Pandas – Chart Visualization

Pandas uses the **matplotlib** for chart visualization

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

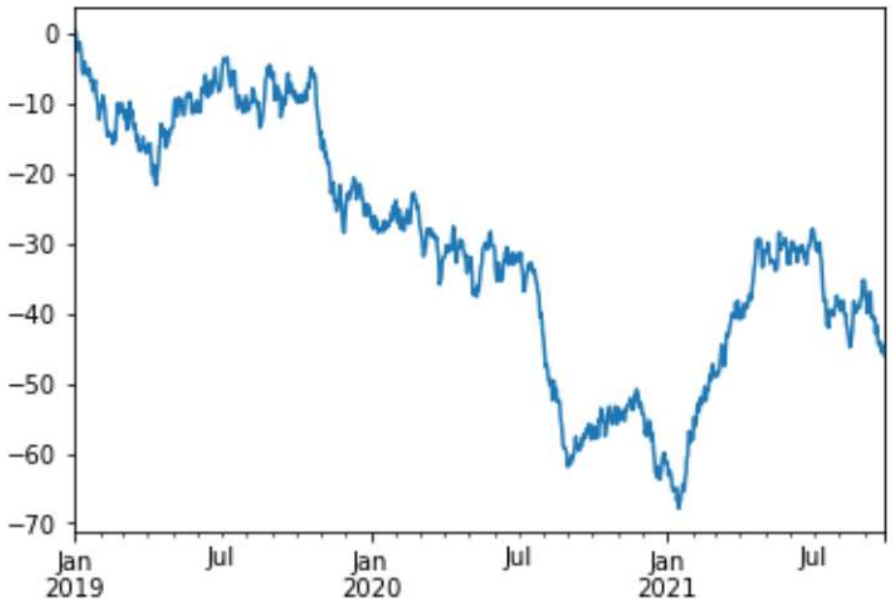
```
In [2]: np.random.seed(123456)
        ts = pd.Series(np.random.randn(1000), index=pd.date_range("1/1/2019", periods=1000))
        ts.head(3)
```

```
Out[2]: 2019-01-01    0.469112
        2019-01-02   -0.282863
        2019-01-03   -1.509059
        Freq: D, dtype: float64
```

```
In [3]: ts = ts.cumsum()
```

```
In [4]: ts.plot()
```

```
Out[4]: <AxesSubplot:>
```



Pandas – Chart Visualization

plot() can be used with a DataFrame to plot all the columns with labels

```
In [5]: df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=list("ABCD"))
        df.head(3)
```

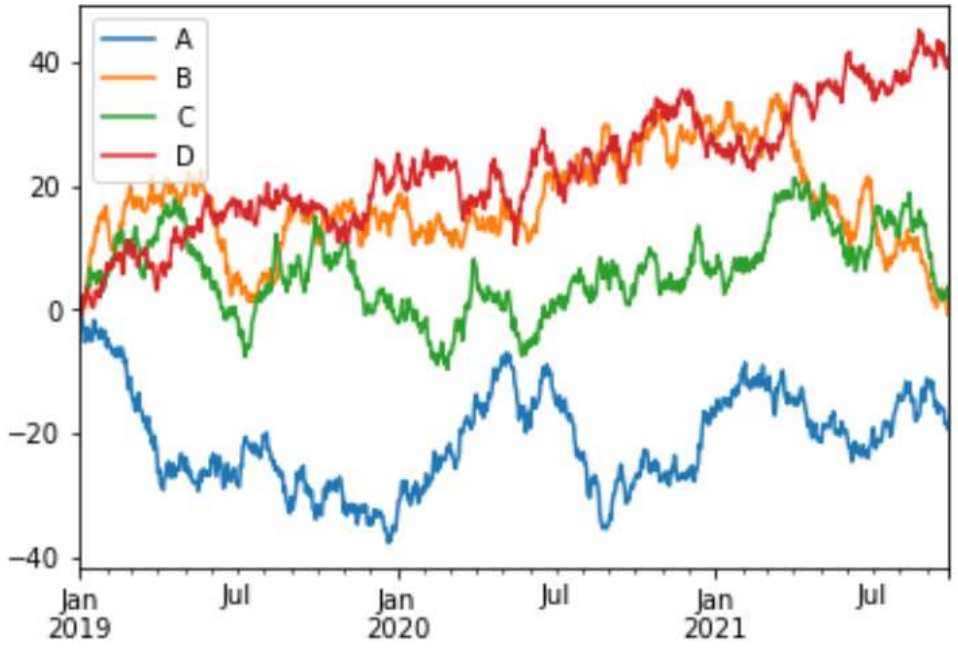
Out[5]:

	A	B	C	D
2019-01-01	-0.218470	-0.061645	-0.723780	0.551225
2019-01-02	-0.497767	0.837519	1.103245	-1.118384
2019-01-03	-0.542980	-0.994002	1.508742	-0.328697

```
In [6]: df = df.cumsum()
        plt.figure()
        df.plot()
```

```
Out[6]: <AxesSubplot:>

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



Pandas – Chart Visualization

The **plot**() x and y keywords can be used to plot one DataFrame column versus another

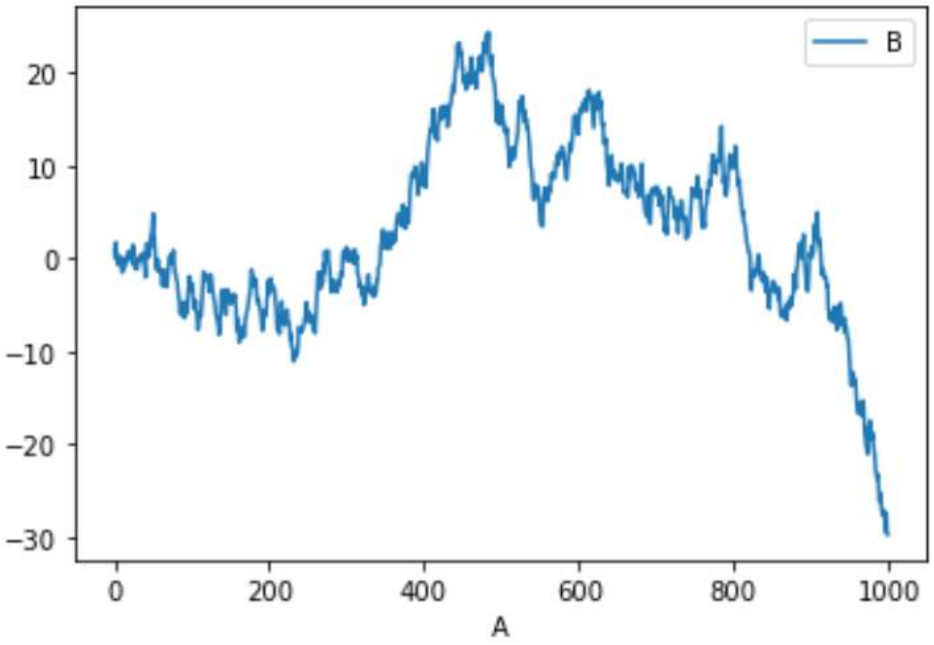
```
In [7]: df3 = pd.DataFrame(np.random.randn(1000, 2), columns=["B", "C"]).cumsum()
        df3["A"] = pd.Series(list(range(len(df))))
        df3.head(3)
```

Out[7]:

	B	C	A
0	0.879831	0.573687	0
1	0.145896	0.149091	1
2	1.782761	1.397438	2

```
In [8]: df3.plot(x="A", y="B")
```

Out[8]: <AxesSubplot:xlabel='A'>



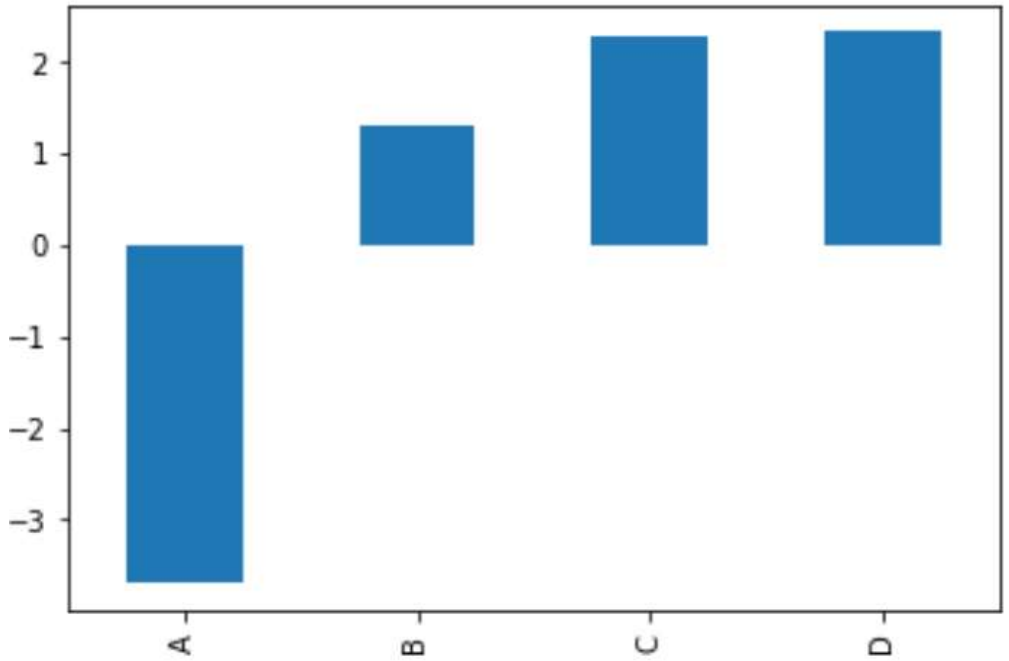
Pandas – Chart Visualization – Plot methods

The plot methods can be selected with the **kind** keyword or using **dataFrame.plot.<kind>**

- '**bar**' or '**barh**' for bar plots
- '**hist**' for histogram
- '**box**' for boxplot
- '**kde**' or '**density**' for density plots
- '**area**' for area plots
- '**scatter**' for scatter plots
- '**hexbin**' for hexagonal bin plots
- '**pie**' for pie plots

```
In [9]: df.iloc[5].plot(kind="bar")
        df.iloc[5].plot.bar()

Out[9]: <AxesSubplot:>
```



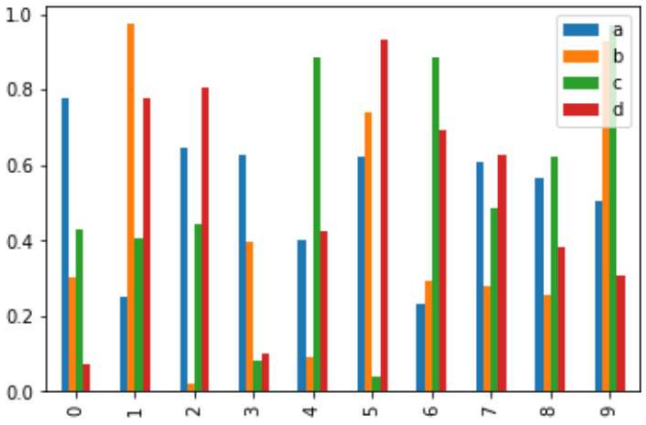
Pandas – Chart – Bar plots

```
In [10]: df2 = pd.DataFrame(np.random.rand(10, 4), columns=["a", "b", "c", "d"])
         df2.head(3)
```

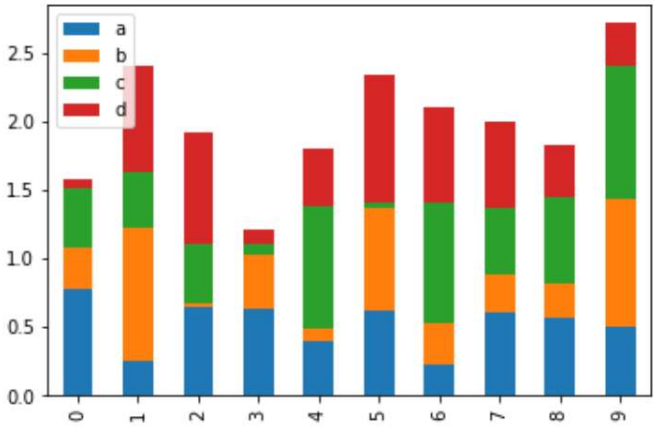
Out[10]:

	a	b	c	d
0	0.776756	0.303773	0.428600	0.070400
1	0.249213	0.973558	0.404596	0.777473
2	0.644560	0.022563	0.444410	0.804606

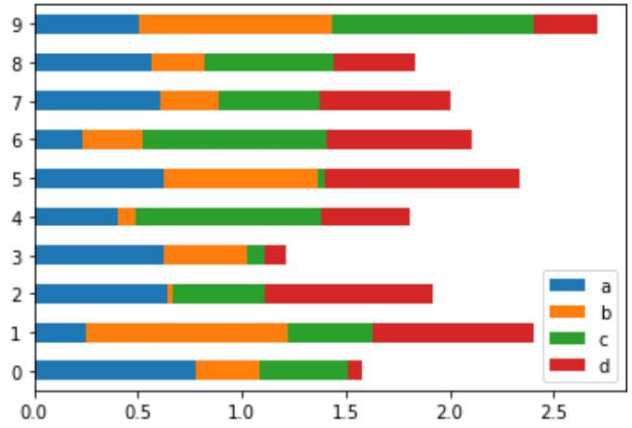
```
In [11]: df2.plot.bar();
```




```
In [12]: df2.plot.bar(stacked=True);
```




```
In [13]: df2.plot.barh(stacked=True);
```



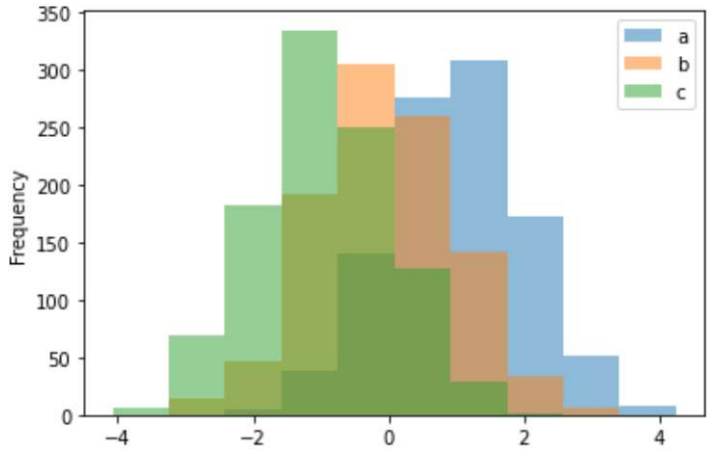
Pandas – Chart - Histogram and Box plots

In [14]:
```
df4 = pd.DataFrame(
    {
        "a": np.random.randn(1000) + 1,
        "b": np.random.randn(1000),
        "c": np.random.randn(1000) - 1,
    },
    columns=["a", "b", "c"],
)
df4.head(3)
```

Out[14]:

	a	b	c
0	1.717808	1.600283	-0.854723
1	0.099041	-0.605731	-0.691854
2	2.180217	-0.561047	-2.043530

In [15]:
```
df4.plot.hist(alpha=0.5);
```

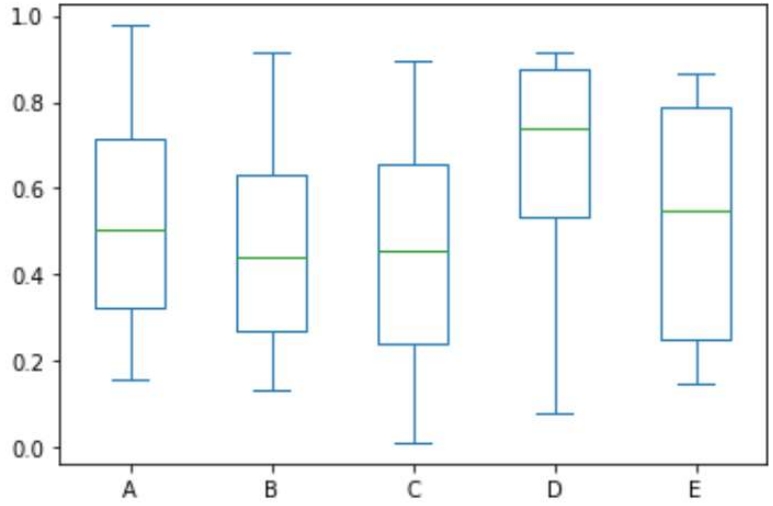


In [16]:
```
df = pd.DataFrame(
    np.random.rand(10, 5),
    columns=["A", "B", "C", "D", "E"])
df.head(3)
```

Out[16]:

	A	B	C	D	E
0	0.177332	0.580773	0.296386	0.077946	0.190391
1	0.157114	0.647679	0.079399	0.711080	0.356732
2	0.489527	0.299715	0.650378	0.768944	0.857699

In [17]:
```
df.plot.box();
```



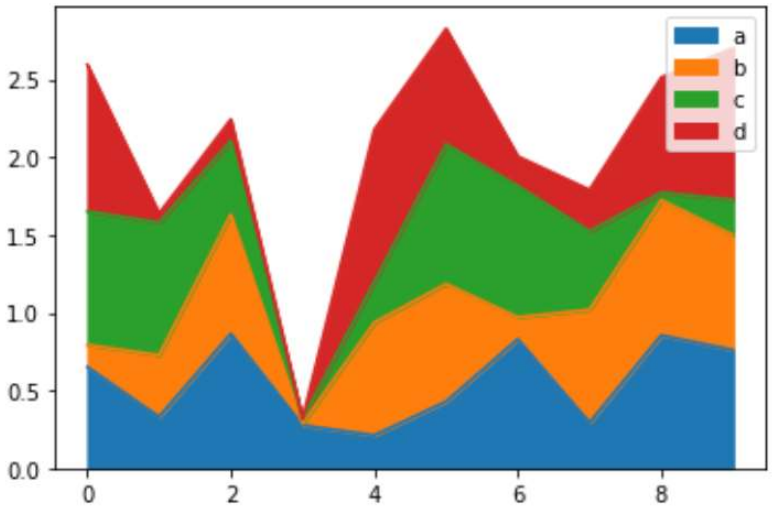
Pandas – Chart - Area and Scatter plot

In [18]:
```
df = pd.DataFrame(
    np.random.rand(10, 4),
    columns=["a", "b", "c", "d"])
df.head(3)
```

Out[18]:

	a	b	c	d
0	0.650358	0.142018	0.859387	0.941305
1	0.328915	0.396671	0.853521	0.058256
2	0.863381	0.767262	0.471649	0.139792

In [19]:
```
df.plot.area();
```

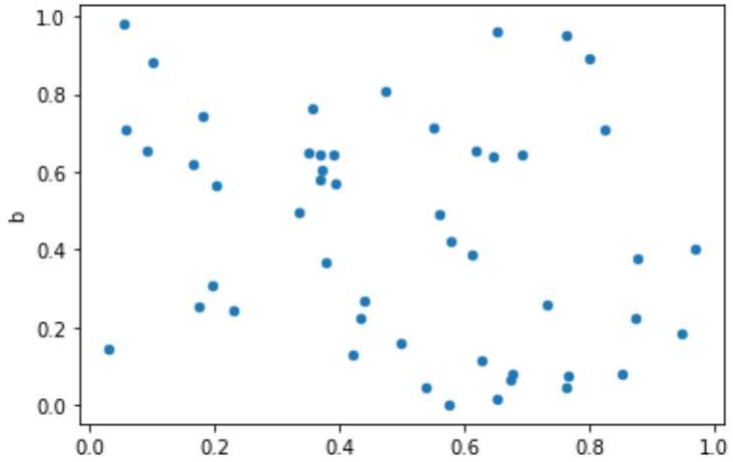


In [20]:
```
df = pd.DataFrame(
    np.random.rand(50, 4),
    columns=["a", "b", "c", "d"])
df["species"] = pd.Categorical(
    ["setosa"] * 20 + ["versicolor"] * 20 + ["virginica"] * 10
)
df.head(3)
```

Out[20]:

	a	b	c	d	species
0	0.472352	0.805964	0.325777	0.035732	setosa
1	0.873763	0.225313	0.174763	0.073953	setosa
2	0.173608	0.255411	0.008679	0.882071	setosa

In [21]:
```
df.plot.scatter(x="a", y="b");
```



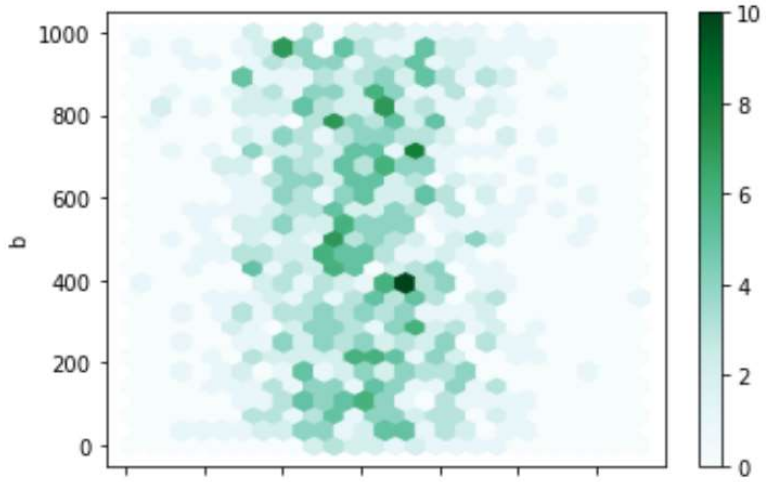
Pandas – Chart – Hexagonal bin and pie plot

In [22]:
```
df = pd.DataFrame(
    np.random.randn(1000, 2),
    columns=["a", "b"])
df["b"] = df["b"] + np.arange(1000)
df.head(3)
```

Out[22]:

	a	b
0	-0.477601	-0.692893
1	1.360552	0.985217
2	0.713865	3.025486

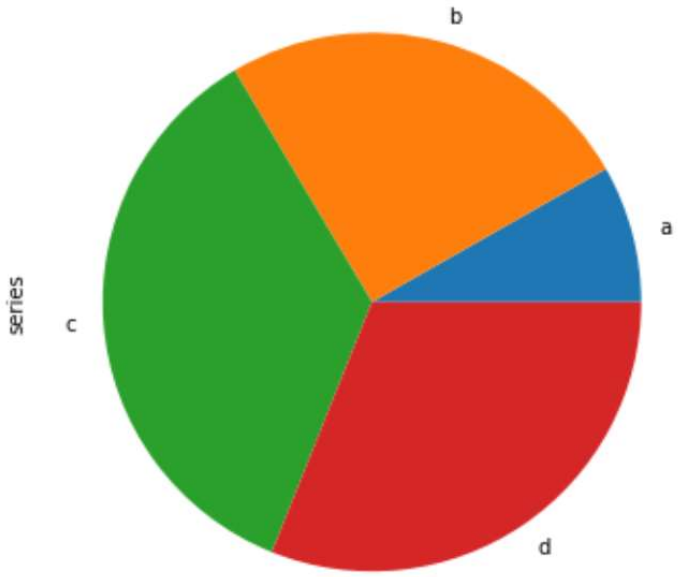
In [23]:
```
df.plot.hexbin(x="a", y="b", gridsize=25);
```



In [24]:
```
series = pd.Series(
    3 * np.random.rand(4),
    index=["a", "b", "c", "d"], name="series")
series.head(3)
```

Out[24]:
```
a    0.408321
b    1.254653
c    1.757288
Name: series, dtype: float64
```

In [25]:
```
series.plot.pie(figsize=(6, 6));
```



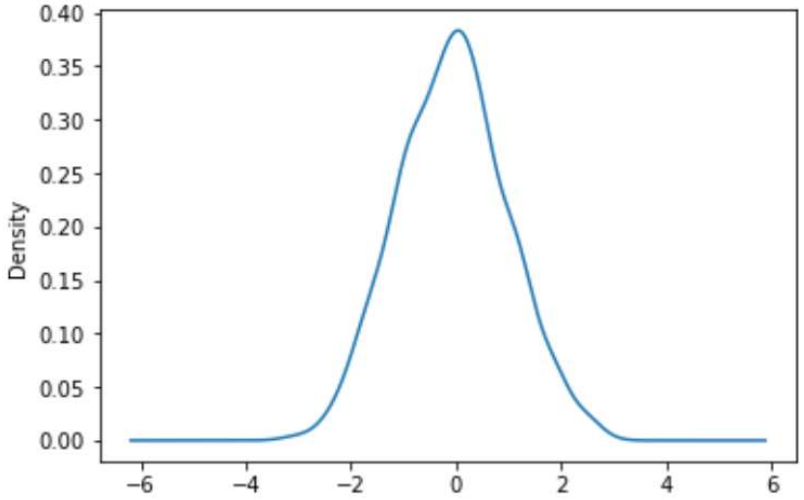
Pandas – Chart – Density bin and matplotlib

If necessary it can be used directly matplotlib

```
In [26]: ser = pd.Series(np.random.randn(1000))
         ser.head(3)
```

```
Out[26]: 0    1.375064
         1   -0.268824
         2   -1.503032
         dtype: float64
```

```
In [27]: ser.plot.kde();
```




```
In [28]: price = pd.Series(
             np.random.randn(150).cumsum(),
             index=pd.date_range("2000-1-1", periods=150, freq="B"),
         )
         ma = price.rolling(20).mean()
         mstd = price.rolling(20).std()
         plt.figure();
         plt.plot(price.index, price, "k");
         plt.plot(ma.index, ma, "b");
         plt.fill_between(
             mstd.index, ma - 2 * mstd, ma + 2 * mstd, color="b", alpha=0.2);
```