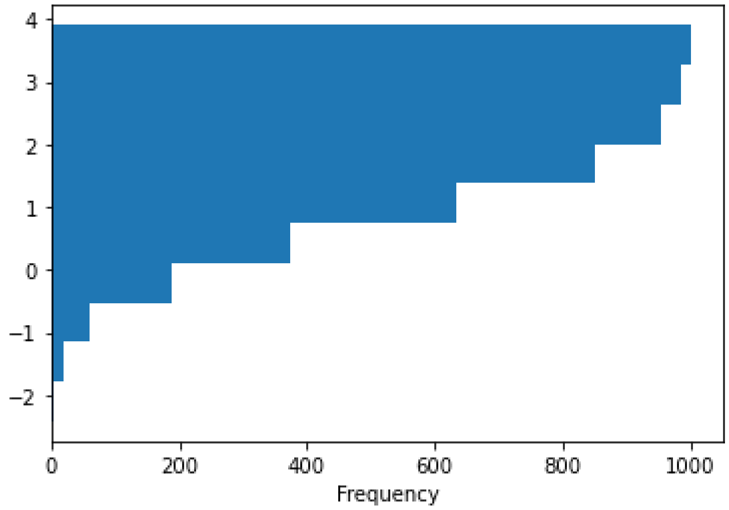
In [1]:

```
# Let’s create a horizontal cumulative histogram of a single column
# 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['a'].plot.hist(orientation='horizontal', cumulative=True)
plt.show()
```

```
            a         b         c
0    0.654367 -0.988079  0.234018
1   -0.794184 -1.070127 -0.459476
2    1.663773 -0.194070 -1.623879
3    0.983369 -0.112327 -0.135126
4    0.086411 -0.719514 -0.896352
..        ...       ...       ...
995  0.321312 -1.166780 -1.179806
996  1.425544  0.620037 -0.562061
997  2.603243  0.667413 -2.437526
998  0.763988 -1.204117 -1.787248
999  1.443270 -0.378097 -1.429902

[1000 rows x 3 columns]
```



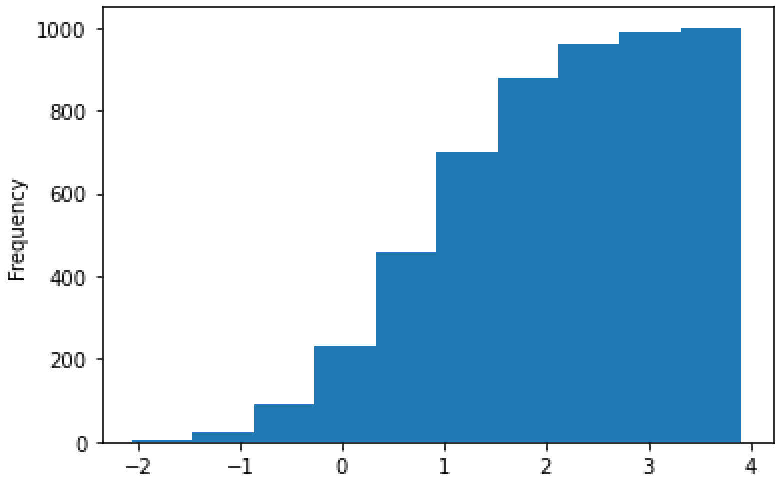
In [2]:

```
# The vertical version of the same histogram can be created 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['a'].plot.hist(orientation='vertical', cumulative=True)
plt.show()
```

```
            a         b         c
0    2.321655  1.720359 -1.304734
1    2.010915 -1.393351 -1.025259
2    0.416948 -0.224408  1.143051
3    1.471939  1.286214 -0.061149
4    0.670227  0.452863  0.140193
..        ...       ...       ...
995  1.052931  0.243746 -2.357155
996  0.562859 -1.547617 -1.027116
997 -0.350921 -0.456365 -1.924521
998  3.490968  1.389278 -2.795724
999  0.891592  0.067493 -1.227199

[1000 rows x 3 columns]
```



In [3]:

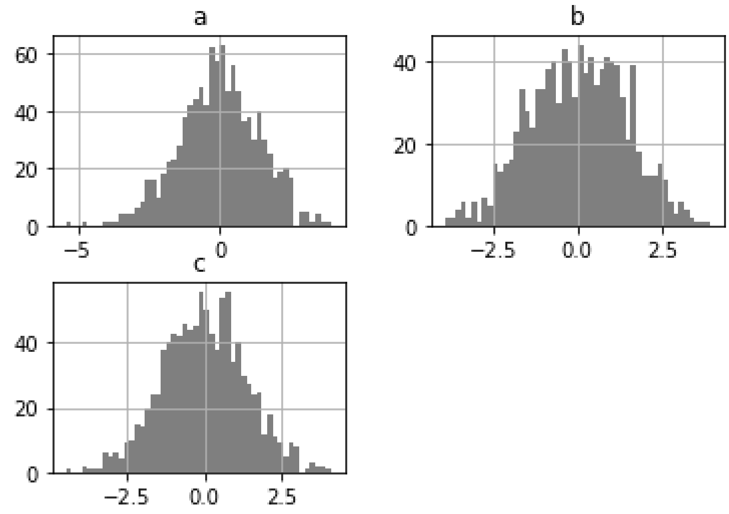
```
# Let’s try a fancy type of histogram next. The routine diff() computes
# the numeric difference between the previous row and the current one.

%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.diff())
# The output will have the first row populated with NaN for all the
# columns (as there is no row before the first one).
# Let’s visualize this dataset, as shown here:
plt.figure()
df4.diff().hist(color='k', alpha=0.5, bins=50)
plt.show()
```

```
            a         b         c
0         NaN       NaN       NaN
1    0.850791 -2.388798 -1.850190
2   -1.043945  0.581283 -0.345531
3    1.000397  0.993921  1.597623
4   -0.071265 -2.235158 -0.733533
..        ...       ...       ...
995 -0.415961  0.049106  2.881210
996 -1.342267 -1.671567 -1.066157
997  1.845456  1.670451 -0.234525
998 -1.252148 -1.596792 -0.974316
999  0.006464  1.581809  1.633959

[1000 rows x 3 columns]

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



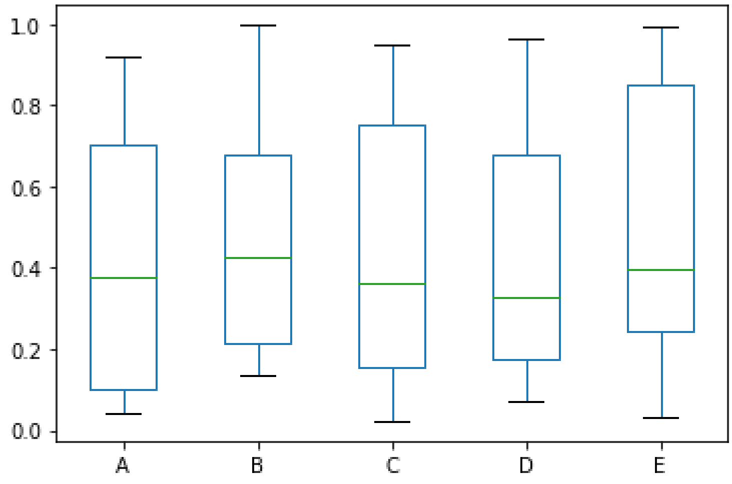
In [4]:

```
# Box Plots
# You can visualize data with box plots as well. Box plots (also spelled
# as boxplots) display the groups of numerical data through their
# quartiles. Let’s create a dataset as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 5),
columns=['A', 'B', 'C', 'D', 'E'])
print(df)
# You can draw box plots as follows:
plt.figure()
df.plot.box()
plt.show()
```

```
          A         B         C         D         E
0  0.920574  0.997594  0.022853  0.806425  0.693858
1  0.040341  0.386341  0.125689  0.199240  0.990741
2  0.795488  0.135734  0.127519  0.168986  0.300124
3  0.489832  0.211627  0.949353  0.761957  0.448372
4  0.773225  0.716052  0.673506  0.070806  0.227286
5  0.273066  0.899826  0.894771  0.962505  0.905162
6  0.332754  0.560277  0.272352  0.408202  0.341721
7  0.046867  0.164570  0.777496  0.248156  0.033872
8  0.419480  0.465745  0.232793  0.428947  0.992872
9  0.040042  0.226108  0.456585  0.093854  0.178119
```

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



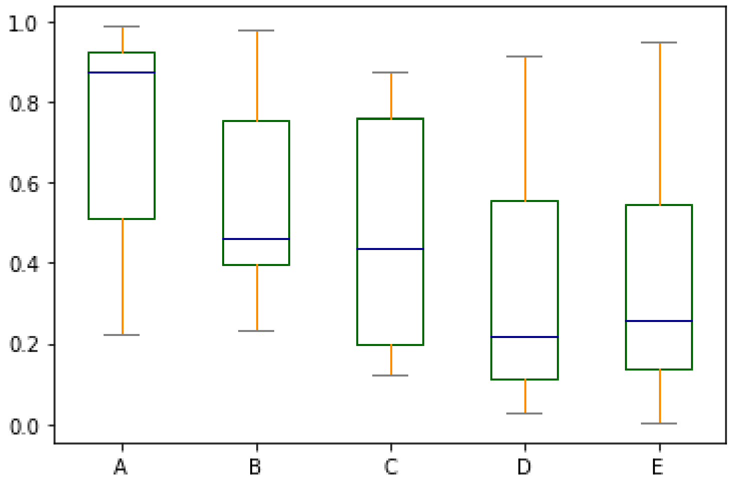
In [5]:

```
# The colors shown here are the default values. You can change them.
# First, you need to create a dictionary as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 5),
columns=['A', 'B', 'C', 'D', 'E'])
print(df)
color = dict(boxes='DarkGreen',
             whiskers='DarkOrange',
             medians='DarkBlue',
             caps='Gray')
print(color)
# Finally, you pass this dictionary as an argument to the routine that
# draws the box plot as follows:
plt.figure()
df.plot.box(color=color, sym='r+')
plt.show()
```

```
          A         B         C         D         E
0  0.223658  0.233932  0.264287  0.035402  0.458119
1  0.892372  0.654849  0.797634  0.916385  0.843019
2  0.667002  0.400265  0.657021  0.110077  0.135222
3  0.856946  0.514980  0.124528  0.570686  0.274876
4  0.983592  0.300815  0.827511  0.126652  0.142149
5  0.307478  0.977545  0.215705  0.026781  0.110791
6  0.914759  0.395985  0.194905  0.186062  0.004509
7  0.928315  0.787442  0.605798  0.511607  0.950247
8  0.463146  0.969012  0.875902  0.247006  0.574153
9  0.988772  0.408929  0.160687  0.588077  0.245871
{'boxes': 'DarkGreen', 'whiskers': 'DarkOrange', 'medians': 'DarkBlue', 'c
aps': 'Gray'}

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



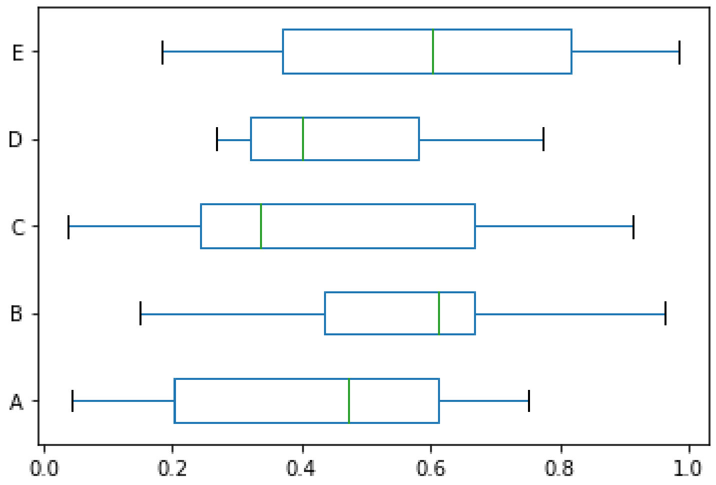
In [6]:

```
# The following example creates a horizontal box plot visualization:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 5),
columns=['A', 'B', 'C', 'D', 'E'])
print(df)
color = dict(boxes='DarkGreen',
             whiskers='DarkOrange',
             medians='DarkBlue',
             caps='Gray')
print(color)
plt.figure()
df.plot.box(vert=False, positions=[1, 2, 3, 4 , 5])
plt.show()
```

```
          A         B         C         D         E
0  0.486189  0.650490  0.237019  0.408413  0.343651
1  0.275932  0.398783  0.913969  0.268478  0.590865
2  0.620479  0.599305  0.335157  0.756712  0.234431
3  0.586924  0.624678  0.508036  0.305624  0.982391
4  0.750206  0.148362  0.258012  0.505351  0.452792
5  0.651112  0.674080  0.721865  0.391796  0.614639
6  0.455314  0.960619  0.718443  0.311652  0.856913
7  0.043087  0.746605  0.173118  0.606734  0.821728
8  0.159616  0.537974  0.332624  0.345142  0.803692
9  0.176294  0.174190  0.037875  0.771327  0.183741
{'boxes': 'DarkGreen', 'whiskers': 'DarkOrange', 'medians': 'DarkBlue', 'c
aps': 'Gray'}

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

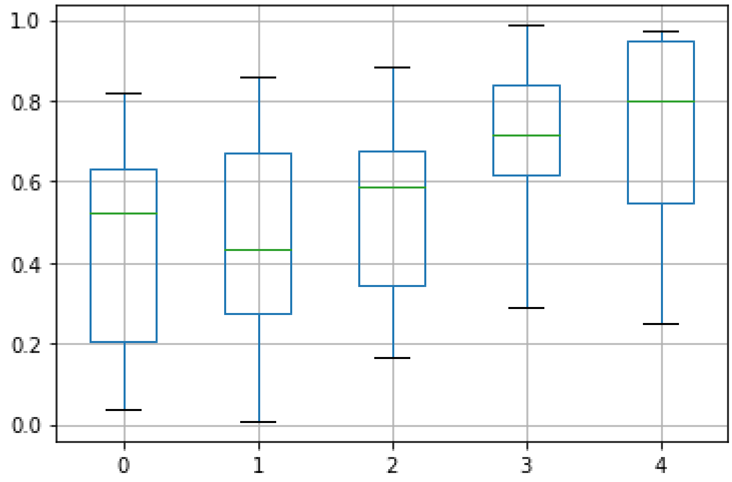


In [7]:

```
# Let’s see another routine, boxplot(), that also creates box plots.
# For that, let’s create another dataset, as shown here:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 5))
print(df)
# You can draw box plots as follows:
plt.figure()
bp = df.boxplot()
plt.show()
```

```
          0         1         2         3         4
0  0.497830  0.594182  0.549120  0.851455  0.970234
1  0.550177  0.857646  0.646986  0.987135  0.971928
2  0.634340  0.114582  0.277561  0.810268  0.953839
3  0.197895  0.508860  0.687032  0.647456  0.689291
4  0.618602  0.731949  0.600448  0.753961  0.677482
5  0.821528  0.008561  0.885188  0.347796  0.249466
6  0.640902  0.285266  0.756421  0.673382  0.331433
7  0.221678  0.275205  0.164943  0.292643  0.934913
8  0.095943  0.699239  0.575979  0.955796  0.506414
9  0.037579  0.362982  0.190272  0.605514  0.913407
```



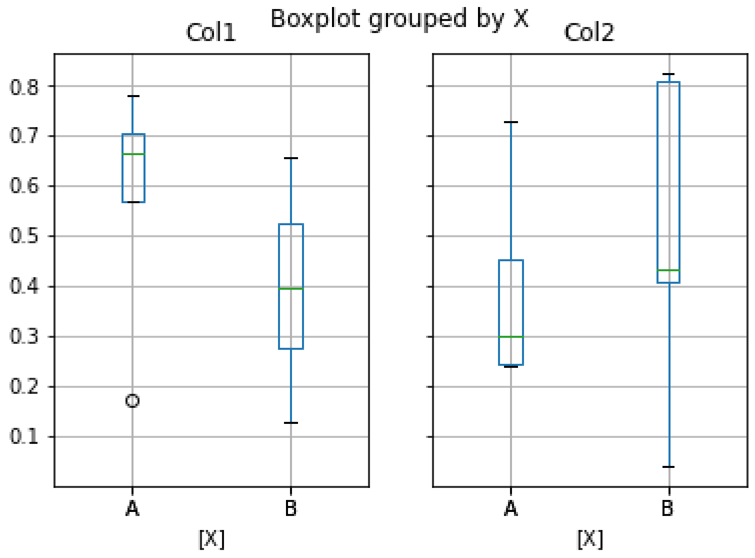
In [8]:

```
# The main advantage of the routine boxplot() is that you can have
# column-wise visualizations in a single output. Let’s create an
# appropriate dataset as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 2), columns=['Col1', 'Col2'] )
df['X'] = pd.Series(['A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B'])
print(df)
# Let’s create column-wise visualizations as follows:
plt.figure();
bp = df.boxplot(by='X')
plt.show()
```

```
       Col1      Col2  X
0  0.780273  0.239481  A
1  0.703489  0.727344  A
2  0.567063  0.453846  A
3  0.172151  0.242670  A
4  0.664194  0.297689  A
5  0.274967  0.807536  B
6  0.656785  0.408087  B
7  0.125485  0.825867  B
8  0.395293  0.039400  B
9  0.522969  0.430718  B


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



In [9]:

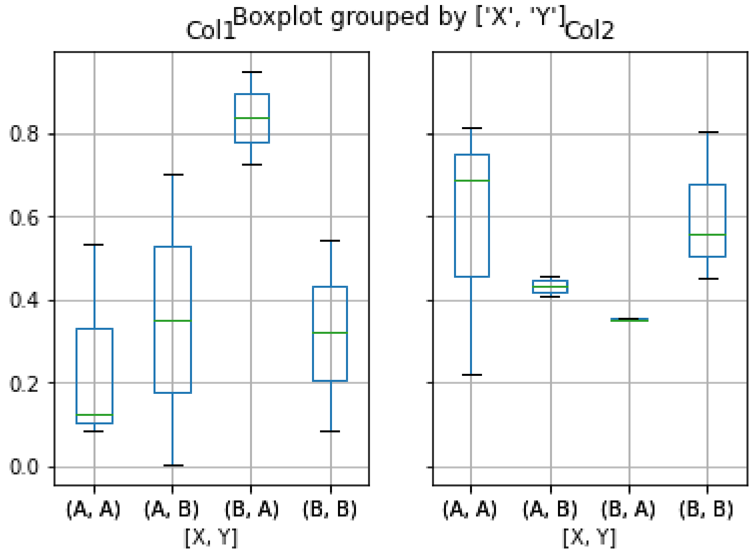
```
# Let’s look at a little more complex example for this. The following is
# the code for a new dataset:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10,3), columns=['Col1', 'Col2', 'Col3'])
df['X'] = pd.Series(['A','A','A','A','A','B','B','B','B','B'])
df['Y'] = pd.Series(['A','B','A','B','A','B','A','B','A','B'])
print(df)
# You can create box plots in groups of multiple columns (this means the
# grouping criteria will have multiple columns).
plt.figure();
bp = df.boxplot(column=['Col1','Col2'], by=['X','Y'])
plt.show()
```

```
       Col1      Col2      Col3  X  Y
0  0.534904  0.814291  0.160496  A  A
1  0.002619  0.407193  0.135888  A  B
2  0.124913  0.218666  0.983652  A  A
3  0.702234  0.457583  0.168760  A  B
4  0.084399  0.690018  0.313036  A  A
5  0.544395  0.556152  0.059888  B  B
6  0.724984  0.356256  0.802166  B  A
7  0.086336  0.806255  0.286251  B  B
8  0.951401  0.348329  0.003955  B  A
9  0.321984  0.451430  0.882893  B  B
```

```
c:\python\lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationW
arning: Creating an ndarray from ragged nested sequences (which is a list-
or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' wh
en creating the ndarray
  return array(a, dtype, copy=False, order=order)
c:\python\lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationW
arning: Creating an ndarray from ragged nested sequences (which is a list-
or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' wh
en creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

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


Boxplot grouped by ['X', 'Y']
Col1
Col2
0.0
0.2
0.4
0.6
0.8
(A, A)
(A, B)
(B, A)
(B, B)
[X, Y]
(A, A)
(A, B)
(B, A)
(B, B)
[X, Y]

In [10]:

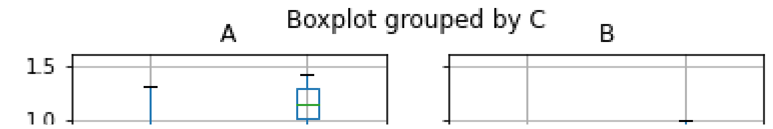
```
# Let’s see a bit more complex example with a dataset that has more
# variation. The following code creates such a dataset:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
np.random.seed(1234)
df_box = pd.DataFrame(np.random.randn(10, 2), columns=['A', 'B'])
df_box['C'] = np.random.choice(['Yes', 'No'], size=10)
print(df_box)
# You can use the routine groupby() in Pandas to group the data and
# visualize it as follows:
plt.figure()
bp = df_box.boxplot(by='C')
plt.show()
```

```
          A         B    C
0  0.471435 -1.190976   No
1  1.432707 -0.312652  Yes
2 -0.720589  0.887163   No
3  0.859588 -0.636524  Yes
4  0.015696 -2.242685   No
5  1.150036  0.991946  Yes
6  0.953324 -2.021255   No
7 -0.334077  0.002118   No
8  0.405453  0.289092   No
9  1.321158 -1.546906   No
```

```
c:\python\lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationW
arning: Creating an ndarray from ragged nested sequences (which is a list-
or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' wh
en creating the ndarray
  return array(a, dtype, copy=False, order=order)
c:\python\lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationW
arning: Creating an ndarray from ragged nested sequences (which is a list-
or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes)
is deprecated. If you meant to do this, you must specify 'dtype=object' wh
en creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

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



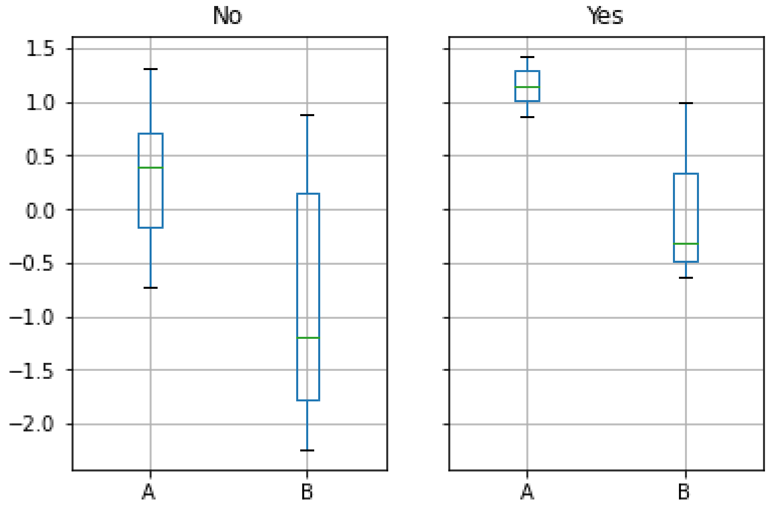
In [11]:

```
# Another example is as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
np.random.seed(1234)
df_box = pd.DataFrame(np.random.randn(10, 2), columns=['A', 'B'])
df_box['C'] = np.random.choice(['Yes', 'No'], size=10)
print(df_box)
# You can use the routine groupby() in Pandas to group the data and
# visualize it as follows:
plt.figure()
bp = df_box.groupby('C').boxplot()
plt.show()
```

```
          A         B    C
0  0.471435 -1.190976   No
1  1.432707 -0.312652  Yes
2 -0.720589  0.887163   No
3  0.859588 -0.636524  Yes
4  0.015696 -2.242685   No
5  1.150036  0.991946  Yes
6  0.953324 -2.021255   No
7 -0.334077  0.002118   No
8  0.405453  0.289092   No
9  1.321158 -1.546906   No

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


No
Yes
A
B
A
B