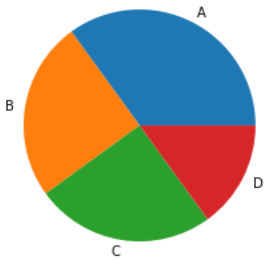
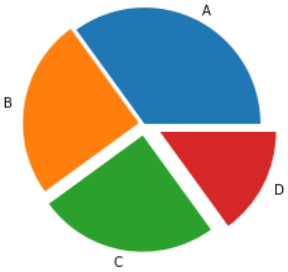
In [1]:
```
# a pie chart is a circle
# that is divided radially depending on the data. Imagine an apple pie or a pizza cut into
# slices. A pie chart fits that description well; however, unlike pizza or pies, which are
# usually divided symmetrically, a pie chart is not necessarily radially symmetrical. It all
# depends on the data to be visualized.

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
# Let's create the data to be visualized, as follows:
data = np.array([35, 25, 25, 15])
# Let's visualize the data with a simple pie chart as follows:
plt.pie(data)
plt.show()
```

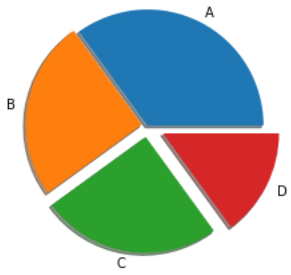
In [2]:
```
# Let's add labels as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
data = np.array([35, 25, 25, 15])
mylabels = ['A', 'B', 'C', 'D']
plt.pie(data,labels = mylabels)
plt.show()
```



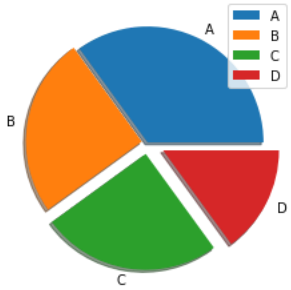
In [3]:
```
# You can even separate the parts of the pie a bit, as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
data = np.array([35, 25, 25, 15])
mylabels = ['A', 'B', 'C', 'D']
explode = [0.0, 0.05, 0.1, 0.15]
plt.pie(data,labels = mylabels,explode = explode)
plt.show()
```



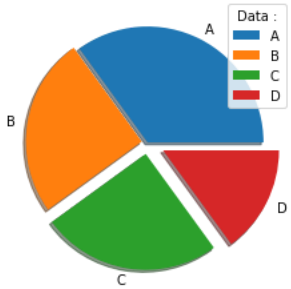
In [4]:
```
# You can also enable shadows as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
data = np.array([35, 25, 25, 15])
mylabels = ['A', 'B', 'C', 'D']
explode = [0.0, 0.05, 0.1, 0.15]
plt.pie(data,labels = mylabels,explode = explode,shadow = True)
plt.show()
```



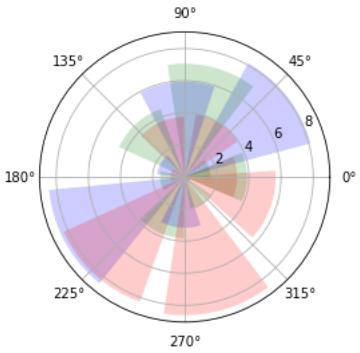
In [5]:
```
# You can also add a legend to the output as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
data = np.array([35, 25, 25, 15])
mylabels = ['A', 'B', 'C', 'D']
explode = [0.0, 0.05, 0.1, 0.15]
plt.pie(data,labels = mylabels,explode = explode,shadow = True)
plt.legend()
plt.show()
```



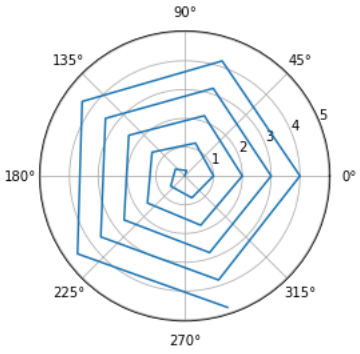
In [6]:
```
# You can add a title for the legend as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
data = np.array([35, 25, 25, 15])
mylabels = ['A', 'B', 'C', 'D']
explode = [0.0, 0.05, 0.1, 0.15]
plt.pie(data,labels = mylabels,explode = explode,shadow = True)
plt.legend(title='Data :')
plt.show()
```



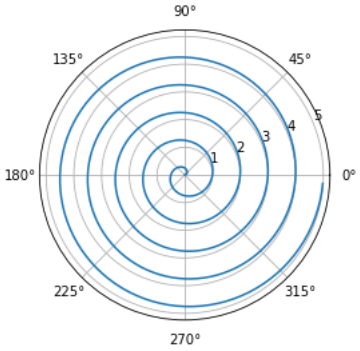
In [7]:
```
# Polar Charts
# You can also create polar graphs that are in the shape of pie charts. However, a
# fundamental difference from the Cartesian (X-Y) coordinate system is that in a polar
# chart the coordinate system is radially arranged, so you need the angle (theta) and
# distance from the origin (r is the radius) to visualize a point or set of points. Let's create a
# dataset as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
N = 20
theta = np.linspace(0.0, 2 * np.pi, N)
r = 10 * np.random.rand(N)
# The set of points can be visualized as follows:
plt.subplot(projection='polar')
plt.bar(theta, r, bottom=0.0,
color=['r', 'g', 'b'], alpha=0.2)
plt.show()
```



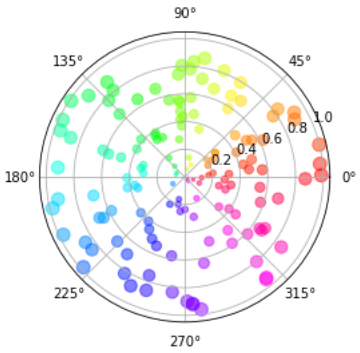
In [8]:
```
# Let's create a simple graph. Let's create the dataset for it as shown here:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
r = np.arange(0, 5, 0.2)
theta = 2 * np.pi * r
plt.subplot(projection='polar')
plt.plot(theta, r)
plt.show()

# This creates a simple linear visualization on a polar graph. As this is a polar graph,
# you will see a spiral-like structure
```



In [9]:
```
# This is not a perfect spiral as the distance between the consecutive points is 0.2. If
# you reduce the distance, then you will get a perfect spiral. Let's tweak the data as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
r = np.arange(0, 5, 0.01)
theta = 2 * np.pi * r
plt.subplot(projection='polar')
plt.plot(theta, r)
plt.show()
```



In [10]:
```
# Let's see a couple of examples of scatter plots on a polar graph. To start, prepare the
# data as shown here:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
N = 150
r = np.random.rand(N)
theta = 2 * np.pi * np.random.rand(N)
size = r * 100
# You can visualize this as follows:
plt.subplot(projection='polar')
plt.scatter(theta, r, c=theta,s=size, cmap='hsv',alpha=0.5)
plt.show()
```



In [11]:

```
# You can also show part of the visualization by setting the start and end angles, as
# follows:

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot(projection='polar')
c = ax.scatter(theta, r, c=theta, s=size, cmap='hsv', alpha=0.5)
ax.set_thetamin(0)
ax.set_thetamax(90)
plt.show()
```