Session 5 Introduction to Plotting

Matplotlib

It is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib as mpl #mpltools provides tools for Matplotlib that make it easier to adjust the style, choose
import matplotlib.pyplot as plt
%matplotlib inline
```

ggplot style

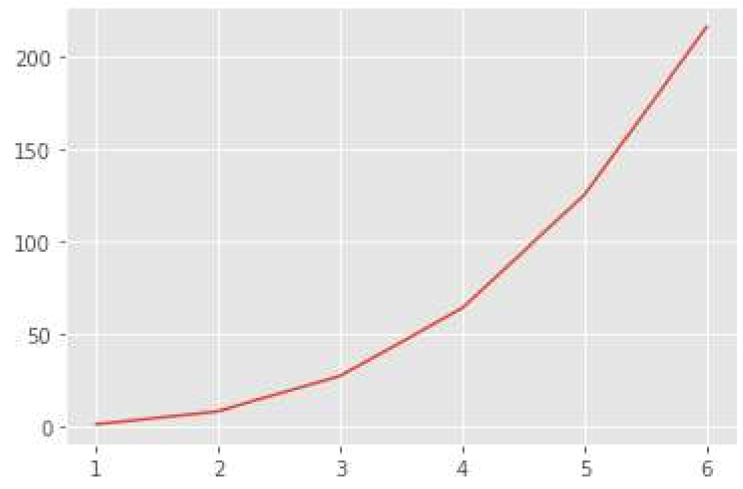
A key feature of mpltools is the ability to set “styles”—essentially, stylesheets that are similar to matplotlibrc files.

This example demonstrates the “ggplot” style, which adjusts the style to emulate ggplot

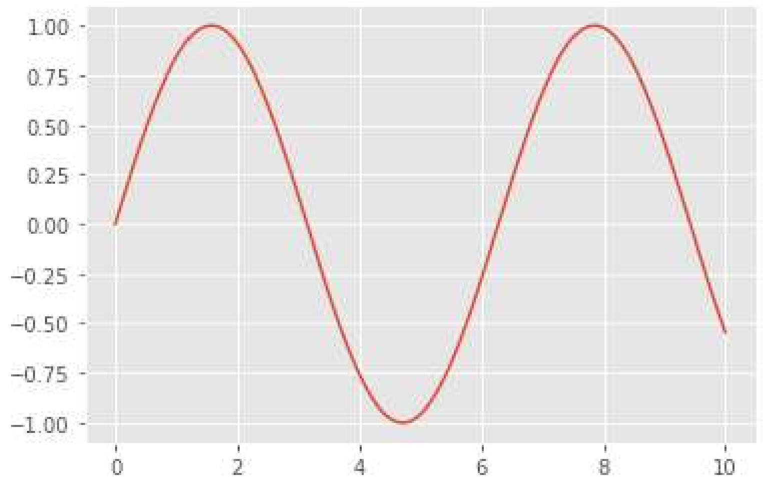
```
plt.style.use('ggplot')
```

Line plot

```
x = np.array([1,2,3,4,5,6])
y = np.power(x,3)
plt.plot(x,y);

```



```
x = np.linspace(0,10,1000)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```



The matplotlib.rcParams

It is essentially a dictionary which stores certain default parameters that are meant to be used when creating objects in matplotlib.rcParams.update will update this dictionary; update is a method of the python dict object. matplotlib.style.use is a shortcut to updating the rcParams, it will load the respective parameters from a file or dictionary.

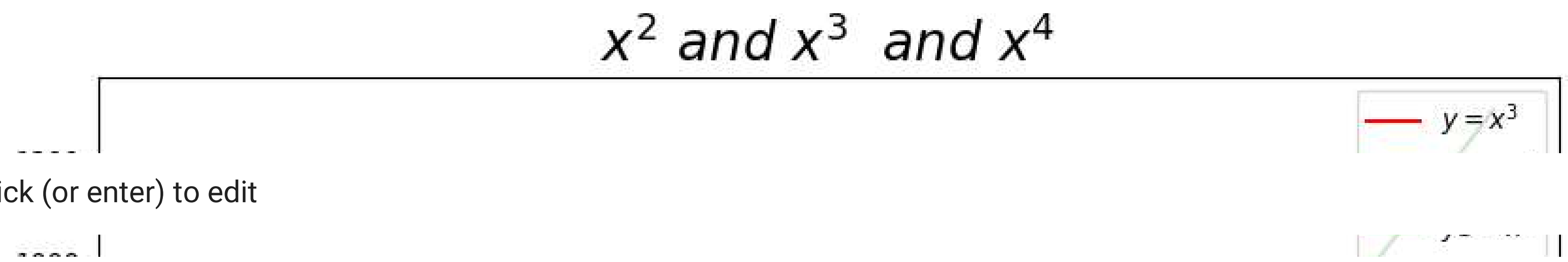
```
mpl.rcParams.update(mpl.rcParamsDefault)
```

```
print(plt.rcParams['figure.figsize'])
```

```
[6.4, 4.8]
```

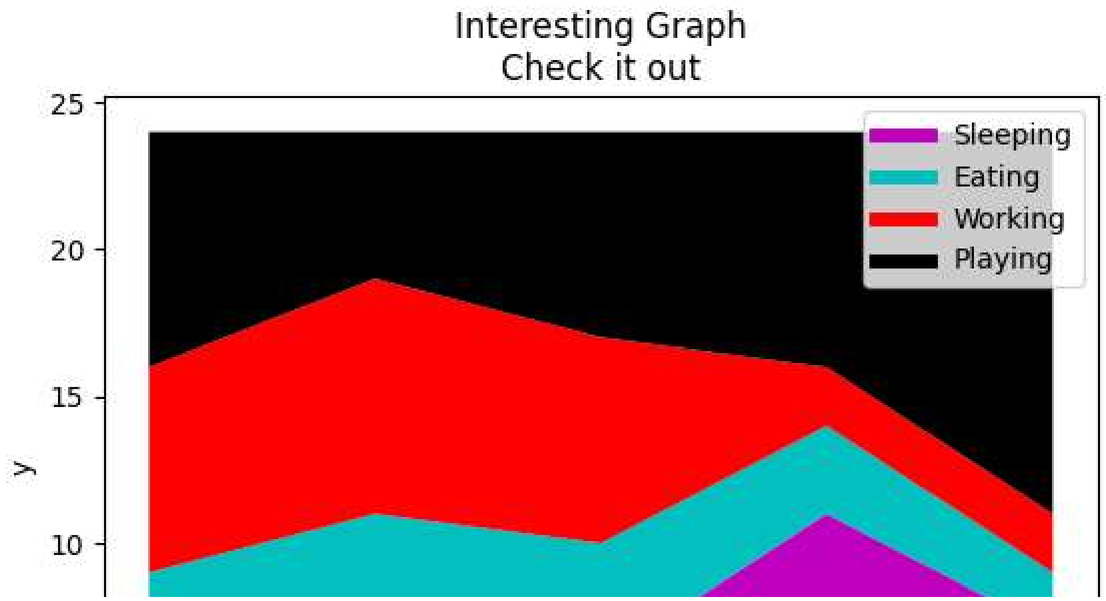
```
plt.figure(figsize=(8,3))
x = np.linspace(0,10,40)
y = np.sin(x)
plt.plot(x,y,'y--', linewidth=3) #g-
plt.xlabel("X Axis")
plt.ylabel("Y Axis  ", rotation="horizontal")
plt.show()
```

```
plt.figure(figsize=(10,5))
x = np.array([1,2,3,4,5,6])
y = np.power(x,3)
y2 = np.power(x,2)
y3=np.power(x,4)
plt.plot(x,y,'r-', label="$y=x^3$") #g-
plt.plot(x,y2,'b-', label="$y2=x^2$")
plt.plot(x,y3,'g-', label="$y3=x^4$")
plt.xlabel("X Axis", fontsize=18)
plt.ylabel("Y Axis", fontsize=18)
plt.title("$x^2$ $and$ $x^3$  $and$ $x^4$ ", fontsize=20)
plt.legend(loc="upper right")
plt.show()
```



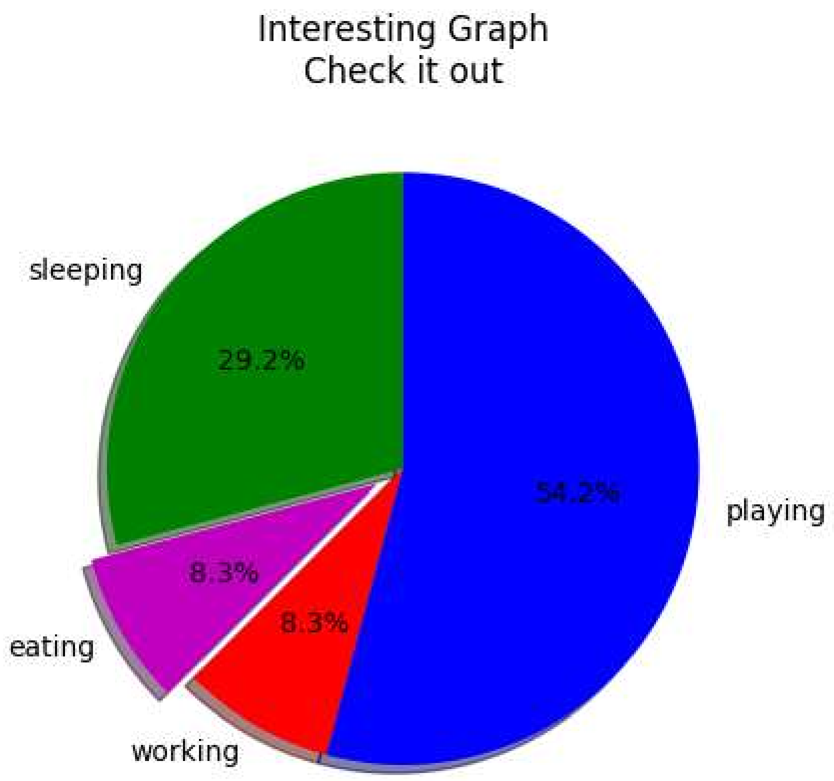
Double-click (or enter) to edit

```
days = [1,2,3,4,5]

sleeping = [7,8,6,11,7]
eating =   [2,3,4,3,2]
working =  [7,8,7,2,2]
playing =  [8,5,7,8,13]


plt.plot([],[],color='m', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)

plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```



Pie charts

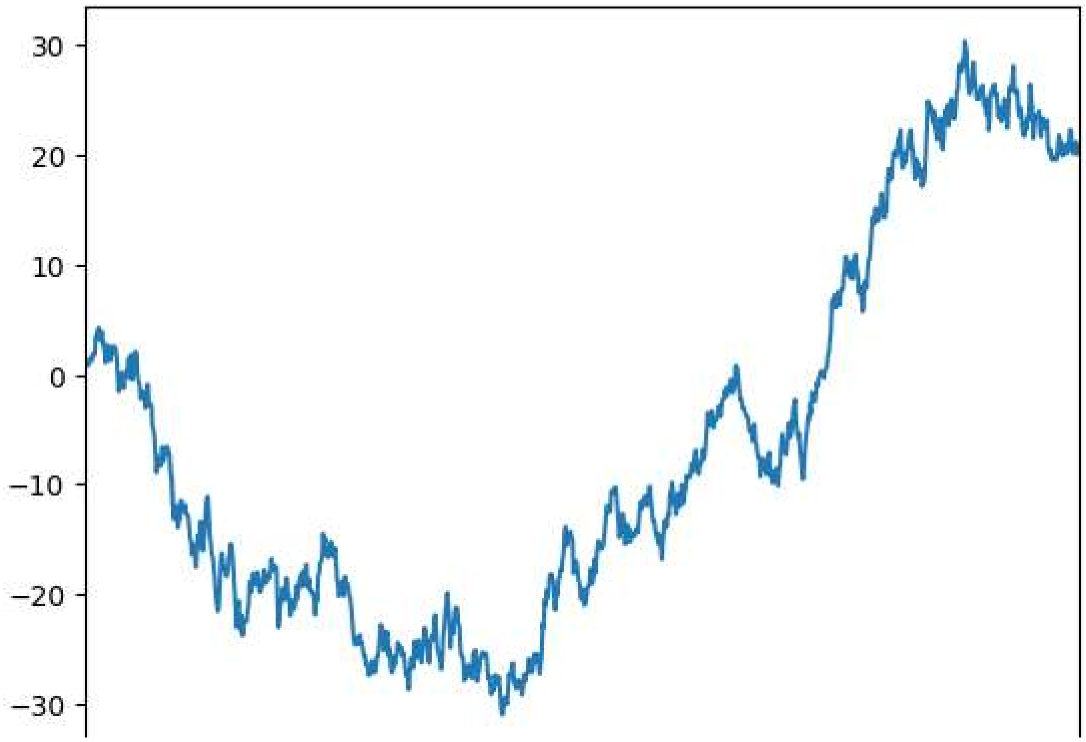
```
slices = [7,2,2,13]
activities = ['sleeping','eating','working','playing']
cols = ['g','m','r','b']

plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow= True,
        explode=(0,0.1,0,0),
        autopct='%1.1f%%')

plt.title('Interesting Graph\nCheck it out')
plt.show()
```



```
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
```

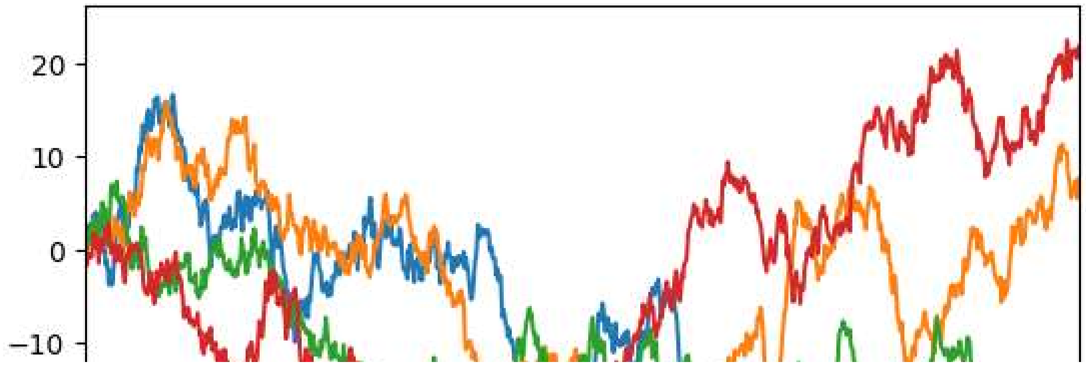
```

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()
ts.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff15bb38190>
```

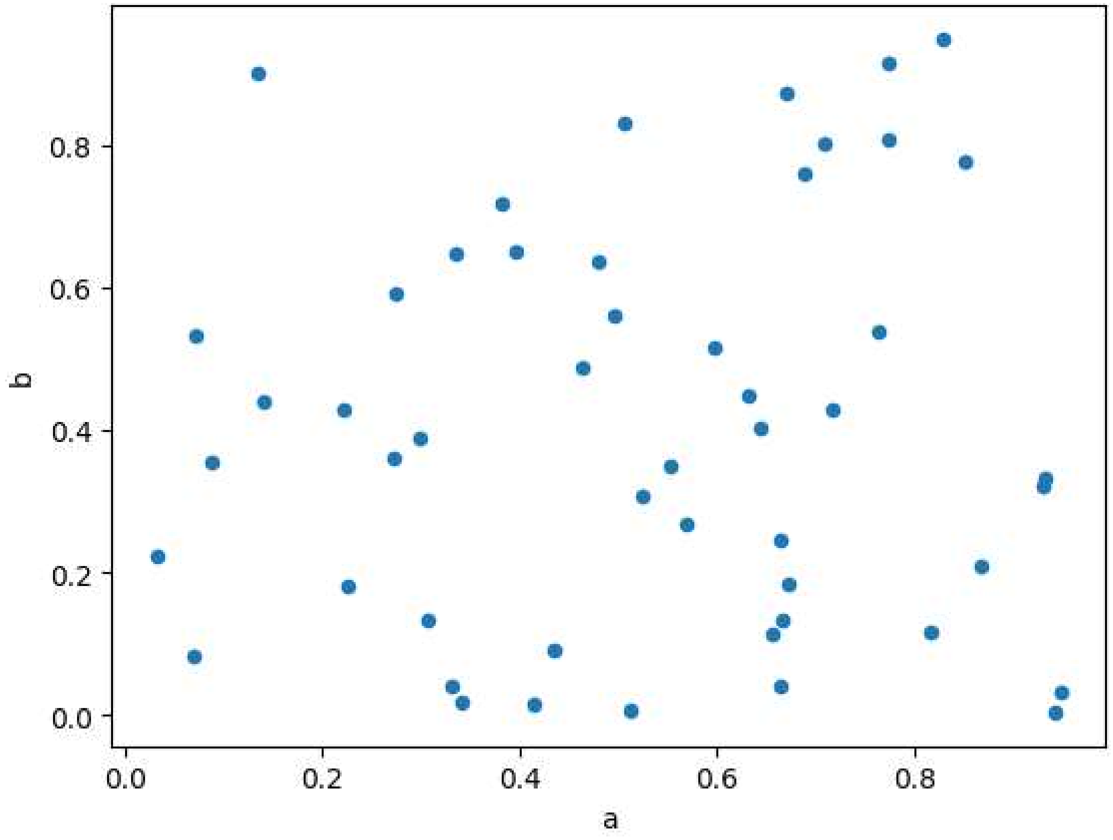


```
df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=list('ABCD'))
df = df.cumsum()
plt.figure(); df.plot();
```

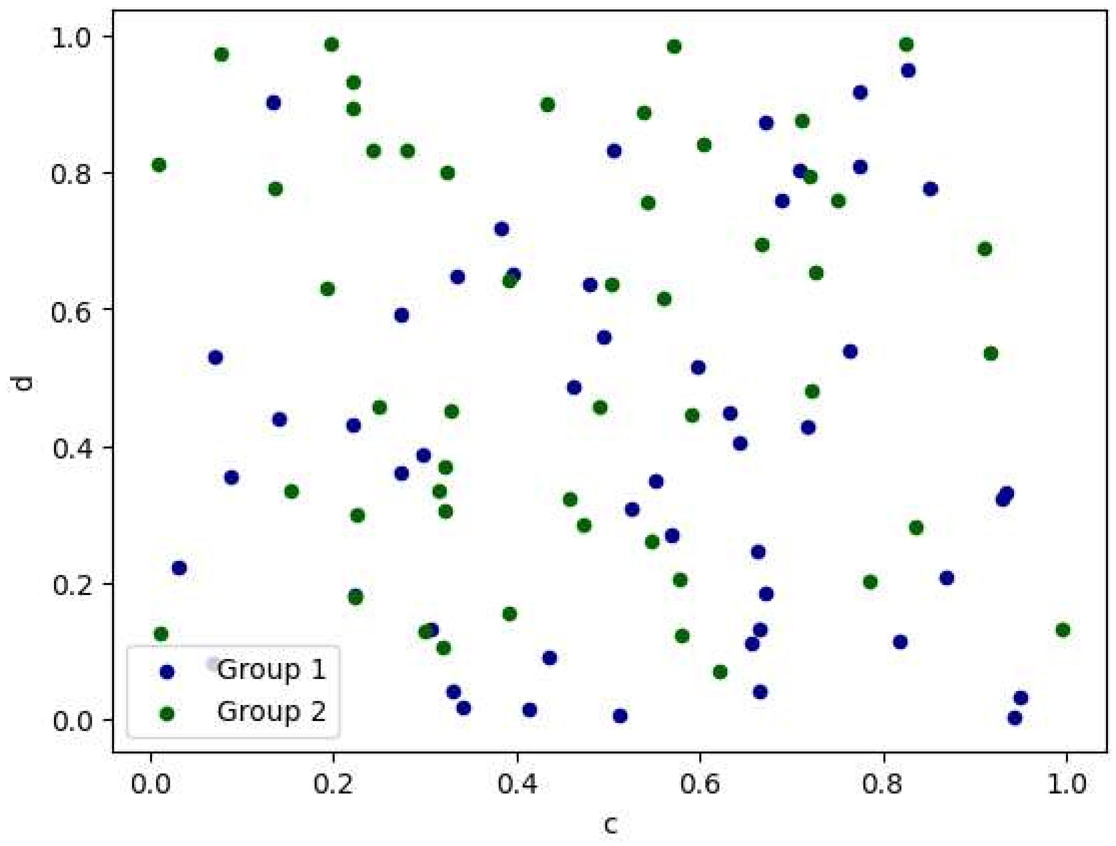
```
<Figure size 640x480 with 0 Axes>
```



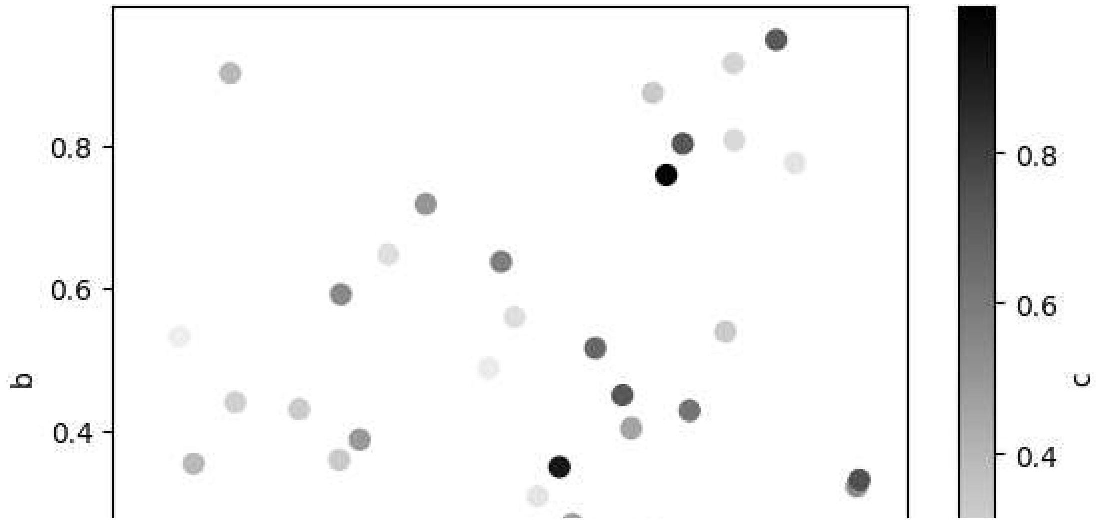
```
df = pd.DataFrame(np.random.rand(50, 4), columns=['a', 'b', 'c', 'd'])
df.plot.scatter(x='a', y='b');
```



```
ax = df.plot.scatter(x='a', y='b', color='DarkBlue', label='Group 1');
df.plot.scatter(x='c', y='d', color='DarkGreen', label='Group 2', ax=ax);
```



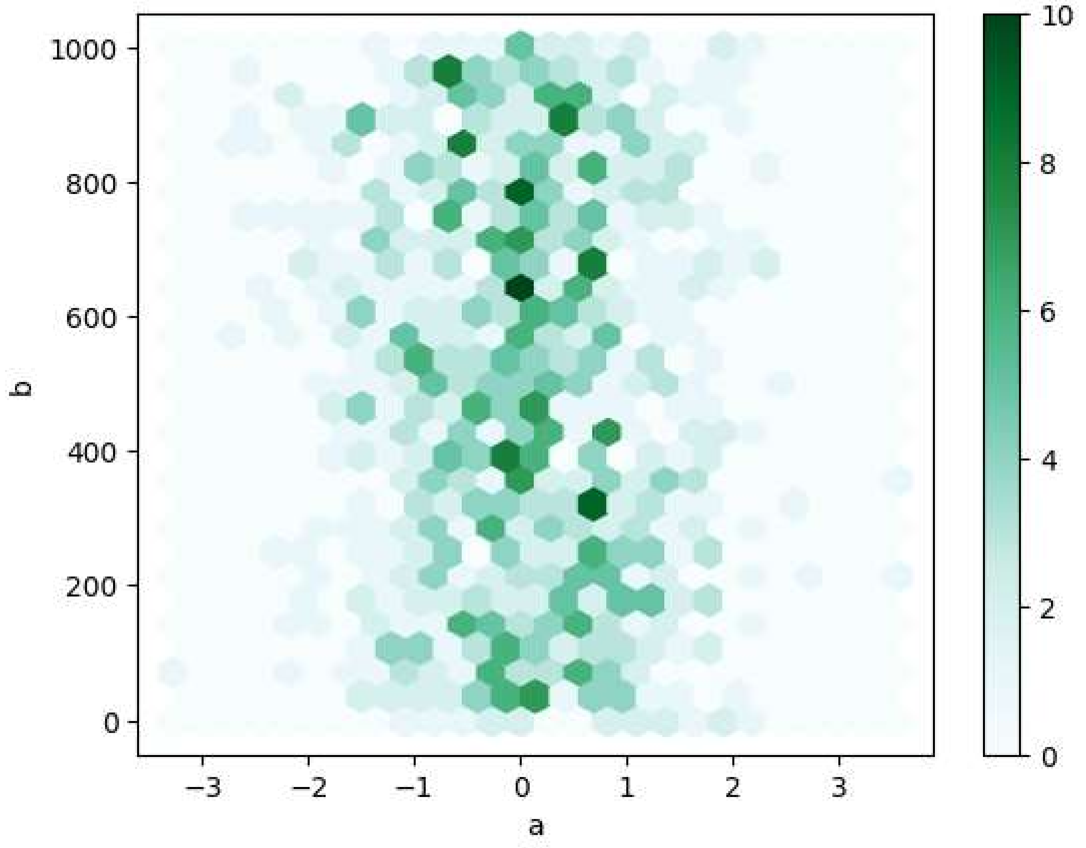
```
df.plot.scatter(x='a', y='b', c='c', s=50);
```



```
df.plot.scatter(x='a', y='b', s=df['c']*200);
```

```
df = pd.DataFrame(np.random.randn(1000, 2), columns=['a', 'b'])
df['b'] = df['b'] + np.arange(1000)
df.plot.hexbin(x='a', y='b', gridsize=25)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff15bf0b4d0>
```



Pair plots using Scatter matrix in Pandas

Checking for collinearity among attributes of a dataset, is one of the most important steps in data preprocessing.

A good way to understand the correlation among the features, is to create scatter plots for each pair of attributes.

Pandas has a function scatter_matrix(), for this purpose.

scatter_matrix() can be used to easily generate a group of scatter plots between all pairs of numerical features.

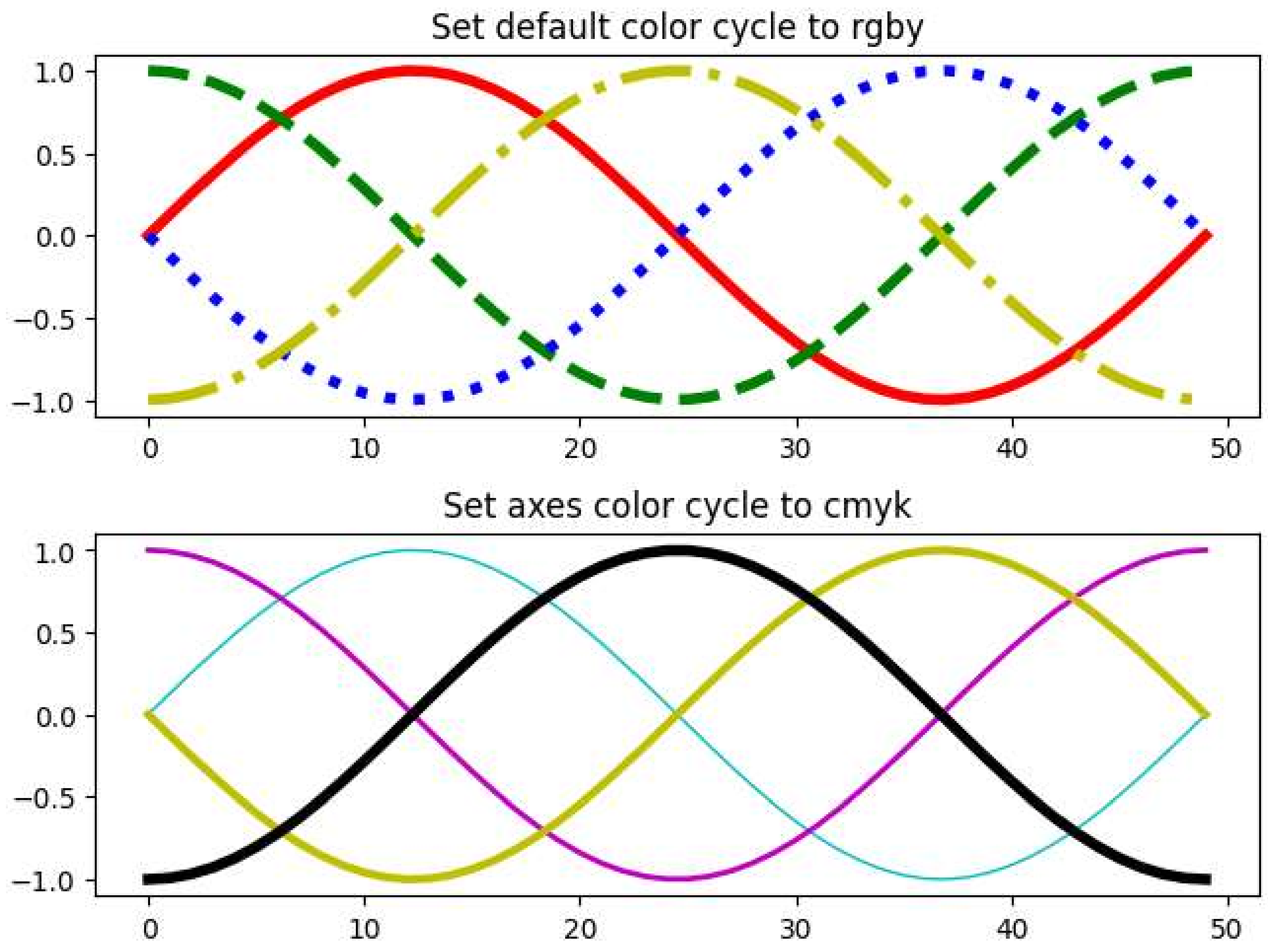
It creates a plot for each numerical feature against every other numerical feature and also a histogram for each of them.

```
from pandas.plotting import scatter_matrix
df = pd.DataFrame(np.random.randn(1000, 4), columns=['a', 'b', 'c', 'd'])
scatter_matrix(df, alpha=0.2, figsize=(6, 6), diagonal='kde')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7ff15bcf1390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159f94dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159f57490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159f0ab10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff159ece1d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159e85850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159e3af50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159dfc550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff159dfc590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159db2d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159d2a950>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159d56b10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7ff159ca3690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159ccab90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159c97610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7ff159c4eb90>]],
      dtype=object)
```

```
from pandas.plotting import radviz
data = pd.read_csv('iris.csv')
```

```
from cycler import cycler
x = np.linspace(0, 2 * np.pi)
offsets = np.linspace(0, 2*np.pi, 4, endpoint=False)
# Create array with shifted-sine curve along each column
yy = np.transpose([np.sin(x + phi) for phi in offsets])

# 1. Setting prop cycle on default rc parameter
plt.rc('lines', linewidth=4)
plt.rc('axes', prop_cycle=(cycler(color=['r', 'g', 'b', 'y']) +
                           cycler(linestyle=['-', '--', ':', '-.'])))
fig, (ax0, ax1) = plt.subplots(nrows=2, constrained_layout=True)
ax0.plot(yy)
ax0.set_title('Set default color cycle to rgby')

# 2. Define prop cycle for single set of axes
#    For the most general use-case, you can provide a cycler to
#    `.set_prop_cycle`.
#    Here, we use the convenient shortcut that we can alternatively pass
#    one or more properties as keyword arguments. This creates and sets
#    a cycler iterating simultaneously over all properties.
ax1.set_prop_cycle(color=['c', 'm', 'y', 'k'], lw=[1, 2, 3, 4])
ax1.plot(yy)
ax1.set_title('Set axes color cycle to cmyk')

plt.show()
```

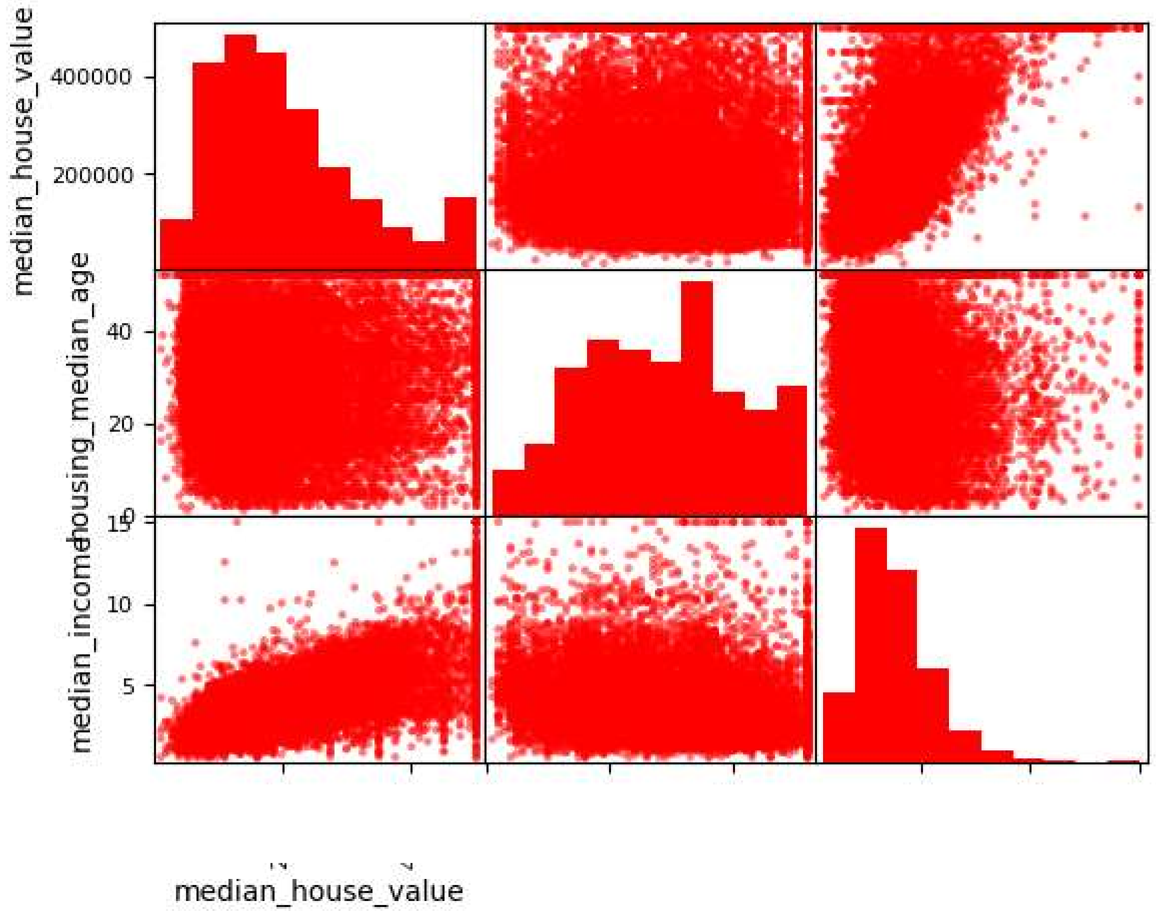


```
import pandas as pd

# loading the dataset
data = pd.read_csv('housing.csv')
```

```

# inspecting the data
data.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  int64  
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# selecting three numerical features
features = ['median_house_value', 'housing_median_age',
      'median_income']

# plotting the scatter matrix
# with the features
scatter_matrix(data[features])
plt.show()

```


median_house_value
400000
200000
housing_median_age
40
20
0
median_income
15
10
5
median_house_value

1

✓ 0s completed at 12:34 PM