# **Applied Plotting, Charting & Data Representation in Python**

**MATPLOT LIB**

1: >>> import matplotlib as mpl

>>> mpl.get\_backend()

'TkAgg'

2: import matplotlib.pyplot as plt

plt.plot(3,2)

plt.show()

3:

plt.plot(2,3,’.’)

plt.show()

4:

#setting backends

>>> from matplotlib.backends.backend\_agg import FigureCanvasAgg

>>> from matplotlib.figure import Figure

>>> fig=Figure()

>>> canvas=FigureCanvasAgg(fig)

>>> ax=fig.add\_subplot(111)

>>> ax.plot(3,2,'.')

[<matplotlib.lines.Line2D object at 0x02D83570>]

>>> canvas.print\_png('test.png')

5

# setting x axis and y axis limits ax.axis(x min,x max,y min, y max)

plt.figure()

plt.plot(3,2,’o’)

ax=plt.gca() #get current access

ax.axis([0,6,0,10]) # setting x axis and y axis limits ax.axis(x min,x max,y min, y max)

6:

#display 2 plots

plt.figure()

plt.plot(1.5,1.5,’o’)

plt.plot(2,2,’o’)

plt.plot(2.5,2.5,’o’)

plt.show()

7:

#to view child objects

>>> ax=plt.gca()

>>> ax.get\_children()

[<matplotlib.spines.Spine object at 0x0B354590>, <matplotlib.spines.Spine object at 0x0B349110>, <matplotlib.spines.Spine object at 0x0B349410>, <matplotlib.spines.Spine object at 0x0B349070>, <matplotlib.axis.XAxis object at 0x0B37C070>, <matplotlib.axis.YAxis object at 0x0B37CDB0>, Text(0.5,1,''), Text(0,1,''), Text(1,1,''), <matplotlib.patches.Rectangle object at 0x007CBAB0>]

>>>

**Scatterplots**

**1:**

**#**simple scatter plot

Import numpy as np

x=np.arange(10) #array with range

y=x

plt.scatter(x,y)

plt.show()

2:

#setting different colors to scatter plot

>>> colors=['green']\*(len(x)-1)

>>> colors.append('red')

>>> colors

['green', 'green', 'green', 'green', 'green', 'green', 'green', 'green', 'green', 'red']

>>> plt.scatter(x,y,s=100,c=colors) #s =size of data point(circle)

<matplotlib.collections.PathCollection object at 0x005D1790>

>>> plt.show()

3:#know zip and unzip

>>> zip\_generator=zip(list(range(10)),list(range(10,20)))

>>> x,y=zip(\*zip\_generator)

>>> x

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

>>> y

(10, 11, 12, 13, 14, 15, 16, 17, 18, 19)

>>>

4:

>>> plt.figure()

<matplotlib.figure.Figure object at 0x0B435970>

>>> plt.scatter(x[:2],y[:2],s=100,c='red',label='tall students')

<matplotlib.collections.PathCollection object at 0x02A256B0>

>>> plt.scatter(x[2:],y[2:],s=100,c='blue',label='short students')

<matplotlib.collections.PathCollection object at 0x02A25910>

>>> plt.show()

>>>

5:

zip\_generator=zip(list(range(1,10)),list(range(2,20,2)))

x,y=zip(\*zip\_generator)

plt.figure()

plt.scatter(x[:2],y[:2],s=100,c='red',label='tall students')

plt.scatter(x[2:],y[2:],s=100,c='blue',label='short students')

plt.xlabel('x -axis')

plt.ylabel('y axis')

plt.title('y= 2x equation')

plt.legend(loc=4,frameon=False,title=’Legend’) #plt.legend()

plt.show()

6:

linear\_data=np.array([1,2,3,4,5])

quadric\_data=linear\_data\*\*2

plt.figure()

plt.plot(linear\_data,'-o',quadric\_data,'-o')

plt.show()

7:

plt.figure()

plt.plot([22,44,55],'--r')

plt.show()

8:

linear\_data=np.array([1,2,3,4,5])

quadric\_data=linear\_data\*\*2

plt.figure()

plt.plot(linear\_data,'-o',quadric\_data,'-o')

plt.legend([‘Baseline’,’Competition’])

plt.show()

9:

#filling color between two plots

linear\_data=np.array([1,2,3,4,5])

quadric\_data=linear\_data\*\*2

plt.figure()

plt.plot(linear\_data,'-o',quadric\_data,'-o')

plt.legend(['Baseline','Competition'])

plt.gca().fill\_between(range(len(linear\_data)),linear\_data,quadric\_data,facecolor='green',alpha=0.25) #fill the color between the values in the figure

plt.show()

10:

#display 2 plots with different lines and points without convert to datetime

plt.figure()

observation\_dates=np.arange('2017-01-01','2017-01-09',dtype='datetime64[D]')

linear\_data=np.array([1,2,3,4,5,6,7,8])

quadric\_data=linear\_data\*\*2

plt.plot(observation\_dates,linear\_data,'-o',observation\_dates,quadric\_data,'--o')

plt.show()

11:

#display 2 plots with different lines and points after convert to datetime

import pandas as pd

plt.figure()

observation\_dates=np.arange('2017-01-01','2017-01-09',dtype='datetime64[D]')

observation\_dates=list(map(pd.to\_datetime,observation\_dates))

linear\_data=np.array([1,2,3,4,5,6,7,8])

quadric\_data=linear\_data\*\*2

plt.plot(observation\_dates,linear\_data,'-o',observation\_dates,quadric\_data,'--o')

plt.show()

12:

# setting title as math equations

ax=plt.gca()

ax.set\_title("Q($x^2$) Vs L($x$) ex(x)")

plt.show()

**Bar Charts:**

13:

#simple bar chart

plt.figure()

linear\_data=np.array([1,2,3,4,5,6,7,8])

xvals=range(len(linear\_data))

plt.bar(xvals,linear\_data,width=0.4)

plt.show()

14:

#display two plots in bar chart side by side

plt.figure()

linear\_data=np.arange(1,9)

quadric\_data=linear\_data\*\*2

xvals=range(len(linear\_data))

plt.bar(xvals,linear\_data,width=0.3,color='green')

new\_vals=[]

for item in xvals:

new\_vals.append(item+0.3)

plt.bar(new\_vals,quadric\_data,width=0.3,color='red')

plt.show()

15 optional because error:

# yerr in bar chart

from random import randint

plt.figure()

linear\_data=np.arange(1,9)

quadric\_data=linear\_data\*\*2

xvals=np.arange(len(linear\_data))

linear\_err=[randint(0,15) for x in range(len(linear\_data))]

print(linear\_err)

plt.bar(xvals,linear\_data,width=0.3,color='green',yerr=linear\_err)

new\_vals=xvals+0.3

#print(new\_vals,type(new\_vals))

plt.bar(new\_vals,quadric\_data,width=0.3,color='red')

plt.show()

16:

# display two plots in same bar

plt.figure()

linear\_data=np.arange(1,9)

quadric\_data=linear\_data\*\*2

xvals=np.arange(len(linear\_data))

plt.bar(xvals,linear\_data,width=0.3,color='green')

plt.bar(xvals,quadric\_data,width=0.3,color='red',bottom=linear\_data)

plt.show()

17:

#setting bar chart as horizontal by using barh()

plt.figure()

linear\_data=np.arange(1,9)

quadric\_data=linear\_data\*\*2

xvals=np.arange(len(linear\_data))

plt.barh(xvals,linear\_data,height=0.3,color='green')

plt.barh(xvals,quadric\_data,height=0.3,color='red',left=linear\_data)

plt.show()

#c2 w3

18:

#subplots

plt.figure()

plt.subplot(1,2,1)

lin\_d=np.arange(1,9)

plt.plot(lin\_d,'-o')

plt.show()

19:

#subplot continues

plt.figure()

plt.subplot(1,2,1)

lin\_d=np.arange(1,9)

plt.plot(lin\_d,’-o’)

exp\_d=lin\_d\*\*2

plt.subplot(1,2,2)

plt.plot(exp\_d,’-x’)

plt.show()

20:

#sub plot with same scaling in y axis

plt.figure()

ax1=plt.subplot(1,2,1)

lin\_d=np.arange(1,9)

plt.plot(lin\_d,'-o')

exp\_d=lin\_d\*\*2

ax2=plt.subplot(1,2,2,sharey=ax1)

plt.plot(exp\_d,'-x')

plt.show()

21:

# subplot is equals

plt.figure()

plt.subplot(1,2,1)==plt.subplot(121)

print(x)

plt.show()

22:

#displaying multiple plots (9 plots)

lin\_d=np.arange(1,9)

fig,((ax1,ax2,ax3),(ax4,ax5,ax6),(ax7,ax8,ax9))=plt.subplots(3,3,sharex=True,sharey=True)

ax5.plot(lin\_d,'-')

plt.show()

23:

# Not worked

lin\_d=np.arange(1,9)

fig,((ax1,ax2,ax3),(ax4,ax5,ax6),(ax7,ax8,ax9))=plt.subplots(3,3,sharex=True,sharey=True)

ax5.plot(lin\_d,'-o')

plt.show()

for ax in plt.gcf().get\_axes():

for label in ax.get\_xticklabels()+ax.get\_yticklabels():

label.set\_visible(True)

plt.gcf().canvas.draw()

**Histograms**

24:#simple hist

a=[1,2,2,3,3,3]

plt.hist(a)

plt.show()

25:

# subplots using gridspec

import matplotlib.gridspec as gridspec

plt.figure()

gspec=gridspec.GridSpec(3,3)

top\_hist=plt.subplot(gspec[0,1:])

side\_hist=plt.subplot(gspec[1:,0])

low\_right=plt.subplot(gspec[1:,1:])

plt.show()

26:

#applying plots

plt.figure()

gspec=gridspec.GridSpec(3,3)

top\_hist=plt.subplot(gspec[0,1:])

side\_hist=plt.subplot(gspec[1:,0])

low\_right=plt.subplot(gspec[1:,1:])

Y=np.random.normal(loc=0.0,scale=1.0,size=10000)

X=np.random.random(size=10000)

low\_right.scatter(X,Y)

top\_hist.hist(X,bins=100)

s=side\_hist.hist(Y,bins=100,orientation='horizontal')

plt.show()

27:

#change plot to right side

plt.figure()

gspec=gridspec.GridSpec(3,3)

top\_hist=plt.subplot(gspec[0,1:])

side\_hist=plt.subplot(gspec[1:,0])

low\_right=plt.subplot(gspec[1:,1:])

Y=np.random.normal(loc=0.0,scale=1.0,size=10000)

X=np.random.random(size=10000)

low\_right.scatter(X,Y)

top\_hist.hist(X,bins=100)

s=side\_hist.hist(Y,bins=100,orientation='horizontal')

top\_hist.clear()

top\_hist.hist(X,bins=100,normed=True)

side\_hist.clear()

side\_hist.hist(Y,bins=100,orientation='horizontal',normed=True)

side\_hist.invert\_xaxis()

plt.show()

28:

#setting x and y axis limits

plt.figure()

gspec=gridspec.GridSpec(3,3)

top\_hist=plt.subplot(gspec[0,1:])

side\_hist=plt.subplot(gspec[1:,0])

low\_right=plt.subplot(gspec[1:,1:])

Y=np.random.normal(loc=0.0,scale=1.0,size=10000)

X=np.random.random(size=10000)

low\_right.scatter(X,Y)

top\_hist.hist(X,bins=100)

s=side\_hist.hist(Y,bins=100,orientation='horizontal')

top\_hist.clear()

top\_hist.hist(X,bins=100,normed=True)

side\_hist.clear()

side\_hist.hist(Y,bins=100,orientation='horizontal',normed=True)

side\_hist.invert\_xaxis()

for ax in [top\_hist,low\_right]:

ax.set\_xlim(0,1)

for ax in [side\_hist,low\_right]:

ax.set\_ylim(-5,5)

plt.show()

**Box and Whisker Plot**

**29:**

#describe function in df

normal\_sample=np.random.normal(loc=0.0,scale=1.0,size=10000)

random\_sample=np.random.random(size=10000)

gamma\_sample=np.random.gamma(2,size=10000)

df=pd.DataFrame({'normal':normal\_sample,'random':random\_sample,'gamma':gamma\_sample})

df.describe()

30:

#simple boxplot

plt.figure()

\_=plt.boxplot(df['normal'],whis='range')

plt.show()

31:

#box plot with multiple boxplot

plt.figure()

\_=plt.boxplot([df['normal'],df['random'],df['gamma']],whis='range')

plt.show()

32:

#hist with df

plt.figure()

\_=plt.hist(df['gamma'],bins=100)

33:

#changing window title in interactive mode

ax = plt.gca()

fig=plt.figure()

data=np.random.rand(10)

plt.plot(data)

plt.gca().set\_title('hello')

def onclick(event):

plt.cla()

plt.plot(data)

fig.canvas.set\_window\_title('Event at pixels {},{}\n and data {},{}'.format(event.x,event.y,event.xdata,event.ydata))

cid=fig.canvas.mpl\_connect('button\_press\_event',onclick)

34:

#changing window title

from random import shuffle

fig=plt.figure()

origins=['china','Brazil','India','USA','Canada','UK','Germany','Iraq','Chile','Mexico']

shuffle(origins)

df=pd.DataFrame({'height':np.random.rand(10),'weight':np.random.rand(10),'origin':origins})

plt.scatter(df['height'],df['weight'],picker=5)

plt.gca().set\_ylabel('weight')

plt.gca().set\_xlabel('height')

def onpick(event):

origin=df.iloc[event.ind[0]]['origin']

fig.canvas.set\_window\_title('Selected Item came from {}'.format(origin))

plt.gcf().canvas.mpl\_connect('pick\_event',onpick)

35:

#to view all styles in plt

print(plt.style.available)

36:

#draw plot from df

df=pd.DataFrame({'A':np.random.randn(365).cumsum(),'B':np.random.rand(365).cumsum()+20,'C':np.random.randn(365).cumsum()-20},index=pd.date\_range('1/1/2017',periods=365))

print(df.head())

df.plot()

plt.show()

37:

#scatter plot from df

df.plot('A','B',kind='scatter')

plt.show()

# c2 w4 docs not prepared