In [1]:

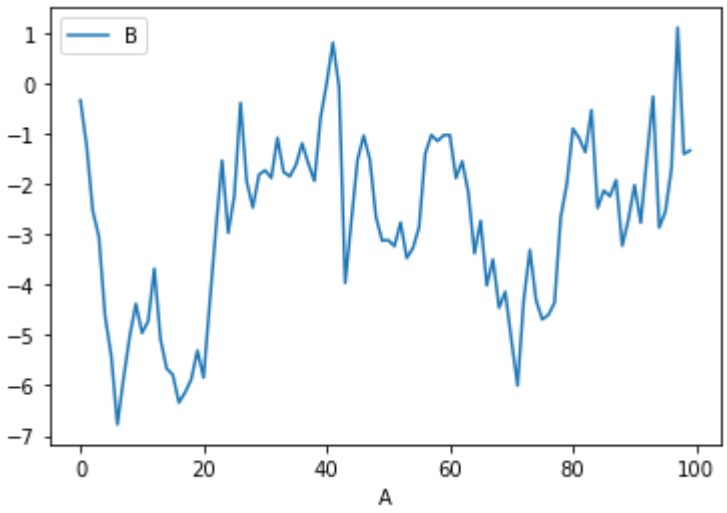
```
# Visualizing Data with Pandas and Matplotlib
# Simple Plots

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
# Let's create some data using the routine cumsum(), as follows:
df1 = pd.DataFrame(np.random.randn(100, 2), columns=['B', 'C']).cumsum()
df1['A'] = pd.Series(list(range(100)))
print(df1)
# Let's use the routine plot() to visualize this data. The plot() routine
# that the dataframe object uses calls Pyplot's plot() by default.
# Here's an example:
plt.figure()
df1.plot(x='A', y='B')
plt.show()
# This code is self-explanatory. We are passing strings that contain the
# names of columns as arguments for the x- and y-axes.
```

```
           B         C   A
0  -0.343834  1.116641   0
1  -1.239760  3.161280   1
2  -2.533521  4.253454   2
3  -3.054156  4.303779   3
4  -4.625432  4.701870   4
..       ...       ...  ..
95 -2.553328 -3.590973  95
96 -1.687302 -3.748553  96
97  1.098492 -4.742418  97
98 -1.407053 -2.953290  98
99 -1.340495 -3.843649  99

[100 rows x 3 columns]

<Figure size 432x288 with 0 Axes>
```



In [2]:

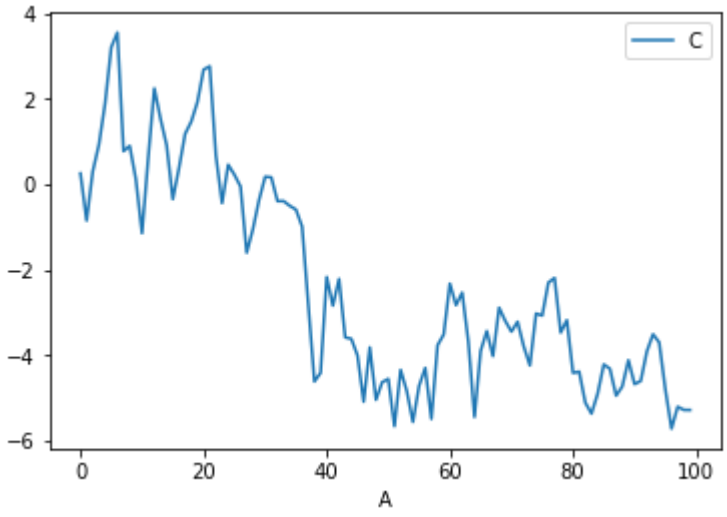
```
# You can use other columns in the visualization as well, as shown here:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df1 = pd.DataFrame(np.random.randn(100, 2), columns=['B', 'C']).cumsum()
df1['A'] = pd.Series(list(range(100)))
print(df1)
plt.figure()
df1.plot(x='A', y='C')
plt.show()
```

```
           B         C   A
0   1.154255  0.258743   0
1  -0.747112 -0.845771   1
2   0.111488  0.306322   2
3  -0.559711  0.933397   3
4   0.198317  1.897596   4
..       ...       ...  ..
95  4.039963 -4.805775  95
96  4.836080 -5.721534  96
97  5.933155 -5.208529  97
98  7.264756 -5.282744  98
99  7.508899 -5.284827  99

[100 rows x 3 columns]

<Figure size 432x288 with 0 Axes>
```



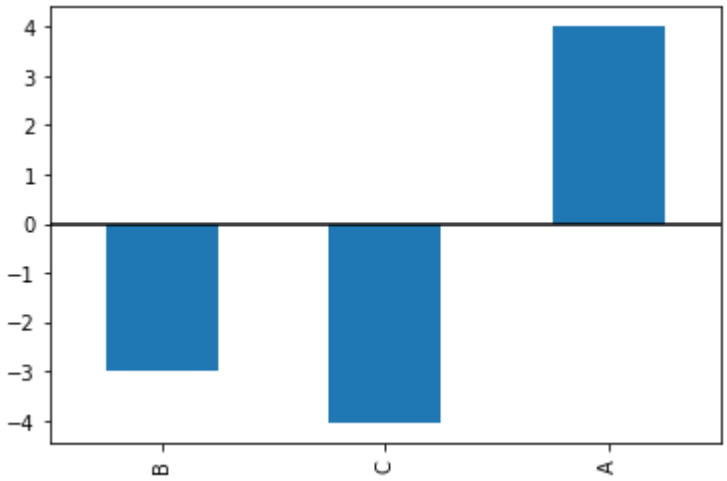
In [3]:

```
# Bar Graphs
# Let's create a simple bar graph using the same dataset.

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df1 = pd.DataFrame(np.random.randn(100, 2), columns=['B', 'C']).cumsum()
df1['A'] = pd.Series(list(range(100)))
# Let's pick a record from this dataframe as follows:
print(df1.iloc[4])
# Let's draw a simple bar graph with this data using the routine bar().
# The following is the code snippet for that:
plt.figure()
df1.iloc[4].plot.bar()
plt.axhline(0, color='k')
plt.show()

# In this code example, we are using axhline() to draw a horizontal line
# corresponding to the x-axis.
```

```
B   -2.998988
C   -4.050124
A    4.000000
Name: 4, dtype: float64
```



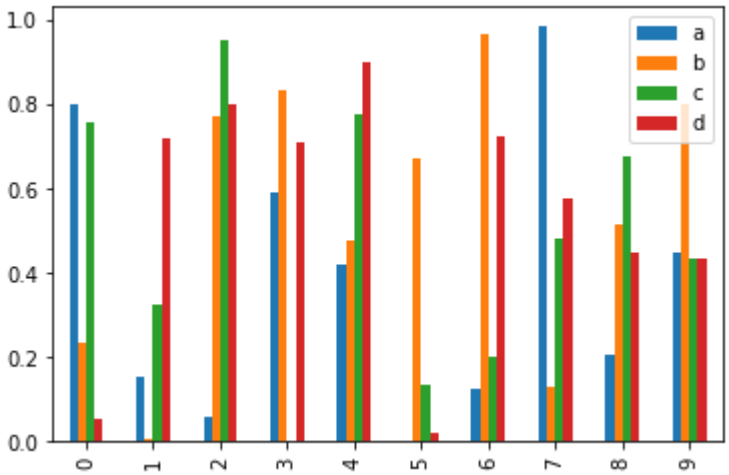
In [5]:

```
# Let's discuss a more complex example of a bar graph. Let's create a new
# dataset as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
print(df2)
# Now, let's visualize the entire dataset as follows:
plt.figure()
df2.plot.bar()
plt.show()
# This will create a bar graph for every row. The graphs will be grouped
# together per the rows.
```

```
          a         b         c         d
0  0.803283  0.236697  0.758849  0.055012
1  0.155658  0.005403  0.325910  0.720494
2  0.059109  0.773092  0.952274  0.801348
3  0.591882  0.836661  0.002564  0.712690
4  0.421494  0.477274  0.776651  0.900227
5  0.002908  0.671110  0.133720  0.020192
6  0.127053  0.969007  0.202270  0.724745
7  0.985087  0.129888  0.480118  0.578185
8  0.204675  0.514762  0.678688  0.451210
9  0.449377  0.803311  0.435828  0.434678
```

```
<Figure size 432x288 with 0 Axes>
```



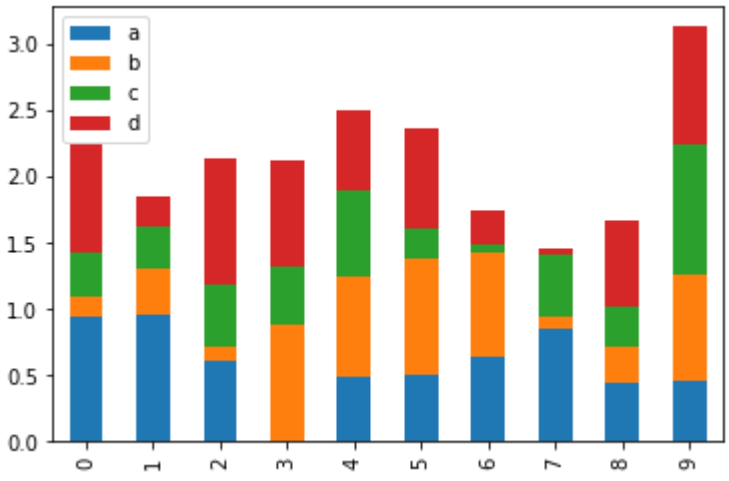
In [6]:

```
# You can see that the indices are represented on the x-axis,
# and magnitudes are marked on the y-axis. This is an unstacked
# vertical bar graph. You can create a stacked variation of it
# by just passing a simple argument as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
print(df2)
plt.figure()
df2.plot.bar(stacked=True)
plt.show()
```

```
          a         b         c         d
0  0.951774  0.138338  0.342677  0.811679
1  0.960864  0.350043  0.320250  0.218014
2  0.615048  0.109400  0.459494  0.959403
3  0.010107  0.872328  0.447025  0.798200
4  0.497622  0.751238  0.655869  0.603599
5  0.499264  0.878784  0.233864  0.752272
6  0.638960  0.788175  0.062799  0.258957
7  0.855493  0.090900  0.468942  0.050155
8  0.442434  0.271596  0.312449  0.651405
9  0.457049  0.798533  0.996021  0.882069
```

```
<Figure size 432x288 with 0 Axes>
```



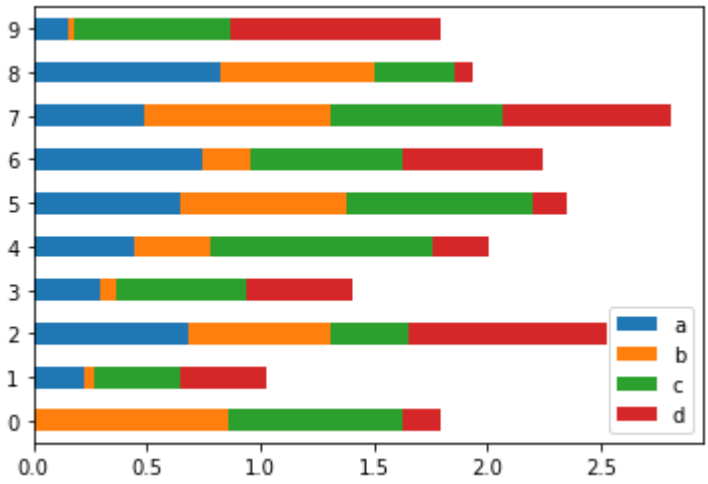
In [7]:

```
# You can even create horizontal stacked and unstacked bar graphs too.
# Let's create a horizontally stacked bar graph with the routine barh()
# as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
print(df2)
plt.figure()
df2.plot.barh(stacked=True)
plt.show()
```

```
          a         b         c         d
0  0.002605  0.859773  0.760981  0.173173
1  0.224693  0.044409  0.382080  0.373566
2  0.685992  0.621645  0.349791  0.874021
3  0.297326  0.067365  0.578103  0.467999
4  0.442011  0.335345  0.986444  0.239788
5  0.651208  0.732150  0.814012  0.151169
6  0.743637  0.212057  0.672336  0.613865
7  0.485146  0.823395  0.761804  0.743118
8  0.821507  0.679194  0.357014  0.079582
9  0.154776  0.028737  0.687861  0.926612
```

```
<Figure size 432x288 with 0 Axes>
```



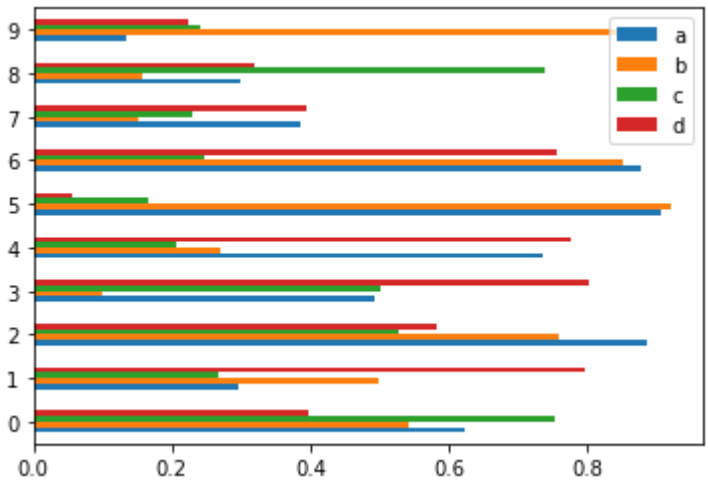
In [8]:

```
# Let's write a code snippet for an unstacked horizontal bar graph by
# omitting the argument as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
print(df2)
plt.figure()
df2.plot.barh()
plt.show()
```

```
          a         b         c         d
0  0.622859  0.542861  0.752348  0.398346
1  0.295247  0.498489  0.265913  0.797327
2  0.886048  0.756910  0.527109  0.582672
3  0.491097  0.100538  0.500892  0.802648
4  0.734677  0.269671  0.205731  0.774573
5  0.905181  0.921427  0.167253  0.055770
6  0.876379  0.852141  0.247298  0.755407
7  0.384604  0.150737  0.229648  0.395332
8  0.299561  0.157530  0.737198  0.319782
9  0.134746  0.874258  0.240163  0.222948
```

```
<Figure size 432x288 with 0 Axes>
```



In [9]:

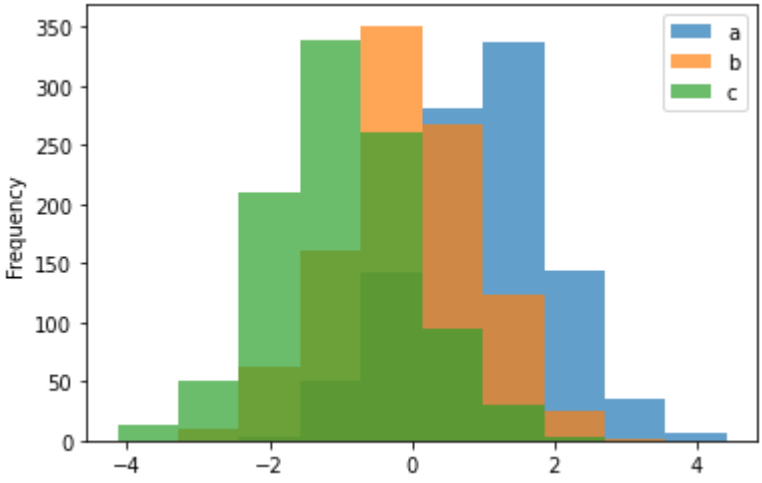
```
# Histograms
# A histogram is a visual representation of the frequency distribution of
# numerical data. It was first used by Karl Pearson.
# We first divide the data into various buckets, or bins. The size of the
# bins depends on the requirements. For integer datasets, you can have
# the smallest bin size, which is 1. Then for each bin, you can list the
# number of occurrences of elements that fall under the bin. Then you can
# show that table as a bar graph.
# You can draw the histogram of a given dataset with Pandas and Matplotlib.
# Let's create a dataset as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df4 = pd.DataFrame({'a': np.random.randn(1000) + 1,
                    'b': np.random.randn(1000),
                    'c': np.random.randn(1000) - 1},
                   columns=['a', 'b', 'c'])
print(df4)
# Let's visualize this dataset as a histogram using the routine hist(),
# as follows:
plt.figure();
df4.plot.hist(alpha=0.7)
plt.show()
# The argument passed to routine decides the opacity (or alpha
# transparency) of the output.
```

```
            a         b         c
0    1.340696  1.522237 -0.642025
1    1.214861 -1.242795 -0.926155
2    3.016867  0.451544 -0.199854
3    0.950689 -0.608912 -0.484600
4    1.268629 -0.850753 -0.530041
..        ...       ...       ...
995  1.207297 -0.202804 -0.528054
996  2.283427  0.056101 -0.031412
997 -0.700049  0.041651 -2.207479
998  2.228445 -0.030855 -1.452964
999  2.876967  0.099098 -2.241635

[1000 rows x 3 columns]


<Figure size 432x288 with 0 Axes>
```


a
b
c
Frequency
350
300
250
200
150
100
50
0
−4
−2
0
2
4

In [10]:

```
# You had to make this transparent in the previous example because the
# histogram was unstacked. Let's create a stacked histogram with the size
# of the buckets as 20, as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df4 = pd.DataFrame({'a': np.random.randn(1000) + 1,
                    'b': np.random.randn(1000),
                    'c': np.random.randn(1000) - 1},
                   columns=['a', 'b', 'c'])
print(df4)
plt.figure();
df4.plot.hist(stacked=True, bins=20)
plt.show()
```

```
            a         b         c
0    1.435559  1.197192 -0.080493
1   -0.839429  1.738201  0.866615
2    0.288004  1.095742  0.909759
3    0.561959  0.364604 -2.432308
4    0.547871 -0.178350  0.576917
..        ...       ...       ...
995  0.192788 -0.146870 -0.069354
996  1.880129 -1.343519 -0.617994
997 -0.250663  1.059606  0.541053
998  1.213552  1.116077 -1.209477
999 -1.037842 -1.711716 -0.741809

[1000 rows x 3 columns]

<Figure size 432x288 with 0 Axes>
```