  9.09090909  9.6969697  10.3030303
 10.90909091 11.51515152 12.12121212 12.72727273 13.33333333 13.93939394
 14.54545455 15.15151515 15.75757576 16.36363636 16.96969697 17.57575758
 18.18181818 18.78787879 19.39393939 20.         20.60606061 21.21212121
 21.81818182 22.42424242 23.03030303 23.63636364 24.24242424 24.84848485
 25.45454545 26.06060606 26.66666667 27.27272727 27.87878788 28.48484848
 29.09090909 29.6969697  30.3030303  30.90909091 31.51515152 32.12121212
 32.72727273 33.33333333 33.93939394 34.54545455 35.15151515 35.75757576
 36.36363636 36.96969697 37.57575758 38.18181818 38.78787879 39.39393939
 40.         40.60606061 41.21212121 41.81818182 42.42424242 43.03030303
 43.63636364 44.24242424 44.84848485 45.45454545 46.06060606 46.66666667
 47.27272727 47.87878788 48.48484848 49.09090909 49.6969697  50.3030303
 50.90909091 51.51515152 52.12121212 52.72727273 53.33333333 53.93939394
 54.54545455 55.15151515 55.75757576 56.36363636 56.96969697 57.57575758
 58.18181818 58.78787879 59.39393939 60.        ]
```

In [6]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

In [8]:

```
import numpy as np
from numpy import genfromtxt


file=genfromtxt(iris.data,delimiter=".",dtype="str")
print(file)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-8ebb8cb6fceb> in <module>
      3
      4
----> 5 file=genfromtxt(iris.data,delimiter=".",dtype="str")
      6 print(file)

NameError: name 'iris' is not defined
```

In [2]:

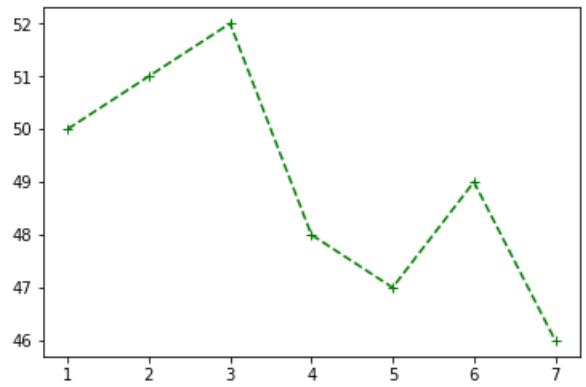
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:

```
x=[1,2,3,4,5,6,7]
y=[50,51,52,48,47,49,46]
plt.plot(x,y,'g+--') #+,--,**,'--r*'
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x10cc230>]
```

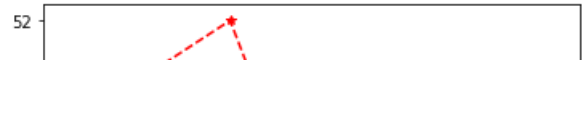


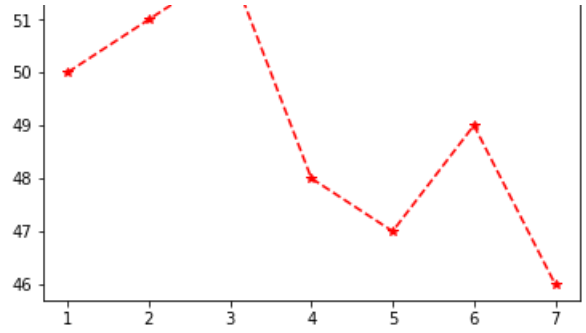
In [7]:

```
plt.plot(x,y,'--r*')
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x1ab2f90>]
```



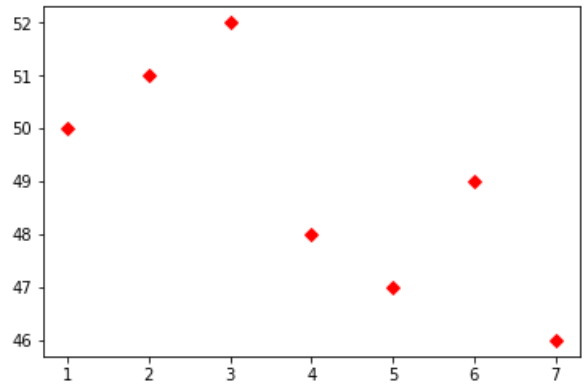


In [8]:

```
plt.plot(x,y,'rD')
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x1af2f90>]
```

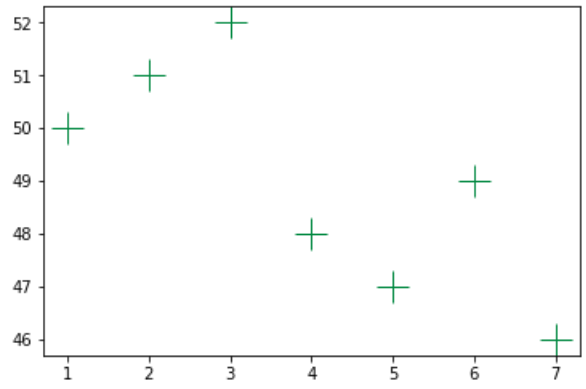


In [12]:

```
plt.plot(x,y,color='#008840',marker='+',linestyle='',markersize=20)
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x1bf6110>]
```

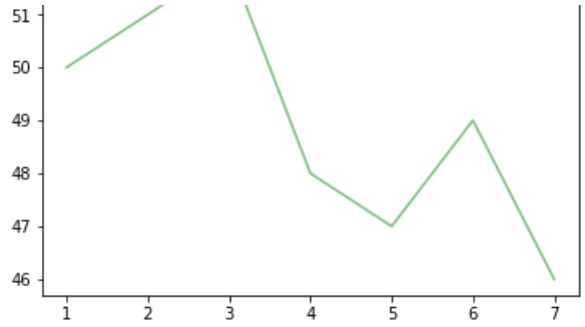


In [13]:

```
plt.plot(x,y,color='green',alpha=0.5)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x1c29ef0>]
```



In [14]:

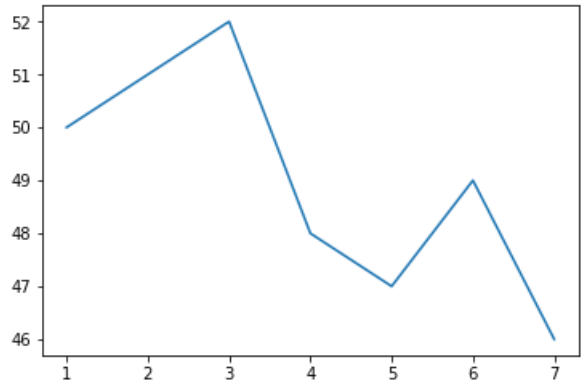
```
days=[1,2,3,4,5,6,7]
max_t=[50,51,52,48,47,49,46]
min_t=[43,42,40,44,33,35,37]
avg_t=[45,48,48,46,40,42,41]
```

In [15]:

```
plt.plot(days,max_t)
```

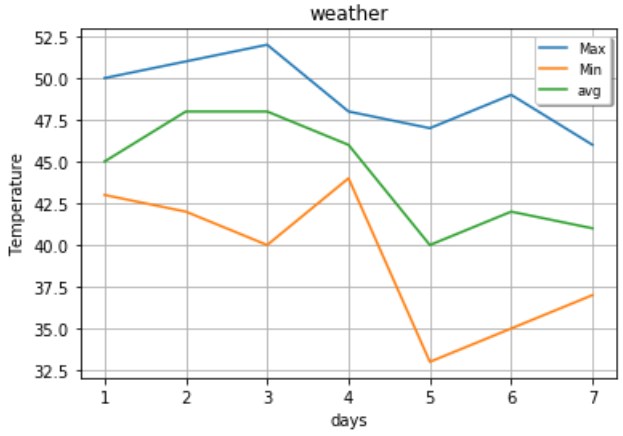
Out[15]:

```
[<matplotlib.lines.Line2D at 0x1c6c190>]
```



In [26]:

```
plt.xlabel("days")
plt.ylabel("Temperature")
plt.title("weather")
plt.plot(days,max_t,label="Max")
plt.plot(days,min_t,label="Min")
plt.plot(days,avg_t,label="avg")
plt.legend(loc="best",shadow=True,fontsize='small')#"upper right"
plt.grid()
```

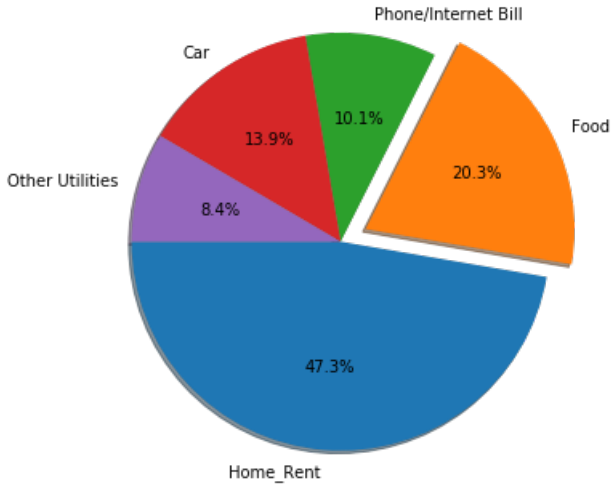


In [27]:

```
exp_vals=[1400,600,300,410,250]
exp_labels=["Home_Rent","Food","Phone/Internet Bill","Car","Other Utilities"]
```

In [44]:

```
plt.pie(exp_vals,labels=exp_labels,radius=1.5 ,autopct="%0.1f%%",shadow=True,explode=[0,0.2,0,0,0],
startangle=180)
#plt.savefig(piechart.png,bbox_inches="tight",pad_inches=2)
plt.show()
```



In [10]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```