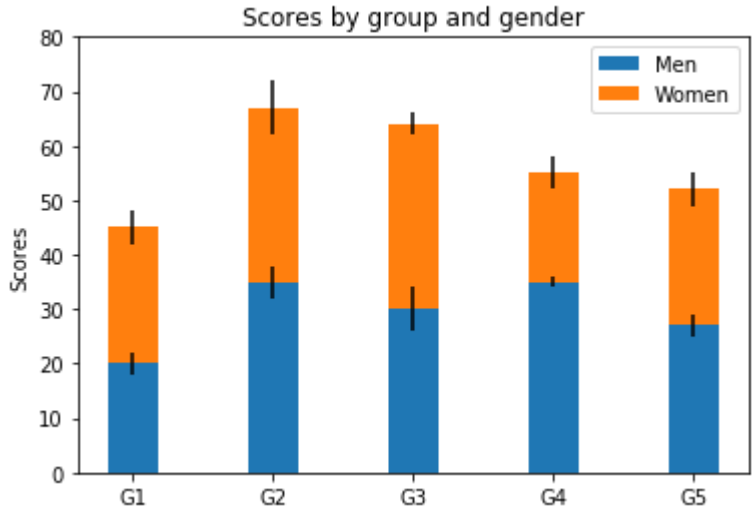
Chapter 5

Using Jupyter Notebook

Working with styles

Loading Stacked Bar Graph

In [3]:
```
# %load https://matplotlib.org/_downloads/a35b828c62f6d56bd84f2ca8d6670a5e/bar_stacked.py
# This command did not load the graph but only the code, I had to re-run for it to display the graph
"""
=================
Stacked Bar Graph
=================

This is an example of creating a stacked bar plot with error bars
using `~matplotlib.pyplot.bar`.  Note the parameters *yerr* used for
error bars, and *bottom* to stack the women's bars on top of the men's
bars.

"""

import numpy as np
import matplotlib.pyplot as plt


N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N)    # the x locations for the groups
width = 0.35       # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind, womenMeans, width,
             bottom=menMeans, yerr=womenStd)

plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```



Displaying an image from the web

In [4]:
```
# Importing image in jpg format
from IPython.display import Image
Embed = Image('https://si.wsj.net/public/resources/images/B3-DM077_RIGHTS_M_20190319163907.jpg')
Embed
```

Out[4]:

In [5]:
```
# Importing image in pdf format
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
from IPython.display import Image
Embed = Image('https://si.wsj.net/public/resources/images/B3-DM077_RIGHTS_M_20190319163907.jpg')
Embed
```

Out[5]:

Chapter 6

Uploading, Streaming, and Sampling Data

Uploading small amounts of data into memory

In [1]:
```
with open("Colors.txt", 'r') as open_file:
    print('Colors.txt content:\n' + open_file.read())
```

```
Colors.txt content:
Color   Value
Red     1
Orange  2
Yellow  3
Green   4
Blue    5
Purple  6
Black   7
White   8
```

Streaming large amounts of data into memory

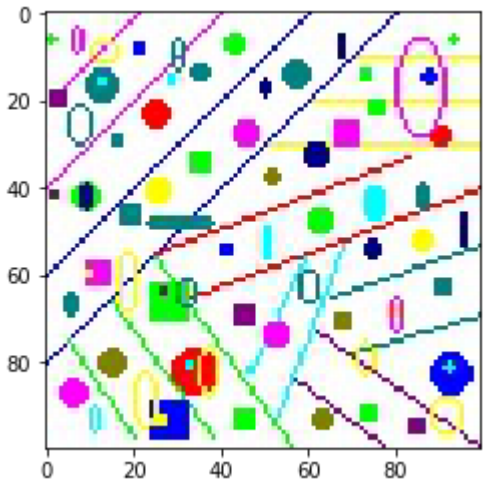
In [2]:
```
with open("Colors.txt", 'r') as open_file:
    for observation in open_file:
        print('Reading Data: ' + observation)
```

```
Reading Data: Color     Value

Reading Data: Red       1

Reading Data: Orange    2

Reading Data: Yellow    3

Reading Data: Green     4

Reading Data: Blue      5

Reading Data: Purple    6

Reading Data: Black     7

Reading Data: White     8
```

Generating variations on image data

In [3]:
```
import matplotlib.pyplot as plt
import matplotlib.image as img
%matplotlib inline

image = img.imread("Colorblk.jpg")
print(image.shape)
print(image.size)
plt.imshow(image)
plt.show()
```

```
(100, 100, 3)
30000
```



Sampling data in different ways

In [2]:
```
n = 2
with open("Colors.txt", 'r') as open_file:
    for j, observation in enumerate(open_file):
        if j % n==0:
            print('Reading Line: ' + str(j) +
            ' Content: ' + observation)
```

```
Reading Line: 0 Content: Color  Value

Reading Line: 2 Content: Orange 2

Reading Line: 4 Content: Green  4

Reading Line: 6 Content: Purple 6

Reading Line: 8 Content: White  8
```

In [12]:
```
from random import random
sample_size = 0.25
with open("Colors.txt", 'r') as open_file:
    for j, observation in enumerate(open_file):
        if random()<=sample_size:
            print('Reading Line: ' + str(j) +
            ' Content: ' + observation)
```

```
Reading Line: 1 Content: Red    1

Reading Line: 7 Content: Black  7

Reading Line: 8 Content: White  8
```

Accessing Data in Structured Flat-File Form

Reading from a text file

In [6]:
```
import pandas as pd
color_table = pd.io.parsers.read_table("Colors.txt")
print(color_table)
```

```
    Color  Value
0     Red      1
1  Orange      2
2  Yellow      3
3   Green      4
4    Blue      5
5  Purple      6
6   Black      7
7   White      8
```

Reading CSV delimited format

In [7]:
```
import pandas as pd
titanic = pd.io.parsers.read_csv("Titanic.csv")
X = titanic[['age']]
print(X)
```

	age
0	29.0000
1	0.9167
2	2.0000
3	30.0000
4	25.0000
5	48.0000
6	63.0000
7	39.0000
8	53.0000
9	71.0000
10	47.0000
11	18.0000
12	24.0000
13	26.0000
14	80.0000
15	9999.0000
16	24.0000
17	50.0000
18	32.0000
19	36.0000
20	37.0000
21	47.0000
22	26.0000
23	42.0000
24	29.0000
25	25.0000
26	25.0000
27	19.0000
28	35.0000
29	28.0000
...	...
1279	14.0000
1280	22.0000
1281	22.0000
1282	9999.0000
1283	9999.0000
1284	9999.0000
1285	32.5000
1286	38.0000
1287	51.0000
1288	18.0000
1289	21.0000
1290	47.0000
1291	9999.0000
1292	9999.0000
1293	9999.0000
1294	28.5000
1295	21.0000
1296	27.0000
1297	9999.0000
1298	36.0000
1299	27.0000
1300	15.0000
1301	45.5000
1302	9999.0000
1303	9999.0000

```
1304    14.5000
1305  9999.0000
1306    26.5000
1307    27.0000
1308    29.0000

[1309 rows x 1 columns]
```

In [8]:
```
X = titanic[['age']].values
print(X)
```

```
[[29.        ]
 [ 0.91670001]
 [ 2.        ]
 ...
 [26.5       ]
 [27.        ]
 [29.        ]]
```

Reading Excel and other Microsoft Office files

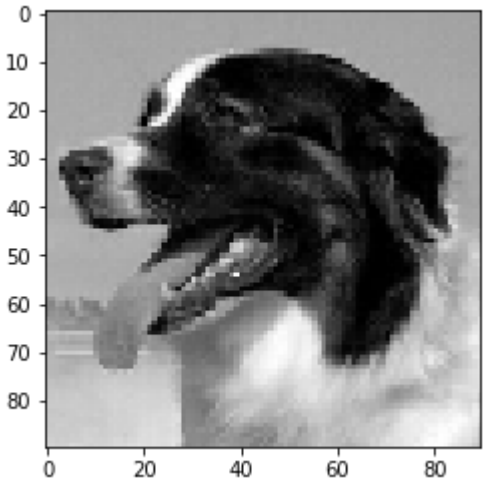
In [9]:
```
import pandas as pd
xls = pd.ExcelFile("Values.xls")
trig_values = xls.parse('Sheet1', index_col=None,
                        na_values=['NA'])
print(trig_values)
```

	Angle (Degrees)	Sine	Cosine	Tangent
0	138.550574	0.661959	-0.749540	-0.883153
1	305.535745	-0.813753	0.581211	-1.400100
2	280.518695	-0.983195	0.182556	-5.385709
3	216.363795	-0.592910	-0.805269	0.736289
4	36.389247	0.593268	0.805005	0.736974
5	31.474311	0.522116	0.852874	0.612184
6	120.121669	0.864962	-0.501838	-1.723588
7	293.947055	-0.913921	0.405892	-2.251634
8	179.882632	0.002048	-0.999998	-0.002048
9	120.927562	0.857818	-0.513954	-1.669056
10	71.349485	0.947487	0.319795	2.962796
11	241.971082	-0.882711	-0.469917	1.878439
12	297.208817	-0.889346	0.457235	-1.945053
13	142.004551	0.615599	-0.788060	-0.781158
14	173.770696	0.108508	-0.994096	-0.109152
15	229.232002	-0.757360	-0.652998	1.159820
16	67.926976	0.926706	0.375788	2.466033
17	261.866575	-0.989941	-0.141479	6.997102
18	59.185450	0.858830	0.512261	1.676547
19	98.029275	0.990197	-0.139679	-7.089086
20	9.336088	0.162225	0.986754	0.164403
21	90.746371	0.999915	-0.013026	-76.761483
22	217.798087	-0.612881	-0.790175	0.775626
23	58.616049	0.853697	0.520771	1.639295
24	197.196367	-0.295647	-0.955297	0.309482
25	331.194892	-0.481832	0.876264	-0.549871
26	6.509875	0.113374	0.993552	0.114110
27	266.390707	-0.998017	-0.062952	15.853513
28	230.323819	-0.769665	-0.638448	1.205525
29	314.224257	-0.716615	0.697469	-1.027452
..	...	...	...	...
42	324.080564	-0.586647	0.809843	-0.724396
43	140.893727	0.630761	-0.775977	-0.812860
44	128.226889	0.785567	-0.618777	-1.269547
45	82.770054	0.992049	0.125852	7.882680
46	303.112455	-0.837600	0.546284	-1.533268
47	164.824273	0.261780	-0.965127	-0.271239
48	218.829827	-0.627009	-0.779012	0.804878
49	28.649593	0.479452	0.877568	0.546341
50	349.336296	-0.185044	0.982730	-0.188296
51	84.889713	0.996025	0.089073	11.182105
52	197.935862	-0.307952	-0.951402	0.323683
53	303.049380	-0.838201	0.545362	-1.536963
54	183.737235	-0.065181	-0.997873	0.065320
55	346.153919	-0.239314	0.970942	-0.246477
56	218.822745	-0.626913	-0.779089	0.804674
57	243.969070	-0.898557	-0.438856	2.047498
58	115.600771	0.901827	-0.432098	-2.087089
59	125.906606	0.809974	-0.586466	-1.381111
60	200.094363	-0.343567	-0.939128	0.365836
61	337.860807	-0.376858	0.926271	-0.406855
62	168.176975	0.204889	-0.978785	-0.209330
63	305.708155	-0.812000	0.583657	-1.391229
64	162.656078	0.298107	-0.954533	-0.312306
65	219.007899	-0.629428	-0.777059	0.810012
66	222.830465	-0.679831	-0.733368	0.926998

```
67       324.199562 -0.584964  0.811059  -0.721234
68       187.948172 -0.138277 -0.990394   0.139619
69       270.678249 -0.999930  0.011837 -84.472139
70       270.779159 -0.999908  0.013598 -73.530885
71       200.213513 -0.345520 -0.938412   0.368196

[72 rows x 4 columns]
```

Sending Data in Unstructured File Form

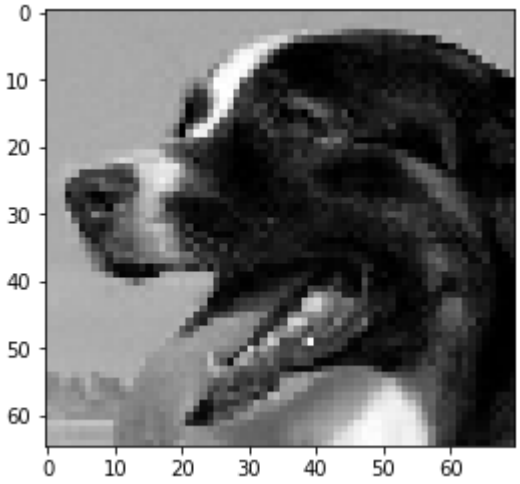
In [10]:
```
from skimage.io import imread
from skimage.transform import resize
from matplotlib import pyplot as plt
import matplotlib.cm as cm

example_file = ("http://upload.wikimedia.org/" +
    "wikipedia/commons/7/7d/Dog_face.png")
image = imread(example_file, as_grey=True)
plt.imshow(image, cmap=cm.gray)
plt.show()
```



In [11]:
```
print("data type: %s, shape: %s" %
      (type(image), image.shape))
```

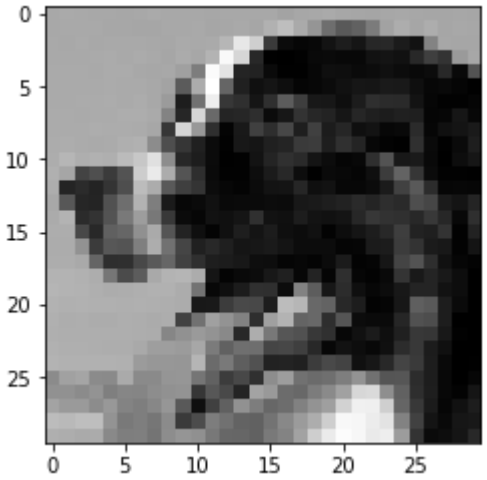
```
data type: <class 'numpy.ndarray'>, shape: (90, 90)
```

In [12]:
```
image2 = image[5:70,0:70]
plt.imshow(image2, cmap=cm.gray)
plt.show()
```



In [13]:
```
image3 = resize(image2, (30, 30), mode='symmetric')
plt.imshow(image3, cmap=cm.gray)
print("data type: %s, shape: %s" %
      (type(image3), image3.shape))
```

```
data type: <class 'numpy.ndarray'>, shape: (30, 30)
```



In [14]:
```
image_row = image3.flatten()
print("data type: %s, shape: %s" %
      (type(image_row), image_row.shape))
```

```
data type: <class 'numpy.ndarray'>, shape: (900,)
```

Managing Data from Relational Databases

In [15]:
```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')
```

Accessing Data from the Web

In [16]:
```
from lxml import objectify
import pandas as pd

xml = objectify.parse(open('XMLData.xml'))
root = xml.getroot()

df = pd.DataFrame(columns=('Number', 'String', 'Boolean'))

for i in range(0,4):
    obj = root.getchildren()[i].getchildren()
    row = dict(zip(['Number', 'String', 'Boolean'],
                   [obj[0].text, obj[1].text,
                    obj[2].text]))
    row_s = pd.Series(row)
    row_s.name = i
    df = df.append(row_s)

print(df)
```

```
  Number  String Boolean
0      1   First    True
1      2  Second   False
2      3   Third    True
3      4  Fourth   False
```

In [ ]:
```

```