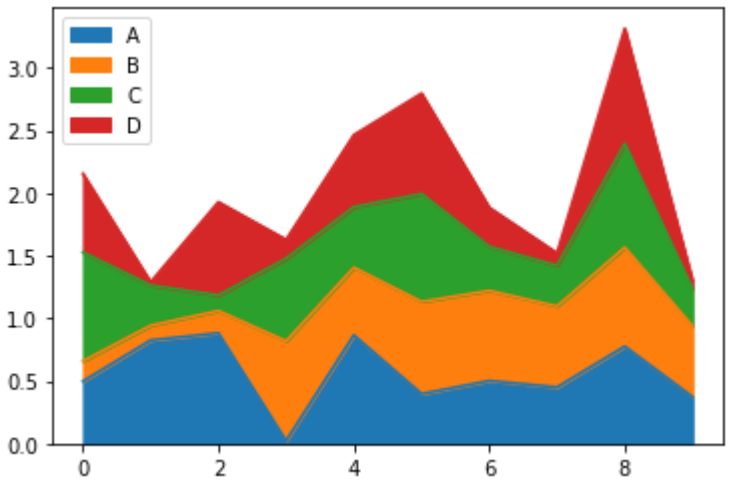
In [1]:

```
# Area Plots
# You can visualize datasets as area plots too. Let’s create a dataset
# with four columns as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
# You can visualize all this data with the routine area() as follows:
plt.figure()
df.plot.area()
plt.show()
```

```
          A         B         C         D
0  0.497164  0.161371  0.868631  0.627363
1  0.824578  0.117927  0.320920  0.025956
2  0.881508  0.178162  0.124437  0.744689
3  0.020615  0.797340  0.660531  0.149511
4  0.864806  0.540782  0.484061  0.574732
5  0.395934  0.736675  0.862174  0.802066
6  0.500430  0.719844  0.352272  0.311633
7  0.449386  0.646761  0.324385  0.100241
8  0.775341  0.790897  0.828344  0.920649
9  0.371931  0.565792  0.300402  0.065326
```

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



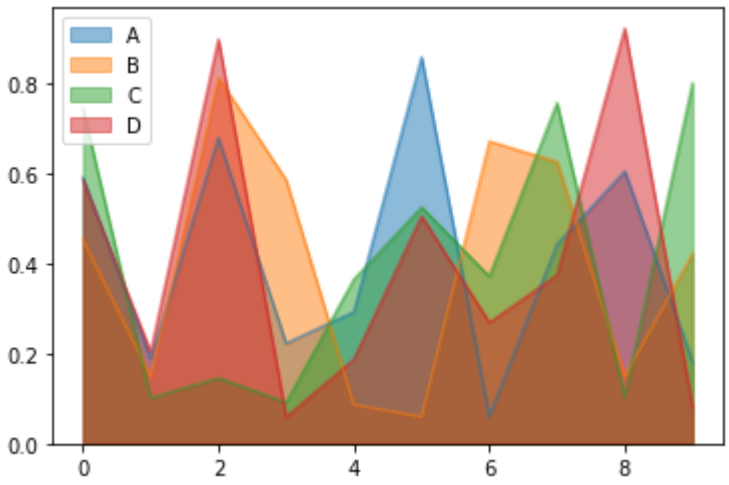
In [2]:

```
# You can also create unstacked area plots by passing an argument to the
# routine area() as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
plt.figure()
df.plot.area(stacked=False)
plt.show()

# The unstacked area plot will be transparent by default so that all the
# individual area plots are visible.
```

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



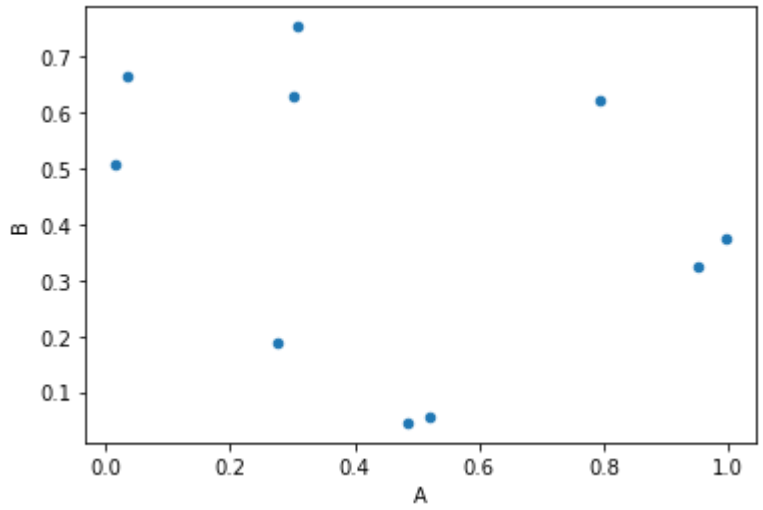
In [3]:

```
# Scatter Plots
# You can also visualize any dataset as a scatter plot. 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, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
# You can visualize columns A and B as a scatter plot as follows:
plt.figure()
df.plot.scatter(x='A', y='B')
plt.show()
```

```
          A         B         C         D
0  0.793298  0.623422  0.797495  0.307030
1  0.307865  0.754022  0.654228  0.099178
2  0.995341  0.375911  0.888086  0.461611
3  0.951597  0.326178  0.256081  0.697208
4  0.519383  0.056432  0.232233  0.947135
5  0.274390  0.188839  0.275490  0.768283
6  0.300123  0.628748  0.504645  0.083323
7  0.485107  0.044968  0.235495  0.150231
8  0.016880  0.508007  0.199427  0.373435
9  0.033854  0.666048  0.252803  0.046986
```

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

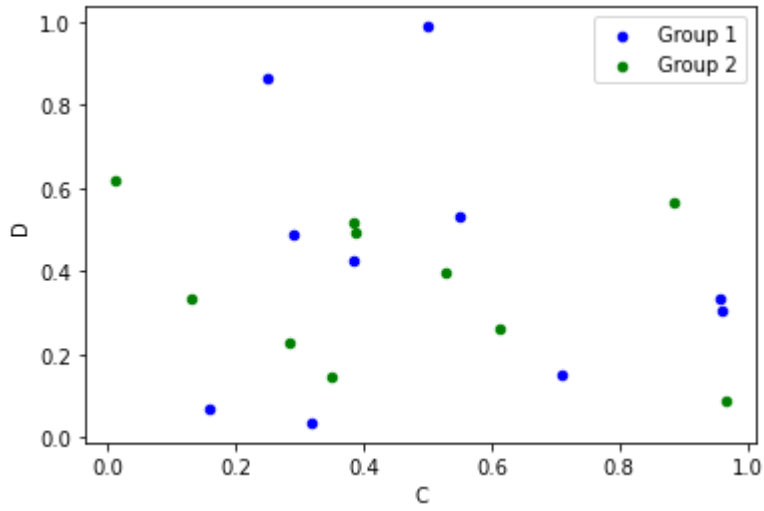


In [4]:

```
# You can visualize multiple groups as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
ax = df.plot.scatter(x='A', y='B',
                     color='Blue',
                     label='Group 1')
plt.figure()
df.plot.scatter(x='C', y='D',
                color='Green',
                label='Group 2',
                ax=ax)
plt.show()
```

```
          A         B         C         D
0  0.158250  0.068270  0.387570  0.494766
1  0.708427  0.151093  0.614231  0.263271
2  0.382490  0.426263  0.282260  0.229730
3  0.549723  0.531597  0.349985  0.144354
4  0.959155  0.304468  0.886052  0.565379
5  0.289317  0.487562  0.012383  0.619153
6  0.317111  0.035296  0.128710  0.333618
7  0.247991  0.863729  0.526713  0.394465
8  0.956150  0.335313  0.966236  0.086027
9  0.500298  0.989002  0.382830  0.515911
```




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

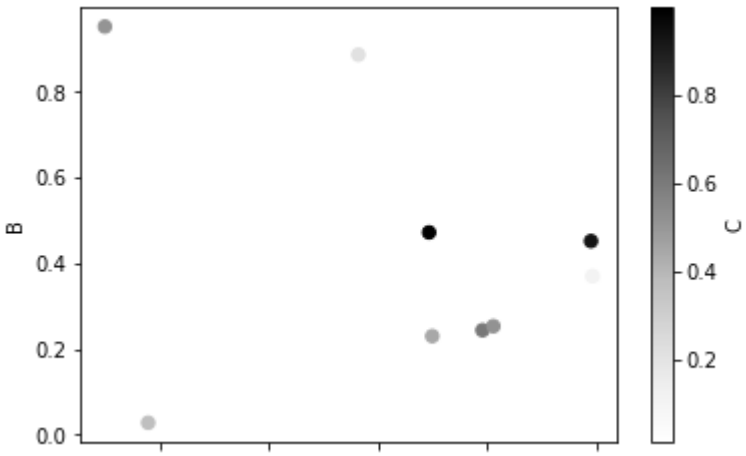
In [5]:

```
# Let's see how to customize the scatter plot. You can customize the
# color and the size of the points. The color or size can be a constant
# or can be variable. The following is an example of variable colors and
# a constant size for the data points. When the color is variable, a
# color bar is added to the output by default.

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
plt.figure()
df.plot.scatter(x='A', y='B', c='C', s=40)
plt.show()
```

```
          A         B         C         D
0  0.993785  0.369531  0.113671  0.033638
1  0.253216  0.613879  0.011975  0.324805
2  0.564459  0.885999  0.214404  0.647565
3  0.699909  0.229516  0.447831  0.547404
4  0.792090  0.243353  0.609524  0.785429
5  0.694053  0.471370  0.998047  0.416216
6  0.990993  0.451163  0.933526  0.232316
7  0.178928  0.027442  0.372009  0.493125
8  0.099541  0.951616  0.508871  0.713078
9  0.811926  0.252643  0.517154  0.244082
```

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



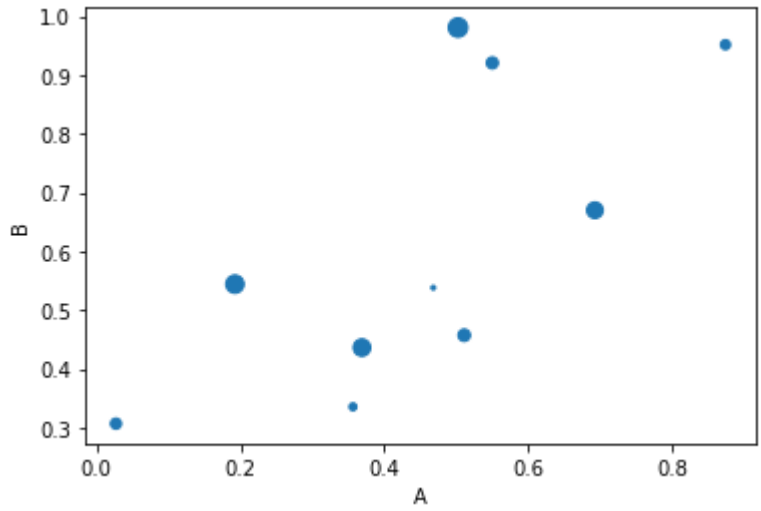
In [6]:

```
# Let's assign the size to be variable as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
plt.figure()
df.plot.scatter(x='A', y='B', s=df['C']*100)
plt.show()
```

```
          A         B         C         D
0  0.467893  0.538154  0.044747  0.541665
1  0.510827  0.457329  0.360694  0.313735
2  0.027107  0.306957  0.286708  0.854560
3  0.692369  0.669935  0.653137  0.569974
4  0.356256  0.335569  0.137873  0.509330
5  0.549891  0.920535  0.339177  0.146543
6  0.502185  0.980663  0.908337  0.623928
7  0.192319  0.544221  0.800807  0.586362
8  0.873758  0.951462  0.232705  0.992833
9  0.368777  0.436280  0.730076  0.025192
```

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



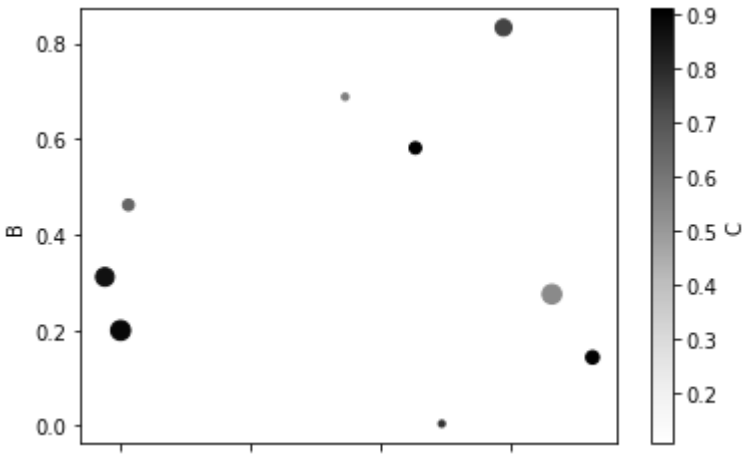
In [7]:

```
# Finally, let's see an example with fully customized variable sizes
# and variable colors as follows:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 4),
                  columns=['A', 'B', 'C', 'D'])
print(df)
plt.figure()
df.plot.scatter(x='A', y='B', c='C', s=df['D']*100)
plt.show()
```

```
          A         B         C         D
0  0.202095  0.198970  0.892807  0.925507
1  0.788580  0.832744  0.739899  0.647391
2  0.214098  0.461490  0.653483  0.309219
3  0.924799  0.142611  0.905728  0.424266
4  0.653568  0.580850  0.912598  0.342449
5  0.862610  0.274749  0.540942  0.899167
6  0.546046  0.687756  0.576026  0.112516
7  0.694001  0.003629  0.777430  0.101172
8  0.407819  0.719169  0.104732  0.534405
9  0.178030  0.310865  0.864474  0.810222
```

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



In [8]:

```
# Hexagonal Bin Plots
# You can also visualize data with hexagonal bin (hexbin) plots.

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(100, 2),
                  columns=['A', 'B'])
df['B'] = df['B'] + np.arange(100)
print(df)
# Let's visualize this data with a hexbin plot as follows:
plt.figure()
df.plot.hexbin(x='A', y='B', gridsize=20)
plt.show()
```

```
           A          B
0  -0.091948  -1.584430
1   0.068766   0.069732
2   1.043884   1.880081
3   1.059820   4.539050
4  -1.066144   3.176690
..       ...        ...
95  1.187480  93.344278
96 -0.511698  95.603423
97 -0.565959  97.101544
98 -0.168269  97.188910
99 -0.177930  97.982401

[100 rows x 2 columns]


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