Unit3_part_2_visuals-1

November 3, 2022

1 Matplotlib, an viable open source alternative to MATLAB, is a cross-platform, data visualization and graphical plotting library for Python, NumPy

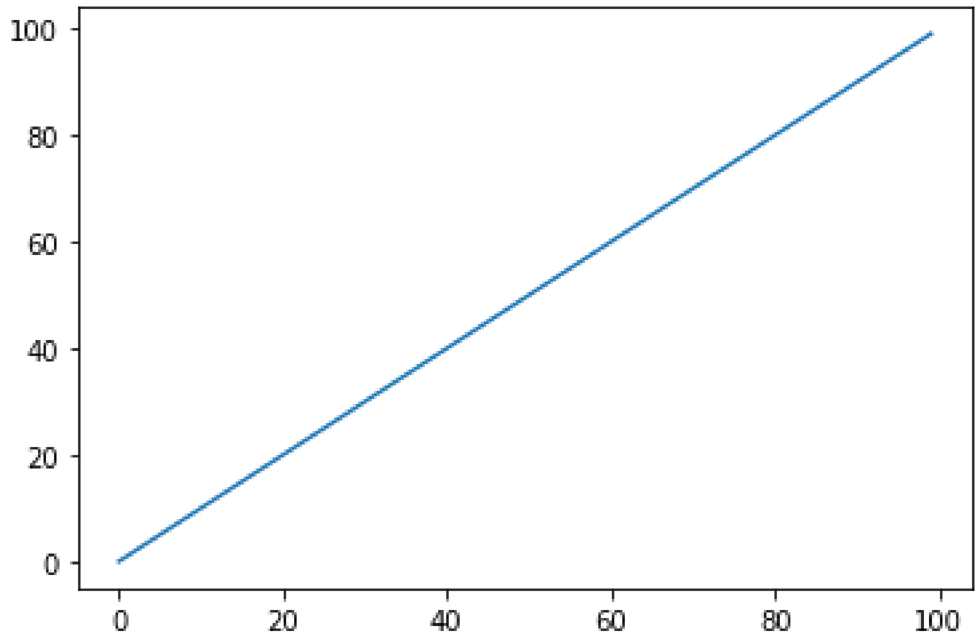
[1]:
```
import matplotlib.pyplot as plt
#import matplotlib
import numpy as np
import pandas as pd
#%matplotlib inline
#%matplotlib --list plt.plot([12,3,4])
```

2 simple line plot: using default values of rcParam

- rcParam: Each time Matplotlib loads, it defines a runtime configuration (rc) containing the default styles for every plot element created

[20]:
```
data = np.arange(100)
data
plt.plot(data)
```

[20]:
```
[<matplotlib.lines.Line2D at 0x2c0cb381580>]
```

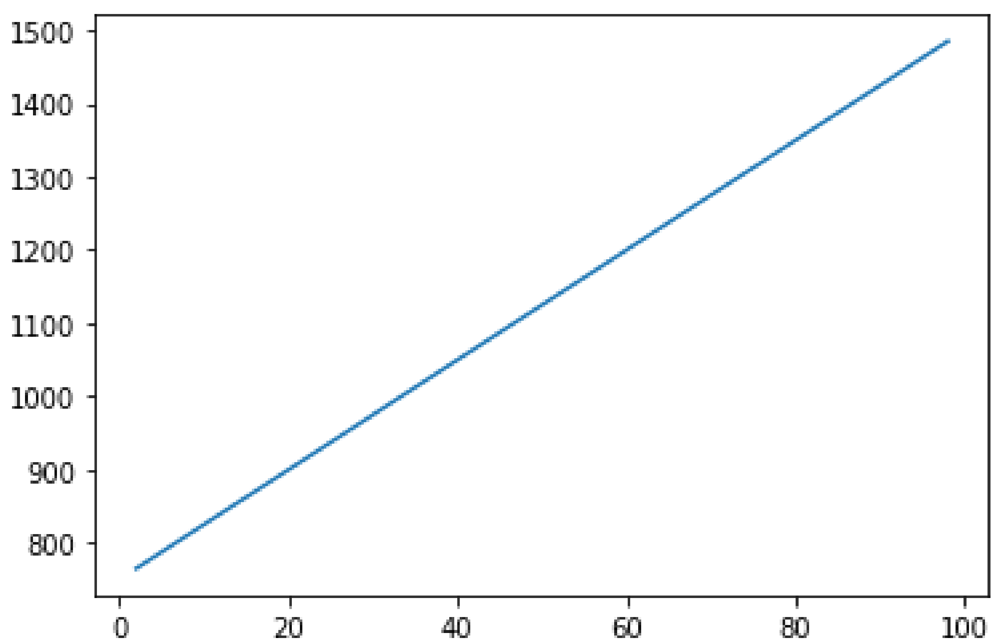



```
[21]: data1d=data[1:50]*2
```

```
[22]: data2d=data[51:100]*15
```

```
[23]: plt.plot(data1d,data2d)
```

```
[23]: [<matplotlib.lines.Line2D at 0x2c0cb31aac0>]
```

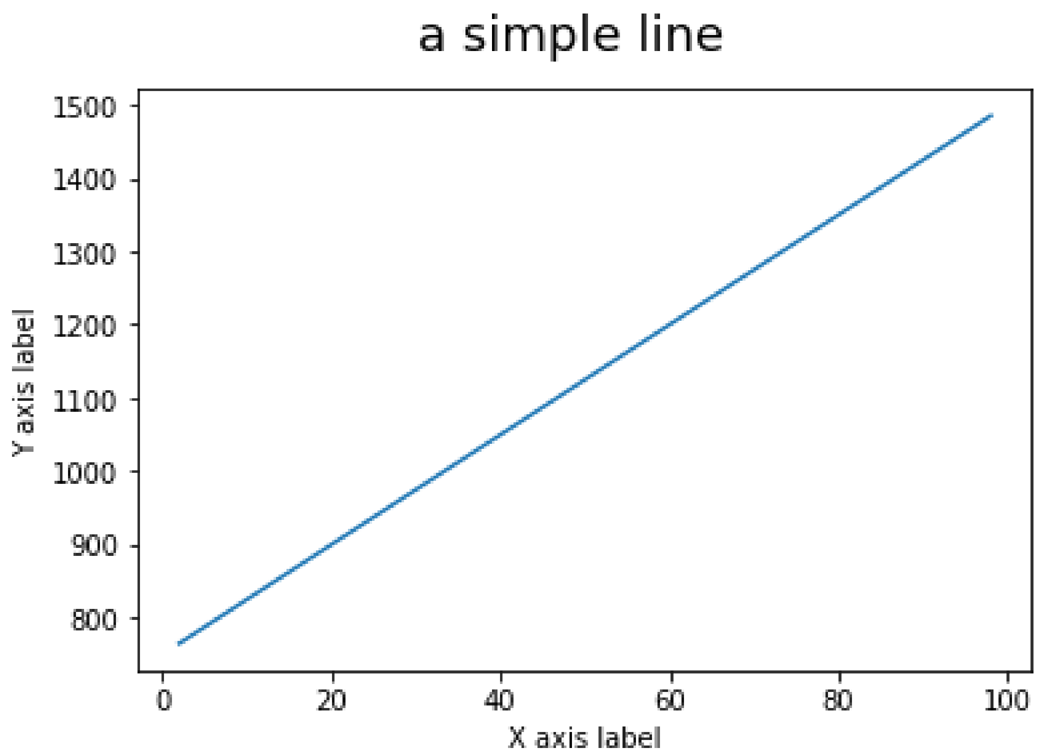


[24]:
```
plt.close()
```

2.1 adding axes labels and fifure title

[25]:
```
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
plt.suptitle('a simple line',fontsize=18)
plt.plot(data1d,data2d)
```

[25]: [<matplotlib.lines.Line2D at 0x2c0c952c970>]



2.2 fig: plots in matplotlib is handled using a figure object

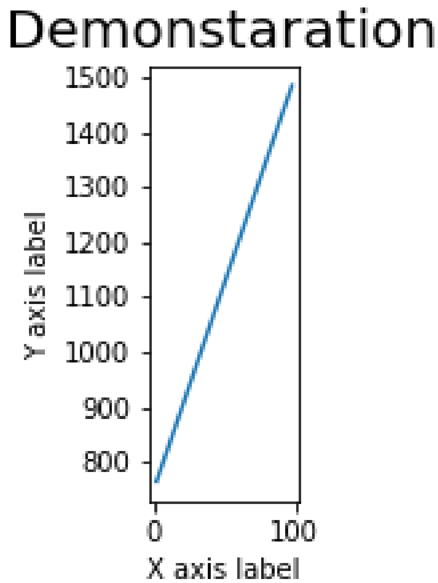
- Creating a new fig object

[36]:
```
```

```
<Figure size 72x216 with 0 Axes>
```

[43]:
```
fig = plt.fi

gure(figsize=[1,3])
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
plt.suptitle('Demonstaration',fontsize=20)
plt.plot(data1d,data2d)
```

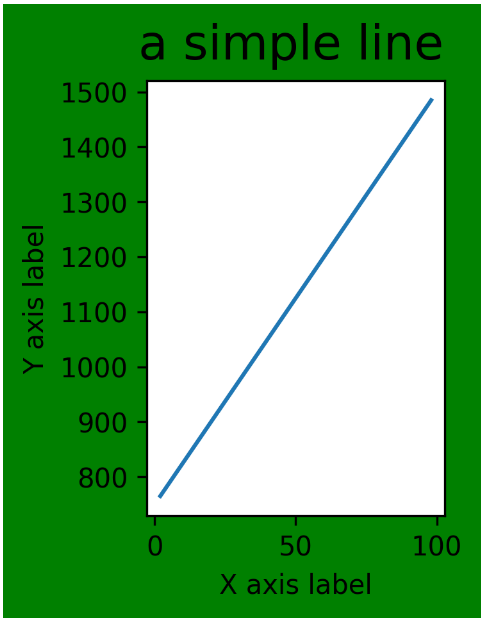
[43]: [<matplotlib.lines.Line2D at 0x2c0c9d41ac0>]



[17]:
```
fig.clear()
```

[44]:
```
fig = plt.figure(figsize=[2,3],facecolor='g',dpi=300)
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
plt.suptitle('a simple line',fontsize=18)
plt.plot(data1d,data2d)
```

[44]: [<matplotlib.lines.Line2D at 0x2c0c9da7520>]

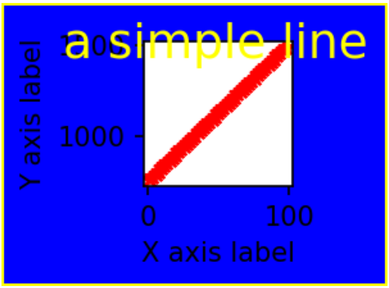


[21]:
```
fig.clear()
```

2.3 styles of line: 'ro': red color circle, 'k-': black dash line and so on,'bo-' : blue..

[47]:
```
fig = plt.
 ↪figure(figsize=[1,1],facecolor='b',dpi=150,edgecolor='yellow',linewidth=1)
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
plt.suptitle('a simple line',fontsize=18,color='yellow')
plt.plot(data1d,data2d,'r*')
```

[47]:
```
[<matplotlib.lines.Line2D at 0x2c0cb7b7fd0>]
```



[48]:
```
A=np.random.randn(10)
```

[49]:
```
A
```

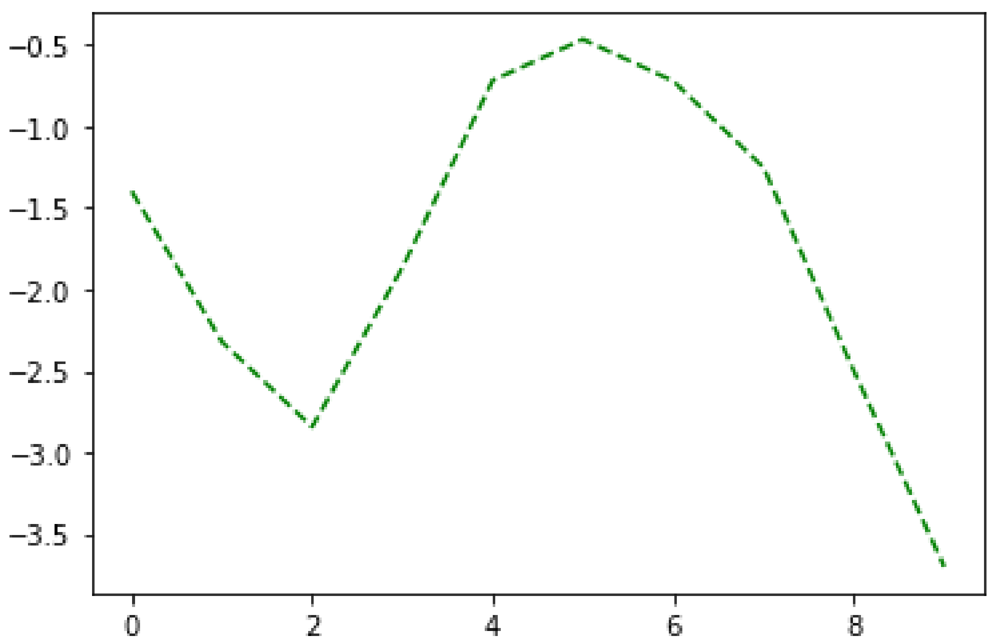
[49]:
```
array([ 0.34776917,  2.06475152, -0.14654592, -1.05925391,  0.22095229,
        0.00373421,  1.40625604,  1.26399893,  0.11223874, -0.88113699])
```

[50]:
```
A.cumsum()
```

[50]:
```
array([0.34776917, 2.41252069, 2.26597477, 1.20672086, 1.42767315,
       1.43140736, 2.8376634 , 4.10166233, 4.21390108, 3.33276408])
```

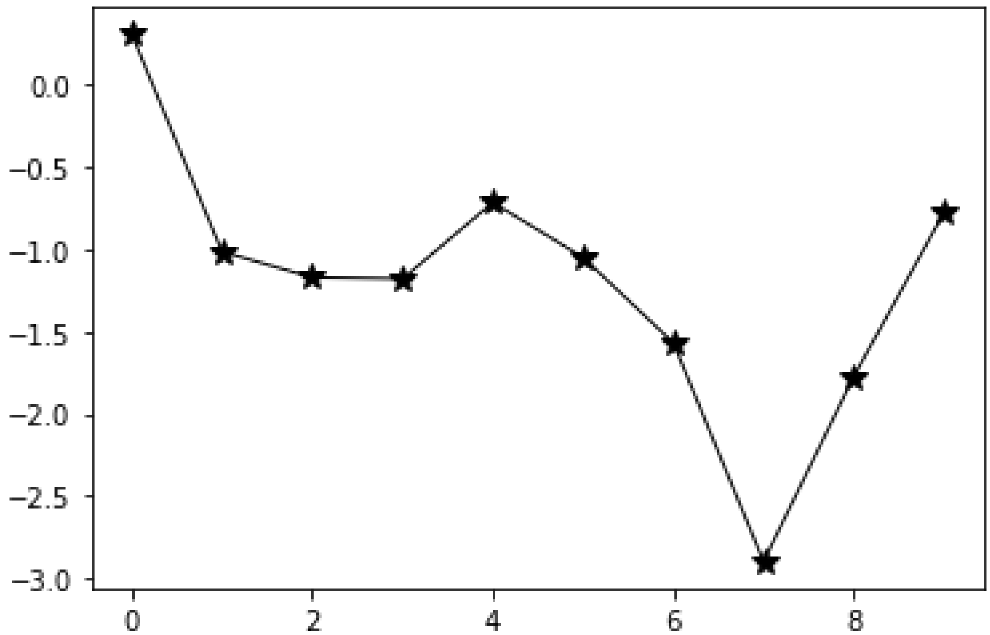
[52]:
```
plt.plot(np.random.randn(10).cumsum(), 'g--',)
```

[52]: [<matplotlib.lines.Line2D at 0x2c0cb84c610>]



[54]:
```
plt.plot(np.random.randn(10).cumsum(), 'k*-',linewidth=1, markersize=10)
```

[54]: [<matplotlib.lines.Line2D at 0x2c0cce83310>]



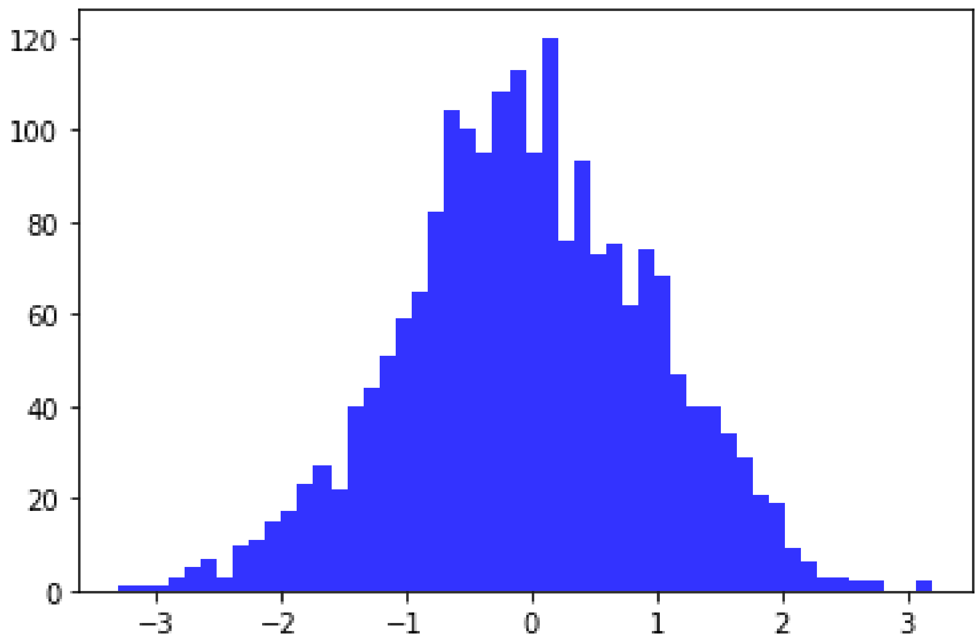
3 Histogram: an accurate graphical representation of the distribution of numerical data used for estimating the probability distribution of a continuous variable (quantitative variable) and was first introduced by Karl Pearson. A bar graph and needs "bins" of the underlying the range of values

3.1 alpha: A tuning parameter of color

[61]:
```
plt.hist(np.random.randn(2000), bins=50, color='b', alpha=0.8)
```

[61]:
```
(array([  1.,   1.,   1.,   3.,   5.,   7.,   3.,  10.,  11.,  15.,  17.,
         23.,  27.,  22.,  40.,  44.,  51.,  59.,  65.,  82., 104., 100.,
         95., 108., 113.,  95., 120.,  76.,  93.,  73.,  75.,  62.,  74.,
         68.,  47.,  40.,  40.,  34.,  29.,  21.,  19.,   9.,   6.,   3.,
          3.,   2.,   2.,   0.,   0.,   2.]),
 array([-3.28496213, -3.15531769, -3.02567326, -2.89602882, -2.76638438,
        -2.63673994, -2.5070955 , -2.37745107, -2.24780663, -2.11816219,
        -1.98851775, -1.85887331, -1.72922888, -1.59958444, -1.46994   ,
        -1.34029556, -1.21065112, -1.08100669, -0.95136225, -0.82171781,
        -0.69207337, -0.56242893, -0.4327845 , -0.30314006, -0.17349562,
        -0.04385118,  0.08579326,  0.21543769,  0.34508213,  0.47472657,
         0.60437101,  0.73401545,  0.86365988,  0.99330432,  1.12294876,
         1.2525932 ,  1.38223764,  1.51188207,  1.64152651,  1.77117095,
         1.90081539,  2.03045983,  2.16010426,  2.2897487 ,  2.41939314,
```

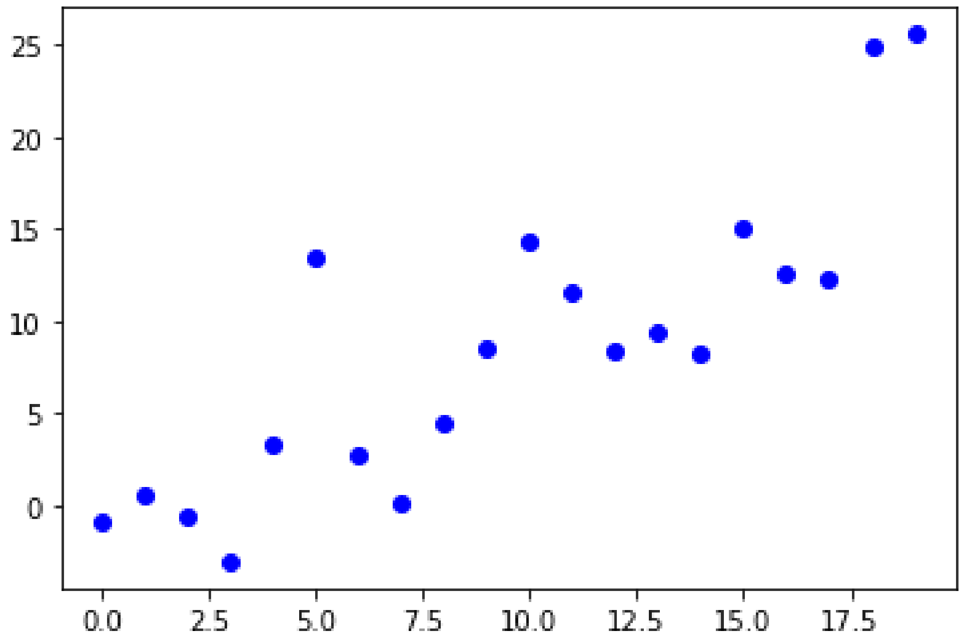
```
        2.54903758,  2.67868202,  2.80832645,  2.93797089,  3.06761533,
        3.19725977]),
 <BarContainer object of 50 artists>)
```



3.2 scatter plot

[63]:
```
plt.scatter(np.arange(20), np.arange(20) + 4* np.random.randn(20),c='blue')
```

[63]:
```
<matplotlib.collections.PathCollection at 0x2c0cd47c370>
```



3.3 The close() function in pyplot module of matplotlib library is used to close a figure window.

[64]:
```
plt.close()
```

[65]:
```
fig.clear()
```

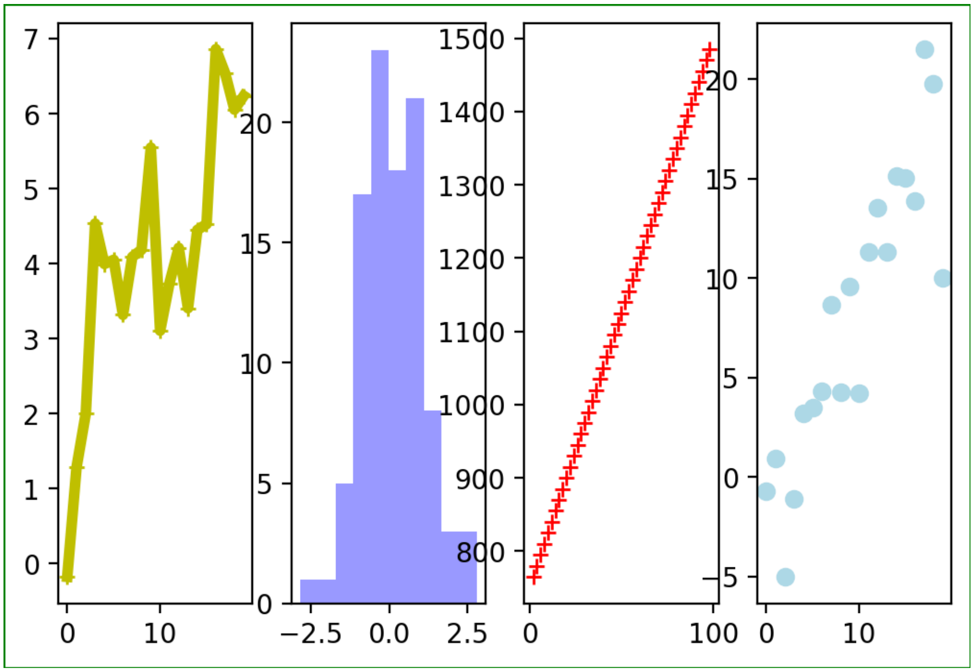
4 subplots: use of figure object: to add many plots in one figure

[ ]:
```
#fig1, ax = plt.subplots(2, 2)
```

[67]:
```
fig = plt.figure(facecolor='w',dpi=200,edgecolor='g',linewidth=1)
ax1 = fig.add_subplot(1, 4, 1)
ax2 = fig.add_subplot(1, 4, 2)
ax3 = fig.add_subplot(1, 4, 3)
ax4 = fig.add_subplot(1, 4, 4)
ax1.plot(np.random.randn(20).cumsum(), 'y+-',linewidth=4)
ax2.hist(np.random.randn(100), bins=10, color='b', alpha=0.4)
ax3.plot(data1d,data2d,'r+')
ax4.scatter(np.arange(20), np.arange(20) + 4* np.random.randn(20),c='lightblue')
fig.savefig('figures4.png')
fig.show()
```

```
C:\Users\SHARAN~1\AppData\Local\Temp/ipykernel_4536/555320540.py:11:
UserWarning: Matplotlib is currently using
```

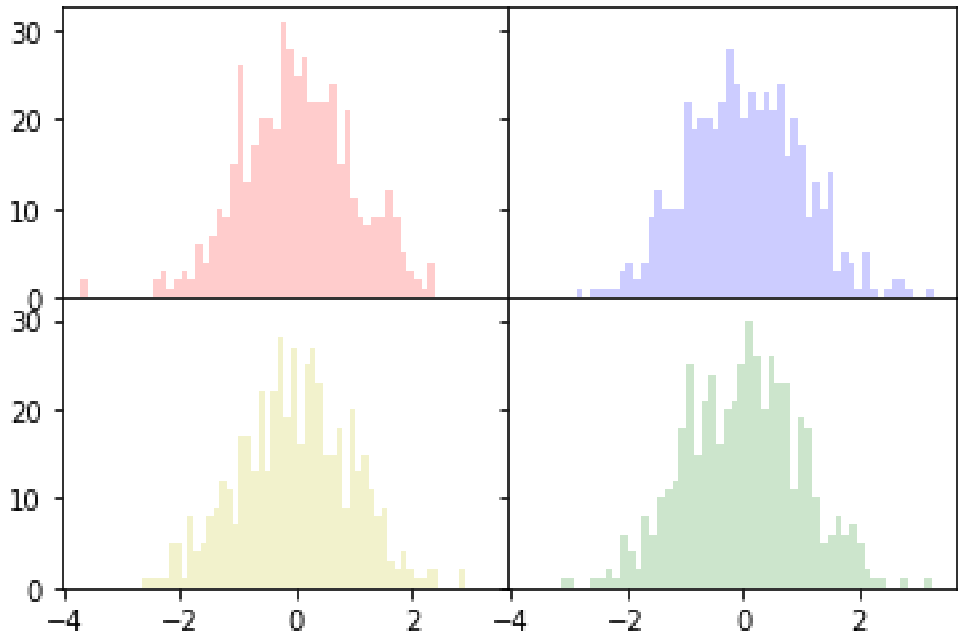
```
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```




```
[90]: fig.clear()
```

5 using single statement to have subplots

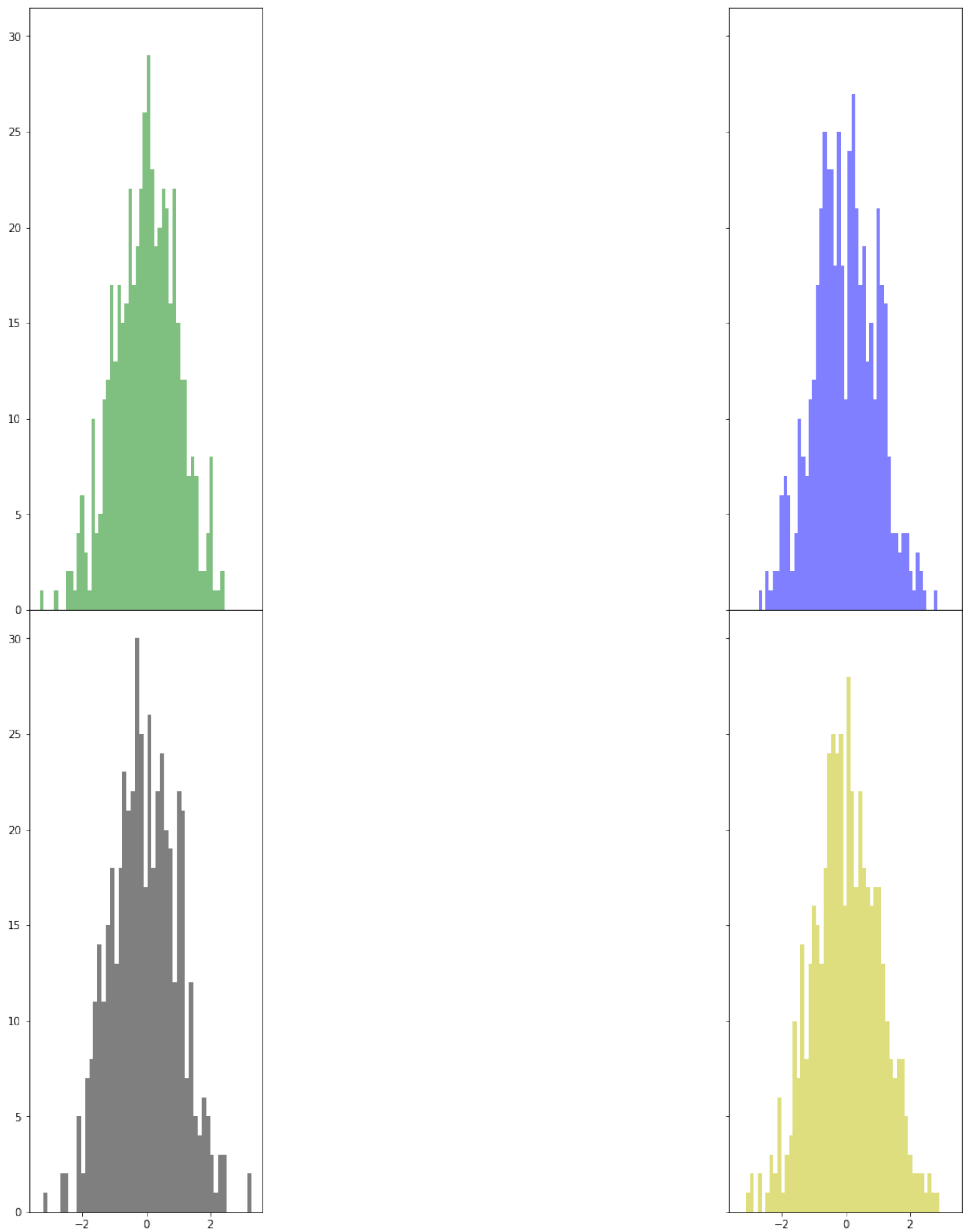
```
[69]: fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
l=['r','b','y','g']
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color=l[2*i+j], alpha=0.
 ↪2)
plt.subplots_adjust(wspace=0, hspace=0)
```



6 Parameters for subplots_adjust()

- left : This parameter is the left side of the subplots of the figure.
- right : This parameter is the right side of the subplots of the figure.
- bottom : This parameter is the bottom of the subplots of the figure.
- top : This parameter is the top of the subplots of the figure.
- wspace : This parameter is the amount of width reserved for space between subplots expressed as a fraction of the average axis width.
- hspace : This parameter is the amount of height reserved for space between subplots expressed as a fraction of the average axis height.

[76]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
c=['g','b','k','y']
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color=c[2*i+j], alpha=0.
 ↪5)
plt.subplots_adjust(left=2,right=4,top=4,wspace=2, hspace=0)
```

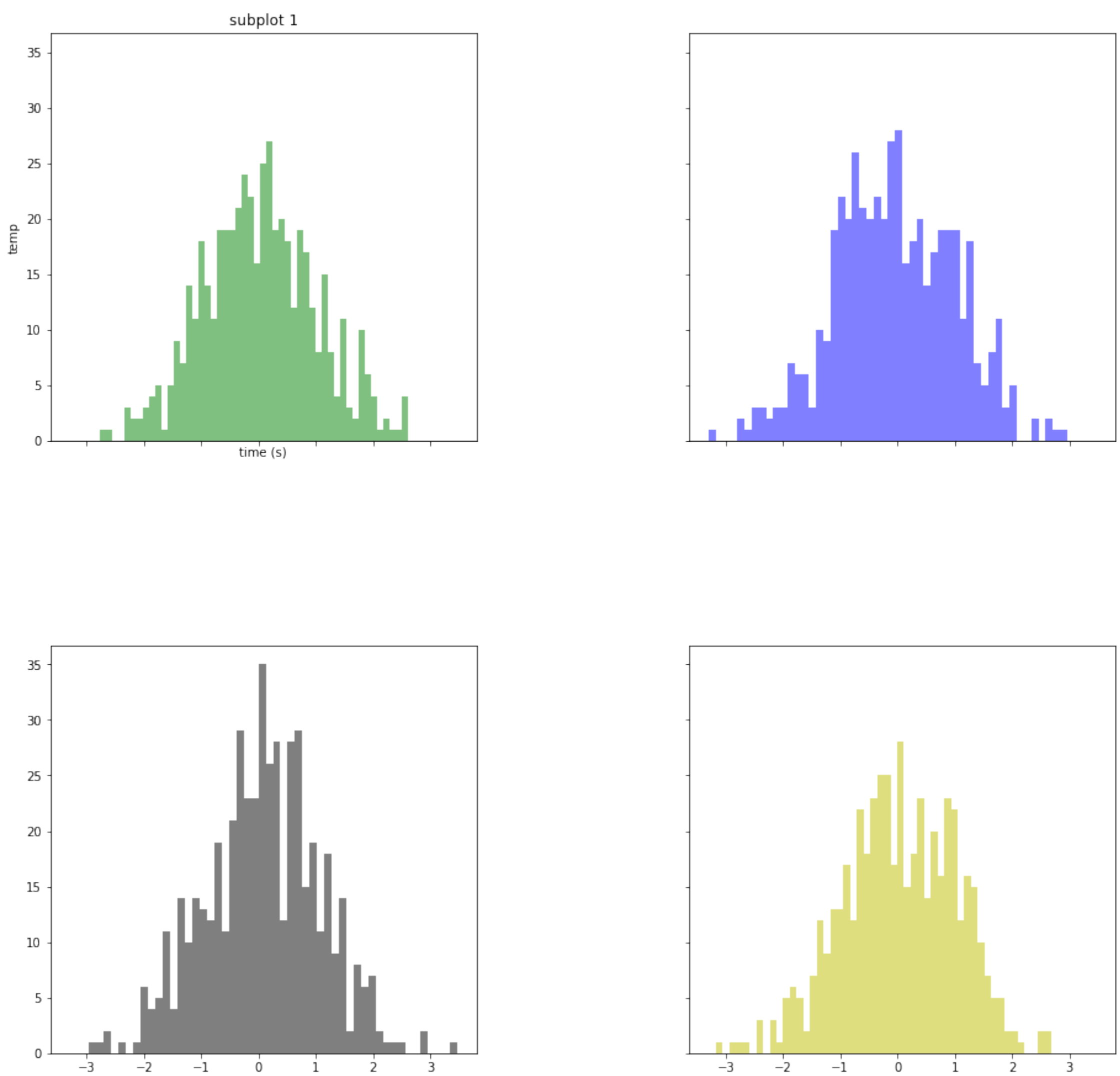


6.1 adding axis labels..

[77]:
```
fig.clear()
```

[78]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
c=['g','b','k','y']
for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(500), bins=50, color=c[2*i+j], alpha=0.
↪5)
plt.subplots_adjust(left=1,right=3,top=3,wspace=0.5, hspace=0.5)
axes[0,0].set_xlabel('time (s)')
axes[0,0].set_title('subplot 1')
axes[0,0].set_ylabel('temp')
```

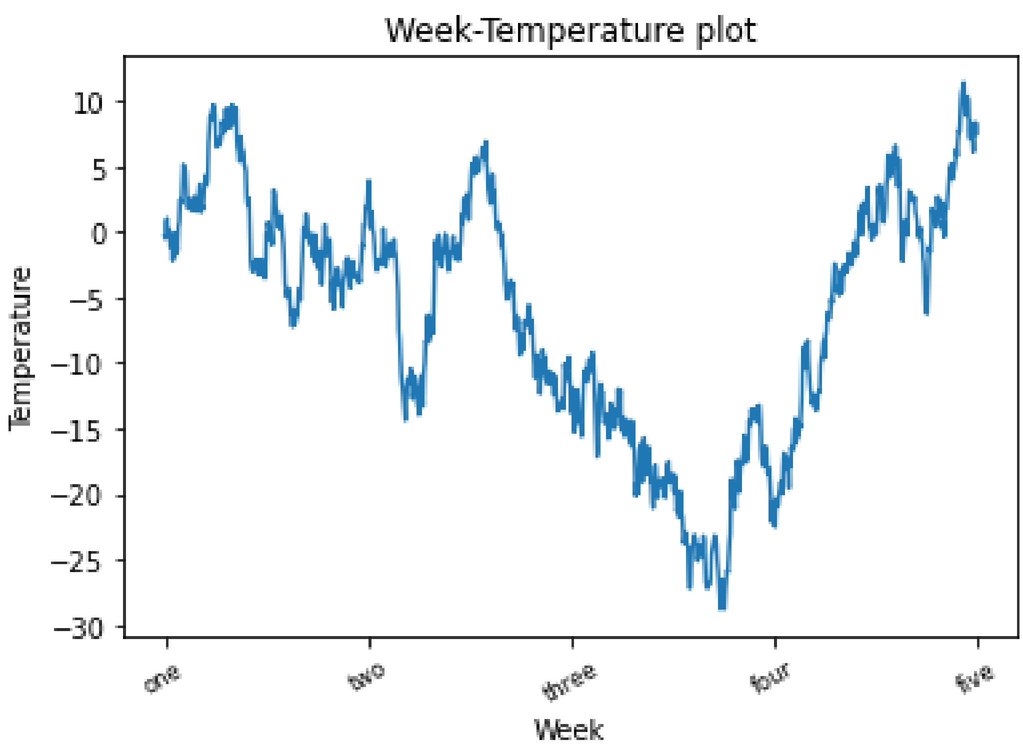
[78]:
```
Text(0, 0.5, 'temp')
```



7 single subplot with example of xtic and xticlabels alongwith

[79]:
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                            rotation=30, fontsize='small')
ax.set_title('Week-Temperature plot')
ax.set_xlabel('Week')
ax.set_ylabel('Temperature')
ax.plot(np.random.randn(1000).cumsum())
```

[79]: [<matplotlib.lines.Line2D at 0x2c0cd78e610>]



8 adding legends and mutltiple lines in same plot

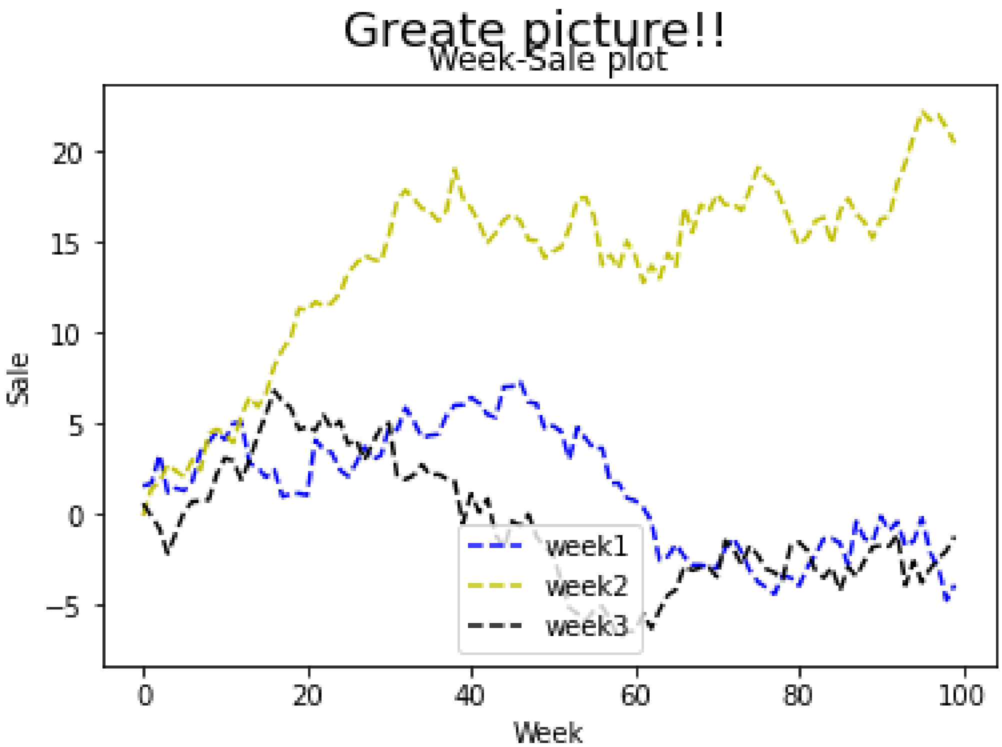
[81]:
```
plt.close()
fig.clear()
```

8.1 adding legend to the plot at the specified location using values like best/upper right/upper left/center/center right etc..,

- usage of operator ** with keyword arguments: take a dictionary of key-value pairs and unpack it into keyword arguments in a function call.

[82]:
```
from numpy.random import randn
fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
#ax.set_title('Week-Sale plot')
#ax.set_xlabel('Week')
#ax.set_ylabel('Sale')
ax.plot(randn(100).cumsum(), 'b--', label='week1')
ax.plot(randn(100).cumsum(), 'y--', label='week2')
ax.plot(randn(100).cumsum(), 'k--', label='week3')
ax.legend(loc='lower center')
props={'title':'Week-Sale plot', 'xlabel':'Week','ylabel':'Sale'}
ax.set(**props)
fig.suptitle('Greate picture!!',fontsize=18)
#fig.show()



.
```

[82]:
```
Text(0.5, 0.98, 'Greate picture!!')
```



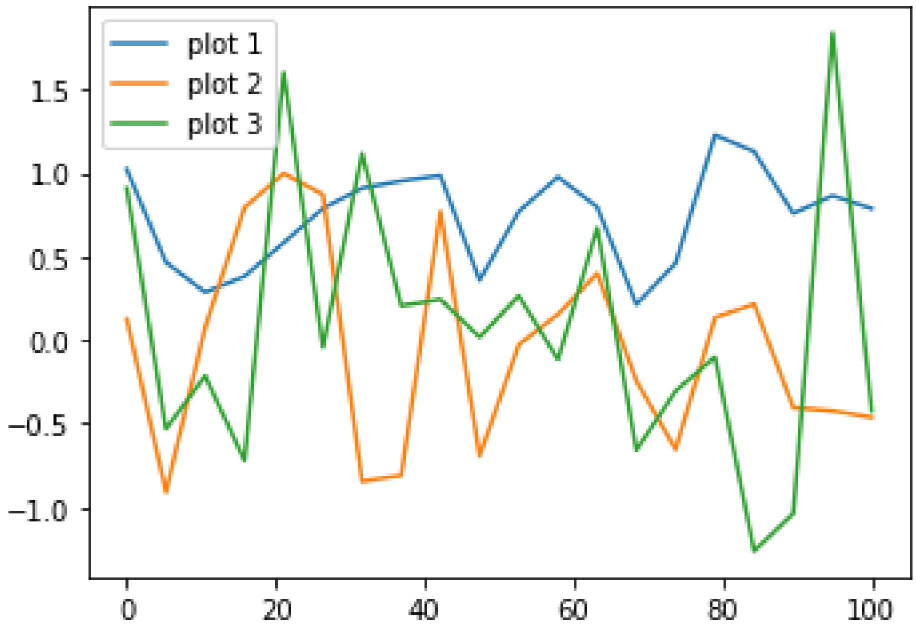
8.2 linspace: returns evenly spaced number over an interval

[68]:
```
x = np.linspace(0.0,150,20)
x
```

[68]:
```
array([  0.        ,   7.89473684,  15.78947368,  23.68421053,
        31.57894737,  39.47368421,  47.36842105,  55.26315789,
        63.15789474,  71.05263158,  78.94736842,  86.84210526,
        94.73684211, 102.63157895, 110.52631579, 118.42105263,
       126.31578947, 134.21052632, 142.10526316, 150.        ])
```

[69]:
```
# generate random data for plotting
x = np.linspace(0.0,100,20)

# now there's 3 sets of points using normal distribution with mentioned std dev␣
 ↪as scale
y1 = np.random.normal(0.6,0.3,size=20)  #mean (loc),sigma,size
y2 = np.random.normal(scale=0.5,size=20) # +ve Scale: stdev
y3 = np.random.normal(scale=0.8,size=20)

# plot the 3 sets
plt.plot(x,y1,label='plot 1')
```

```
plt.plot(x,y2, label='plot 2')
plt.plot(x,y3, label='plot 3')

# call with no parameters
plt.legend()

plt.show()
```



[7]:
```
y2.mean()
```

[7]: -0.05725550002123653

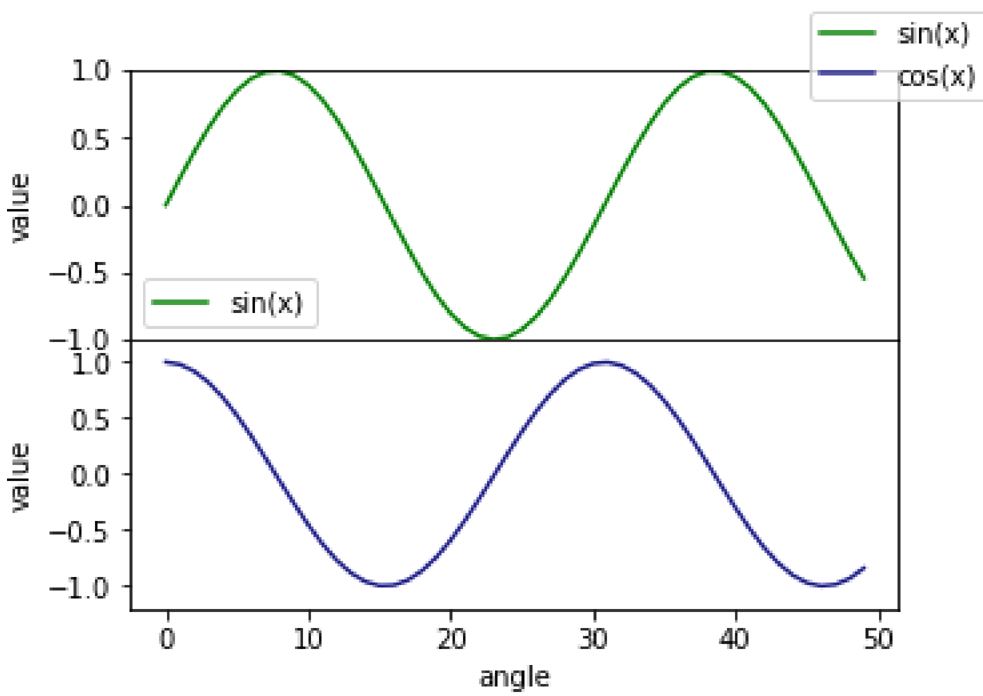
9 add_axes(): used to add multiple axes of the fig. consists of parameters:

- dimensions: [left, bottom, width, height]

[71]:
```
from turtle import color


fig = plt.figure()
axis1 = fig.add_axes([0.1, 0.5, 0.8, 0.4],
                   xticklabels=[], ylim=(-1.2, 1.2))
axis2 = fig.add_axes([0.1, 0.1, 0.8, 0.4],
                   ylim=(-1.2, 1.2))
```

```
x = np.linspace(0, 10)
axis1.plot(np.sin(x),label="sin(x)",color='g')
axis2.plot(np.cos(x),label="cos(x)",color='#0f0f80f0')
#axis2.set_xlabel('angle')
#axis2.set_ylabel('value')
#axis1.set_ylabel('value')
props={ 'xlabel':'angle','ylabel':'value'}
axis2.set(**props)
axis1.set(**props)
fig.legend()
axis1.legend()
#plt.text(0,0,'Here',fontsize=10,family='monospace',color='blue',style='italic')
plt.show()
```

```
c:\Users\Sharanjit Kaur\AppData\Local\Programs\Python\Python39\lib\site-
packages\IPython\core\pylabtools.py:134: UserWarning: This figure includes Axes
that are not compatible with tight_layout, so results might be incorrect.
  fig.canvas.print_figure(bytes_io, **kw)
```

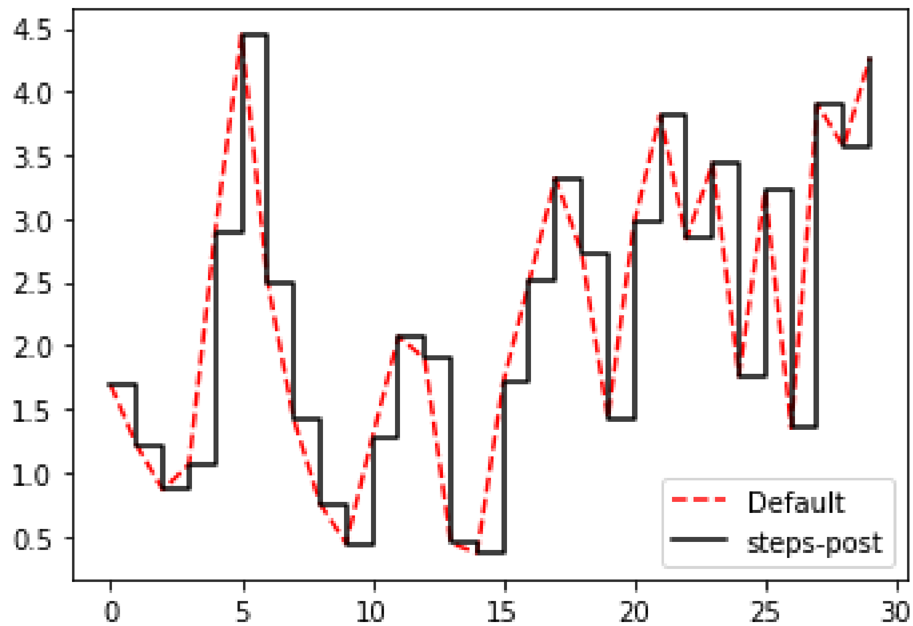


[72]:
```
plt.close()
```

10 Example on setting and drawing

[74]:
```
data = np.random.randn(30).cumsum()
plt.plot(data, 'r--', label='Default')
plt.plot(data, 'k-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')

#plt.legend(loc='upper center')
```

[74]: <matplotlib.legend.Legend at 0x21101076670>

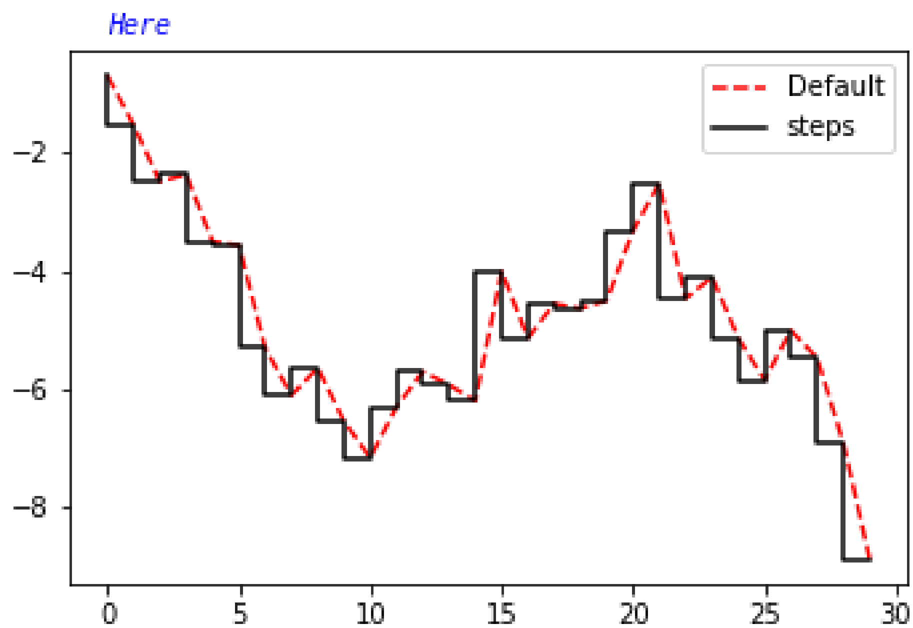


11 annotating within plot, family indicatesFONTNAME:[] 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace']

- rcParam: Each time Matplotlib loads, it defines a runtime configuration (rc) containing the default styles for every plot element created. This configuration can be adjusted at any time using the plt.
- text(x,y,data): text in axis coords by default ((0, 0) is lower-left and (1, 1) is upper-right).

[76]:
```
data = np.random.randn(30).cumsum()
plt.rcParams["figure.figsize"] = [5.00, 3.50]
plt.rcParams["figure.autolayout"] = True
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(data, 'r--', label='Default')
```

```
ax.plot(data, 'k-', drawstyle='steps', label='steps')
ax.legend(loc='best')
ax.text(0,0,'Here',fontsize=10,family='monospace',color='blue',style='italic')
plt.show()
#plt.legend(loc='upper center')
```



[ ]:
```
?plt.text
```

12 Annotations and Drawing on a Subplot

13 method asof(): used to get value at the specified index value (where). In case index value is missing then value at just before index value is returned. Sim, for dataframe()

- asof() returns single or multiple values with exact match. Where uses condition to get elements meeting that condition

[77]:
```
s=pd.Series(np.arange(10)*2,index=range(0,10,1))
s
```

[77]:
```
0    0
1    2
2    4
3    6
4    8
```

```
5     10
6     12
7     14
8     16
9     18
dtype: int32
```

[78]:
```
s.asof(7)
```

[78]:
```
14
```

13.1 in case index mentioned is larger than last index in series then return valu at latgest index. if index -1 is passed then NAN is returned

[81]:
```
s.asof(17)
```

[81]:
```
nan
```

[84]:
```
s.where(s > 15,0)
```

[84]:
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8     16
9     18
dtype: int32
```

[66]:
```
s.asof([19,-1])  #-1 is less than first index so NaN is returned
```

[66]:
```
 19    18.0
-1      NaN
dtype: float64
```

13.2 plotting time series data

[2]:
```
data = pd.read_csv('spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']
spx
```

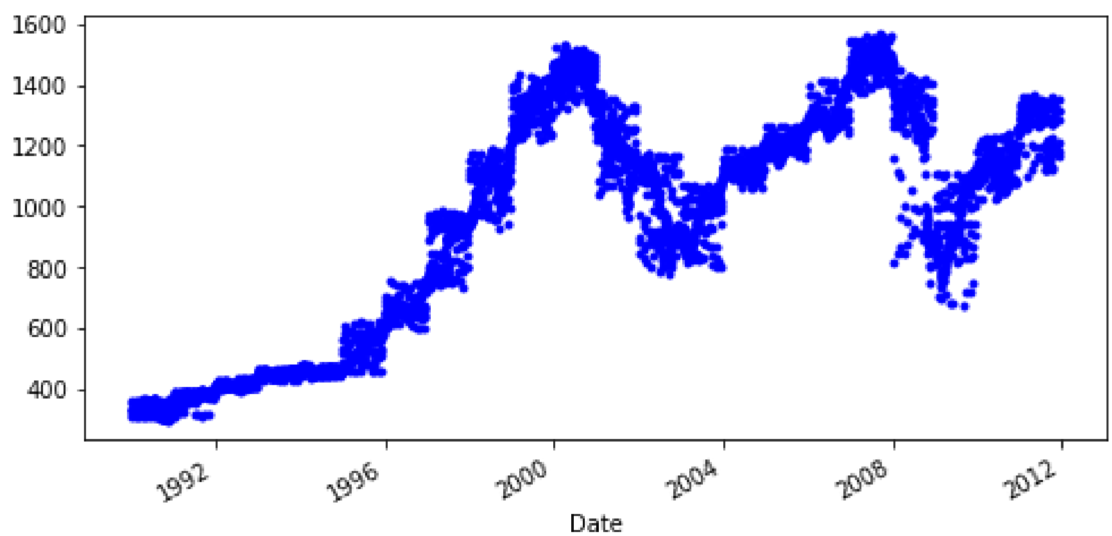
[2]:
```
Date
1990-01-02     328.79
1990-02-02     330.92
1990-05-02     331.85
```

```
1990-06-02     329.66
1990-07-02     333.75
                …
2011-10-10    1194.89
2011-11-10    1195.54
2011-12-10    1207.25
2011-10-13    1203.66
2011-10-14    1224.58
Name: SPX, Length: 5472, dtype: float64
```

[3]:
```
s1=spx.sort_index()
```

[11]:
```
from datetime import datetime
fig = plt.figure()
plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True
ax = fig.add_subplot(1, 1, 1)
s1.plot(ax=ax, style='b.')
```

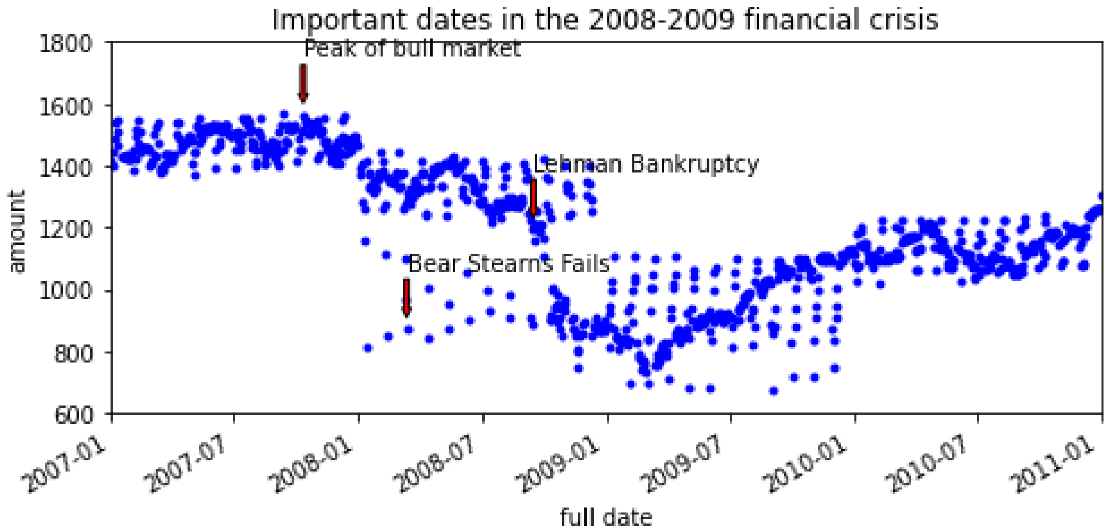
[11]: <AxesSubplot:xlabel='Date'>



[12]:
```
crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]
for date, label in crisis_data:
    print(s1.asof(date)+80)
```

```
1642.47
950.74
1272.7
```

[14]:
```
from datetime import datetime



fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
s1.plot(ax=ax, style='b.')

crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]
#xy:arrow position, xytext:text position
for date, label in crisis_data:
    ax.annotate(label, xy=(date, s1.asof(date)+40),
                xytext=(date, s1.asof(date) + 250),
                arrowprops=dict(facecolor='red', headwidth=4, width=2,
                                headlength=4),
                horizontalalignment='left', verticalalignment='top')
propsyahoo={'xlim':['1/1/2007', '1/1/2011'], 'ylim':[600, 1800],'title':
 ↪'Important dates in the 2008-2009 financial crisis','ylabel':
 ↪'amount','xlabel':'full date'}
#ax.set_xlim(['1/1/2007', '1/1/2011'])
#ax.set_ylim([600, 1800])
#ax.set_title('Important dates in the 2008-2009 financial crisis')

ax.set(**propsyahoo)
fig.savefig('yahoo.jpg',dpi=300,bbox_inches='tight')
```



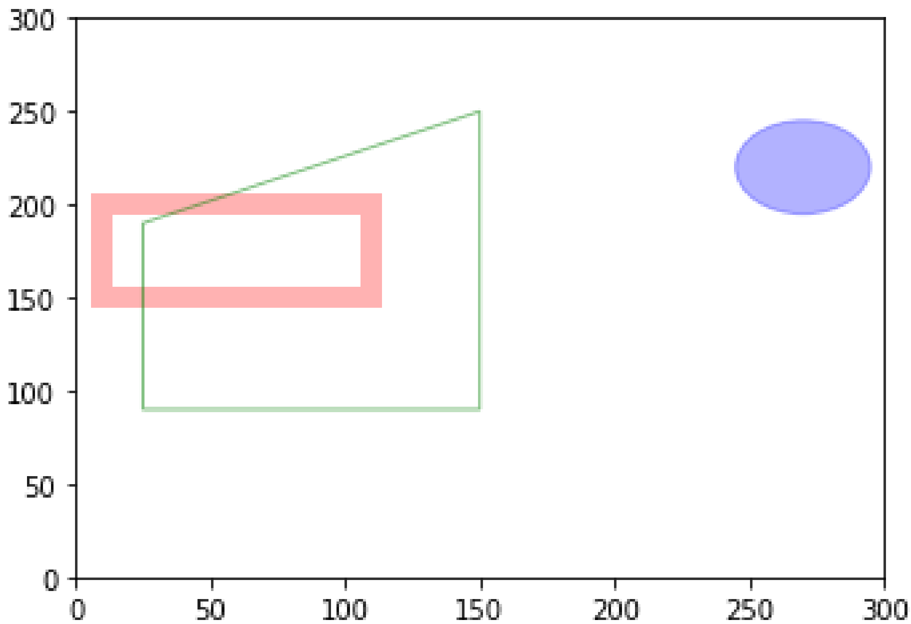
14 drawing shapes using patch object

matplotlib.patches.Rectangle(xy, width, height, angle=0.0, **kwargs)

[51]:
```
fig.clear()
```

[97]:
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

rect = plt.Rectangle((10, 150), 100, 50,  alpha=0.3,fill=False, fc='b',
                        ec ='r',lw = 8)
circ = plt.Circle((270, 220), 25, color='b', alpha=0.3)
pgon = plt.Polygon([[25, 90], [25, 190],[150,250],[150, 90]],
                   color='g', alpha=0.5,fill=False)

ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
plt.xlim([0, 300])
plt.ylim([0, 300])
```

[97]: (0.0, 300.0)



15 scatterplot using iris data and storing fig as svg:Scalable Vector Graphics used for sharing graphics contents on the Internet.

[15]:
```
df = pd.read_csv('Iris1.csv')
df.columns
```

[15]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

[16]:
```
df.Species.unique()
```

[16]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

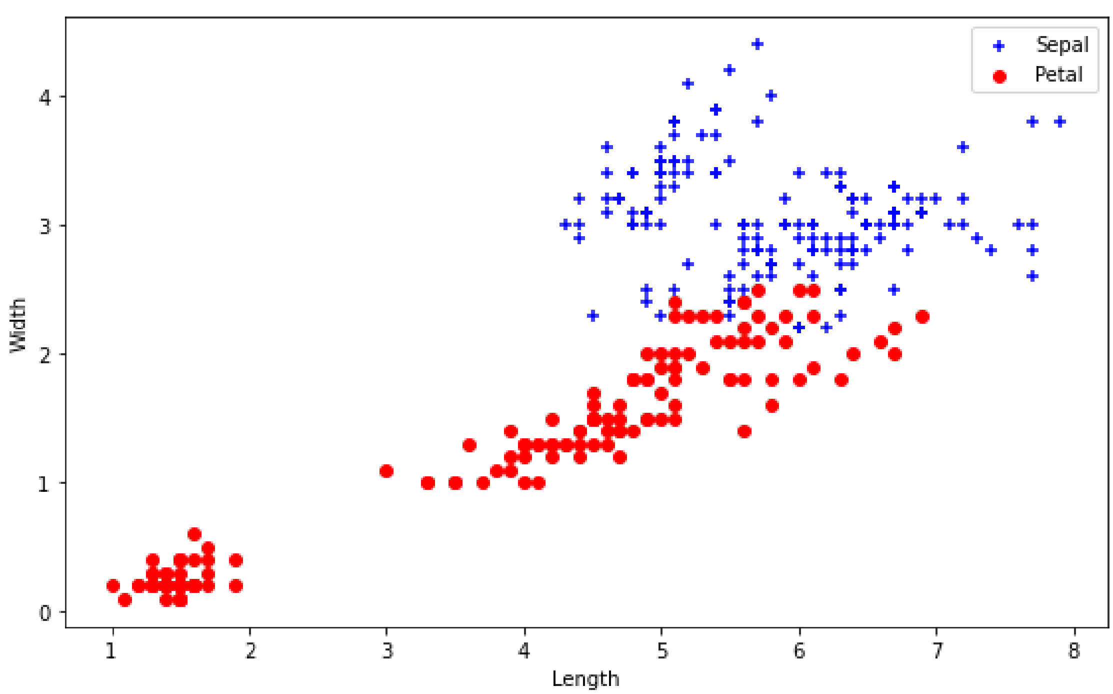
[17]:
```
df.head(20)
```

[17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa

8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa

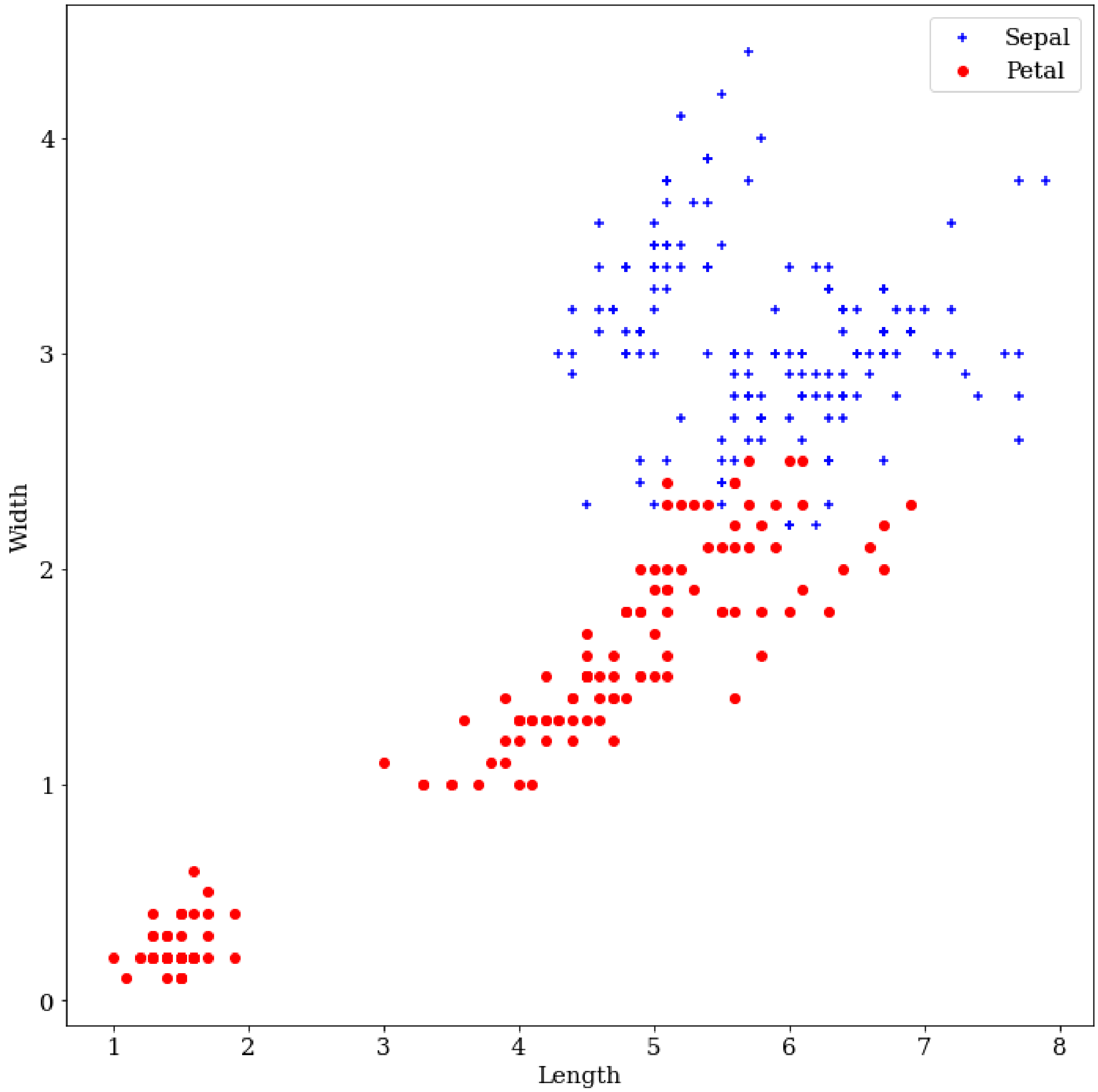
[18]:
```
fig, ax = plt.subplots(1, figsize=(8, 5))
ax.scatter(x = df['SepalLengthCm'], y =␣
 ↪df['SepalWidthCm'],color='b',marker='+',label='Sepal')
ax.set_xlabel("Length")
ax.set_ylabel("Width")
ax.scatter(x = df['PetalLengthCm'], y =␣
 ↪df['PetalWidthCm'],color='r',marker='o',label='Petal')
plt.legend()
plt.savefig("scatter-iris.svg",dpi=400,bbox_inches='tight')
plt.show()
```



16 setting runtime configuration parameters for plt for all the plots plotted unless mention others explicitly

[19]:
```
plt.rc('figure', figsize=(10, 10))
font_options = {'family' : 'serif',
                'weight' : 'normal', 'size':15}
plt.rc('font', **font_options)
```

[20]:
```
fig, ax = plt.subplots(1)
ax.scatter(x = df['SepalLengthCm'], y =␣
 ↪df['SepalWidthCm'],color='b',marker='+',label='Sepal')
ax.set_xlabel("Length")
ax.set_ylabel("Width")
ax.scatter(x = df['PetalLengthCm'], y =␣
 ↪df['PetalWidthCm'],color='r',marker='o',label='Petal')
plt.legend()
plt.savefig("scatter-iris.svg",dpi=400,bbox_inches='tight')
plt.show()
```



[77]:
```
plt.close('all')
```

17 Plotting with Pandas using iris data df

[23]:
```
df.columns
```

[23]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

[21]:
```
len(df[['SepalLengthCm','SepalWidthCm','PetalLengthCm']])
```

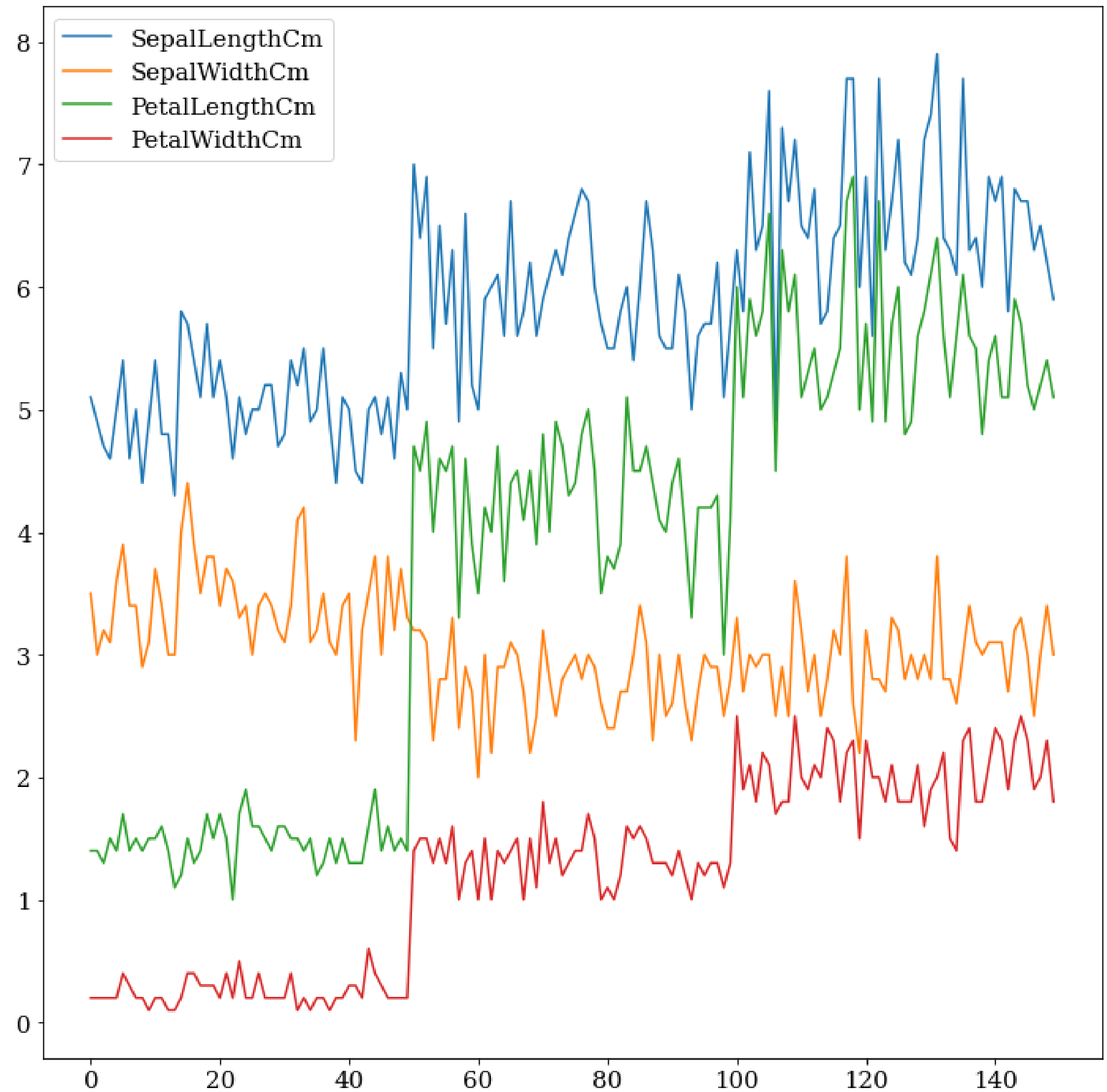
[21]: 150

17.1 kind=['area', 'bar', 'barh', 'density', 'hist', 'kde', 'line', 'pie']

alpha: opacity, styile:tic marks

[24]:
```
df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].plot()
```

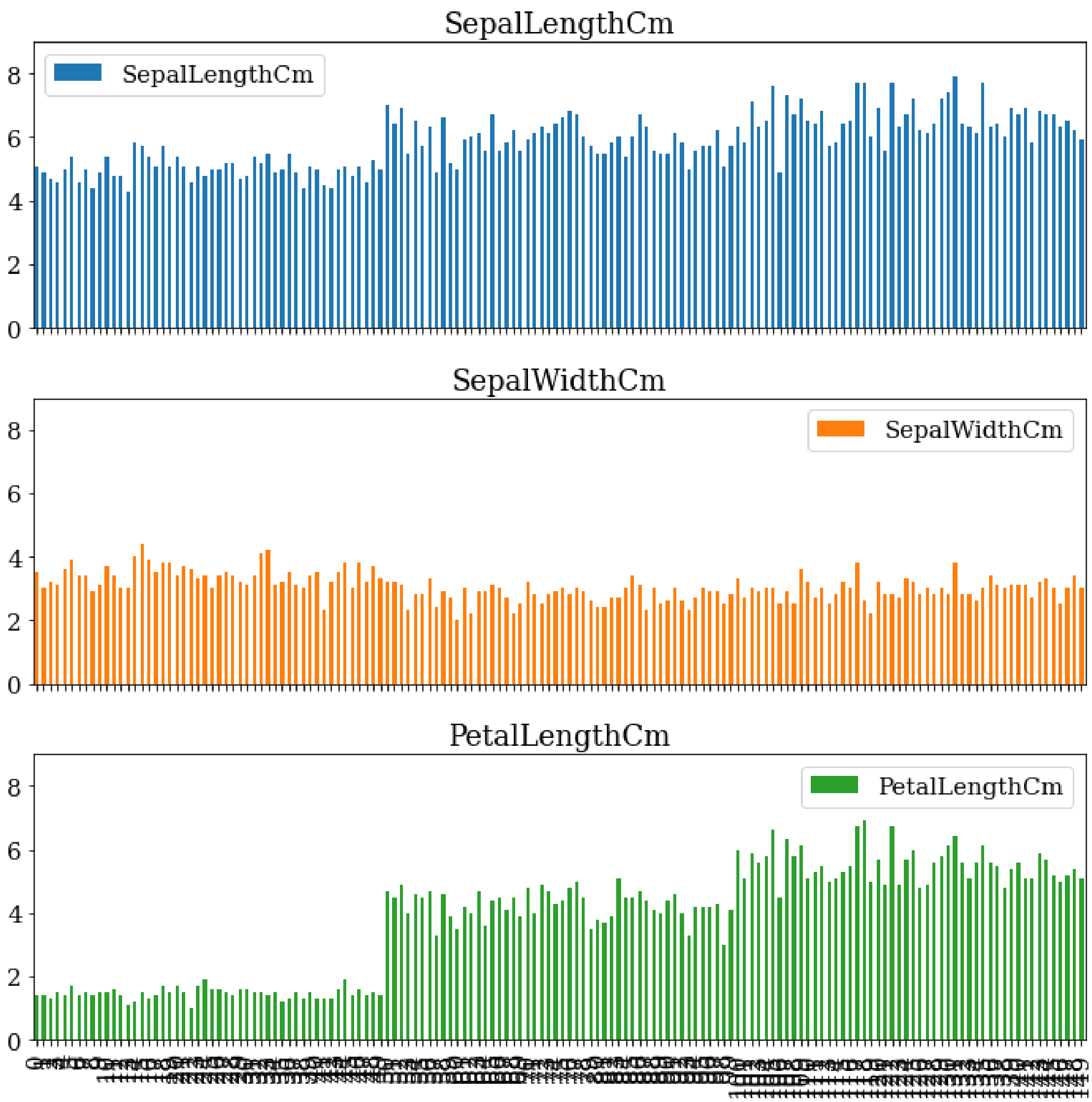
[24]: <AxesSubplot:>



17.2 subplots=True indicates each plotting in seperate subplot

[25]:
```
#df[['SepalLengthCm','SepalWidthCm','PetalLengthCm']].
↪plot(xlim=[1,160],ylim=[0,9])
df[['SepalLengthCm','SepalWidthCm','PetalLengthCm']].
↪plot(style='+',xlim=[1,160],ylim=[0,9],kind='bar',subplots=True)
```

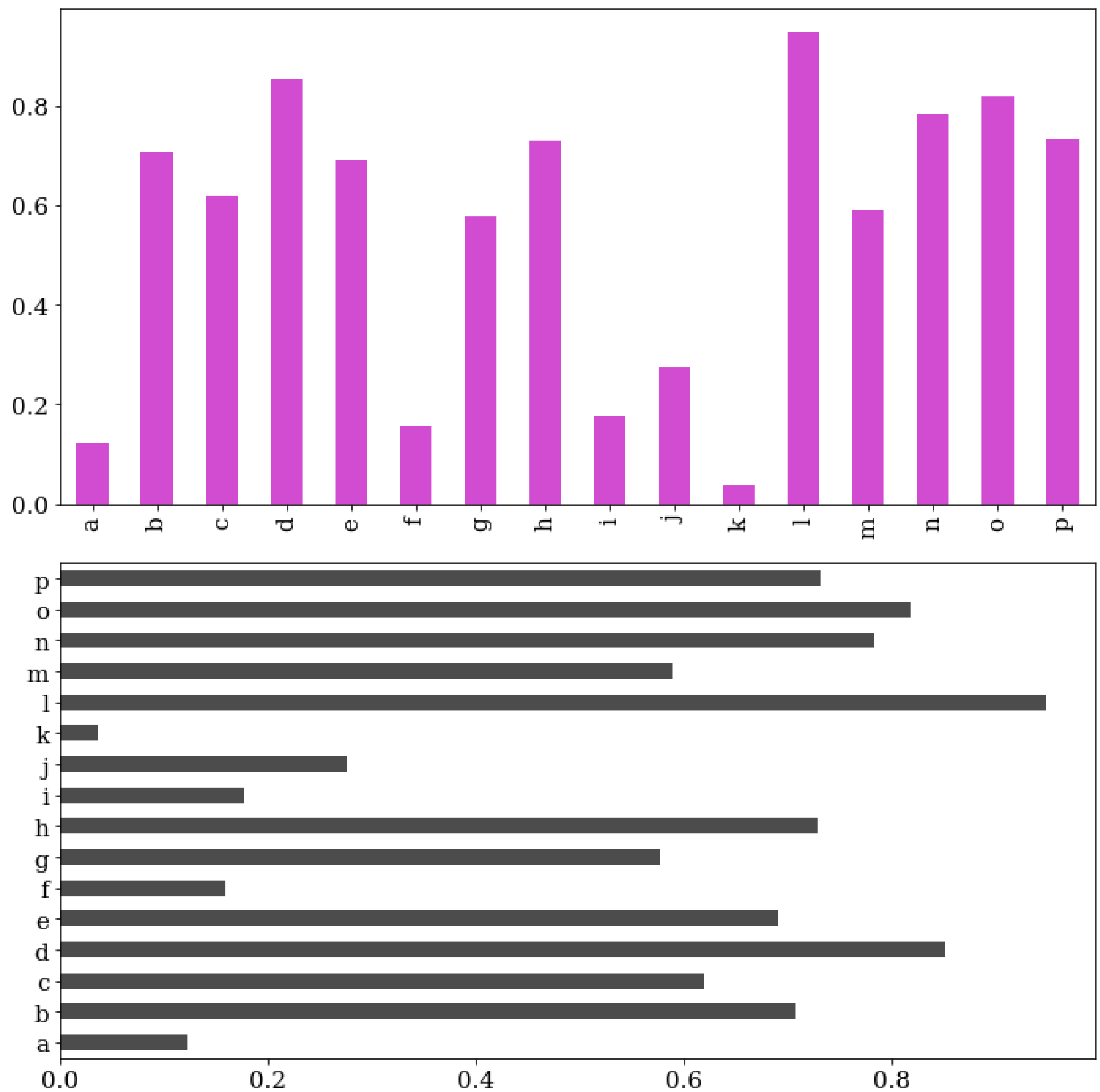
[25]:
```
array([<AxesSubplot:title={'center':'SepalLengthCm'}>,
       <AxesSubplot:title={'center':'SepalWidthCm'}>,
       <AxesSubplot:title={'center':'PetalLengthCm'}>], dtype=object)
```



18 barplot

[29]:
```
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='m', alpha=0.7) #vertical bar
data.plot.barh(ax=axes[1], color='k', alpha=0.7) #horizontal
```

[29]: <AxesSubplot:>



[30]:
```
import os
os.getcwd()
```

[30]: 'd:\\cs(h)Vsem Data analysis and visulaization 2021\\programs\\pandas'

[ ]:
```
xlsdf1 = pd.read_csv('titanictrain.csv')
xlsdf1
```

18.1 compare number of male and female passengers

[32]:
```
S=xlsdf1['Sex']
D=S.value_counts()
type(D)
```

[32]: pandas.core.series.Series

[72]:
```
D
```

[72]:
```
male      577
female    314
Name: Sex, dtype: int64
```

[33]:
```
D["male"]
```

[33]: 577

[34]:
```
len(D.index)
```

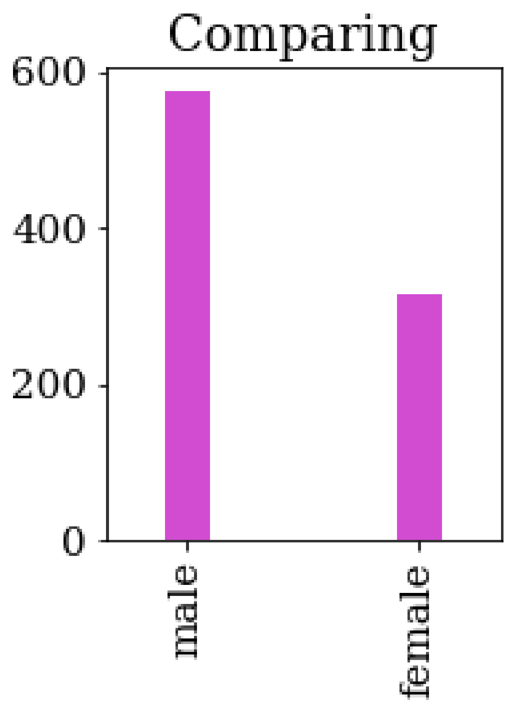
[34]: 2

[35]:
```
D.index[0]
```

[35]: 'male'

[51]:
```
plt.rc('figure', figsize=(3, 4))
plt.title("Comparing")
D.plot.bar(color='m', alpha=0.7,width=0.2) #vertical bar
```

[51]: <AxesSubplot:title={'center':'Comparing'}>



18.2 comparing Min and Max age of Male/Femal passengers

[65]:
```
maleageMax=xlsdf1[xlsdf1['Sex']=='male'].Age.max()
femaleageMax=xlsdf1[xlsdf1['Sex']=='female'].Age.max()
print(maleageMax,femaleageMax)
maleageMin=xlsdf1[xlsdf1['Sex']=='male'].Age.min()
femaleageMin=xlsdf1[xlsdf1['Sex']=='female'].Age.min()
print(maleageMin,femaleageMin)
```

```
80.0 63.0
0.42 0.75
```

[68]:
```
DF=pd.DataFrame(np.zeros(4).
  ↪reshape(2,2),columns=['Max','Min'],index=['Male','Female'])
DF.iloc[0]=[maleageMax,maleageMin]
DF.iloc[1]=[femaleageMax,femaleageMin]
```

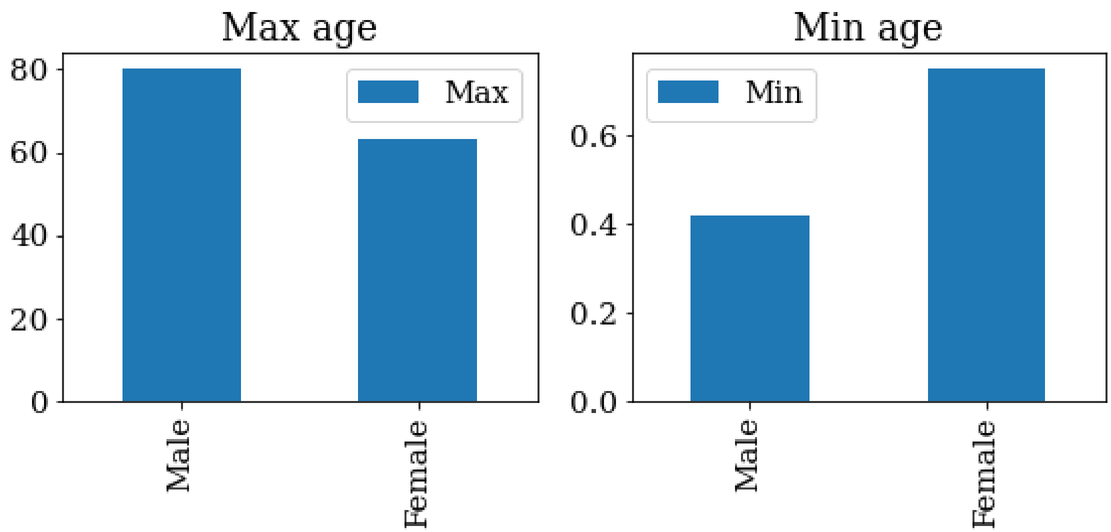
[69]:
```
DF
```

[69]:

	Max	Min
Male	80.0	0.42
Female	63.0	0.75

[75]:
```
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
DF[['Max']].plot.bar(ax=axs[0])
axs[0].set_title('Max age')
DF[['Min']].plot.bar(ax=axs[1])
axs[1].set_title('Min age')
```

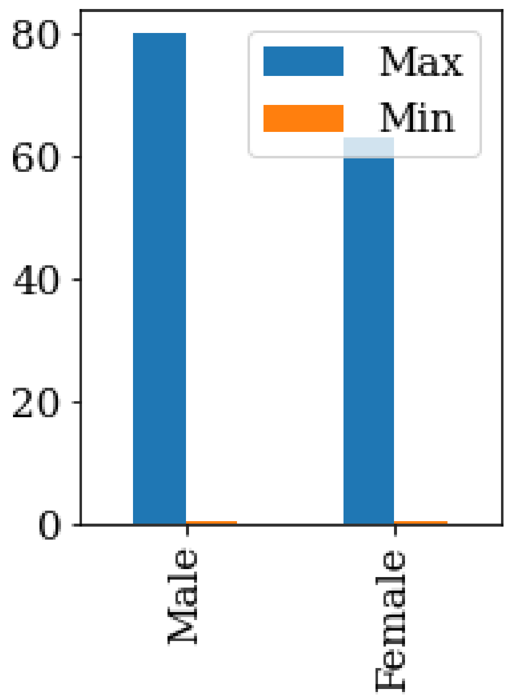
[75]: Text(0.5, 1.0, 'Min age')



18.3 plots all column in bar chart

[79]:
```
DF.plot.bar()
```

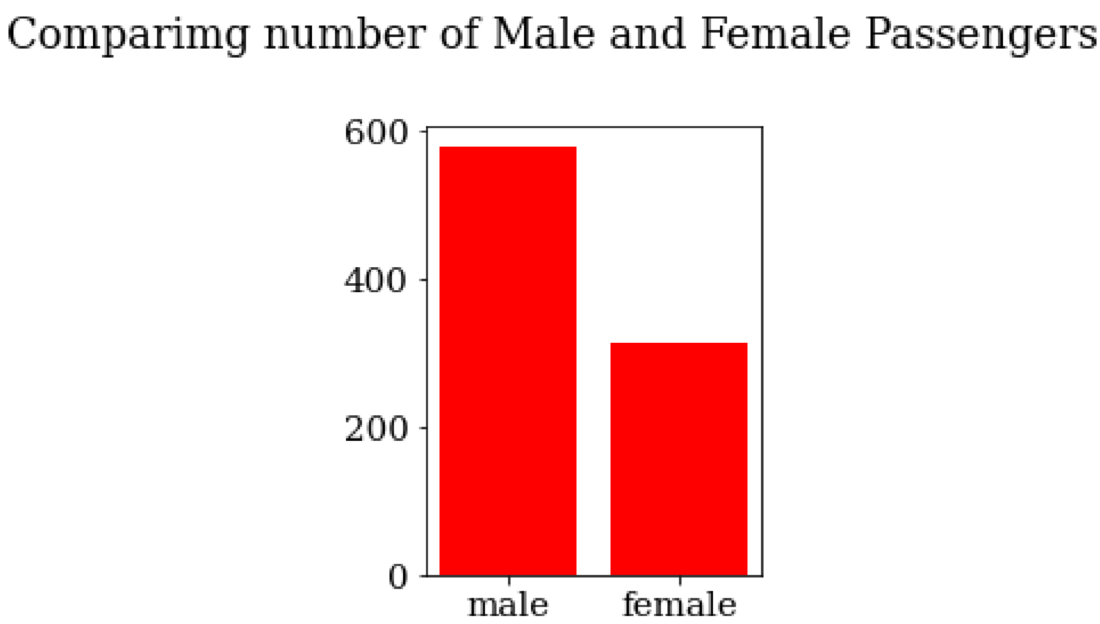
[79]: <AxesSubplot:>



18.4 Example for plotting list values

[78]:
```
data = {D.index[0]: D[0], D.index[1]: D[1]}
names = list(data.keys())
values = list(data.values())
plt.bar(names, values,color='r')
plt.suptitle('Comparimg number of Male and Female Passengers')
```

[78]:
```
Text(0.5, 0.98, 'Comparimg number of Male and Female Passengers')
```



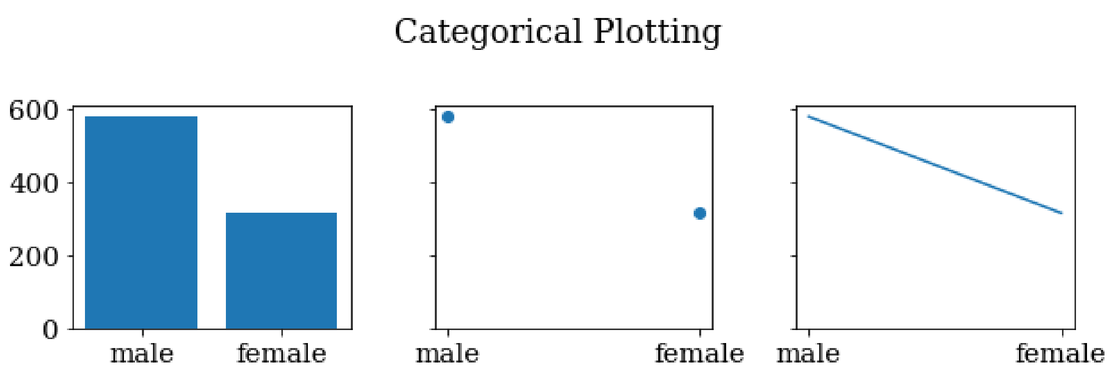
[74]:
```
D.index.values
```

[74]: array(['male', 'female'], dtype=object)

18.5 using D series for plotting multiple plots

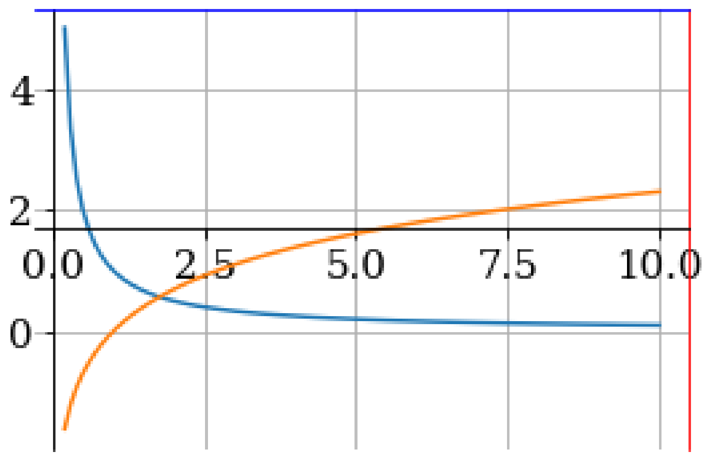
[80]:
```
names = D.index.values
values = D.values

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```

[80]: Text(0.5, 0.98, 'Categorical Plotting')



19 axes spines

[86]:
```
x = np.linspace(0.2,10,100)
fig, ax = plt.subplots(figsize=(4,4,))
ax.plot(x, 1/x)
ax.plot(x, np.log(x))
ax.set_aspect('equal')
ax.grid(True, which='both')

# set the x-spine (see below for more info on `set_position`)
ax.spines['left'].set_position('zero')

# turn off the right spine/ticks
ax.spines['right'].set_color('none')
ax.yaxis.tick_left()

# set the y-spine
ax.spines['bottom'].set_position('center')

# turn off the top spine/ticks
ax.spines['top'].set_color('blue')
ax.spines['right'].set_color('red')
ax.xaxis.tick_bottom()
```


4
2
0
0.0
2.5
5.0
7.5
10.0