Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations.
- It is a ploting library for python and its numeriacal mathematics extension numpy
- It provides an object oriented API (Application programing APT) for embedding plots into applications using general purpose GUI toolkits like, tkinter,QT,WXpython.
- It is a plotting library for 2d graphics.
- It is used for creating static, animated and intrective visulization in python.
- This is a kind of replacemment for MatLab
- matlab.engine.start_matlab, command for matlab
- Matplotlib is built on numpy array, which is multidimensional data visulization for 2 d plots
- import matplotlib.pyplot as plt, recommendation

Some basic type of plots

- line Plot function- plot()
- bar graph function -bar()
- histogram function - hist()
- scatter plot function - scatter()
- area plot function - stackplot()
- pieplot or piechart -function pie()

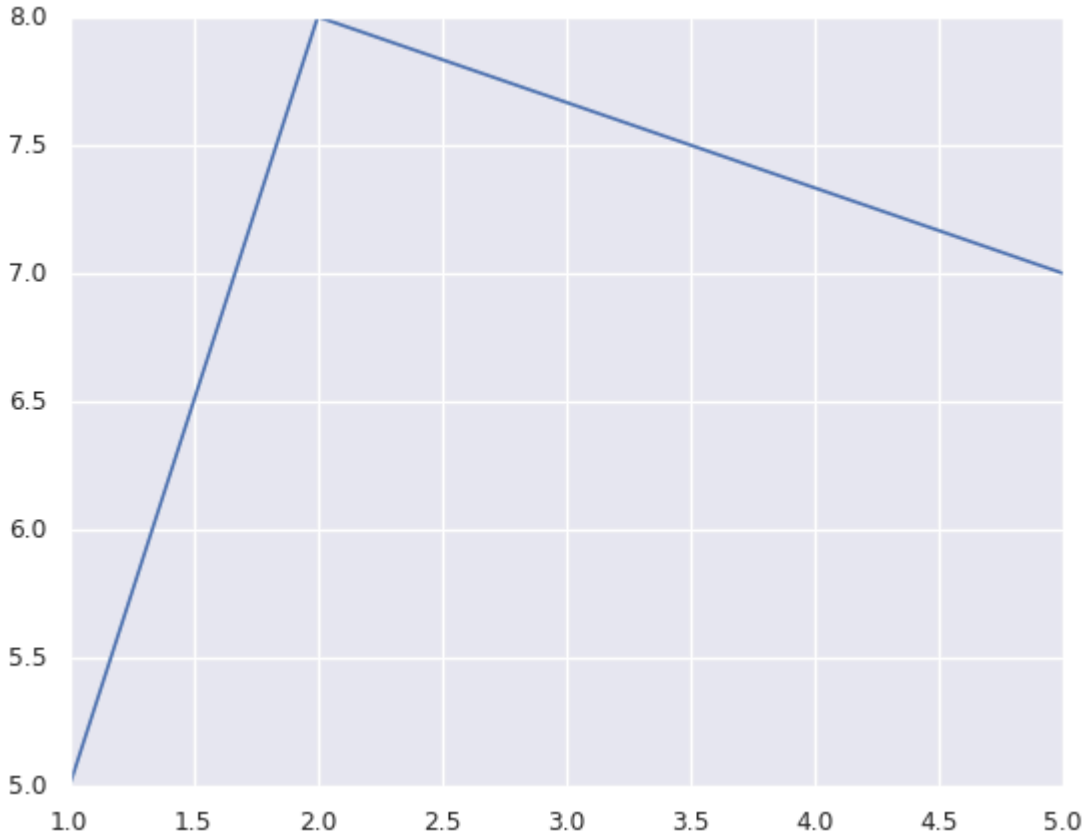
Line Plot or Line Chart

- This is a kind of graph or curve which represent the information in the form of datapoints which are connected through straight lines.
- We use plot function as plot(), it has two argument, init, plot (x,y) where x indicates x coordianted to the x axis and y indicates y coordinated to y axis.

In [117…
```
import matplotlib.pyplot as plt # Importing library method 1
```

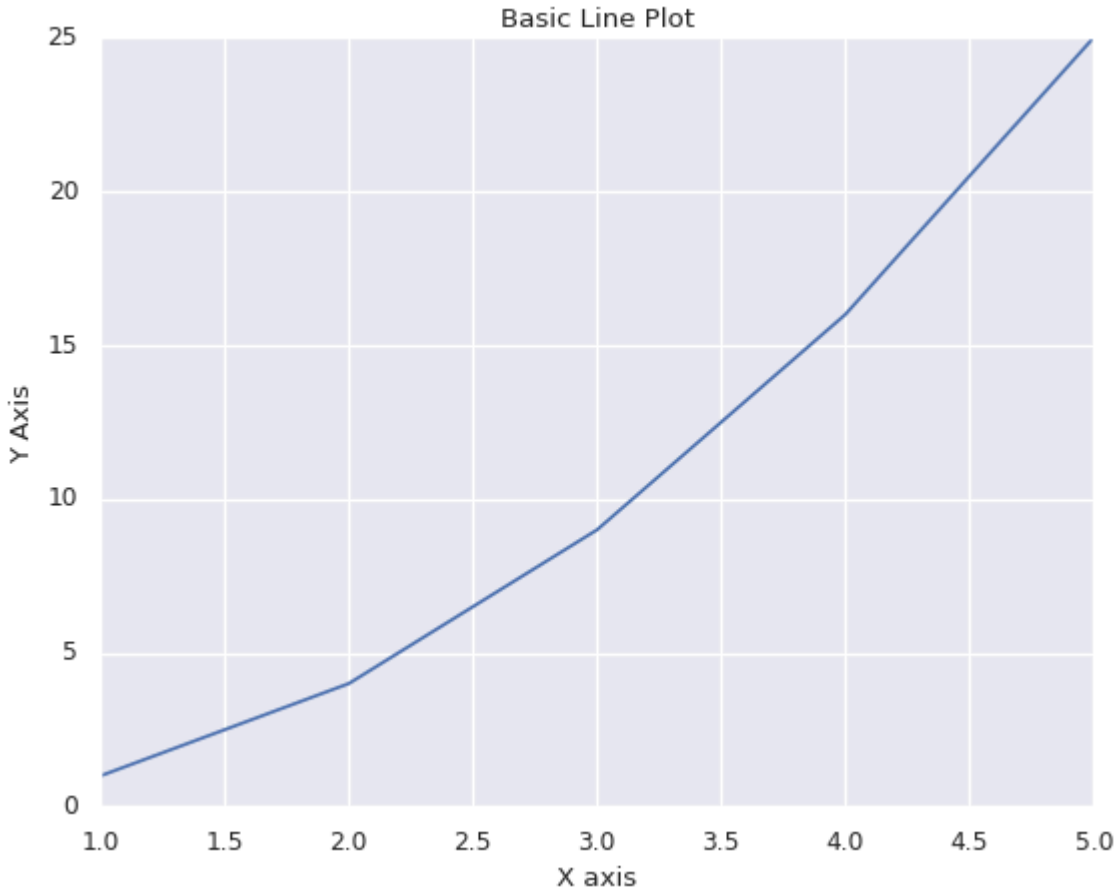
In [118…
```
from matplotlib import pyplot as plt # Importing library method 2
```

In [119…
```
plt.plot([1,2,5],[5,8,7])
plt.show()
```



In [120…

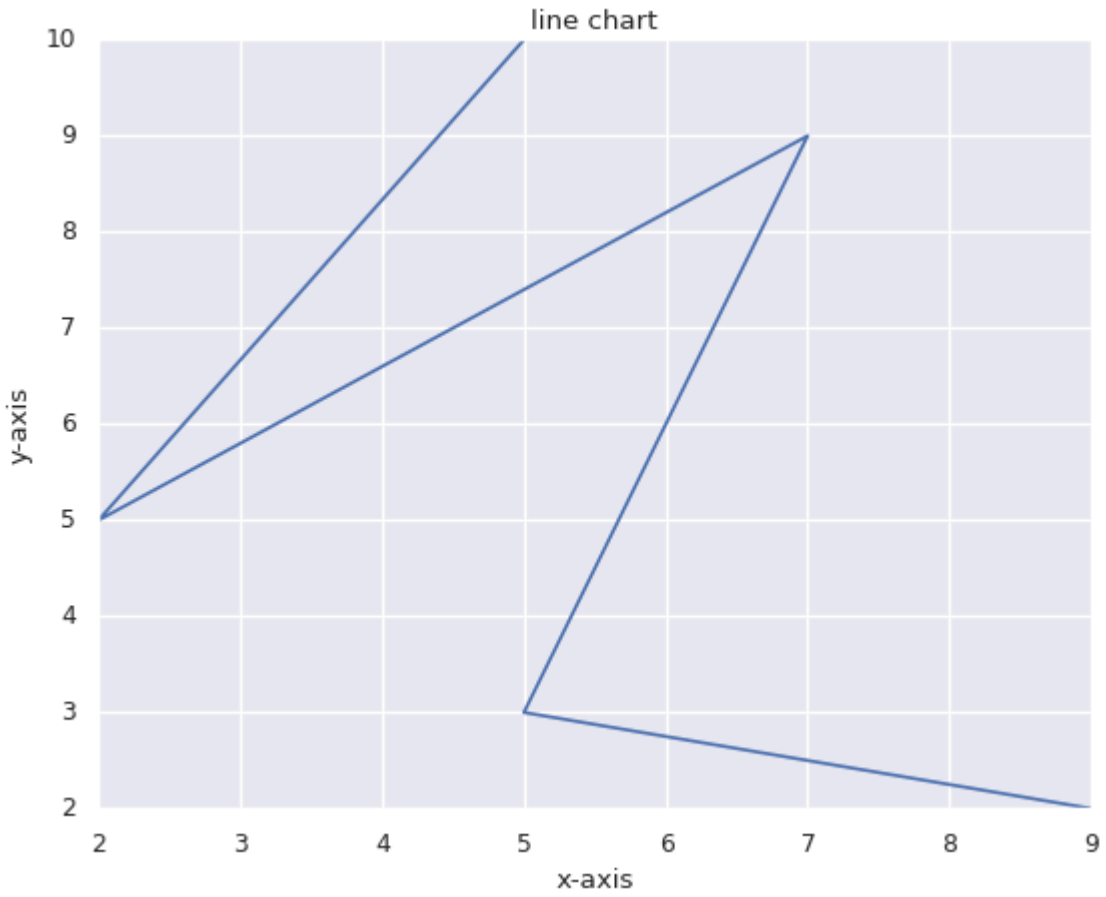
```
x=[1,2,3,4,5]
y=[1,4,9,16,25]

##create a line plot
plt.plot(x,y)
plt.xlabel('X axis')
plt.ylabel('Y Axis')
plt.title("Basic Line Plot")
plt.show()
```



In [121…

```
x=[5,2,7,5,9]
y=[10,5,9,3,2]
plt.plot(x,y)
plt.title("line chart")
```

```
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```



In [122…

```
import numpy as np
from matplotlib import pyplot as plt
x=np.arange(0,np.pi,0.1)
print(x)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1]
```

In [123…

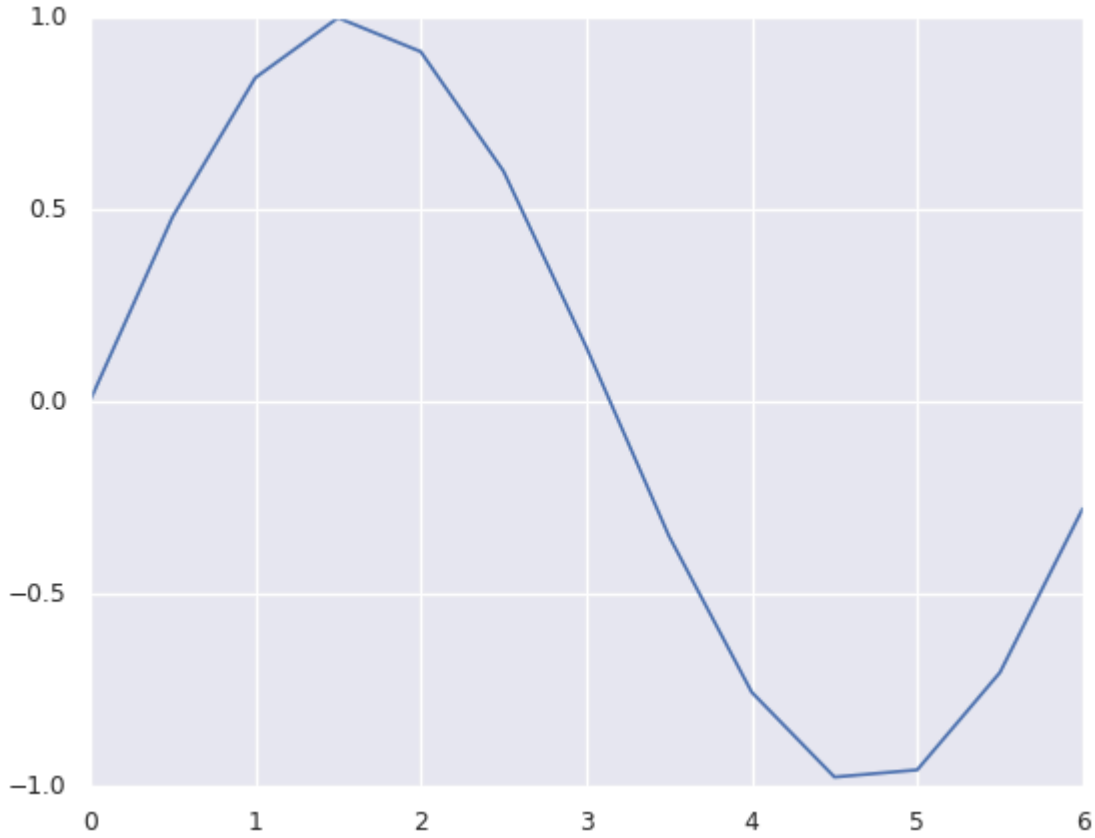
```
y=np.sin(x)
print(y)
```

```
[0.         0.09983342 0.19866933 0.29552021 0.38941834 0.47942554
 0.56464247 0.64421769 0.71735609 0.78332691 0.84147098 0.89120736
 0.93203909 0.96355819 0.98544973 0.99749499 0.9995736  0.99166481
 0.97384763 0.94630009 0.90929743 0.86320937 0.8084964  0.74570521
 0.67546318 0.59847214 0.51550137 0.42737988 0.33498815 0.23924933
 0.14112001 0.04158066]
```

In [124…

```
x=np.arange(0,2*np.pi,0.5)
print(x)
y=np.sin(x)
print(y)
plt.plot(x,y)
plt.show()
```

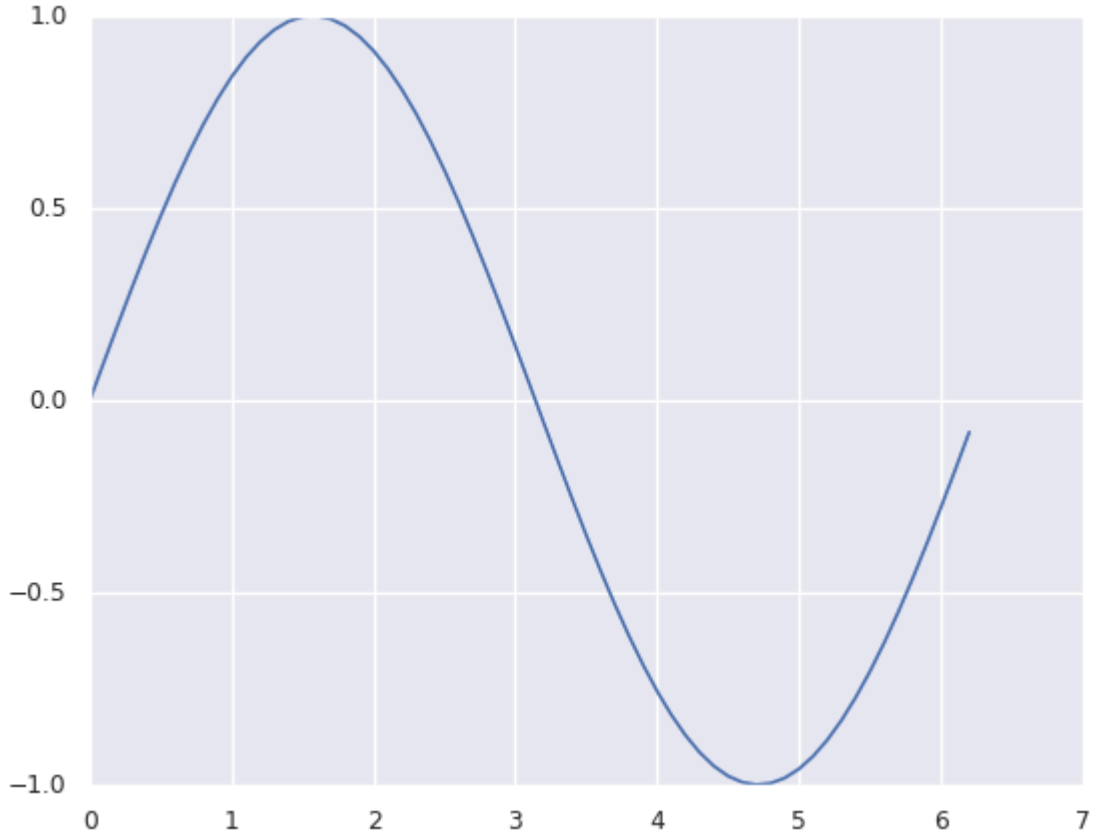
```
[0.  0.5 1.  1.5 2.  2.5 3.  3.5 4.  4.5 5.  5.5 6. ]
[ 0.          0.47942554  0.84147098  0.99749499  0.90929743  0.59847214
  0.14112001 -0.35078323 -0.7568025  -0.97753012 -0.95892427 -0.70554033
 -0.2794155 ]
```



In [125…

```
x=np.arange(0,2*np.pi,0.1)
print(x)
y=np.sin(x)
print(y)
plt.plot(x,y)
plt.show()
```

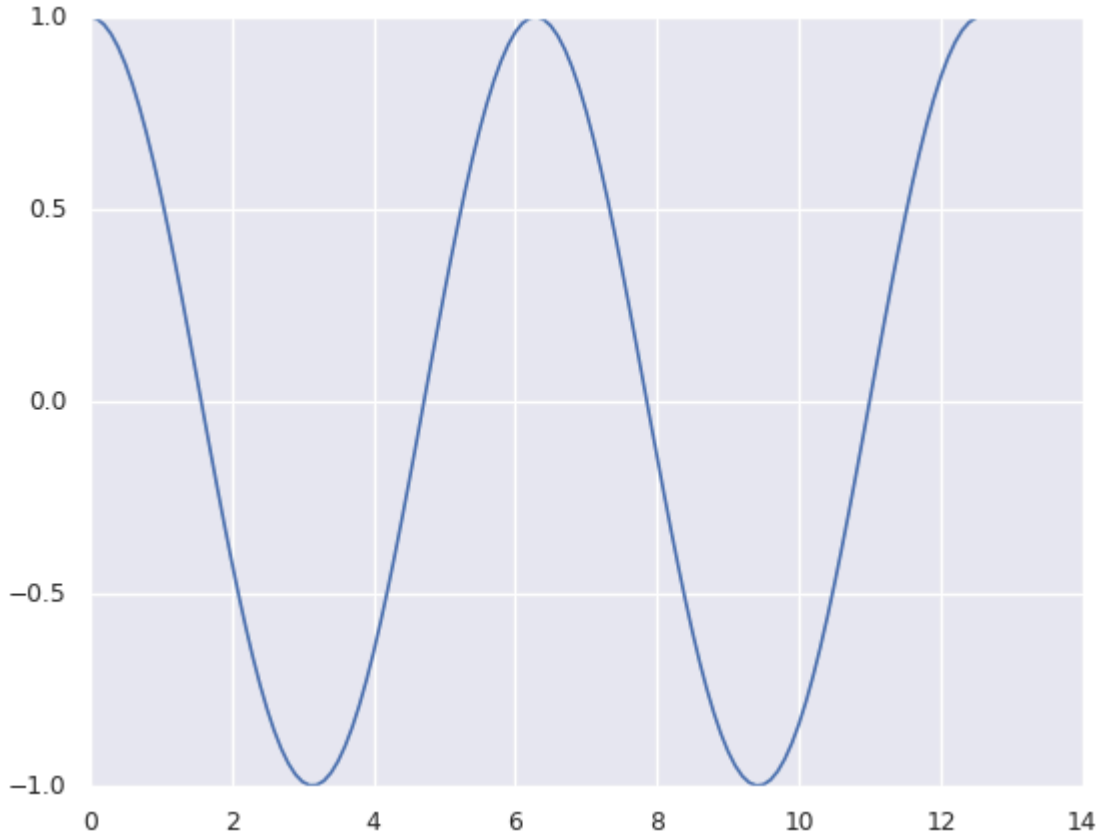
```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2]
[ 0.          0.09983342  0.19866933  0.29552021  0.38941834  0.47942554
  0.56464247  0.64421769  0.71735609  0.78332691  0.84147098  0.89120736
  0.93203909  0.96355819  0.98544973  0.99749499  0.9995736   0.99166481
  0.97384763  0.94630009  0.90929743  0.86320937  0.8084964   0.74570521
  0.67546318  0.59847214  0.51550137  0.42737988  0.33498815  0.23924933
  0.14112001  0.04158066 -0.05837414 -0.15774569 -0.2555411  -0.35078323
 -0.44252044 -0.52983614 -0.61185789 -0.68776616 -0.7568025  -0.81827711
 -0.87157577 -0.91616594 -0.95160207 -0.97753012 -0.993691   -0.99992326
 -0.99616461 -0.98245261 -0.95892427 -0.92581468 -0.88345466 -0.83226744
 -0.77276449 -0.70554033 -0.63126664 -0.55068554 -0.46460218 -0.37387666
 -0.2794155  -0.1821625  -0.0830894 ]
```



In [126…

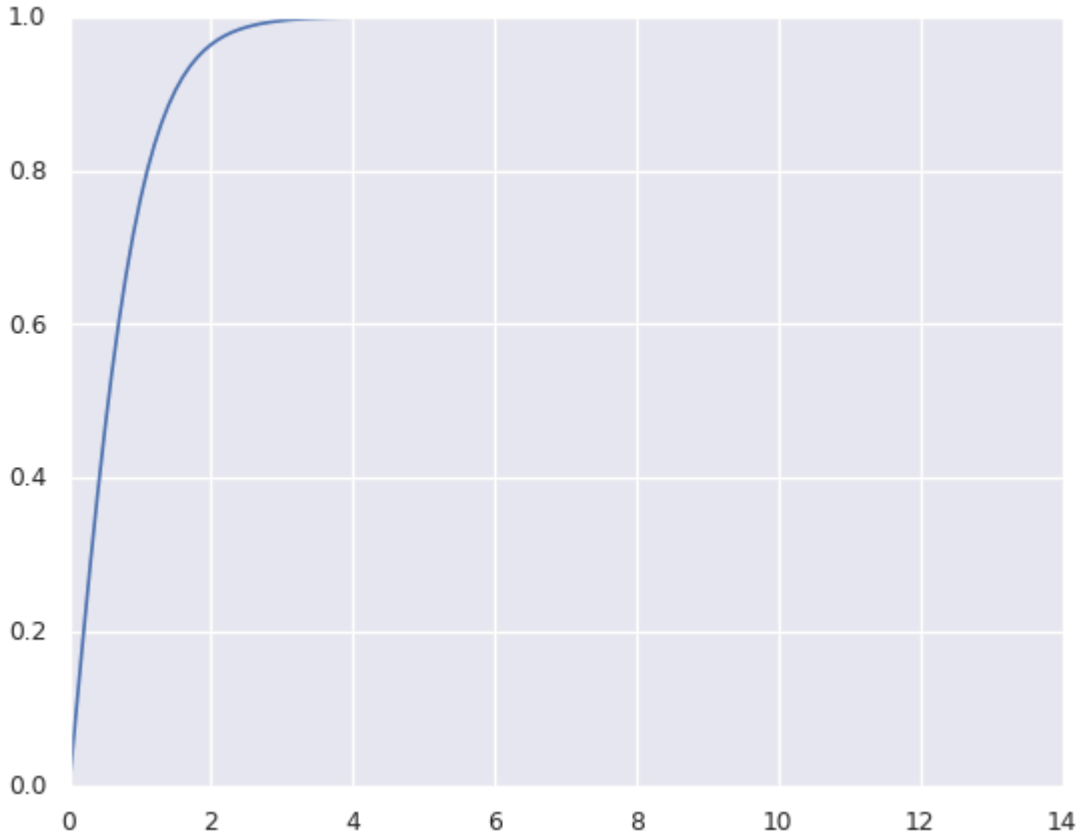
```
x=np.arange(0,4*np.pi,0.1)
print(x)
y=np.cos(x)
print(y)
plt.plot(x,y)
plt.show()
```

```
[ 0.   0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.   1.1  1.2  1.3
  1.4  1.5  1.6  1.7  1.8  1.9  2.   2.1  2.2  2.3  2.4  2.5  2.6  2.7
  2.8  2.9  3.   3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.   4.1
  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.   5.1  5.2  5.3  5.4  5.5
  5.6  5.7  5.8  5.9  6.   6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9
  7.   7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.   8.1  8.2  8.3
  8.4  8.5  8.6  8.7  8.8  8.9  9.   9.1  9.2  9.3  9.4  9.5  9.6  9.7
  9.8  9.9 10.  10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9 11.  11.1
 11.2 11.3 11.4 11.5 11.6 11.7 11.8 11.9 12.  12.1 12.2 12.3 12.4 12.5]
[ 1.          0.99500417  0.98006658  0.95533649  0.92106099  0.87758256
  0.82533561  0.76484219  0.69670671  0.62160997  0.54030231  0.45359612
  0.36235775  0.26749883  0.16996714  0.0707372  -0.02919952 -0.12884449
 -0.22720209 -0.32328957 -0.41614684 -0.5048461  -0.58850112 -0.66627602
 -0.73739372 -0.80114362 -0.85688875 -0.90407214 -0.94222234 -0.97095817
 -0.9899925  -0.99913515 -0.99829478 -0.98747977 -0.96679819 -0.93645669
 -0.89675842 -0.84810003 -0.79096771 -0.7259323  -0.65364362 -0.57482395
 -0.49026082 -0.40079917 -0.30733287 -0.2107958  -0.11215253 -0.01238866
  0.08749898  0.18651237  0.28366219  0.37797774  0.46851667  0.55437434
  0.63469288  0.70866977  0.77556588  0.83471278  0.88551952  0.92747843
  0.96017029  0.98326844  0.9965421   0.99985864  0.99318492  0.97658763
  0.95023259  0.91438315  0.86939749  0.8157251   0.75390225  0.68454667
  0.60835131  0.52607752  0.43854733  0.34663532  0.25125984  0.15337386
  0.05395542 -0.04600213 -0.14550003 -0.24354415 -0.33915486 -0.43137684
 -0.51928865 -0.6020119  -0.67872005 -0.74864665 -0.81109301 -0.86543521
 -0.91113026 -0.9477216  -0.97484362 -0.99222533 -0.99969304 -0.99717216
 -0.98468786 -0.96236488 -0.93042627 -0.88919115 -0.83907153 -0.78056818
 -0.71426565 -0.64082642 -0.56098426 -0.47553693 -0.38533819 -0.29128928
 -0.19432991 -0.09542885  0.0044257   0.10423603  0.20300486  0.29974534
  0.39349087  0.48330476  0.56828963  0.64759634  0.72043248  0.7860703
  0.84385396  0.89320611  0.93363364  0.96473262  0.9861923   0.99779828]
```



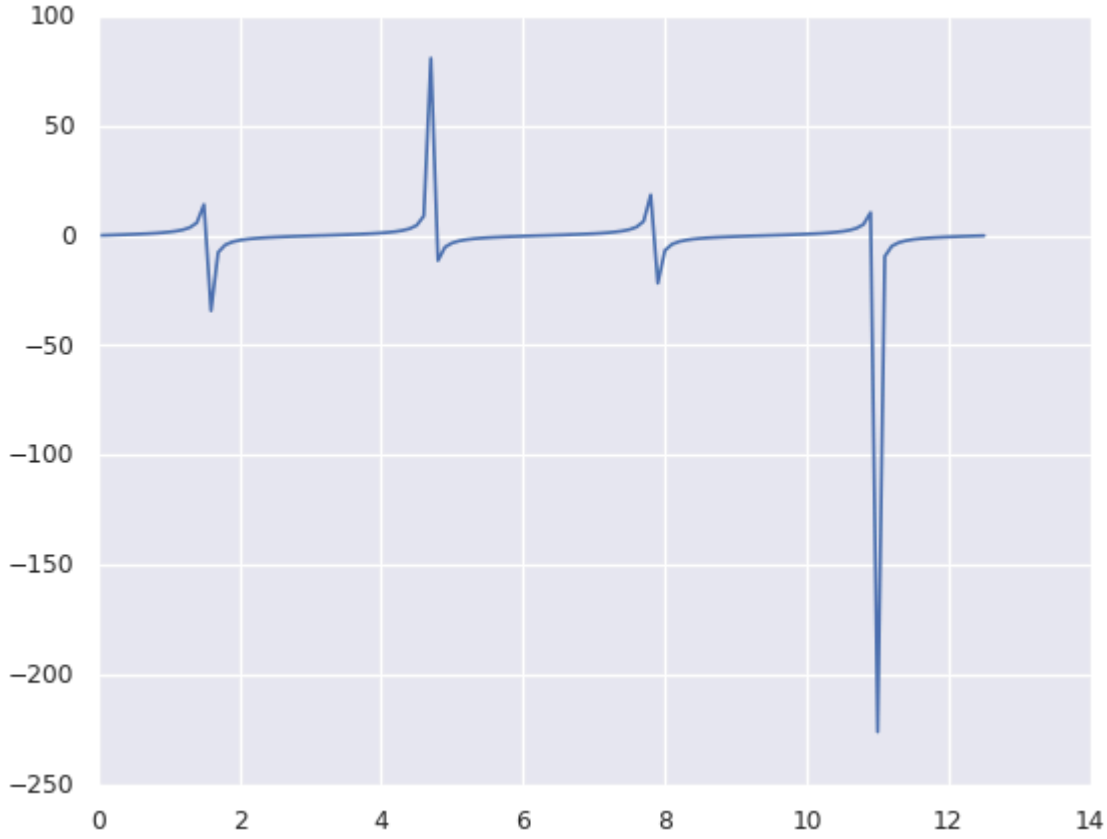
In [127…

```
x=np.arange(0,4*np.pi,0.1)
#print(x)
y=np.tanh(x)
#print(y)
plt.plot(x,y)
plt.show()
```



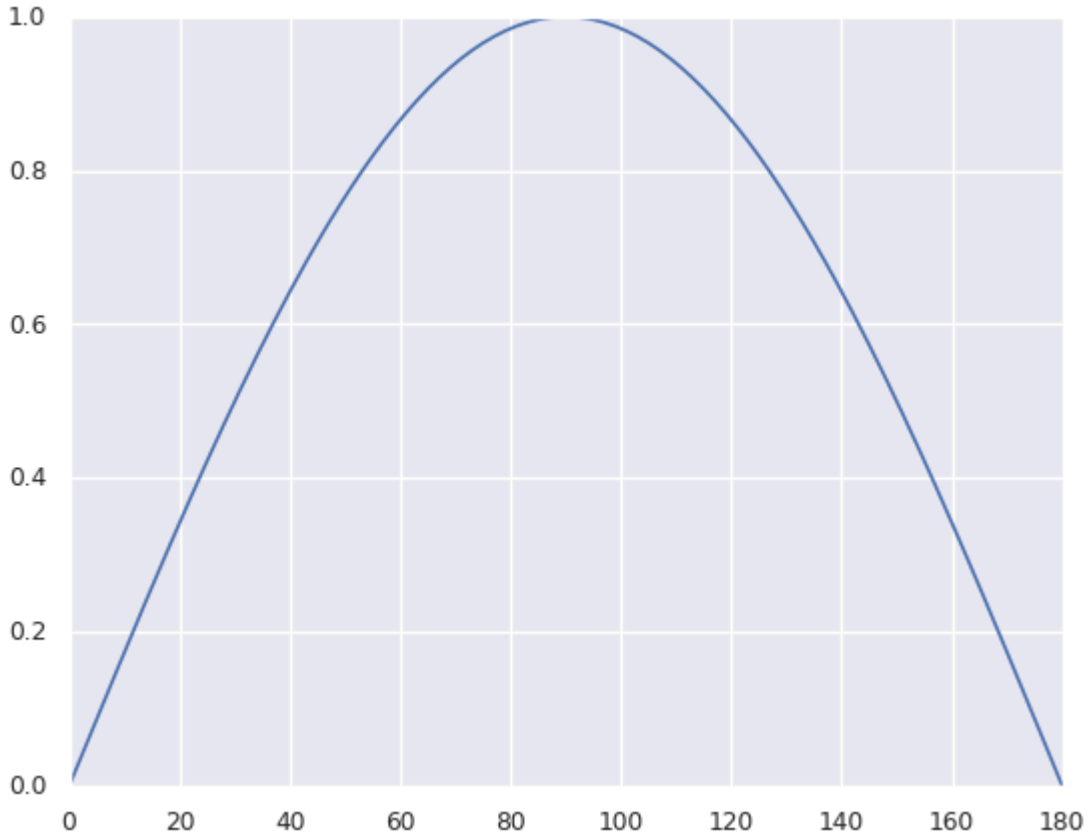
In [128…

```
x=np.arange(0,4*np.pi,0.1)
#print(x)
y=np.tan(x)
#print(y)
plt.plot(x,y)
plt.show()
```



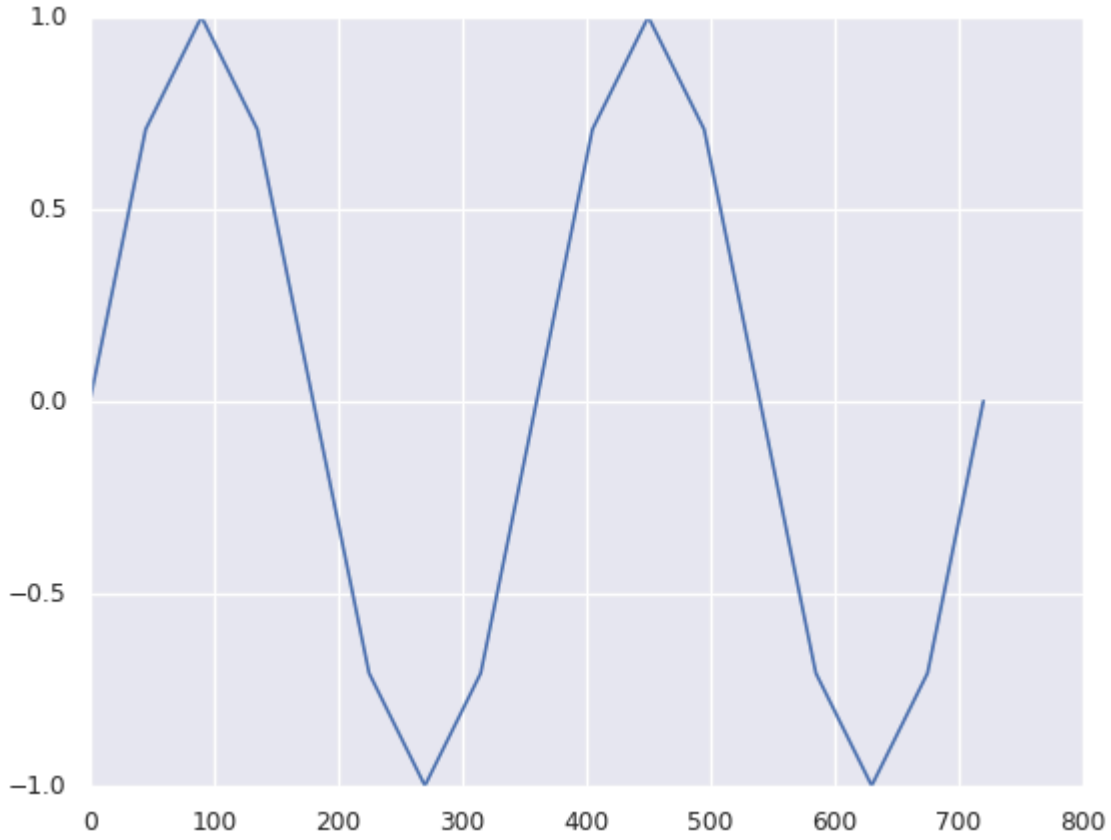
In [129…

```
degrees=np.arange(0,181)
sine=np.sin(np.radians(degrees))
plt.plot(degrees, sine)
plt.show()
#print(degrees)
#print(sine)
```



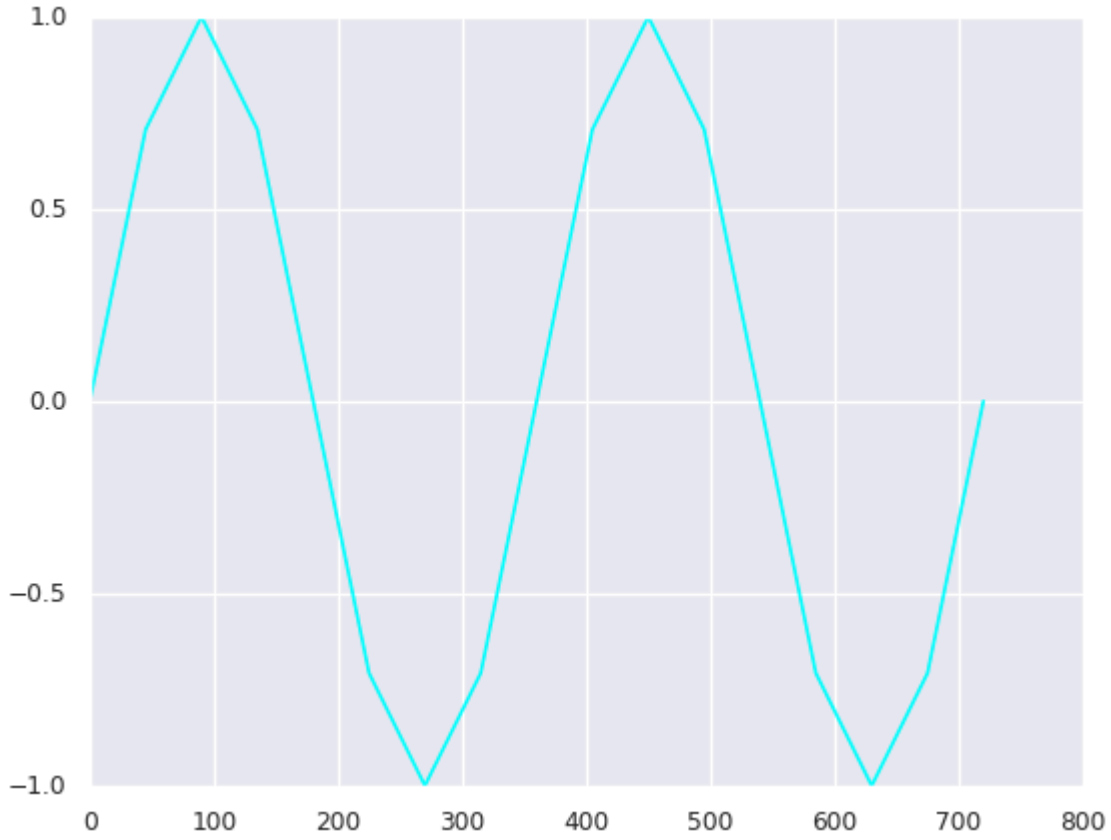
In [130…

```
degrees=np.arange(0,721,45)
sine=np.sin(np.radians(degrees))
plt.plot(degrees,sine)
plt.show()
```



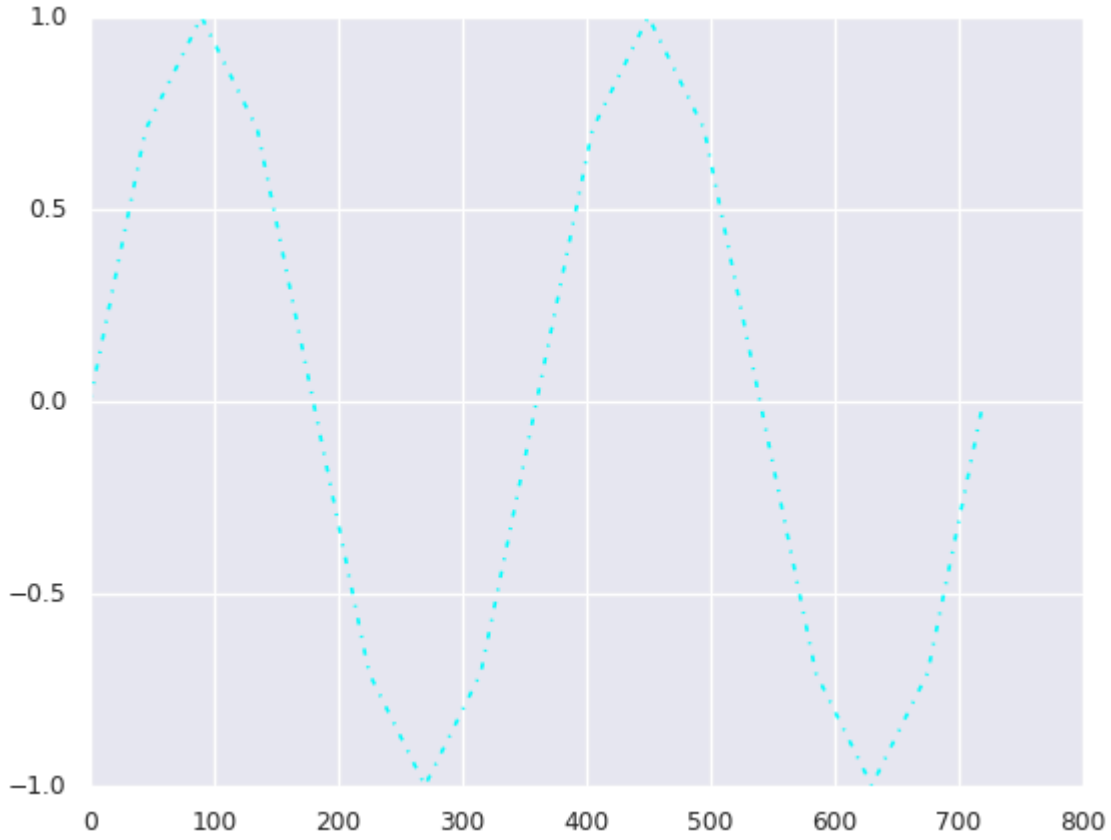
In [131…

```
degrees=np.arange(0,721,45)
sine=np.sin(np.radians(degrees))
plt.plot(degrees,sine,c="cyan") # changing the color of line
plt.show()
```



In [132…

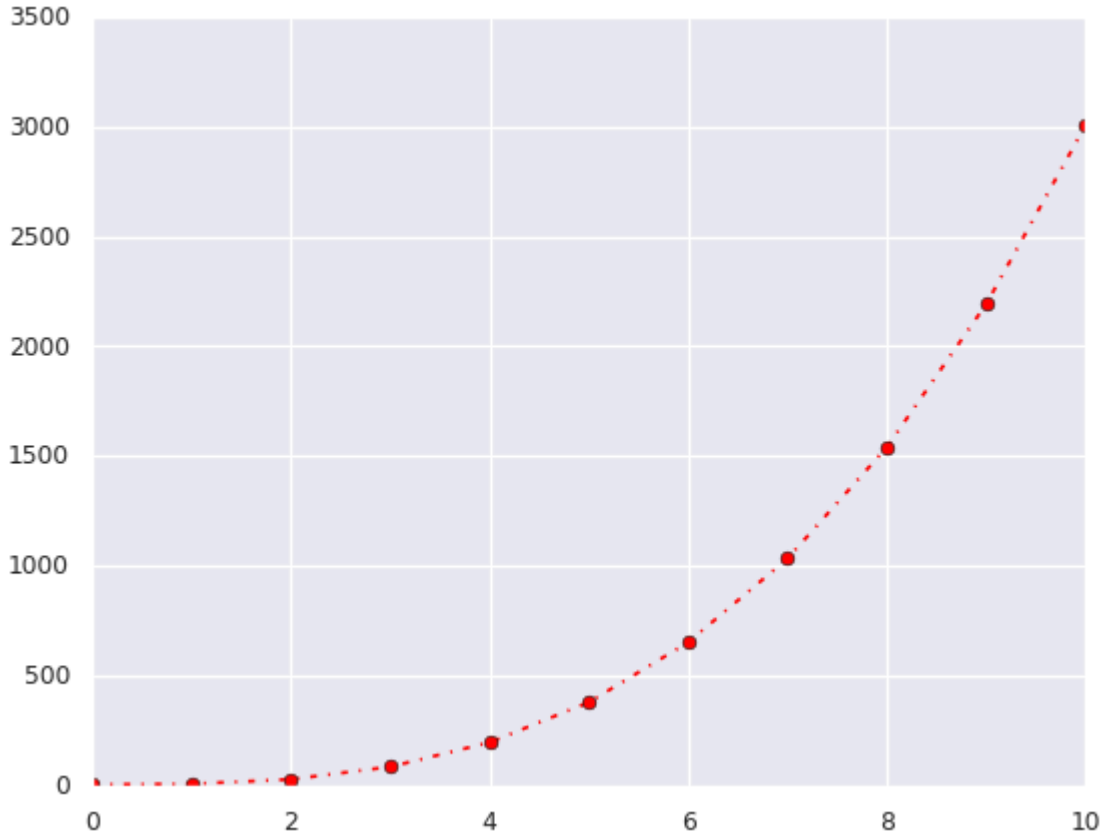
```
# Line color and pattern change
degrees=np.arange(0,721,45)
sine=np.sin(np.radians(degrees))
plt.plot(degrees,sine,c="cyan",ls="dashdot") #ls= line state or line style. c= color of the line in graph.
plt.show()
```



In [133…

```
x=np.arange(0,11)
y=3*x**2*x+5
print(y)
plt.plot(x,y,marker="o",c="red",ls="dashdot")
plt.show()
```

```
[   5    8   29   86  197  380  653 1034 1541 2192 3005]
```



We can plot more than one line into line chart, we can also make it colourful

Markers

'.' – point marker, 'o'- circle, '*'- star, '^'- triangle_up, 'p'- pentagon, 's'- square, '+'-plus, 'D'- Diamond, '|'- vline .... And many more.

Colors

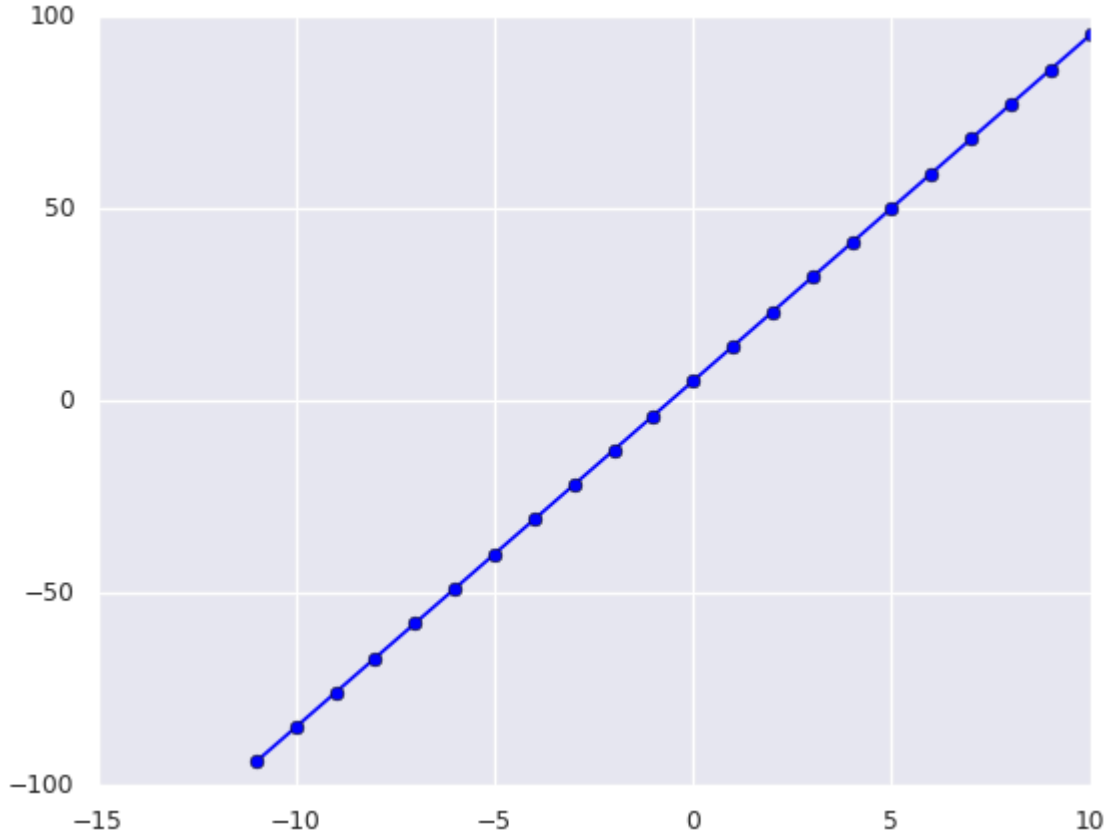
'b' -blue , 'g'- green , 'r'-red , 'y'- yellow , 'k'- black , 'm'- magenta, 'w'- white, 'c'- cyan

Line styles

'-' (solid line), '- -' (Dashed line) , '-.' (Dash dot line) , ':' (Dotted line)

In [134…

```
x=np.arange(-11,11)
y=3*x+6*x+5
plt.plot(x,y,marker="o",c="blue")
plt.show()
```



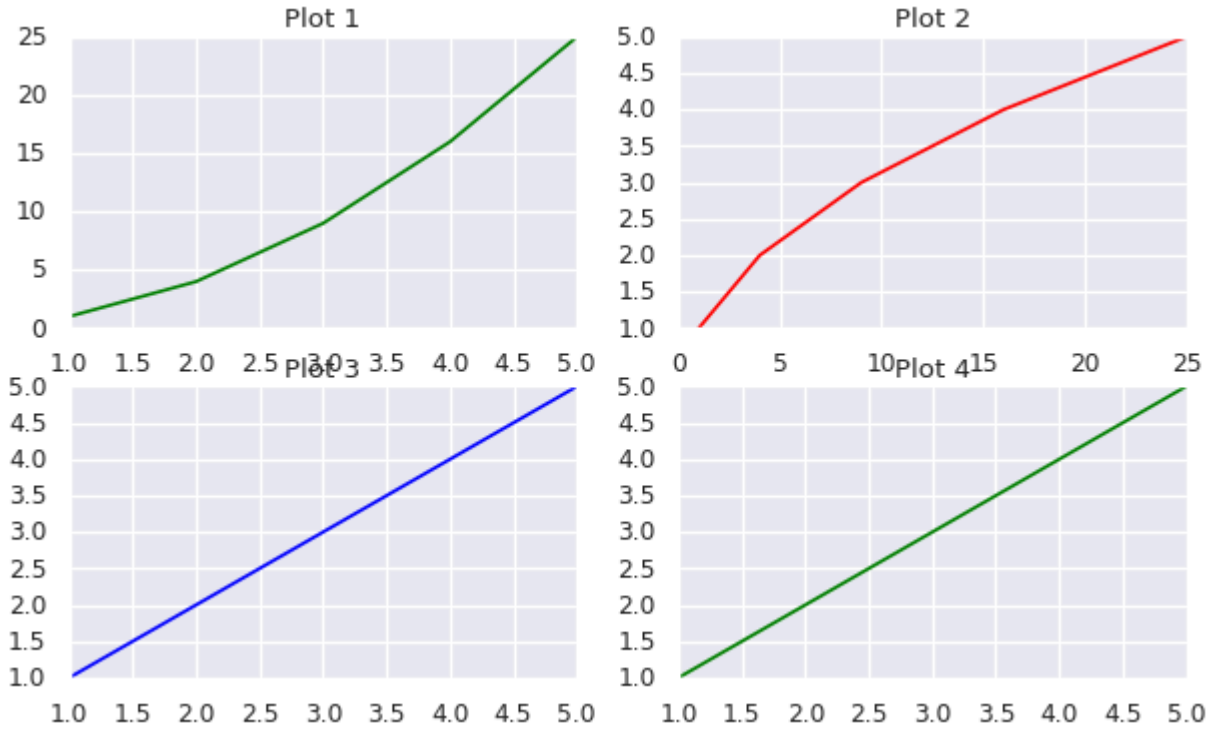
Creating a multi line graph

In [135…

```
## Multiple Plots
## Sample data
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 9, 16, 25]
y2 = [1, 2, 3, 4, 5]

plt.figure(figsize=(9,5))

plt.subplot(2,2,1)
plt.plot(x,y1,color='green')
plt.title("Plot 1")

plt.subplot(2,2,2)
plt.plot(y1,x,color='red')
plt.title("Plot 2")

plt.subplot(2,2,3)
plt.plot(x,y2,color='blue')
plt.title("Plot 3")

plt.subplot(2,2,4)
plt.plot(x,y2,color='green')
plt.title("Plot 4")
```

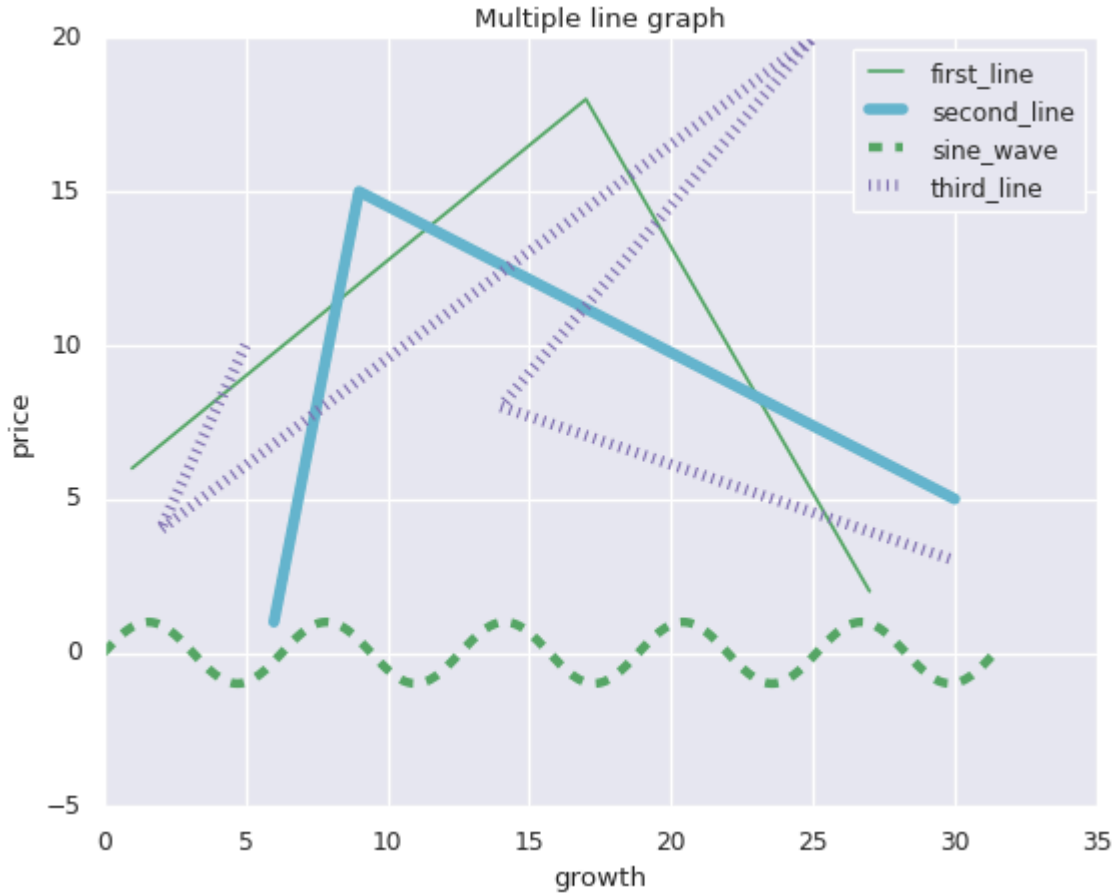
Out[135]:

```
Text(0.5, 1.0, 'Plot 4')
```



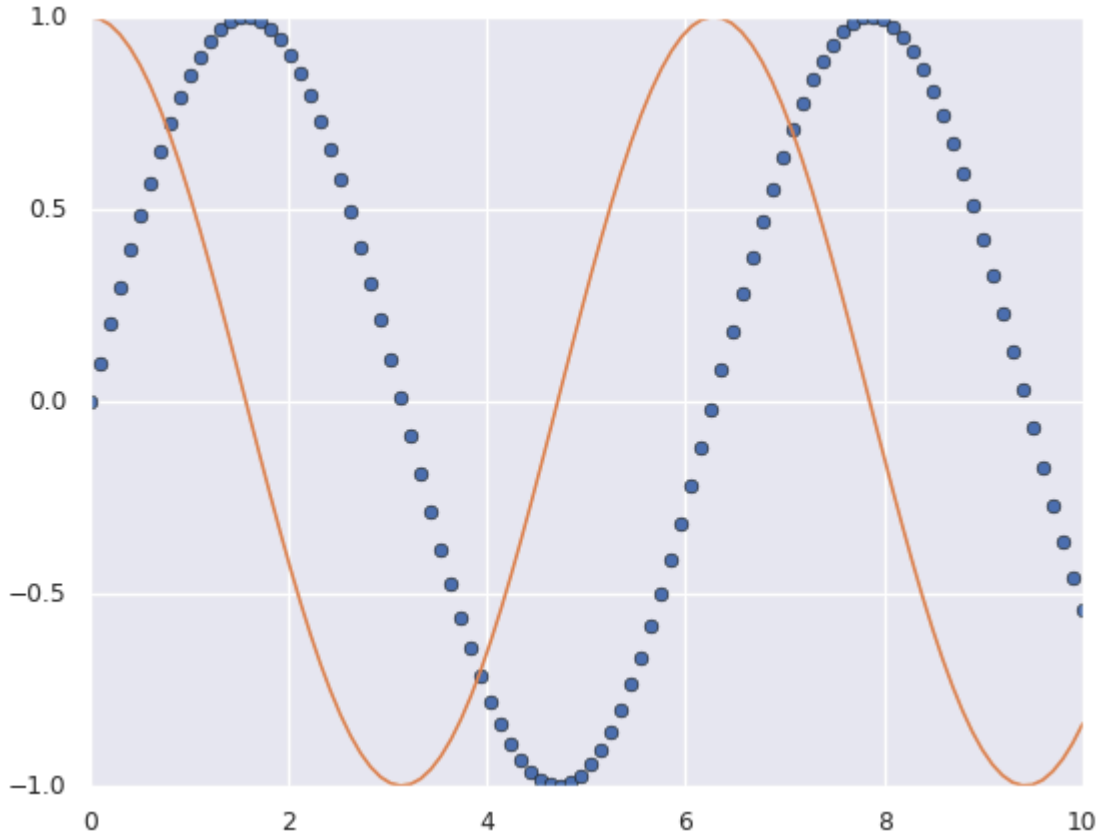
In [136…

```
import math
x1=[1,17,27]
y1=[6,18,2]
x2=[6,9,30]
y2=[1,15,5]
x3=[5,2,25,14,30]
y3=[10,4,20,8,3]
x=np.arange(0,math.pi*10,0.10)
y=np.sin(x)
plt.plot(x1,y1,"g",label="first_line")
plt.plot(x2,y2,"c",label="second_line",linewidth=5)
plt.plot(x,y,"g--",label="sine_wave",linewidth=4)
plt.plot(x3,y3,"m:",label="third_line",linewidth=5)
plt.title("Multiple line graph")
plt.xlabel("growth")
plt.ylabel("price")
plt.legend()
plt.show()
```



In [137…

```
x=np.linspace(0,10,100) #will create an array of 100 values with spaced between 1 and 10
plt.plot(x,np.sin(x),"o")
plt.plot(x,np.cos(x),"-")
plt.show()
```

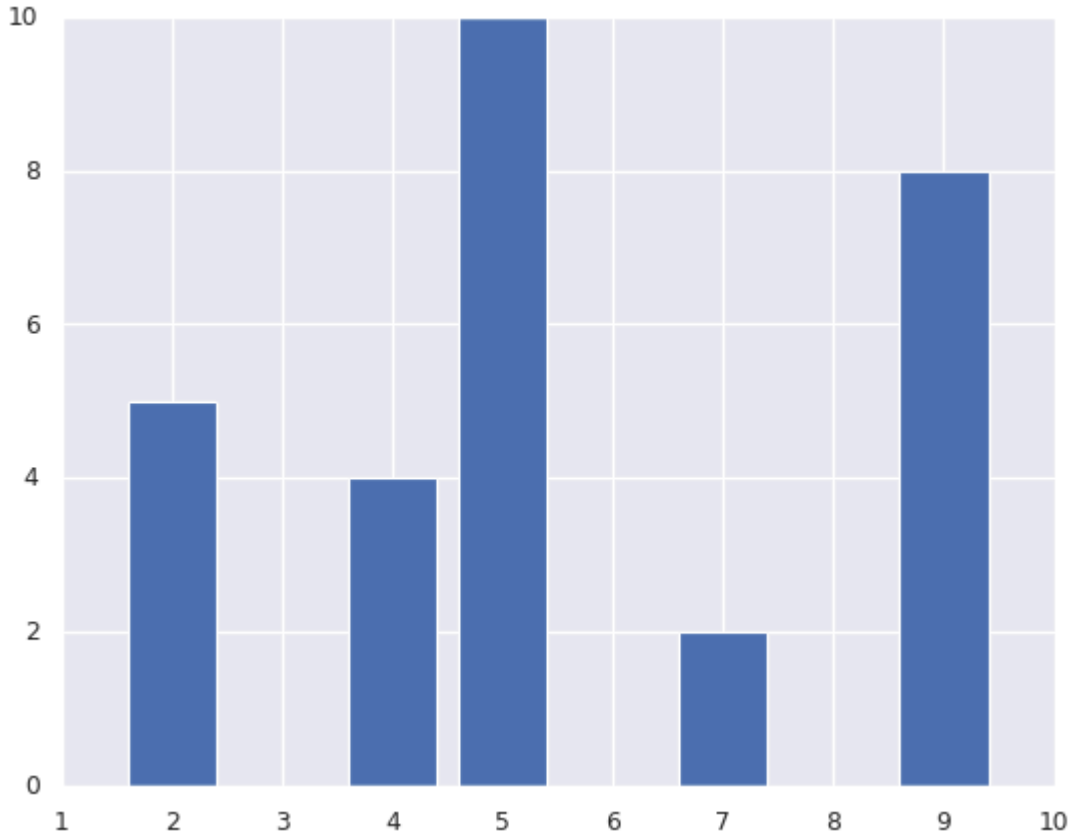


Bar Plot or Bar Graph

- Bar plot or Bar gharph or Bar Graph
- It is a graphichal dispaly of data using rectangular bar of diferent heights.

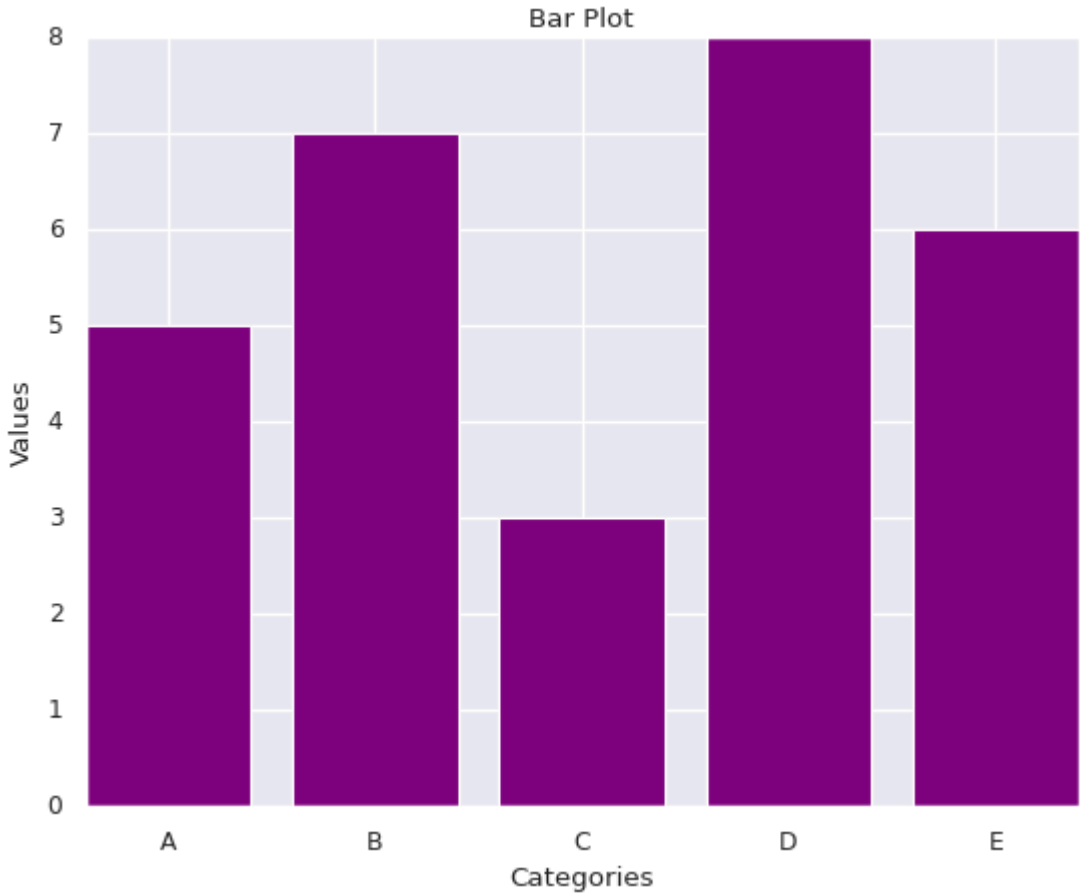
In [138…

```
from matplotlib import pyplot as plt
x=[5,2,9,4,7]
y=[10,5,8,4,2]
plt.bar(x,y)
plt.show()
```



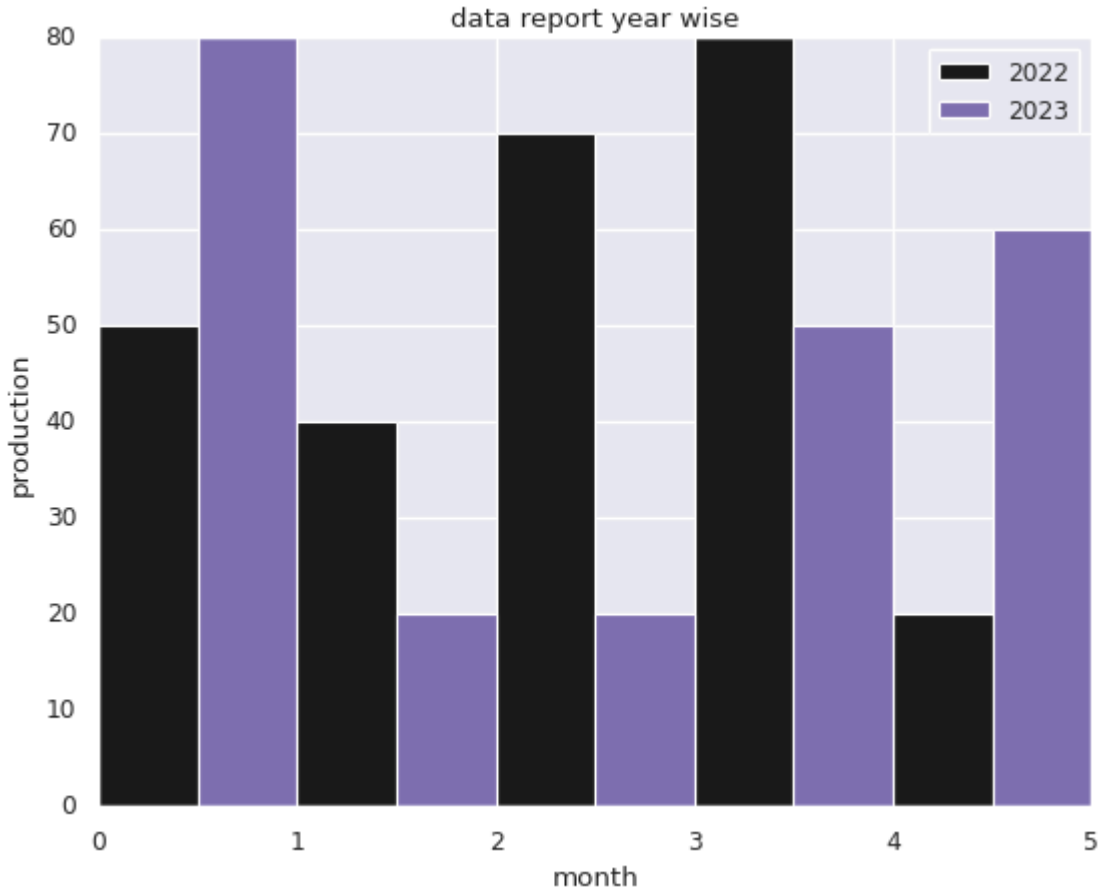
In [139…

```
categories=['A','B','C','D','E']
values=[5,7,3,8,6]

##create a bar plot
plt.bar(categories,values,color='purple')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot')
plt.show()
```



In [140…
```
#relational plots, distributed plots and categorical plot
#under relational plot we have line plot and scatter plot
```

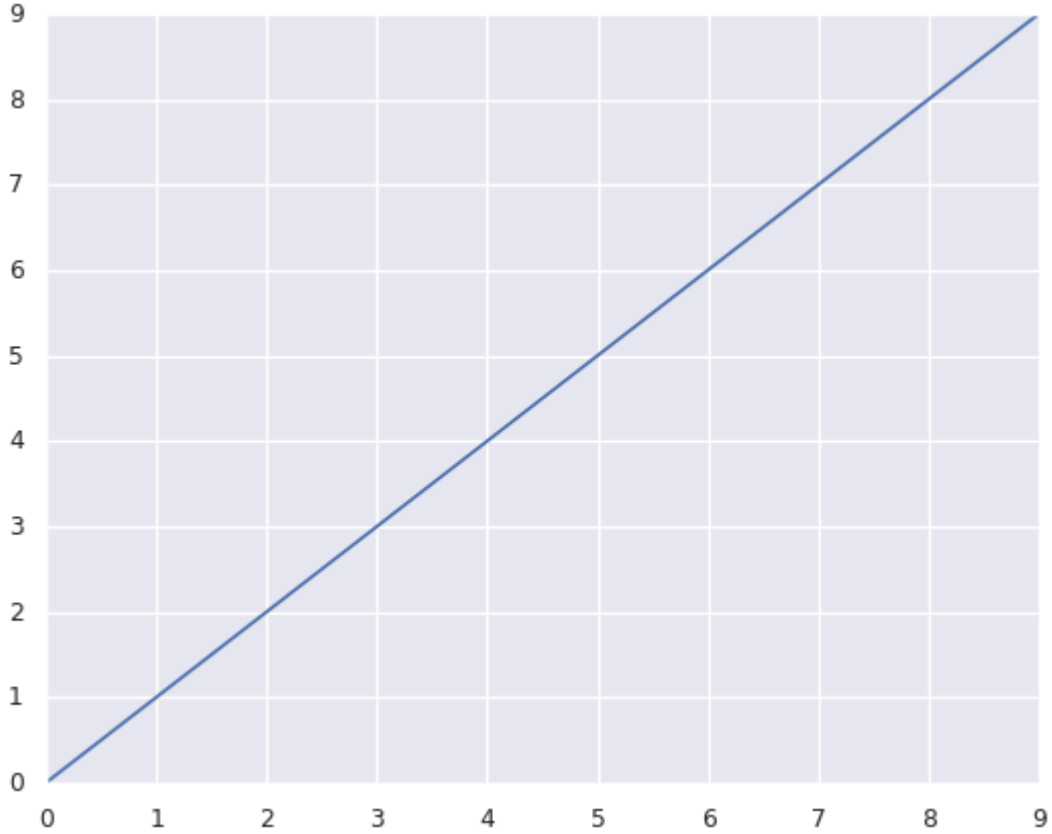
In [141…
```
plt.bar([0.25,1.25,2.25,3.25,4.25],[50,40,70,80,20],label="2022",color="k",width=0.5) # random data
plt.bar([0.75,1.75,2.75,3.75,4.75],[80,20,20,50,60],label="2023",color="m",width=0.5) # random data
plt.legend()
plt.xlabel("month")
plt.ylabel("production")
plt.title("data report year wise")
plt.show()
```



Creating Sub Plots

In [142…

```
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
data=np.arange(10)
plt.plot(data)
plt.show()
```



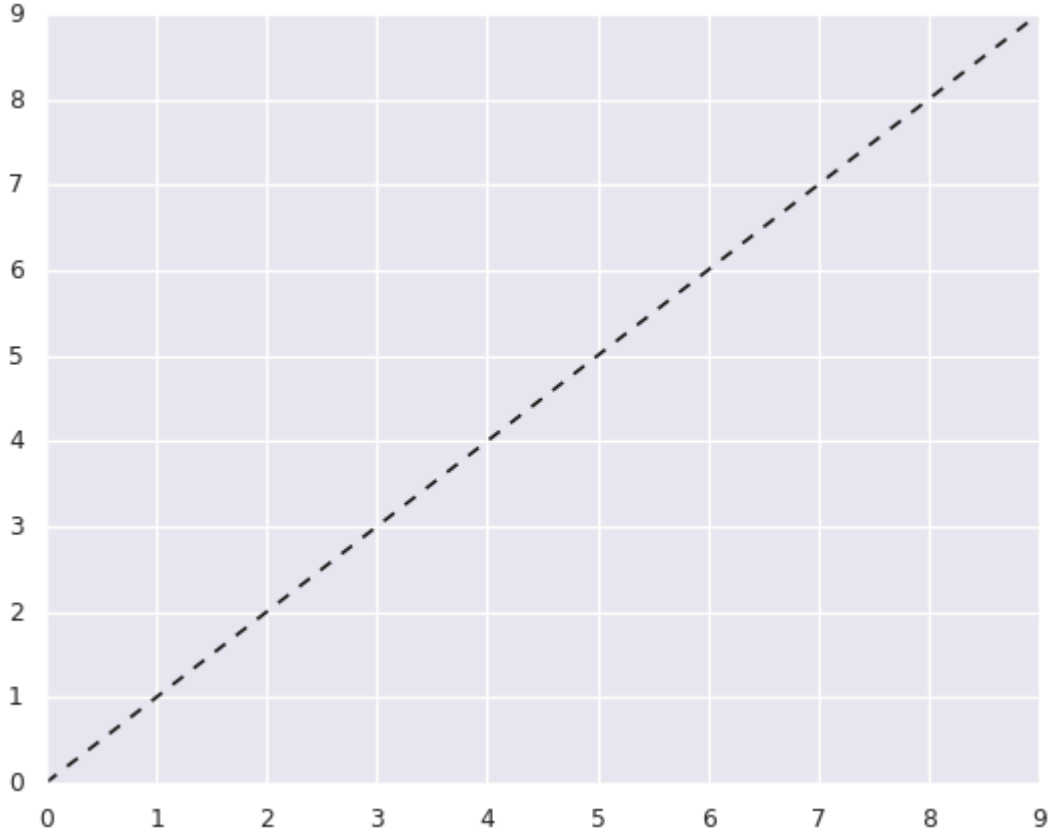
In [143…
```
#figure and subplots
fig=plt.figure()
plt.show()
```

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

In [144…
```
p1=fig.add_subplot(2,2,1) #creating four subplot in the fig
p2=fig.add_subplot(2,2,2)
p3=fig.add_subplot(2,2,3)
p4=fig.add_subplot(2,2,4)
```

In [145…
```
plt.plot(np.arange(10),"k--") #plot function only
p1.scatter(np.arange(30),np.arange(30)+np.random.randn(30))
```

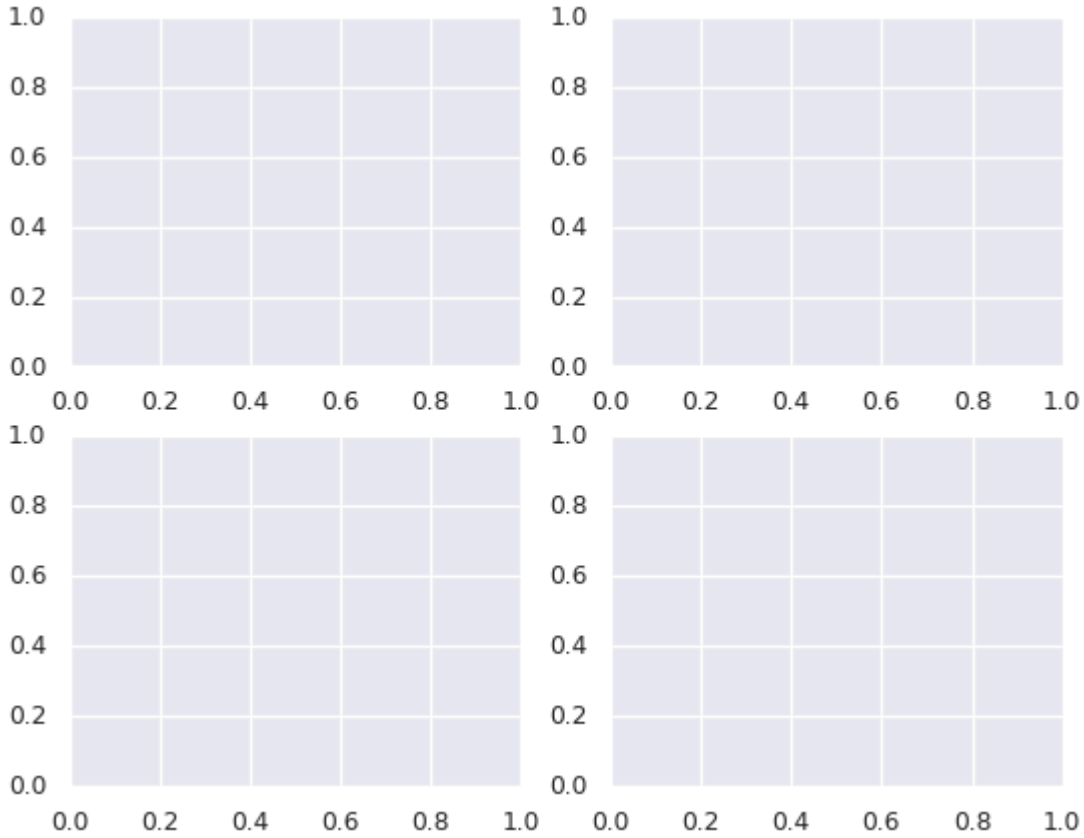
Out[145]:
```
<matplotlib.collections.PathCollection at 0x79c69e638400>
```



In [146…

```
fig,axes=plt.subplots(2,2)
print(axes)
```

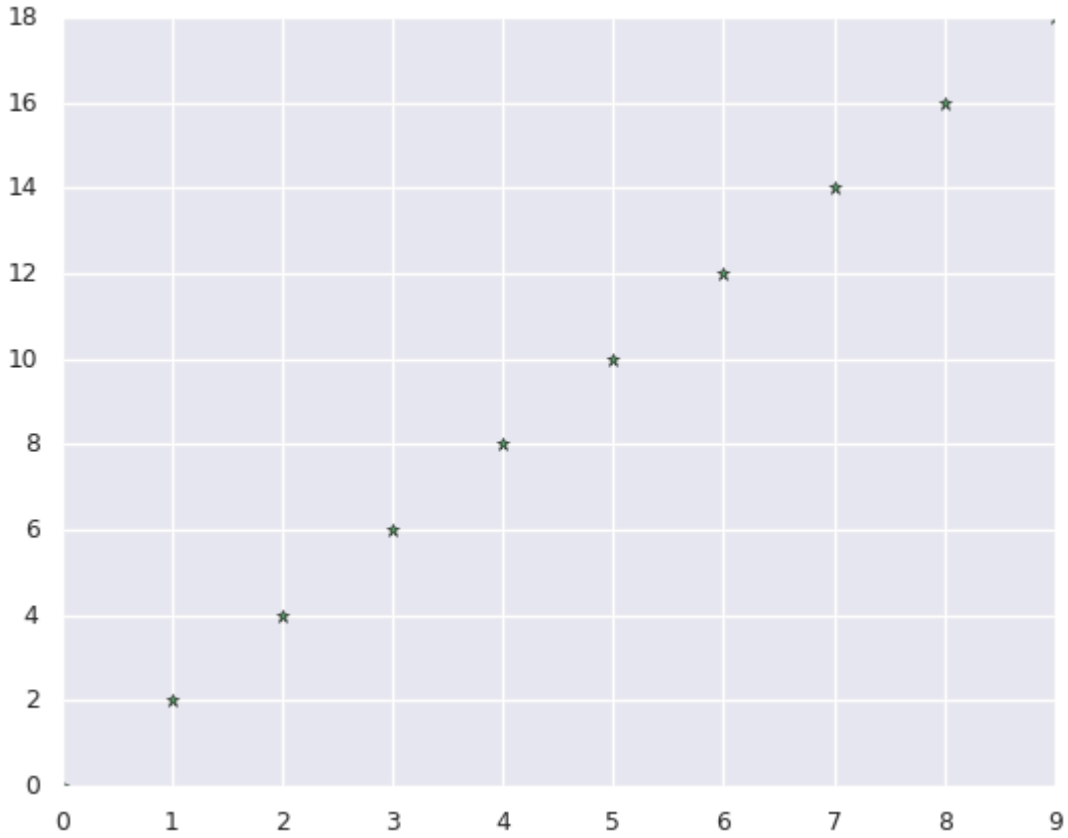
```
[[<Axes: > <Axes: >]
 [<Axes: > <Axes: >]]
```



In [147…

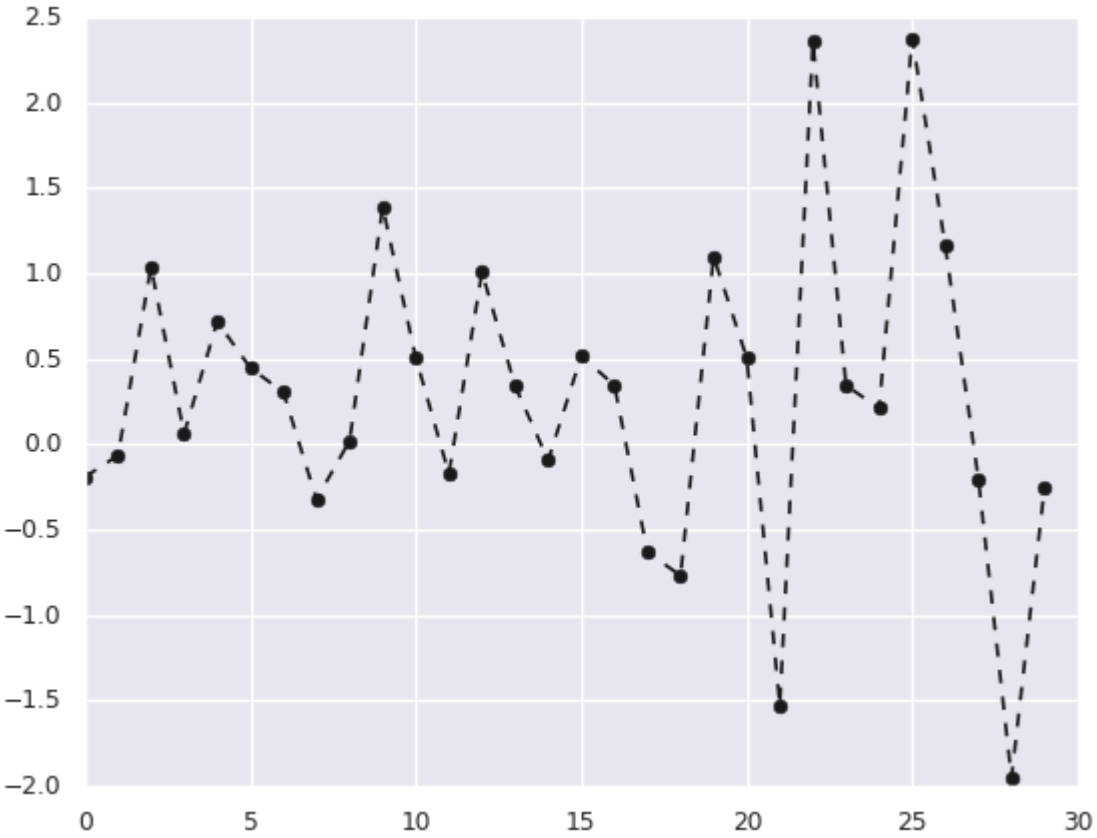
```
fig,p=plt.subplots(1,1)
print(p)
x=np.arange(10)
y=np.arange(10)*2
plt.plot(x,y,"g*")
plt.show()
```

```
Axes(0.125,0.1;0.775x0.8)
```



In [148…

```
fig,p=plt.subplots(1,1)
plt.plot(np.random.randn(30),"ko--",label="one")
plt.show()
```



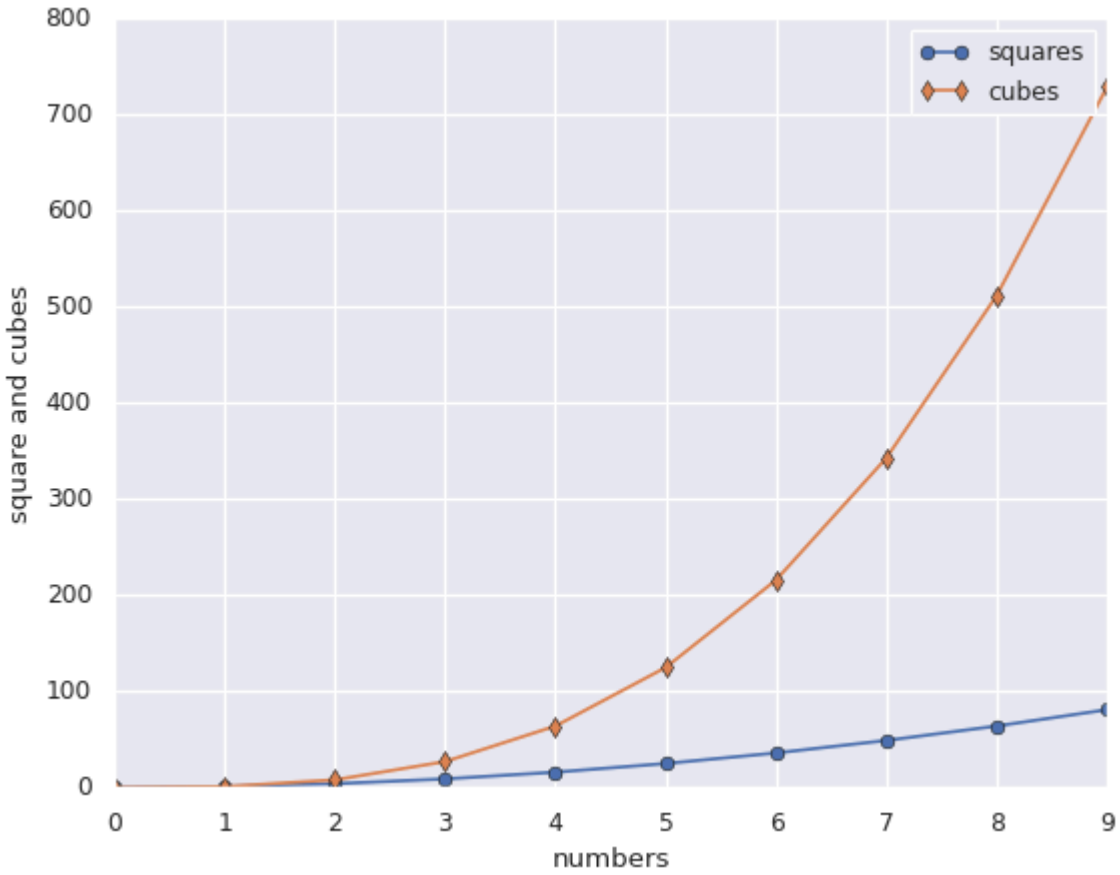
Setting labels and legends

In [149…
```
p.set_xlabel("xaxis")
p.set_ylabel("yaxis")
p.set_title("myplot")
p.legend(loc="best") #search automatically according to its best loaction
```

Out[149]:
```
<matplotlib.legend.Legend at 0x79c69bf08790>
```

In [150…
```
#two plots inside a figure
plt.figure()
x=np.arange(10)
y=x**2
```

```
z=x**3
plt.plot(y,"o-",z,"-d")
plt.xlabel("numbers")
plt.ylabel("square and cubes")
plt.legend(["squares","cubes"])
plt.show()
```

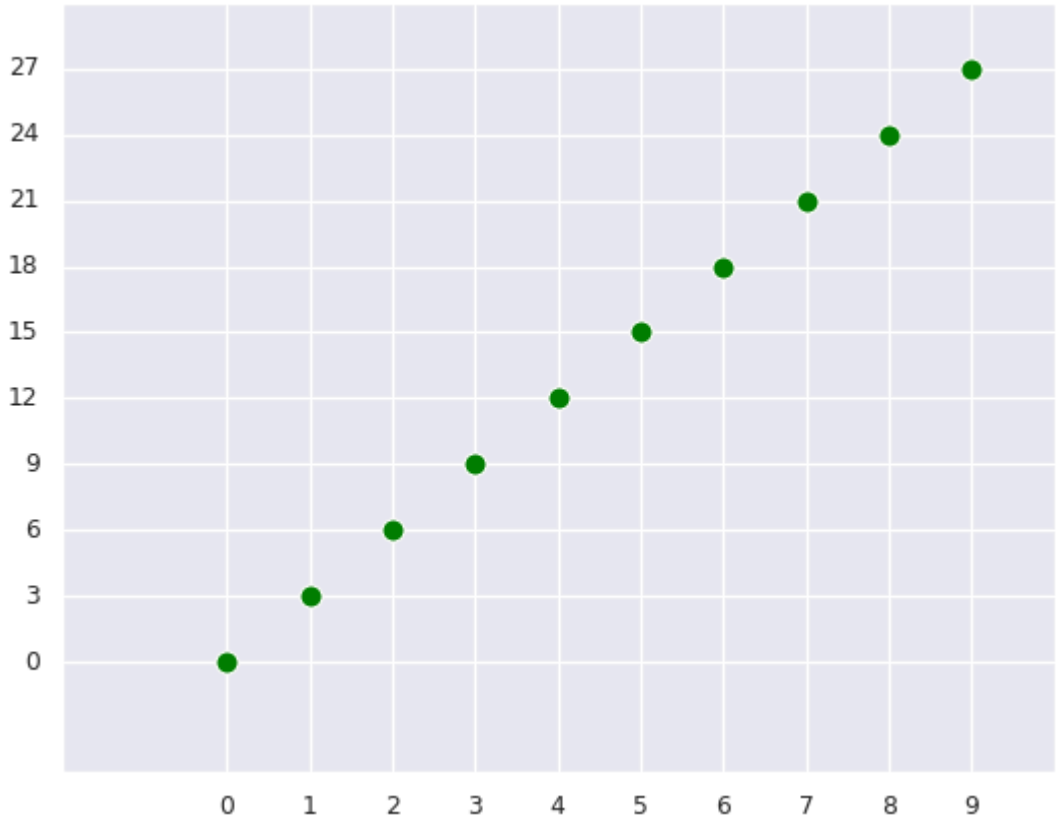


Scatter plot

In [151…

```
plt.figure()
x=np.arange(10)
y=np.arange(10)*3
plt.scatter(x,y,s=100,c="green") #sample =s size of the bullets
```

```
plt.xticks(np.arange(0,10,1)) # ticks
plt.yticks(np.arange(0,30,3))
plt.show()
```

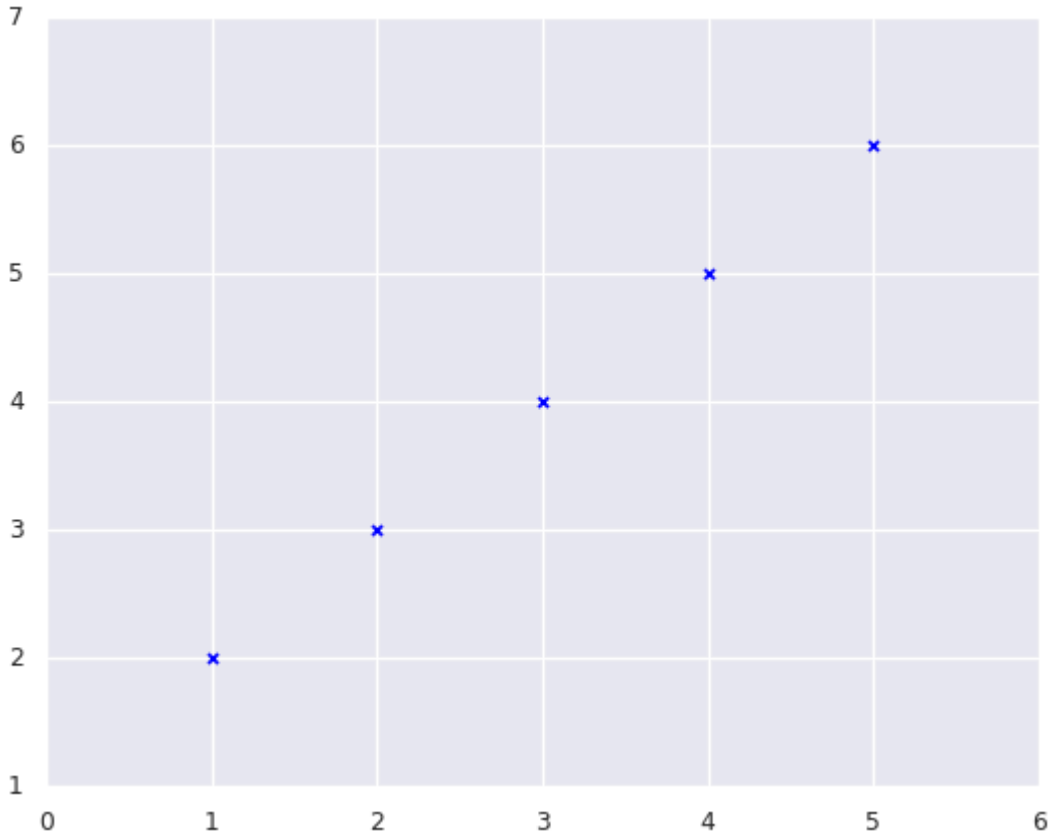


In [152…

```
x = [1, 2, 3, 4, 5]
y = [2, 3, 4, 5, 6]

plt.scatter(x,y,color="blue",marker='x')
```

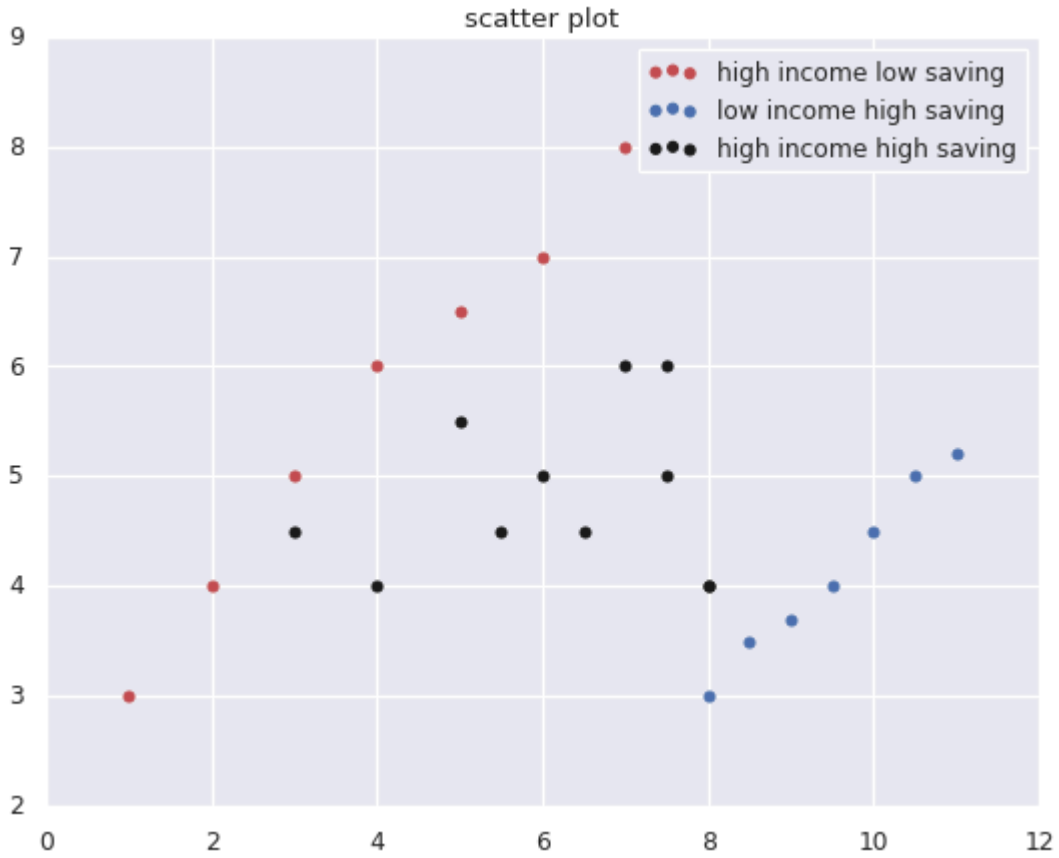
Out[152]:

```
<matplotlib.collections.PathCollection at 0x79c69d2066e0>
```



In [153…

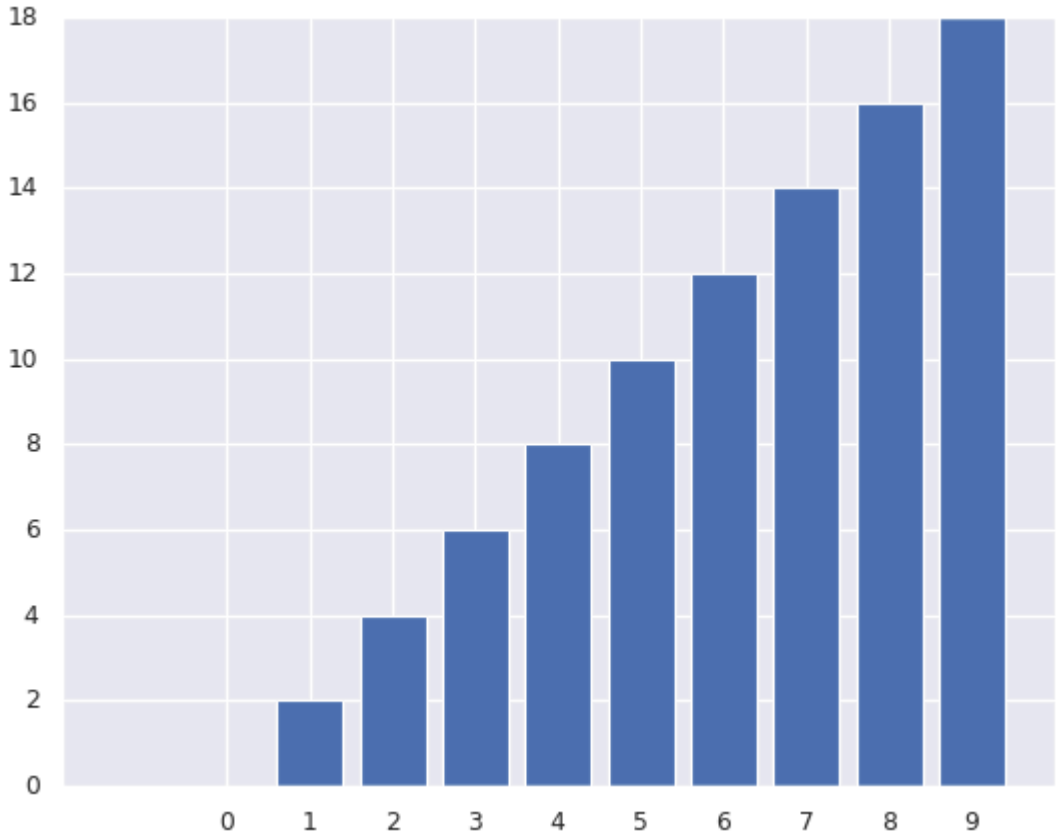
```
plt.figure()
x=[1,2,3,4,5,6,7]
y=[3,4,5,6,6.5,7,8]
x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]
x2=[4,3,6,7,7.5,8,6.5,8,7.5,5,5.5]
y2=[4,4.5,5,6,5,4,4.5,4,6,5.5,4.5]
plt.scatter(x,y,label="high income low saving",color="r")
plt.scatter(x1,y1,label="low income high saving",color="b")
plt.scatter(x2,y2,label="high income high saving", color="k")
plt.title("scatter plot")
plt.legend()
plt.show()
```



In [154…

```
plt.figure()
x = np.arange(10)
y = x * 2
plt.bar(x,y,width = 0.8)
plt.xticks(x)
plt.yticks(y)
```

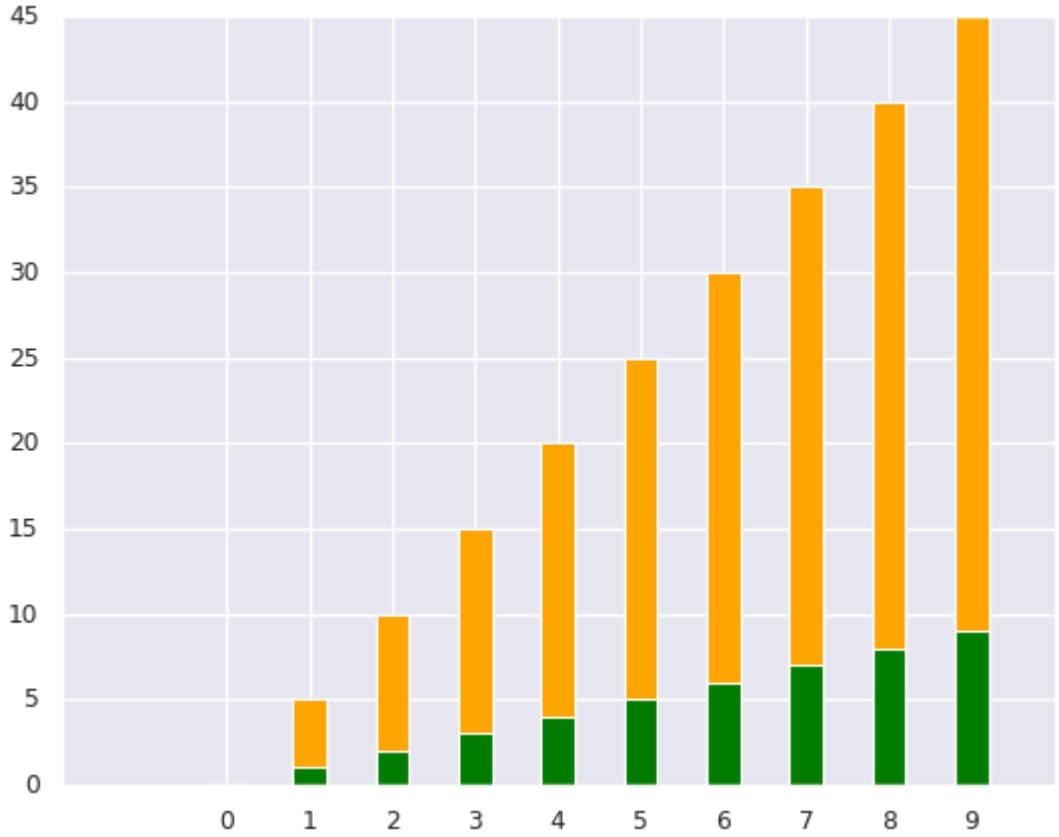
Out[154]:
```
([<matplotlib.axis.YTick at 0x79c69cabe500>,
  <matplotlib.axis.YTick at 0x79c69cabfd30>,
  <matplotlib.axis.YTick at 0x79c69c589600>,
  <matplotlib.axis.YTick at 0x79c69bc37130>,
  <matplotlib.axis.YTick at 0x79c69bc34250>,
  <matplotlib.axis.YTick at 0x79c69bc36fb0>,
  <matplotlib.axis.YTick at 0x79c69bc35b70>,
  <matplotlib.axis.YTick at 0x79c69c086f80>,
  <matplotlib.axis.YTick at 0x79c69bc35900>,
  <matplotlib.axis.YTick at 0x79c69bc34bb0>],
 [Text(0, 0, '0'),
  Text(0, 2, '2'),
  Text(0, 4, '4'),
  Text(0, 6, '6'),
  Text(0, 8, '8'),
  Text(0, 10, '10'),
  Text(0, 12, '12'),
  Text(0, 14, '14'),
  Text(0, 16, '16'),
  Text(0, 18, '18')])
```



In [155…

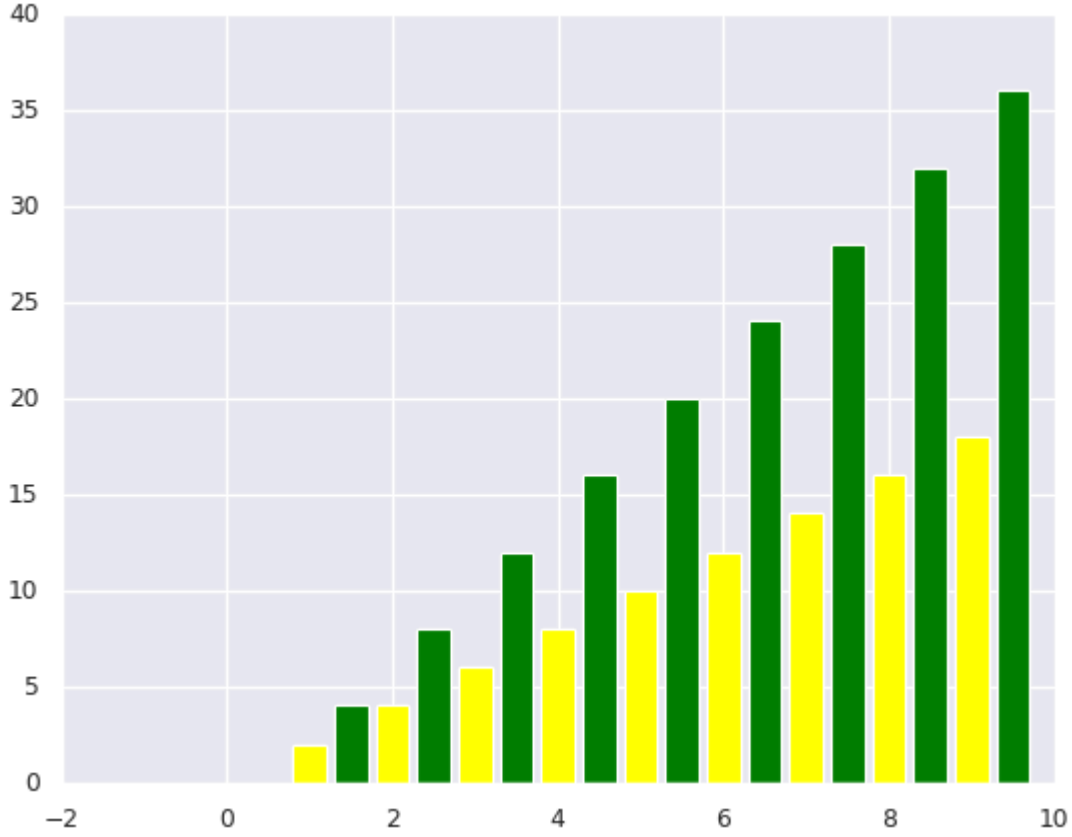
```
plt.figure()
z = x * 4
plt.bar(x,y,width = 0.4,color = 'green')
plt.bar(x,z,width = 0.4,bottom = x,color = 'orange')
plt.xticks(x)
```

Out[155]:
```
([<matplotlib.axis.XTick at 0x79c69c198c70>,
  <matplotlib.axis.XTick at 0x79c69c19a050>,
  <matplotlib.axis.XTick at 0x79c69c1998d0>,
  <matplotlib.axis.XTick at 0x79c69c12d300>,
  <matplotlib.axis.XTick at 0x79c69c12d9f0>,
  <matplotlib.axis.XTick at 0x79c69c2aa1a0>,
  <matplotlib.axis.XTick at 0x79c69c12c970>,
  <matplotlib.axis.XTick at 0x79c69c2ab340>,
  <matplotlib.axis.XTick at 0x79c69c2abeb0>,
  <matplotlib.axis.XTick at 0x79c69c2aa860>],
 [Text(0, 0, '0'),
  Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5'),
  Text(6, 0, '6'),
  Text(7, 0, '7'),
  Text(8, 0, '8'),
  Text(9, 0, '9')])
```



In [156…
```
plt.figure()
x = np.arange(10)
y = x * 2
plt.bar(x,y,width = 0.4,color = 'yellow')
plt.bar(x+0.5,y*2,width = 0.4,color='green')
```

Out[156]:
```
<BarContainer object of 10 artists>
```



Histograms

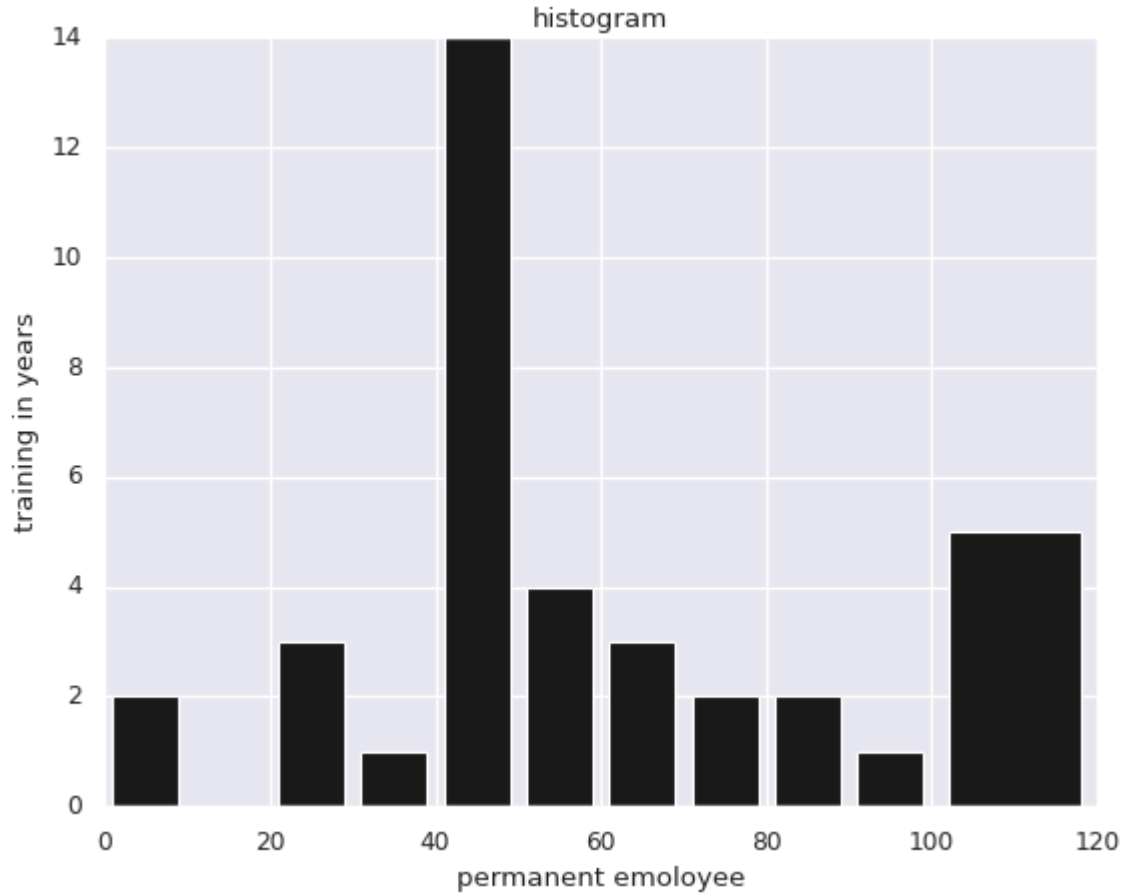
- Histograms are used to represent the distribution of a dataset. They divide the data into bins and count the number of data points in each bin.
- This is a graphical representation of numerical data and it is graphing tool that summarises discreate and continous data.
- Histogram are used to illustrate the major features of the distributuion of data.
- Histogram is a graph tha shows thee frequency of numerical data using rectangles.

When to use histogram?

- Data should be numerical data.
- Histogram is use to check the shape of the data distribution
- Use to check weather the process from one period to another.
- Use to determine weather the output is different when it involves to a more processes.
- Use to analyse weather a given process meets the customer requierments.

In [157…

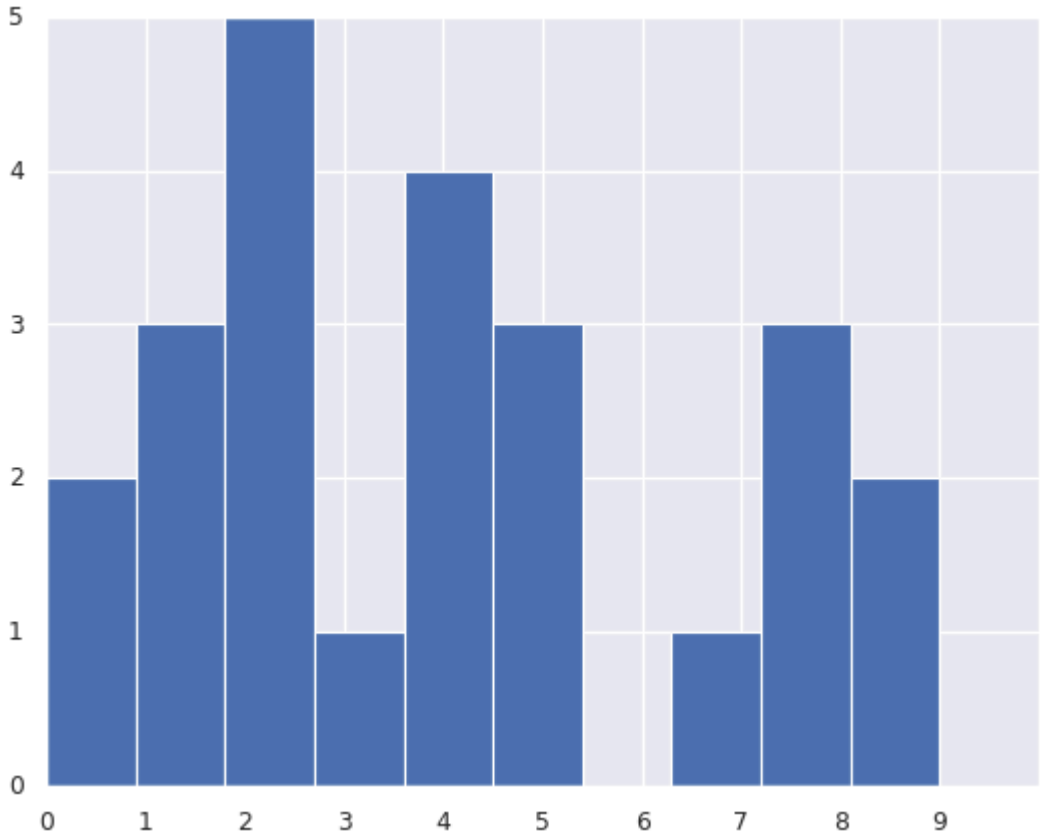
```
plt.figure()
population_age=[22,55,62,45,21,22,34,42,42,4,2,102,95,85,55,110,120,70,65,55,111,115,80,75,65,54,44,43,42,48,40,40,40,41,42,45,49
bins=[0,10,20,30,40,50,60,70,80,90,100,120]
plt.hist(population_age,bins,color="k",rwidth=0.8) #rwidth is the relative width
plt.xlabel("permanent emoloyee")
plt.ylabel("training in years")
plt.title("histogram")
plt.show()
```



In [158…

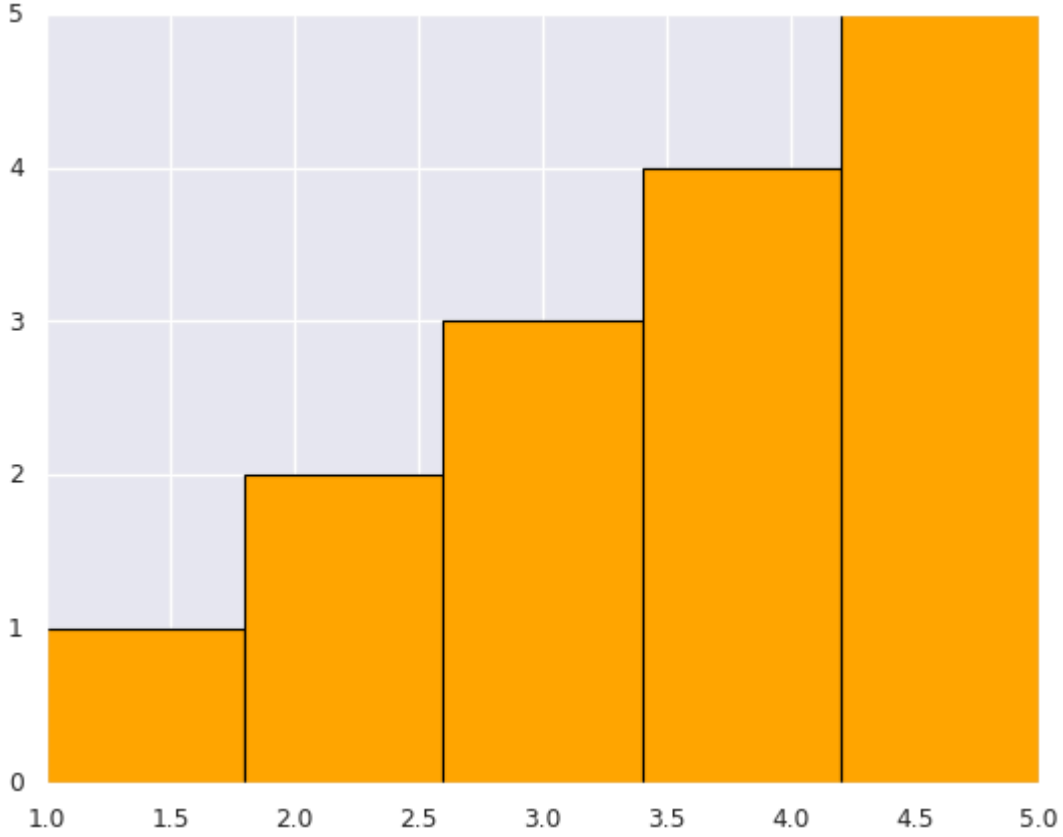
```
plt.figure()
plt.hist([[1,2,3,4,5,2,4,7,8,4,5,2,1,8,8,0,9,0,9,2,5,2,1,4]])
plt.xticks(range(10))
```

Out[158]:
```
([<matplotlib.axis.XTick at 0x79c69b910df0>,
  <matplotlib.axis.XTick at 0x79c69b910dc0>,
  <matplotlib.axis.XTick at 0x79c6c47357b0>,
  <matplotlib.axis.XTick at 0x79c69babd360>,
  <matplotlib.axis.XTick at 0x79c69babc7f0>,
  <matplotlib.axis.XTick at 0x79c69babf0a0>,
  <matplotlib.axis.XTick at 0x79c69babffa0>,
  <matplotlib.axis.XTick at 0x79c69babf460>,
  <matplotlib.axis.XTick at 0x79c69babe920>,
  <matplotlib.axis.XTick at 0x79c69baf5f90>],
 [Text(0, 0, '0'),
  Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5'),
  Text(6, 0, '6'),
  Text(7, 0, '7'),
  Text(8, 0, '8'),
  Text(9, 0, '9')])
```



In [159…
```
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
plt.hist(data,bins=5,color='orange',edgecolor='black')
```

Out[159]:
```
(array([1., 2., 3., 4., 5.]),
 array([1. , 1.8, 2.6, 3.4, 4.2, 5. ]),
 <BarContainer object of 5 artists>)
```



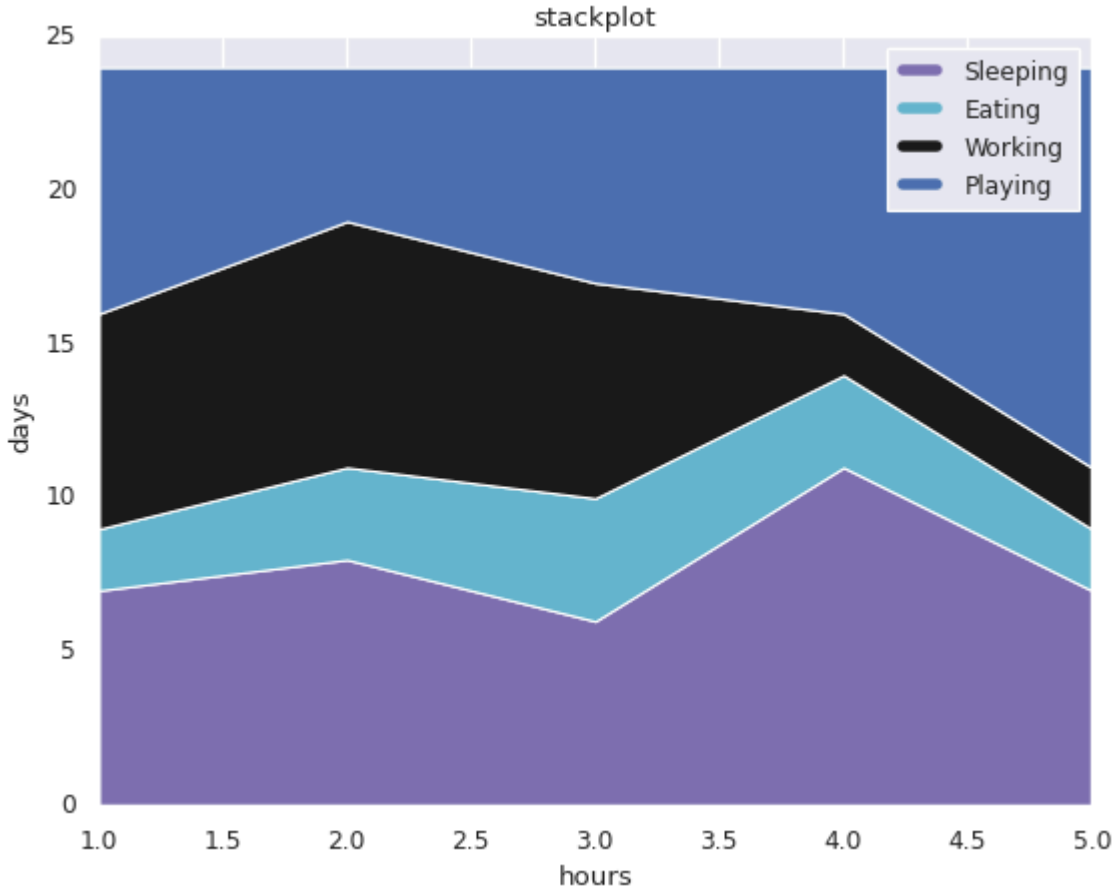
Area Plotting

- Area plot is made by plotting a series of data points over time, connecting those data points with line segments and then filling in the area bt the line and the x axis with color or shading.
- Function - stackplot()

In [160…

```
plt.figure()
days = [1,2,3,4,5]
sleeping =[7,8,6,11,7]
eating = [2,3,4,3,2]
working =[7,8,7,2,2]
```

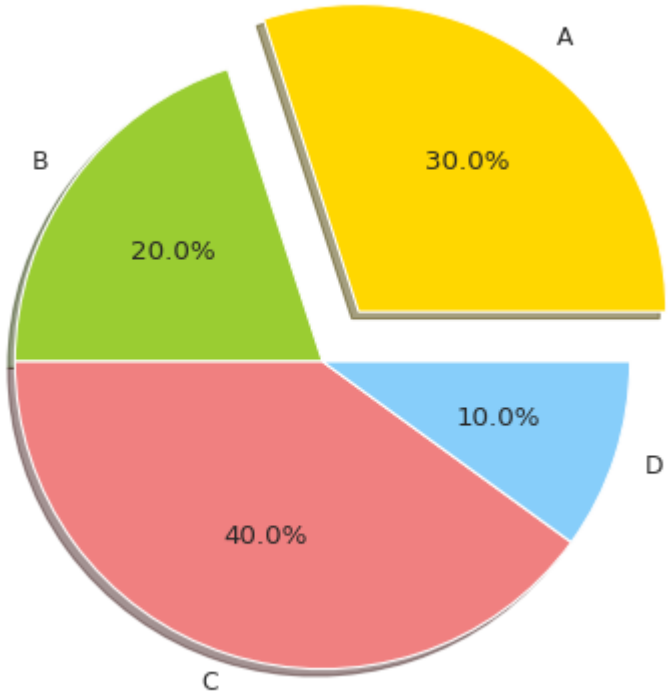
```
playing=[8,5,7,8,13]
plt.plot([],[],color='m', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='k', label='Working', linewidth=5)
plt.plot([],[],color='b', label='Playing', linewidth=5)
plt.stackplot(days,sleeping,eating,working,playing,colors=["m","c","k","b"])
plt.xlabel("hours")
plt.ylabel("days")
plt.title("stackplot")
plt.legend()
plt.show()
```



Pie Chart

In [161…
```
labels=['A','B','C','D']
sizes=[30,20,40,10]
colors=['gold','yellowgreen','lightcoral','lightskyblue']
explode=(0.2,0,0,0) ##move out the 1st slice

##create apie chart
plt.pie(sizes,explode=explode,labels=labels,colors=colors,autopct="%1.1f%%",shadow=True)
```

Out[161]:
```
([<matplotlib.patches.Wedge at 0x79c69b8459f0>,
  <matplotlib.patches.Wedge at 0x79c69b845930>,
  <matplotlib.patches.Wedge at 0x79c69b846b60>,
  <matplotlib.patches.Wedge at 0x79c69b847430>],
 [Text(0.764120788592483, 1.051722121304293, 'A'),
  Text(-0.8899187482945419, 0.6465637025335369, 'B'),
  Text(-0.3399185762739153, -1.046162206115244, 'C'),
  Text(1.0461622140716127, -0.3399185517867209, 'D')],
 [Text(0.47022817759537416, 0.6472136131103341, '30.0%'),
  Text(-0.4854102263424773, 0.3526711104728383, '20.0%'),
  Text(-0.1854101325130447, -0.5706339306083149, '40.0%'),
  Text(0.5706339349481523, -0.18541011915639322, '10.0%')])
```

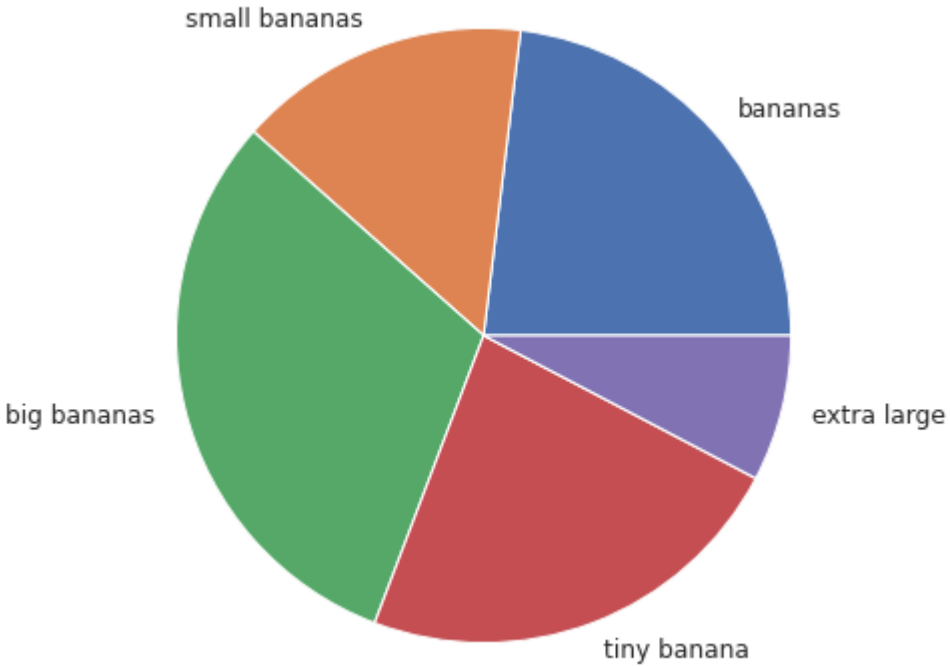


In [162…
```
plt.figure()
x=np.array([10,30,20,15,25])
plt.pie(x)
```

Out[162]:
```
([<matplotlib.patches.Wedge at 0x79c69b8bd0c0>,
  <matplotlib.patches.Wedge at 0x79c69b8bcfd0>,
  <matplotlib.patches.Wedge at 0x79c69b8bd990>,
  <matplotlib.patches.Wedge at 0x79c69b8bde10>,
  <matplotlib.patches.Wedge at 0x79c69b8be290>],
 [Text(1.0461621663333946, 0.3399186987098808, ''),
  Text(-5.149471629032507e-08, 1.0999999999999988, ''),
  Text(-1.0999999999999954, -1.0298943258065002e-07, ''),
  Text(-0.49938943041996464, -0.9801072373902886, ''),
  Text(0.7778175685419842, -0.777817350068405, '')])
```

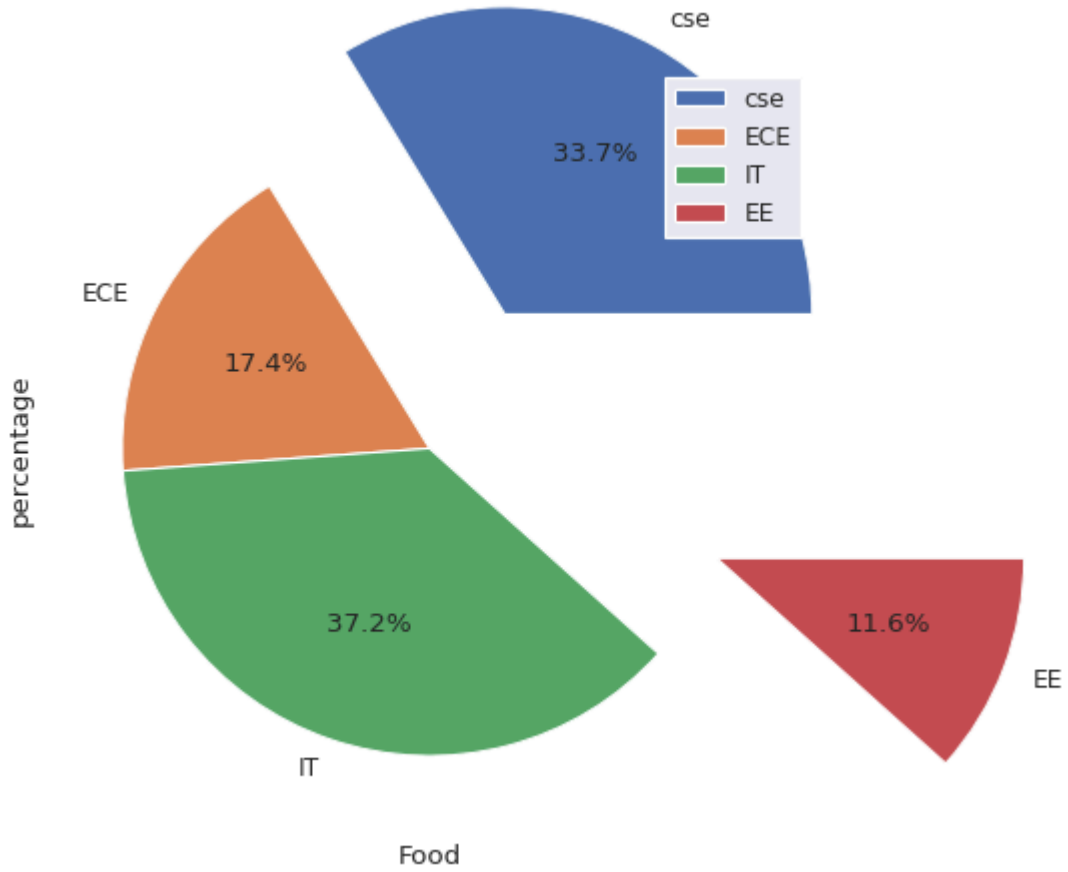
In [163…
```
y=np.array([30,20,40,30,10])
food=["bananas","small bananas","big bananas","tiny banana","extra large"]
plt.pie(y,labels=food)
```

Out[163]:
```
([<matplotlib.patches.Wedge at 0x79c69b71c430>,
  <matplotlib.patches.Wedge at 0x79c69b71c340>,
  <matplotlib.patches.Wedge at 0x79c69b71cbb0>,
  <matplotlib.patches.Wedge at 0x79c69b71d060>,
  <matplotlib.patches.Wedge at 0x79c69b71d4e0>],
 [Text(0.8233618203614949, 0.7294349270298247, 'bananas'),
  Text(-0.3900654016728485, 1.0285178571214986, 'small bananas'),
  Text(-1.0680359783135132, -0.2632473153289448, 'big bananas'),
  Text(0.3900654979696379, -1.0285178206009358, 'tiny banana'),
  Text(1.0680360337692223, -0.2632470903362633, 'extra large')])
```



In [164…

```
branches=["cse","ECE","IT","EE"]
students_counts=[29,15,32,10]
exp=(0.5,0,0,1)
plt.pie(students_counts,labels=branches,explode=exp,autopct="%1.1f%%") #autopct help in determining the percentage of the branch.
plt.xlabel("Food")
plt.ylabel("percentage")
plt.legend()
plt.show()
```



Data Visulization using Seaborn

What is Python Seaborn?

- Seaborn is a library for making statistical graphics in Python.
- It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

Seaborn is a powerful Python library built on top of Matplotlib, specifically designed for statistical data visualization. Here are some of the key functionalities Seaborn offers:

1. Visualizing Distributions

distplot() (Deprecated) / histplot(): Visualizes the distribution of a single variable with options for histograms, kernel density estimation (KDE), or both. kdeplot(): Plots the kernel density estimate of a continuous variable. rugplot(): Adds small tick marks at each observation along the x-axis, providing a visual representation of data points.

1. Categorical Plots barplot(): Creates bar charts, showing mean values with confidence intervals. countplot(): Displays the count of observations in each categorical bin using bars. boxplot(): Draws box-and-whisker plots, useful for displaying distributions of data and detecting outliers. violinplot(): Combines aspects of boxplot and KDE to display both distribution shape and statistical summary. **stripplot() & swarmplot(): Show individual data points for categories, with swarmplot avoiding overlap.
2. Relational Plots scatterplot(): A simple scatter plot to visualize the relationship between two continuous variables. lineplot(): Visualizes the relationship between two variables, particularly useful for time series or sequential data. relplot(): A higher-level interface that can create scatter or line plots, allowing for easy faceting (splitting by additional categorical variables).
3. Matrix Plots heatmap(): Creates a color-coded matrix to visualize data, typically used for correlation matrices or displaying values across two dimensions. clustermap(): Generates a clustered heatmap with hierarchical clustering of rows and columns.
4. Pairwise Relationships pairplot(): Visualizes pairwise relationships between variables in a dataset by plotting scatter plots or KDE plots for all variable combinations. jointplot(): Combines a scatter plot with marginal histograms (or KDE) to show the relationship between two variables along with their distributions.
5. Regression Plots regplot(): Fits and visualizes a simple linear regression model between two variables. lmplot(): A higher-level interface for drawing regression plots with options for faceting by categorical variables. residplot(): Visualizes the residuals (errors) from a linear regression model.
6. Facet Grids FacetGrid(): Allows you to create multiple plots for subsets of data by mapping categorical variables to rows, columns, or color. catplot(): A simplified interface for drawing categorical plots on a FacetGrid.
7. Customizable Aesthetics Seaborn makes it easy to style plots with options like changing color palettes (sns.set_palette()), themes (sns.set_theme()), and adding grid lines.

In [165…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# enable inline plotting where graphs or plots will be displayed just below the cell
#you are writting the plotting commands
```

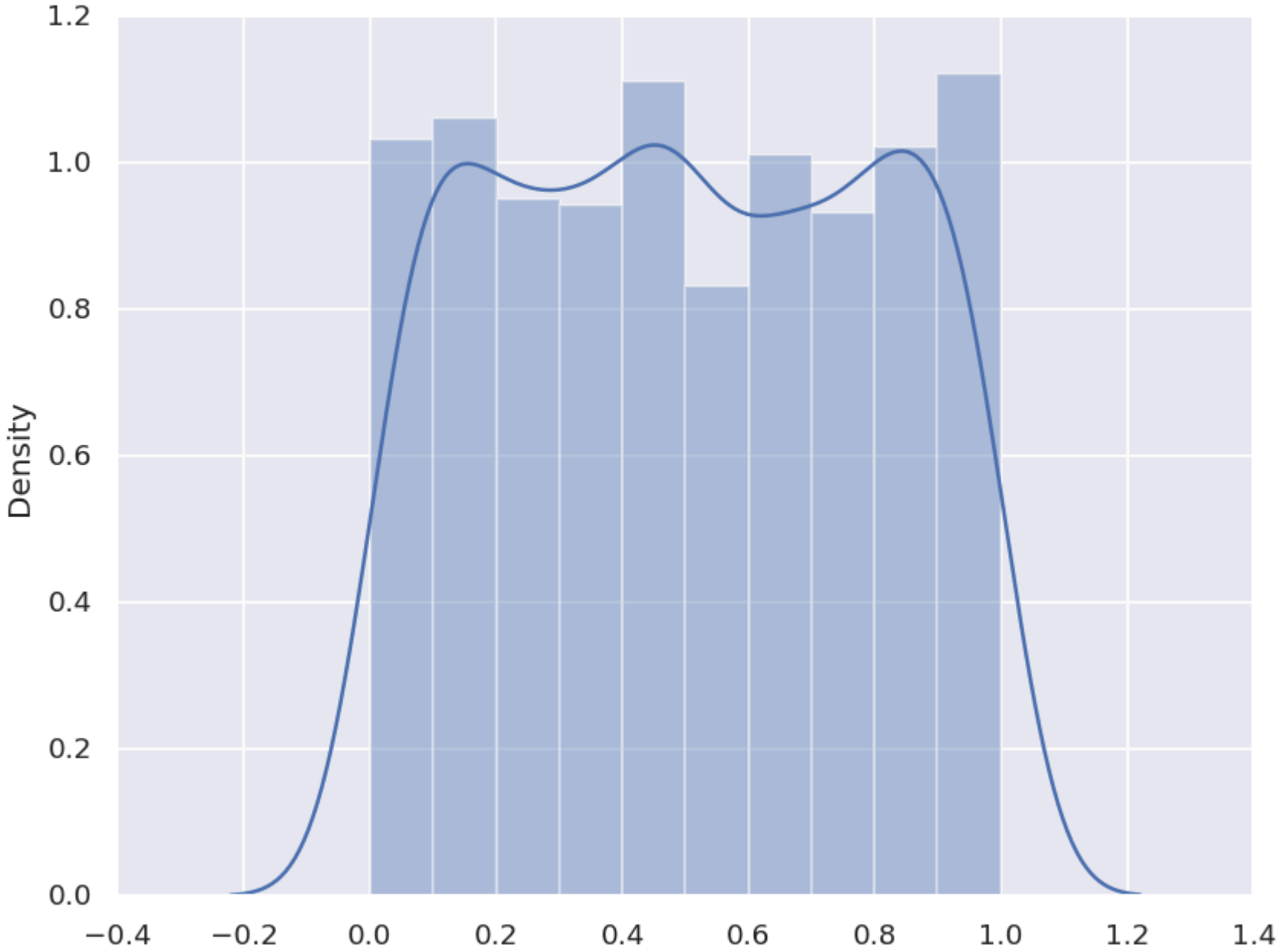
In [166…

```
x=np.random.random(1000)
plt.figure(dpi=120)
sns.distplot(x) #distributed plot (distribution plot)
```

```
<ipython-input-166-1016b1e8f586>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x) #distributed plot (distribution plot)
```

Out[166]:

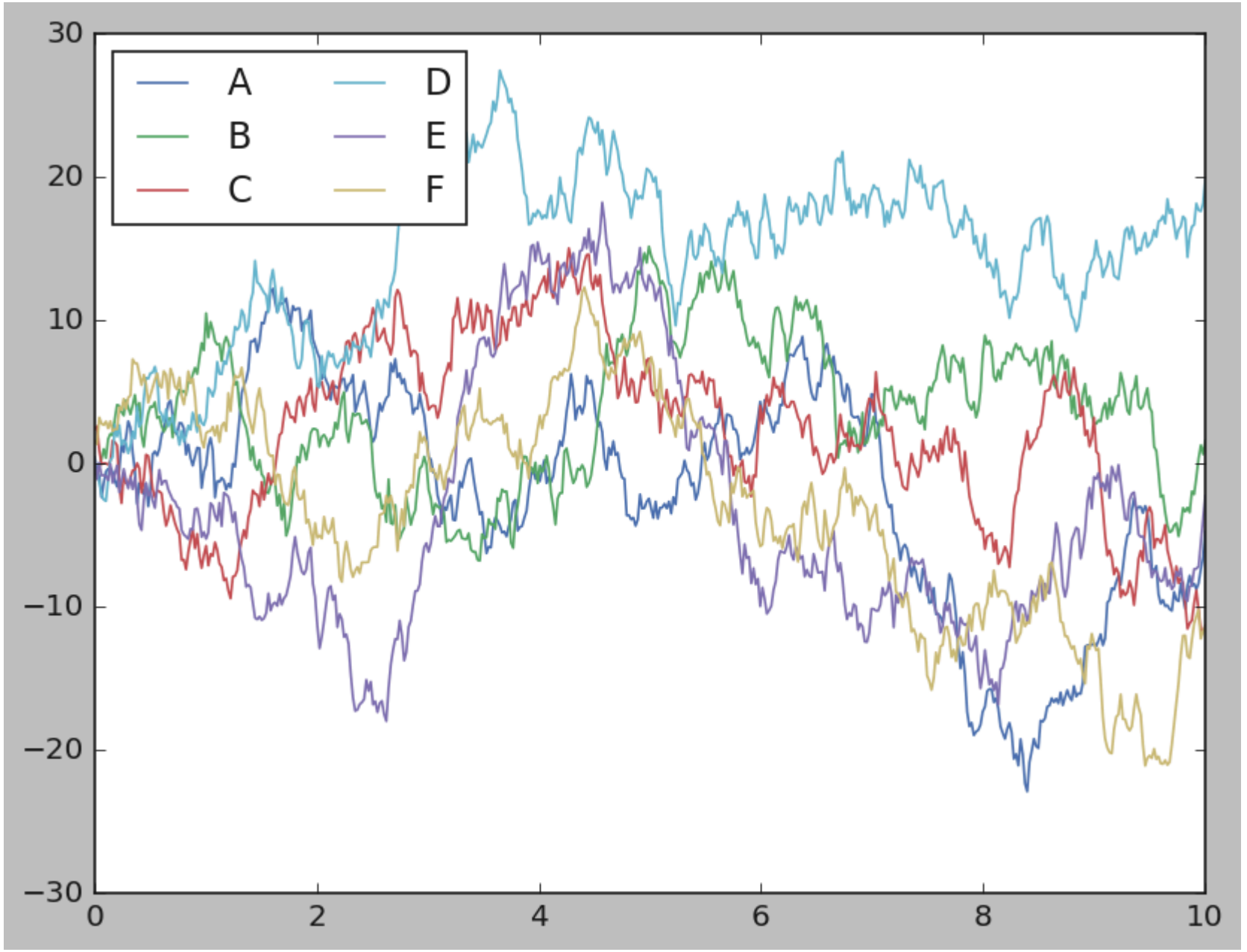
```
<Axes: ylabel='Density'>
```


1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

In [167…
```
import matplotlib.pyplot as plt
plt.style.use("classic")
%matplotlib inline
```

In [168…
```
rng=np.random.RandomState(5)
x=np.linspace(0,10,500)
y=np.cumsum(rng.randn(500,6),0) #range of randn -1 to 1
```

In [169…
```
plt.figure(dpi=120)
plt.plot(x,y)
plt.legend("ABCDEF",ncol=2,loc="upper left")
plt.show()
```



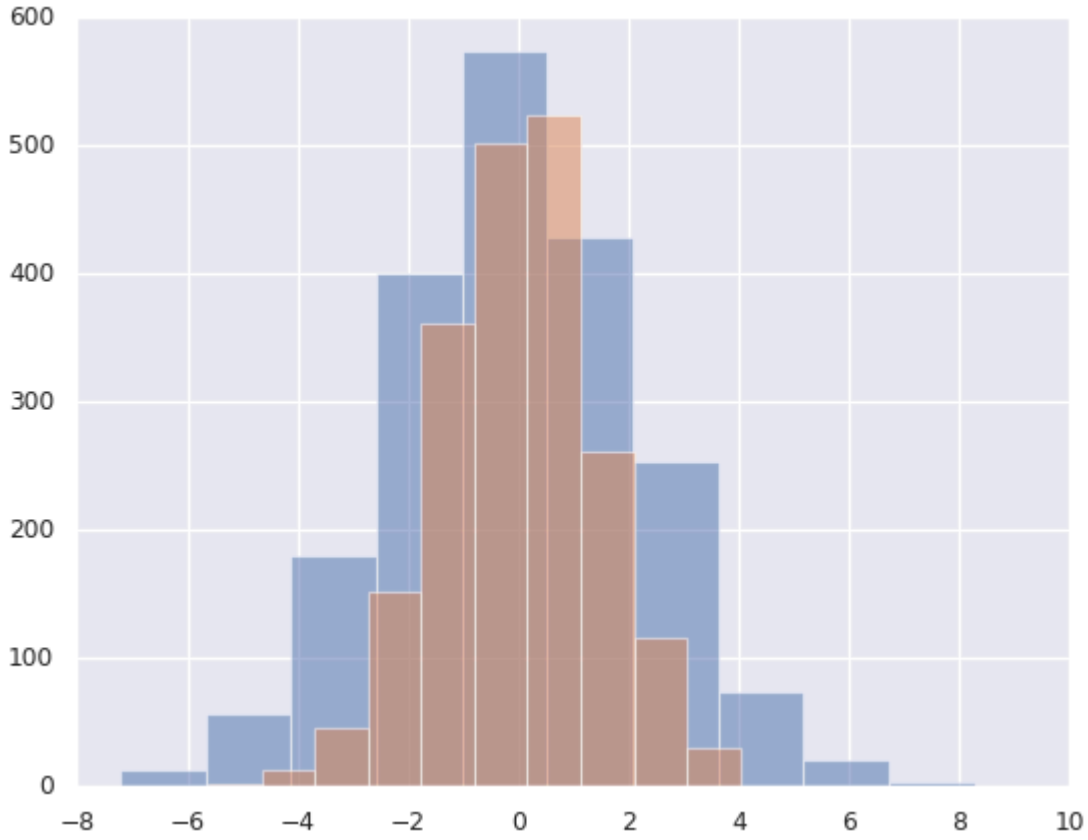
In [170…

```
sns.set()
plt.plot(x,y)
```

```
plt.legend("ABCDEF",ncol=2,loc="upper left")
plt.show()
```

In [171…

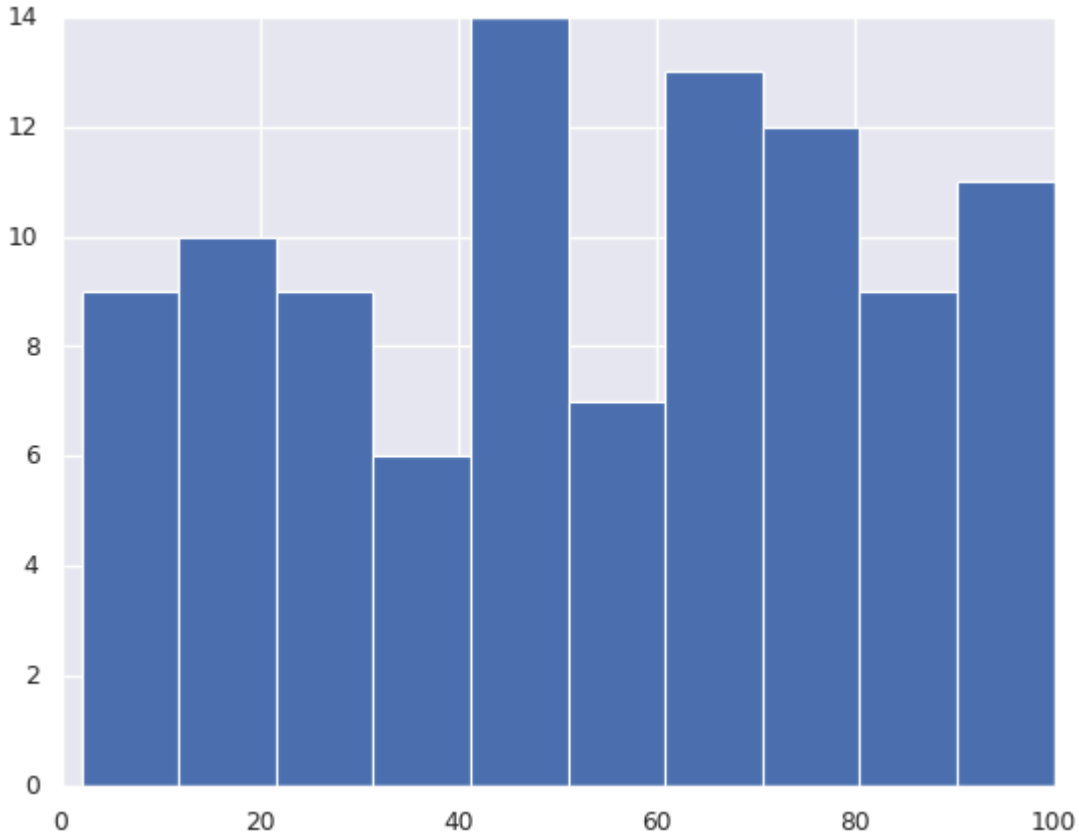
```
#histogram, kde, densities

data=np.random.multivariate_normal([0,0],[[5,2],[2,2]],size=2000)
data1=pd.DataFrame(data,columns=["x","y"])

for item in "xy":
  plt.hist(data1[item],alpha=0.5)
```



In [172…
```
"""bar chart, box plot, line plot, bar horizontal - categorical plot
histo, kde, heatmap - distribution plot"""
```

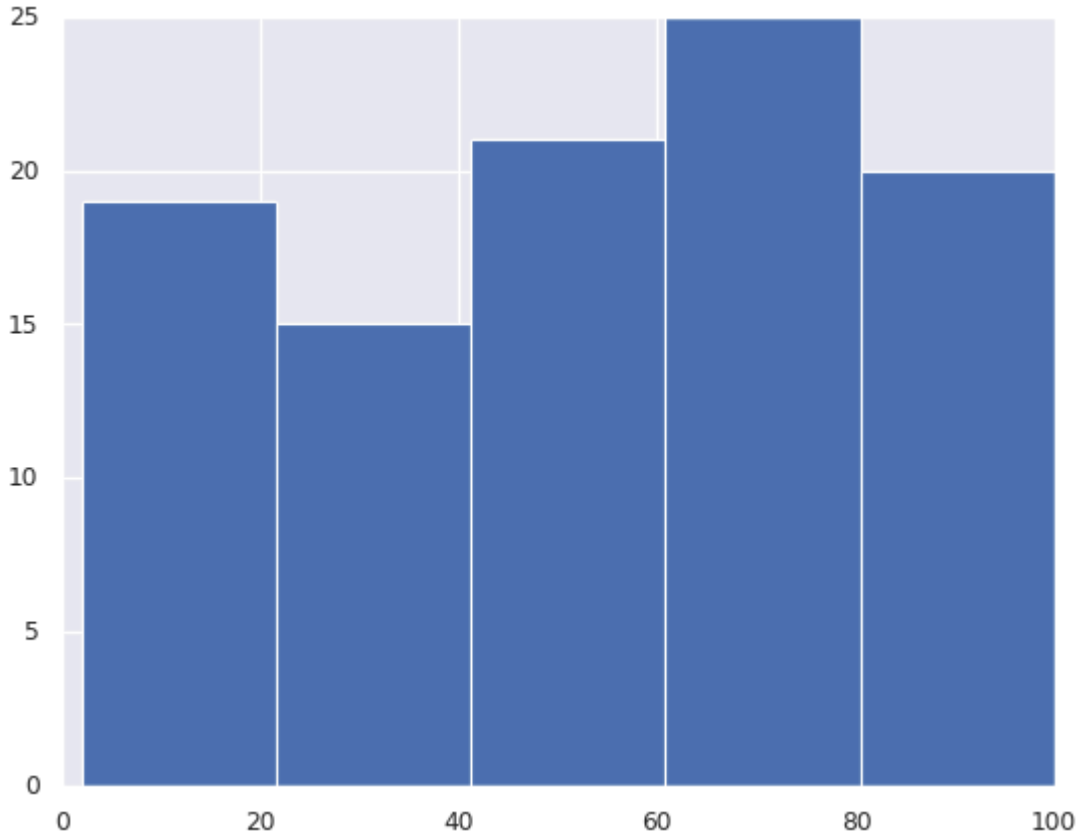
Out[172]:
```
'bar chart, box plot, line plot, bar horizontal - categorical plot\nhisto, kde, heatmap - distribution plot'
```

In [173…
```
marks=np.random.randint(1,101,100)
marks
plt.hist(x=marks)
plt.show()
```



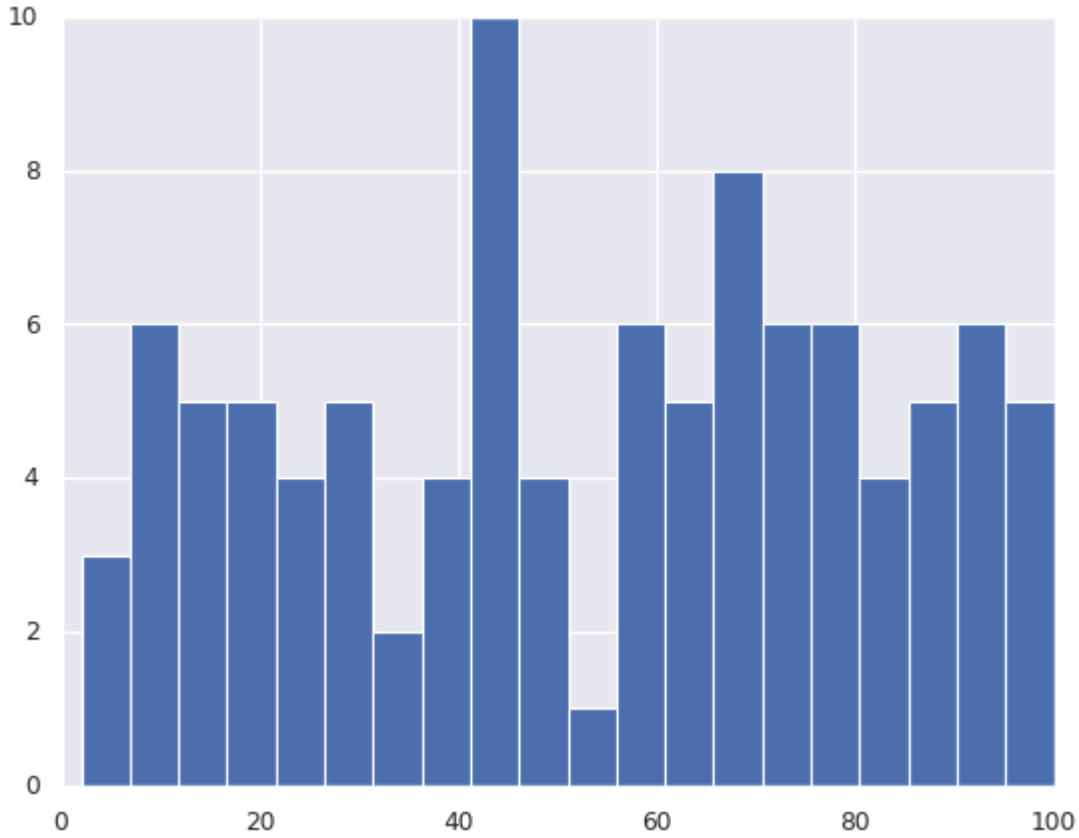
In [174…

```
plt.hist(x=marks,bins=5)
plt.show()
```



In [175…

```
plt.hist(x=marks,bins=20)
plt.show()
```

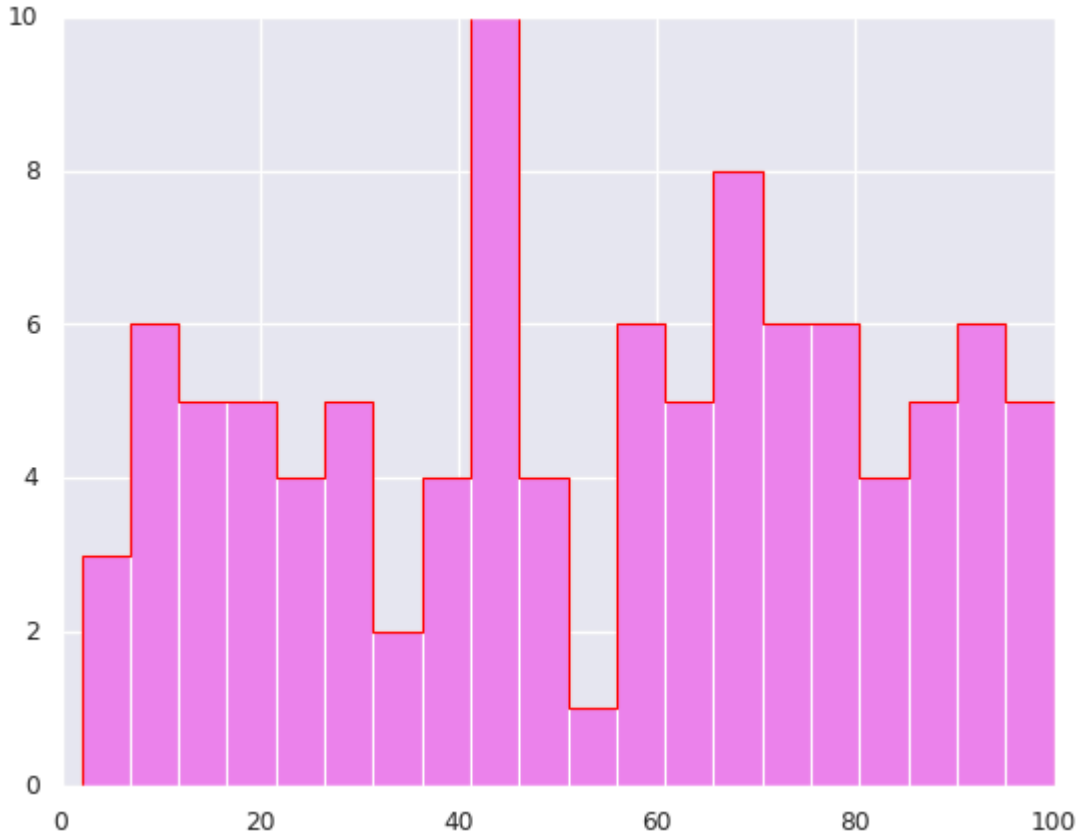


In [176…

```
plt.hist(x=marks,bins=20,histtype="bar",color="violet")
plt.show()
```

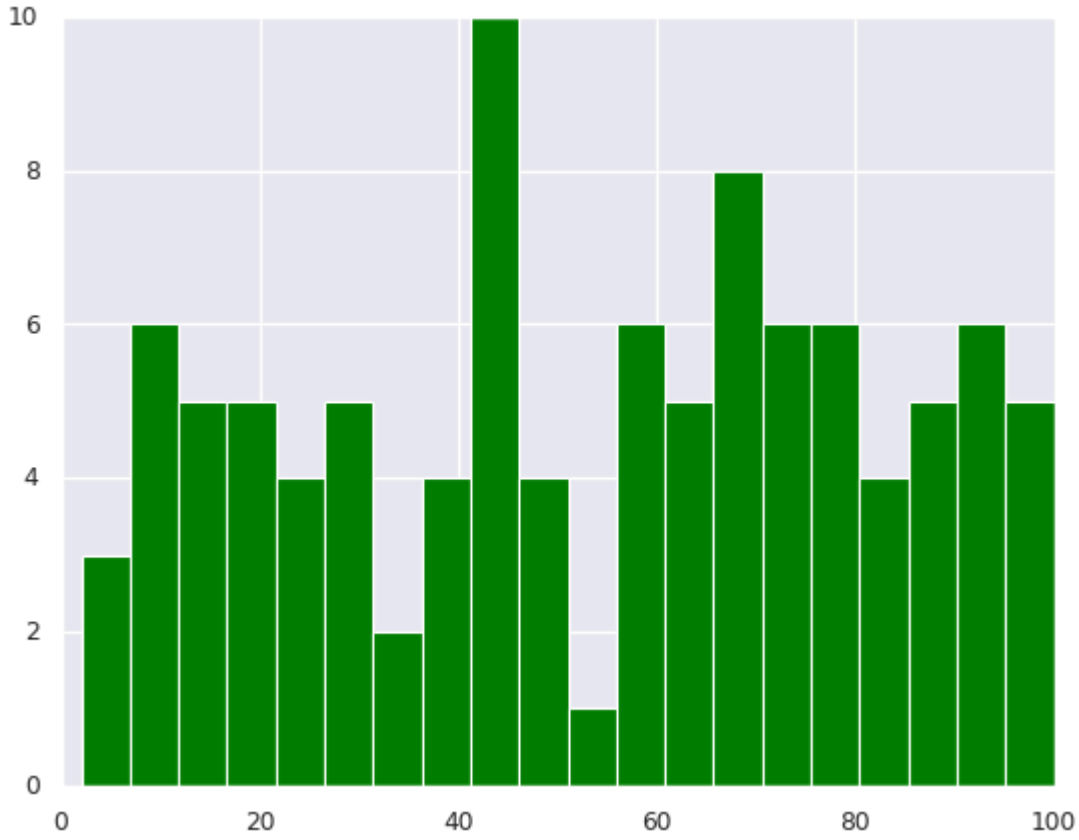
In [177…

```
plt.hist(x=marks,bins=20,histtype="bar",color="violet")
plt.hist(x=marks,bins=20,histtype="step",color="red")
plt.show()
```



In [178…

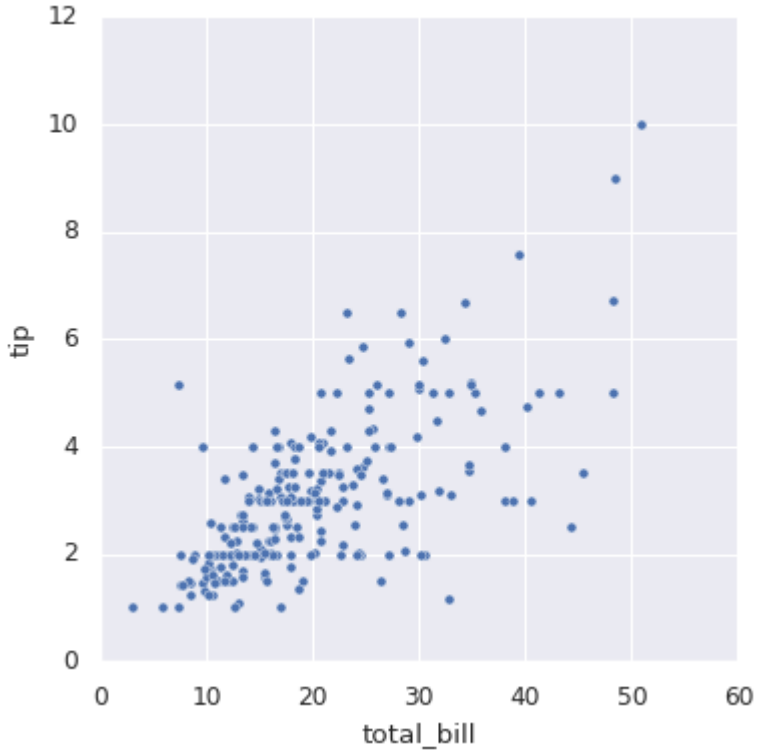
```
plt.hist(x=marks,bins=20,histtype="bar",color="violet") #overclashing into each other
plt.hist(x=marks,bins=20,histtype="bar",color="green")
plt.show()
```



In [179…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(style="darkgrid")
```

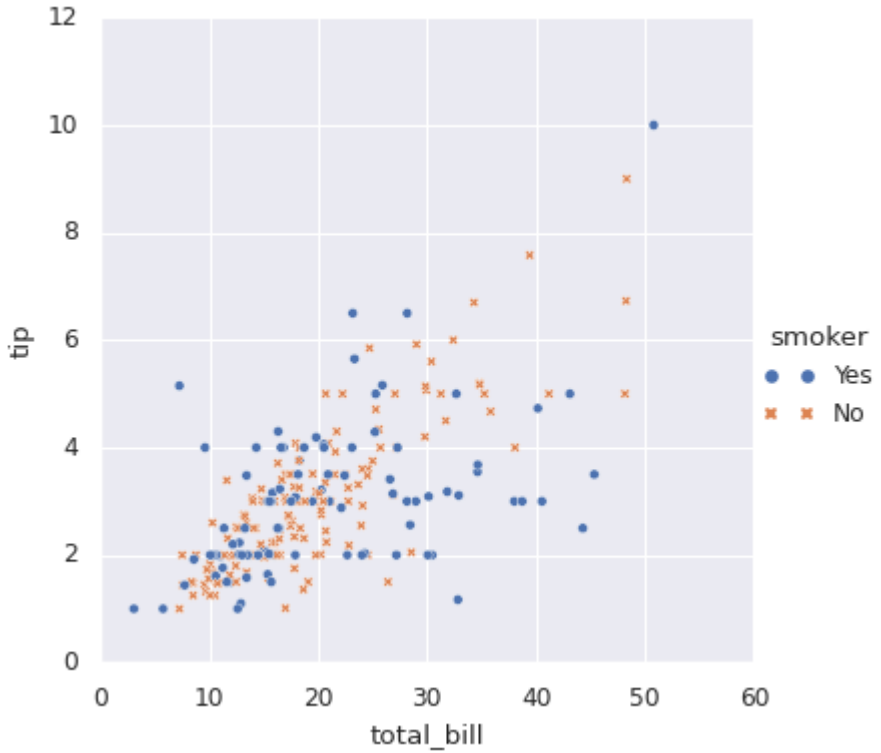
In [180…
```
tips = sns.load_dataset("tips")
sns.relplot(data=tips, x="total_bill", y="tip")
```

Out[180]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b3bd270>
```



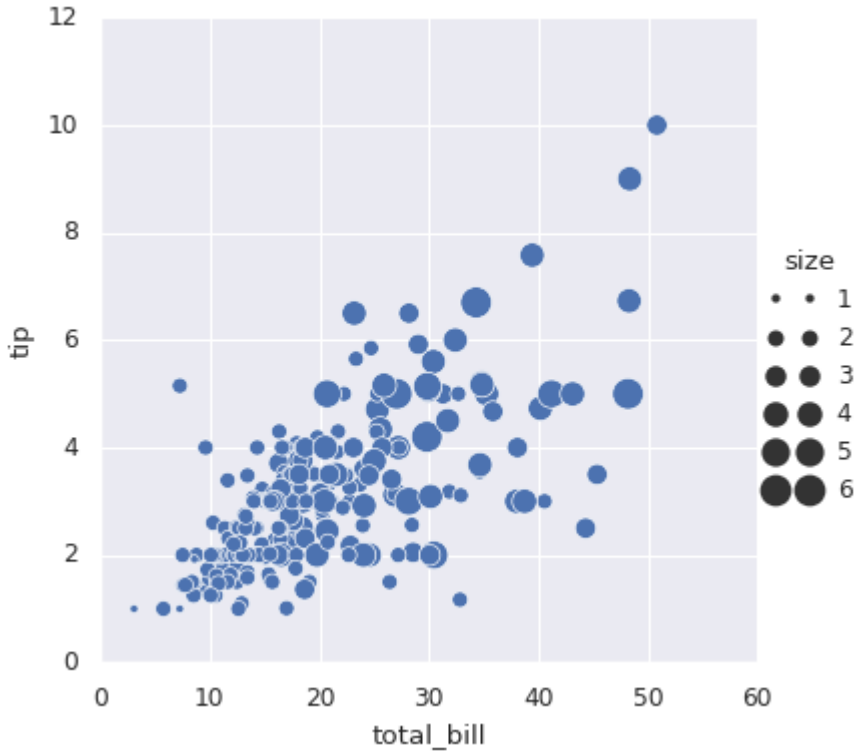
In [181…
```
sns.relplot(
    data=tips,
    x="total_bill", y="tip", hue="smoker", style="smoker"
)
```

Out[181]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b2357e0>
```



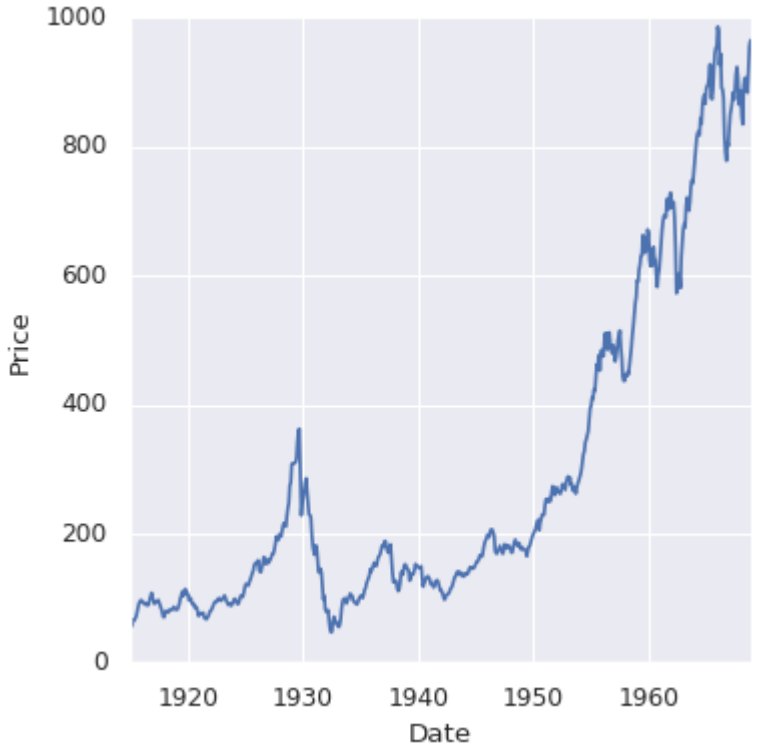
In [182…
```
sns.relplot(
    data=tips, x="total_bill", y="tip",
    size="size", sizes=(15, 200)
)
```

Out[182]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b2b1720>
```



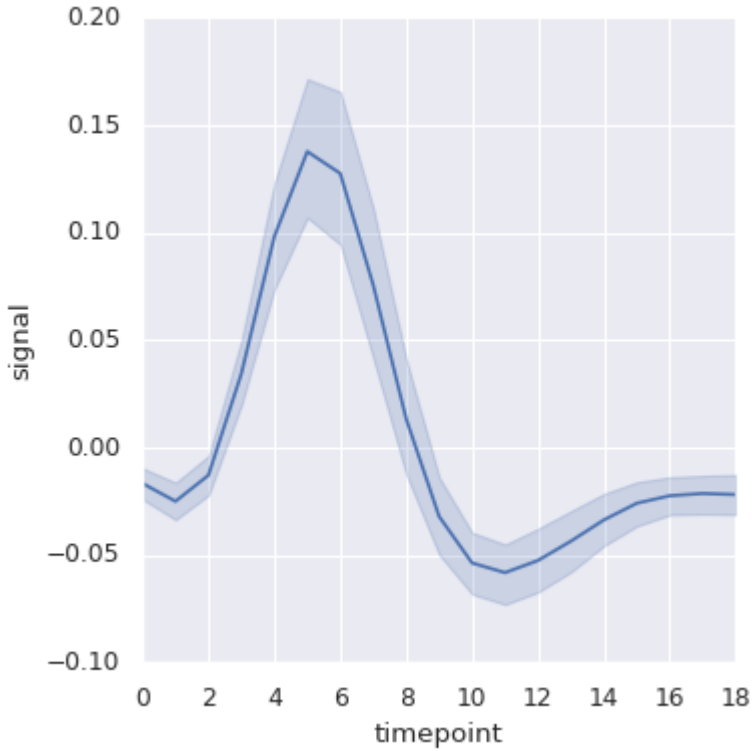
In [183…
```
dowjones = sns.load_dataset("dowjones")
sns.relplot(data=dowjones, x="Date", y="Price", kind="line")
```

Out[183]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b122c20>
```



In [184…
```
fmri = sns.load_dataset("fmri")
sns.relplot(data=fmri, x="timepoint", y="signal", kind="line")
```

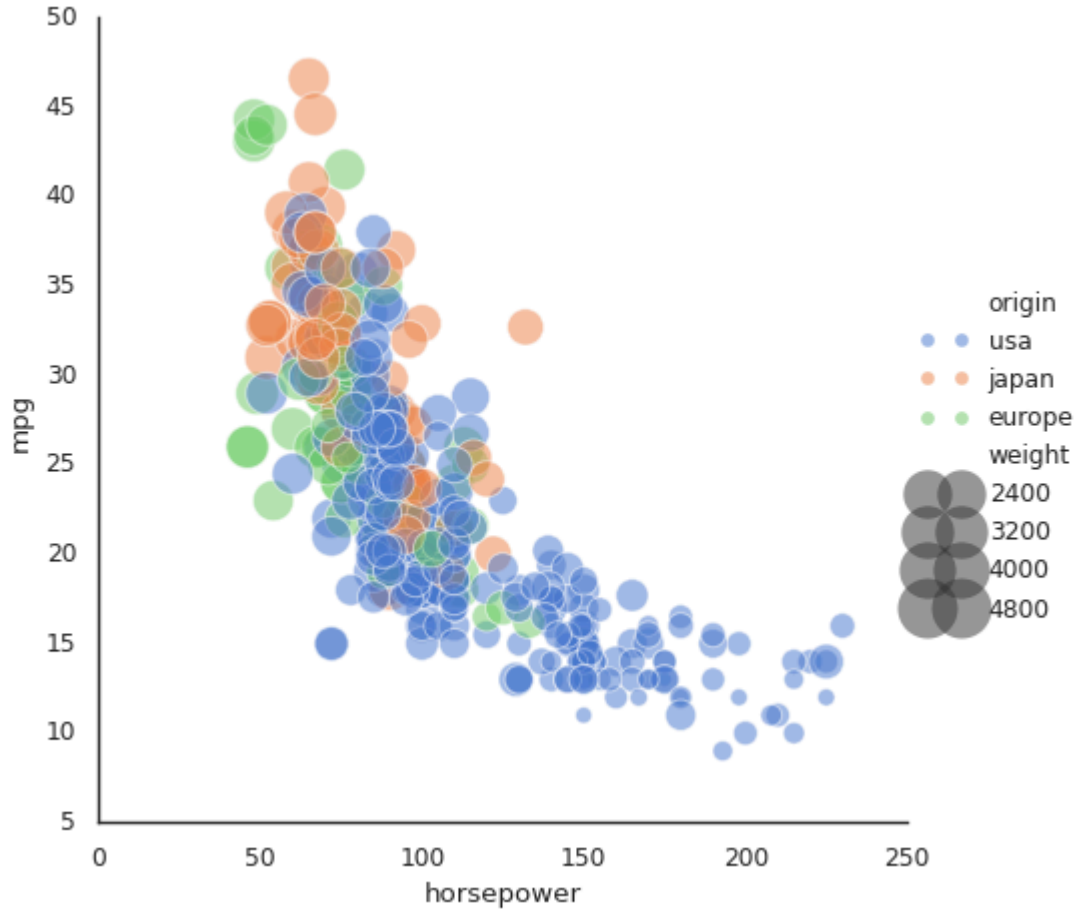
Out[184]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b1b57e0>
```



Some other Plots

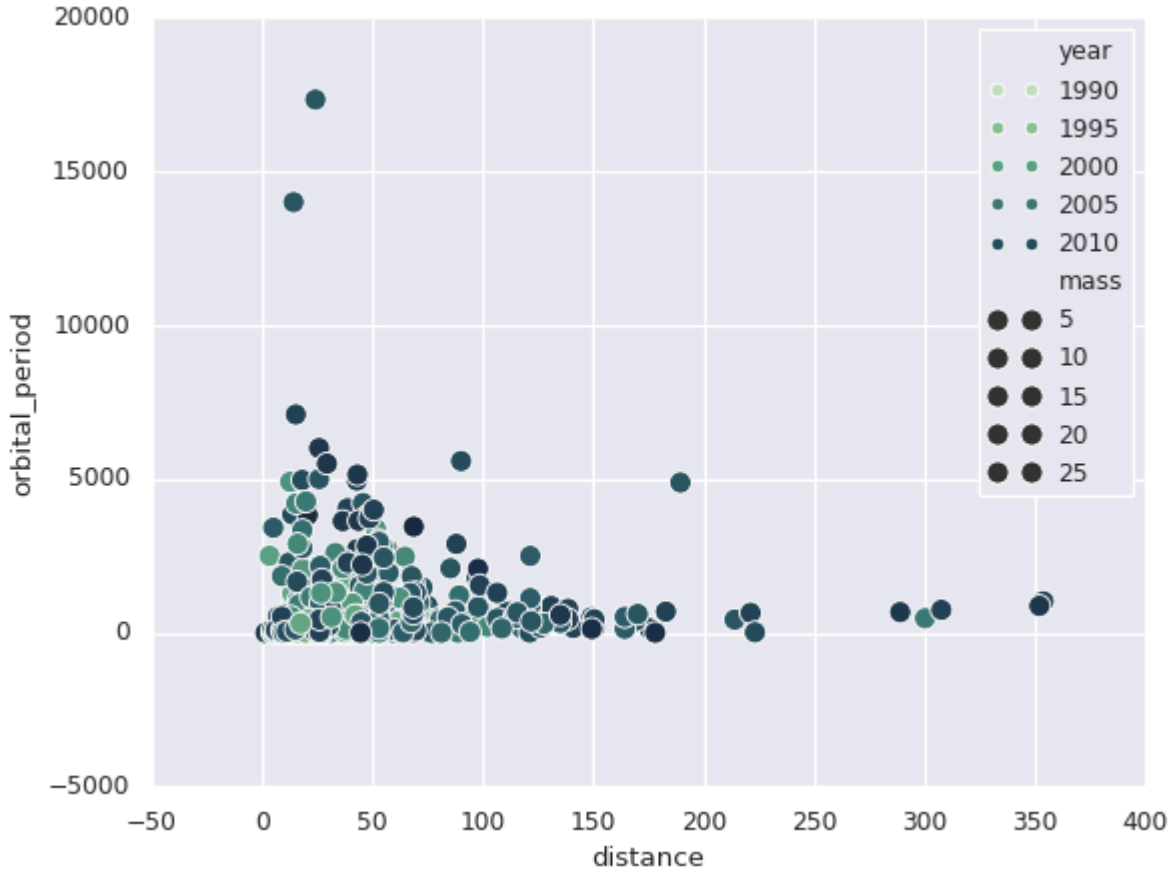
In [185…
```
#1.replot()
import seaborn as sns
sns.set(style="white")

# Load the example mpg dataset
mpg = sns.load_dataset("mpg")

# Plot miles per gallon against horsepower with other semantics
sns.relplot(x="horsepower", y="mpg", hue="origin", size="weight",
            sizes=(400, 40), alpha=.5, palette="muted",
            height=6, data=mpg)
```

Out[185]:
```
<seaborn.axisgrid.FacetGrid at 0x79c69b74fac0>
```



In [186…

```
#2. seaborn.scatterplot()

#Draws a scatter plot with the possibility of several semantic groupings.
import seaborn as sns
sns.set()

# Load the example iris dataset
planets = sns.load_dataset("planets")

cmap = sns.cubehelix_palette(rot=-.5, as_cmap=True)
ax = sns.scatterplot(x="distance", y="orbital_period",
                     hue="year", size="mass",
                     palette=cmap, sizes=(100, 100),
                     data=planets)
```

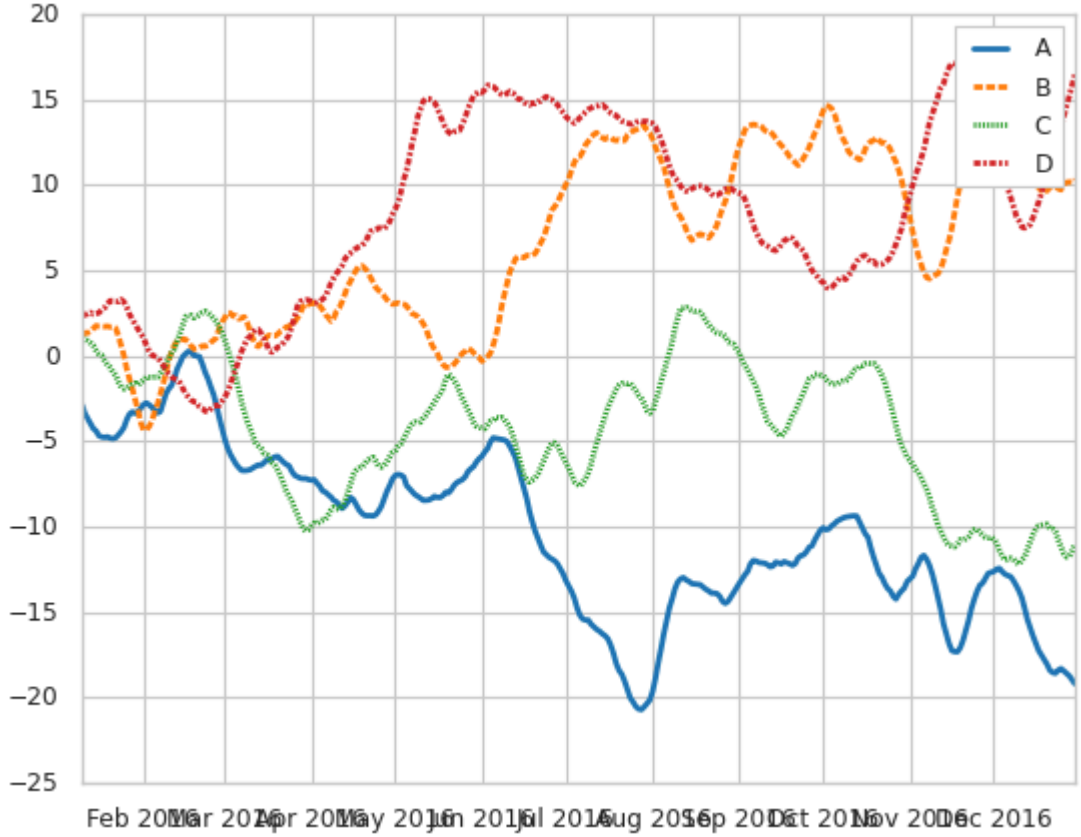


In [187…

```
#3. seaborn.lineplot()

sns.set(style="whitegrid")

rs = np.random.RandomState(365)
values = rs.randn(365, 4).cumsum(axis=0)
dates = pd.date_range("1 1 2016", periods=365, freq="D")
data = pd.DataFrame(values, dates, columns=["A", "B", "C", "D"])
data = data.rolling(10).mean()

sns.lineplot(data=data, palette="tab10", linewidth=2.5)
```

Out[187]:

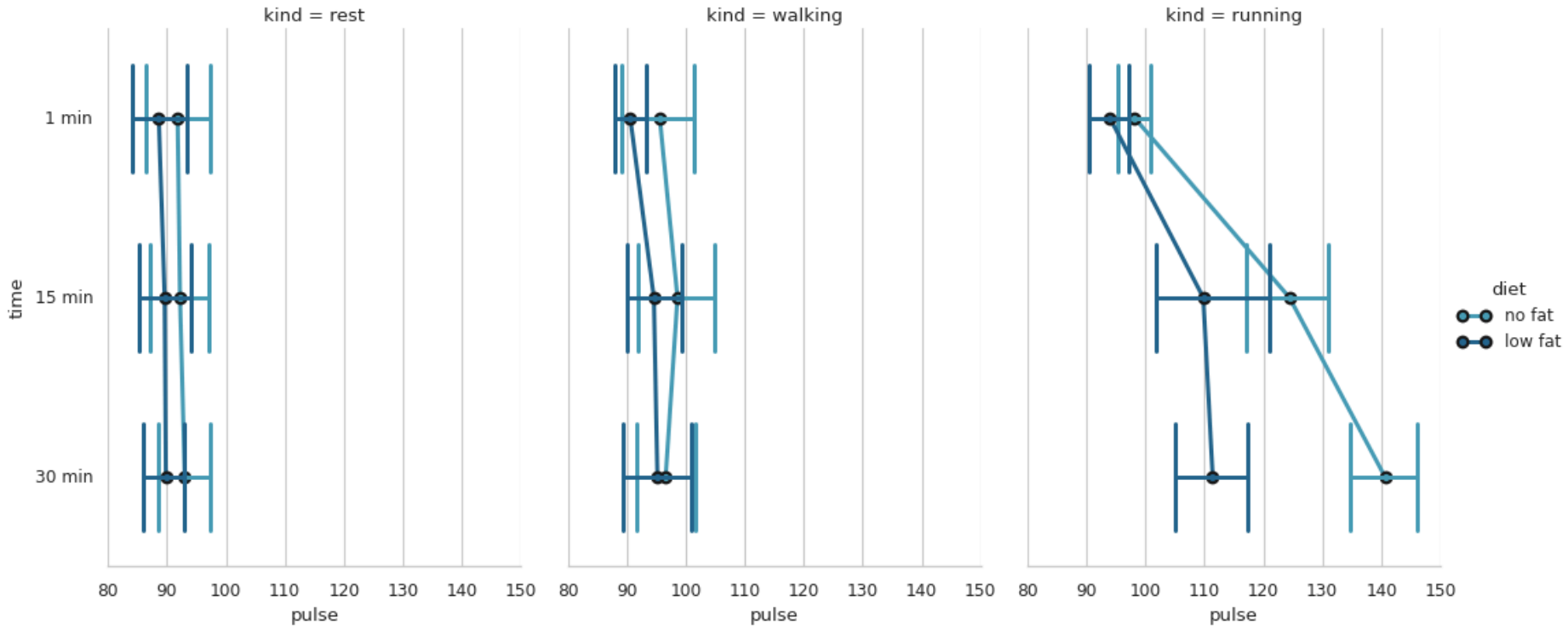
```
<Axes: >
```



In [188…

```
#4. seaborn.catplot()
import seaborn as sns
sns.set(style="whitegrid")

# Load the example exercise dataset
df = sns.load_dataset("exercise")

# Draw a pointplot to show pulse as a function of three categorical factors
g = sns.catplot(x="pulse", y="time", hue="diet", col="kind",
                capsize=.6, palette="YlGnBu_d", height=6, aspect=.75,
                kind="point", data=df)
g.despine(left=True)
```

Out[188]: <seaborn.axisgrid.FacetGrid at 0x79c6df3b9ba0>



In [189…

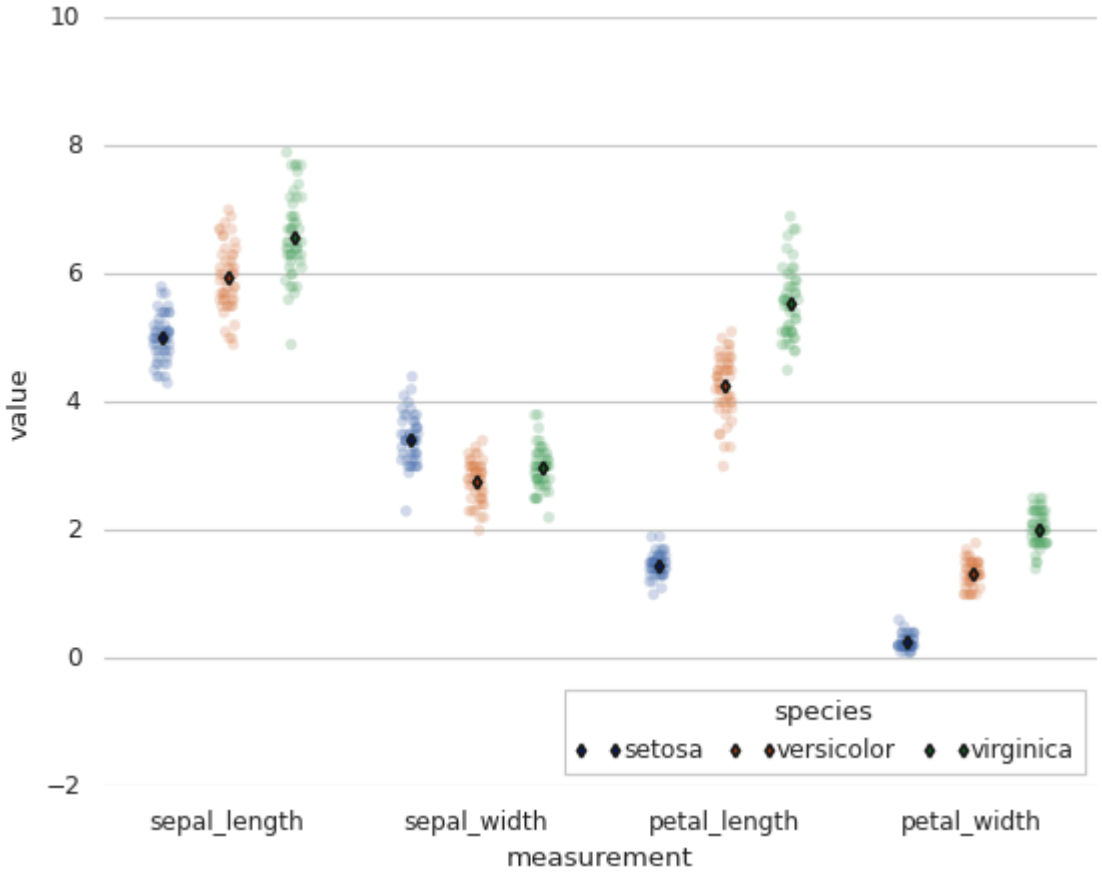
```
#5. seaborn.stripplot()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="whitegrid")
iris = sns.load_dataset("iris")

# "Melt" the dataset to "long-form" or "tidy" representation
iris = pd.melt(iris, "species", var_name="measurement")

# Initialize the figure
f, ax = plt.subplots()
sns.despine(bottom=True, left=True)

# Show each observation with a scatterplot
sns.stripplot(x="measurement", y="value", hue="species",
              data=iris, dodge=True, jitter=True,
              alpha=.25, zorder=1)
```

```
# Show the conditional means
sns.pointplot(x="measurement", y="value", hue="species",
              data=iris, dodge=.532, join=False, palette="dark",
              markers="d", scale=.75, ci=None)

# Improve the legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles[3:], labels[3:], title="species",
          handletextpad=0, columnspacing=1,
          loc="lower right", ncol=3, frameon=True)
```

```
<ipython-input-189-90ad2c1fbe6e>:22: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.pointplot(x="measurement", y="value", hue="species",
<ipython-input-189-90ad2c1fbe6e>:22: UserWarning:

The `scale` parameter is deprecated and will be removed in v0.15.0. You can now control the size of each plot element using matp
lotlib `Line2D` parameters (e.g., `linewidth`, `markersize`, etc.).

  sns.pointplot(x="measurement", y="value", hue="species",
<ipython-input-189-90ad2c1fbe6e>:22: UserWarning:

The `join` parameter is deprecated and will be removed in v0.15.0. You can remove the line between points with `linestyle='non
e'`.

  sns.pointplot(x="measurement", y="value", hue="species",
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
```

Out[189]:

```
<matplotlib.legend.Legend at 0x79c69bb2eb30>
```



In [190…

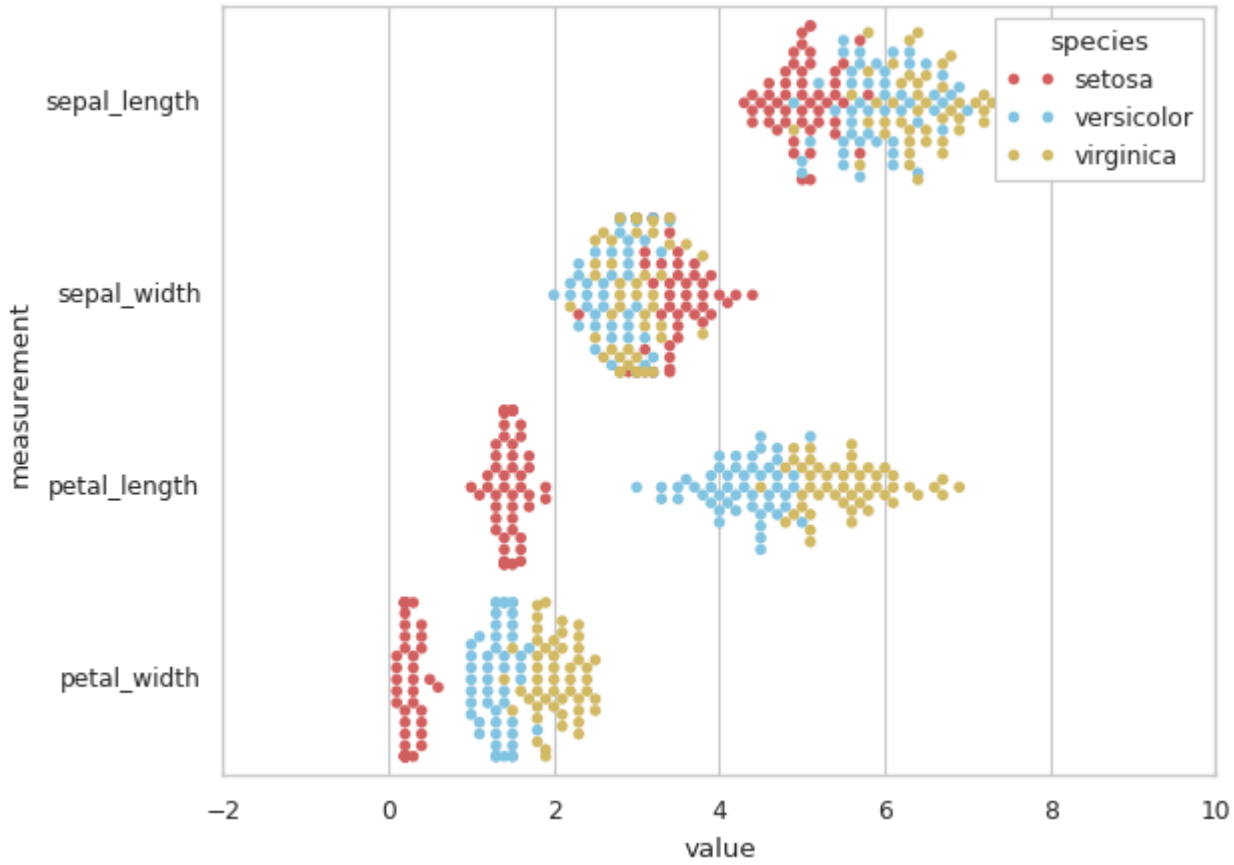
```
#6. seaborn.swarmplot()
import pandas as pd
import seaborn as sns
sns.set(style="whitegrid", palette="muted")

# Load the example iris dataset
iris = sns.load_dataset("iris")

# "Melt" the dataset to "long-form" or "tidy" representation
iris = pd.melt(iris, "species", var_name="measurement")

# Draw a categorical scatterplot to show each observation
sns.swarmplot(x="value", y="measurement", hue="species",
              palette=["r", "c", "y"], data=iris)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 20.0% of the points cannot be placed; you may
want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 6.0% of the points cannot be placed; you may w
ant to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 15.3% of the points cannot be placed; you may
want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[190]:

```
<Axes: xlabel='value', ylabel='measurement'>
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 27.3% of the points cannot be placed; you may
want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 7.3% of the points cannot be placed; you may w
ant to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 24.7% of the points cannot be placed; you may
want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

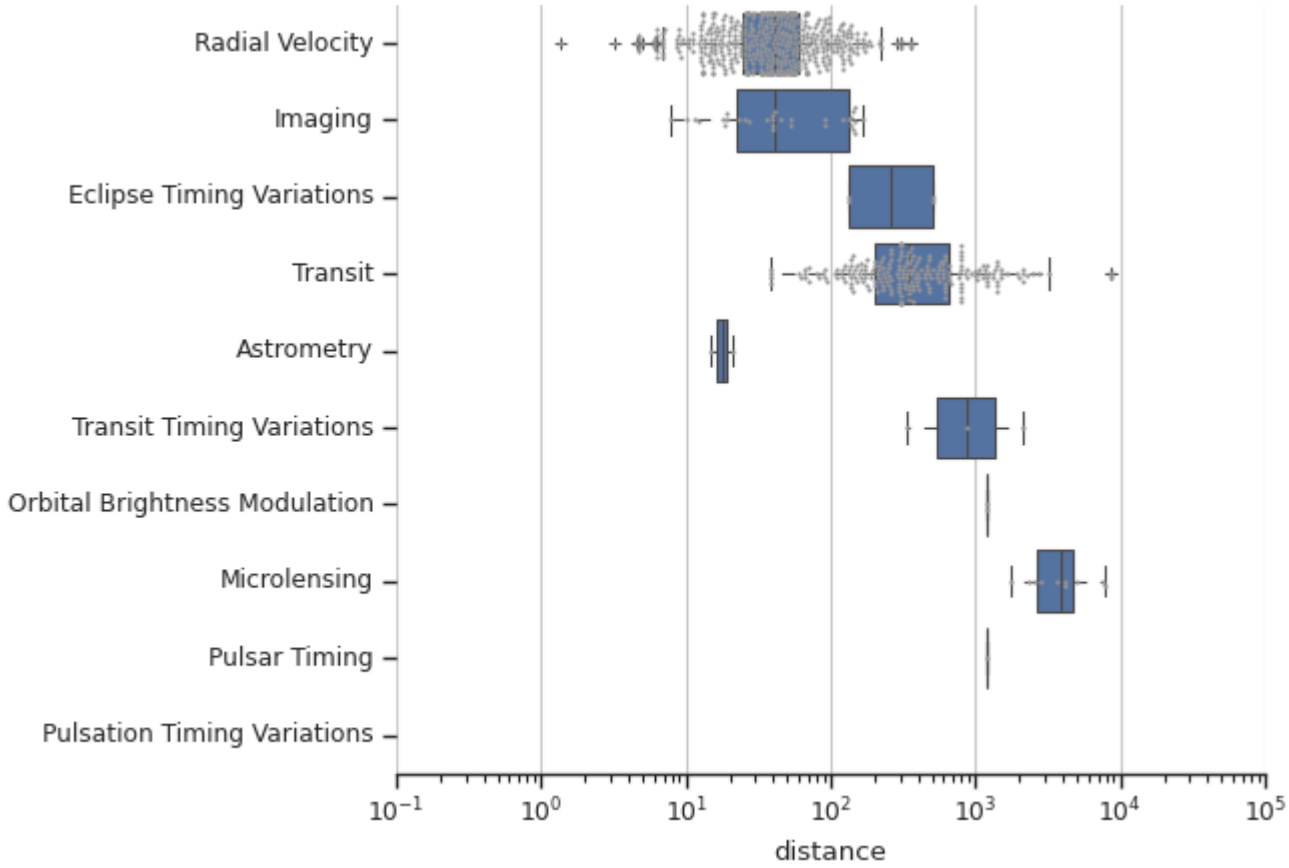


In [191…

```
#7.seaborn.boxplot()
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="ticks")

# Initialize the figure with a logarithmic x axis
f, ax = plt.subplots(figsize=(7, 6))
ax.set_xscale("log")

# Load the example planets dataset
planets = sns.load_dataset("planets")

# Plot the orbital period with horizontal boxes
sns.boxplot(x="distance", y="method", data=planets)
```

```
# Add in points to show each observation
sns.swarmplot(x="distance", y="method", data=planets, size=2, color=".6", linewidth=0)

# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 22.6% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 23.6% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

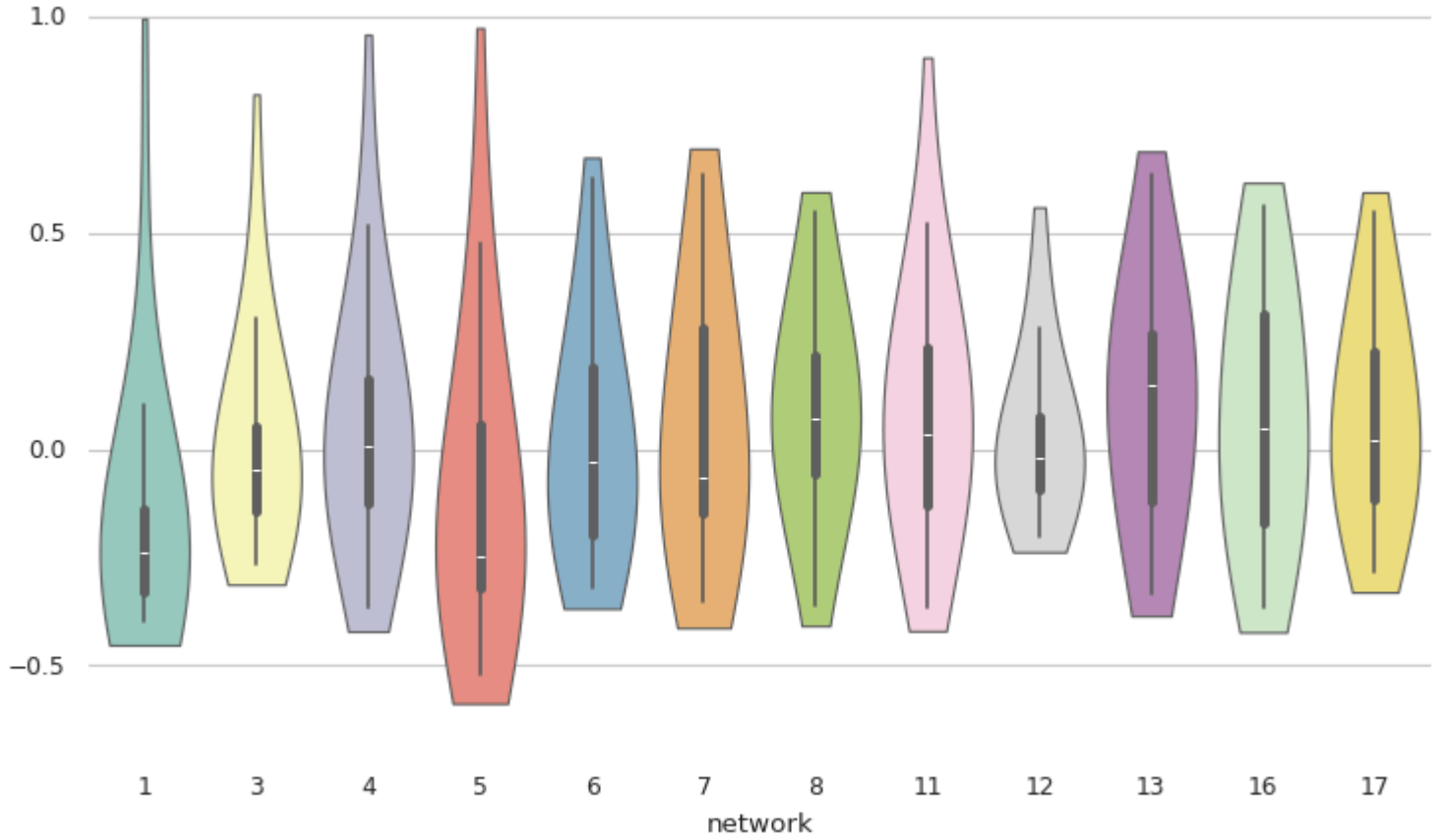


In [192…

```
#8. seaborn.violinplot()
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="whitegrid")

# Load the example dataset of brain network correlations
df = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)

# Pull out a specific subset of networks
used_networks = [1, 3, 4, 5, 6, 7, 8, 11, 12, 13, 16, 17]
used_columns = (df.columns.get_level_values("network")
                          .astype(float)
                          .isin(used_networks))
df = df.loc[:, used_columns]
```

```
# Compute the correlation matrix and average over networks
corr_df = df.corr().groupby(level="network").mean()
corr_df.index = corr_df.index.astype(int)
corr_df = corr_df.sort_index().T

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 6))

# Draw a violinplot with a narrower bandwidth than the default
sns.violinplot(data=corr_df, palette="Set3", bw=1, cut=.2, linewidth=1)

# Finalize the figure
ax.set(ylim=(-.7, 1.05))
sns.despine(left=True, bottom=True)
```

```
<ipython-input-192-e226cd965667>:25: FutureWarning:

The `bw` parameter is deprecated in favor of `bw_method`/`bw_adjust`.
Setting `bw_method=1`, but please see docs for the new parameters
and update your code. This will become an error in seaborn v0.15.0.

  sns.violinplot(data=corr_df, palette="Set3", bw=1, cut=.2, linewidth=1)
```

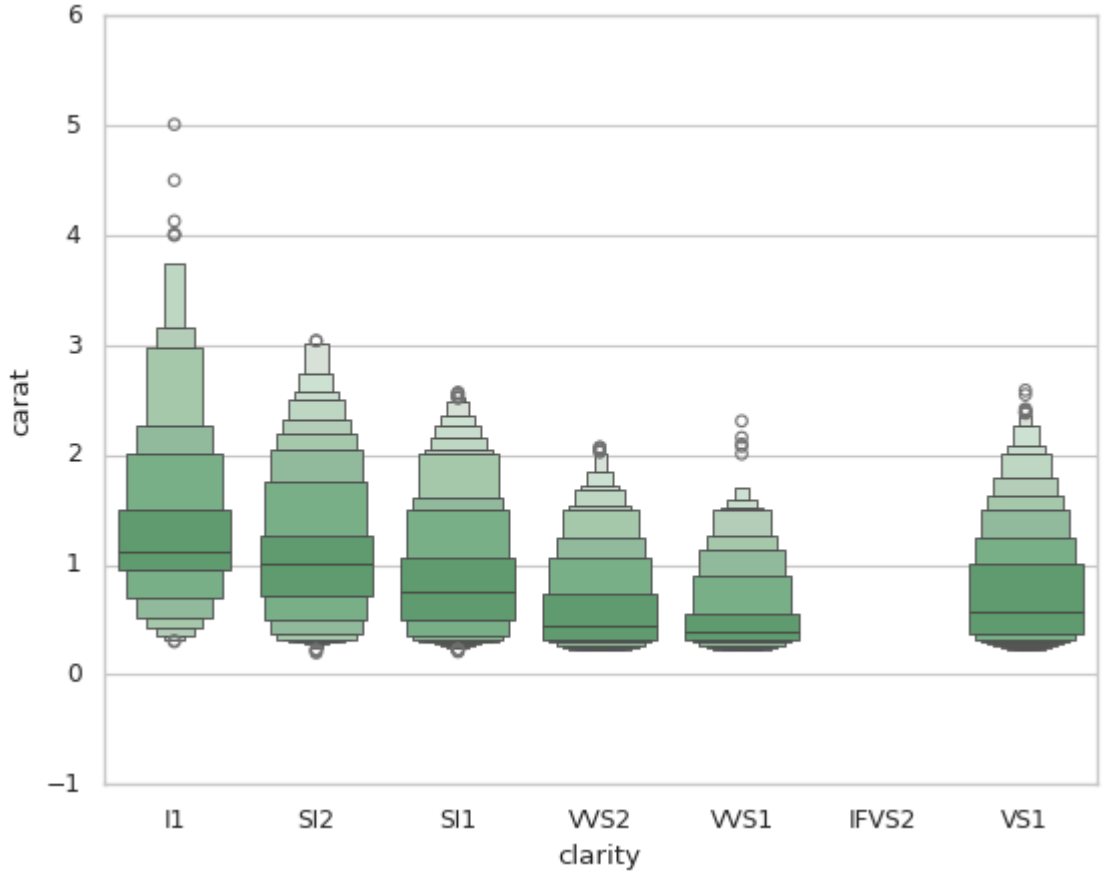


In [193…

```
#9. seaborn.boxenplot()
import seaborn as sns
sns.set(style="whitegrid")

diamonds = sns.load_dataset("diamonds")
clarity_ranking = ["I1", "SI2", "SI1", "VVS2", "VVS1", "IF" "VS2", "VS1"]

sns.boxenplot(x="clarity", y="carat",
              color="g", order=clarity_ranking,
              scale="linear", data=diamonds)
```

```
<ipython-input-193-ffd9d83b45de>:8: FutureWarning:

The `scale` parameter has been renamed to `width_method` and will be removed in v0.15. Pass `width_method='linear' for the same effect.
  sns.boxenplot(x="clarity", y="carat",
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```

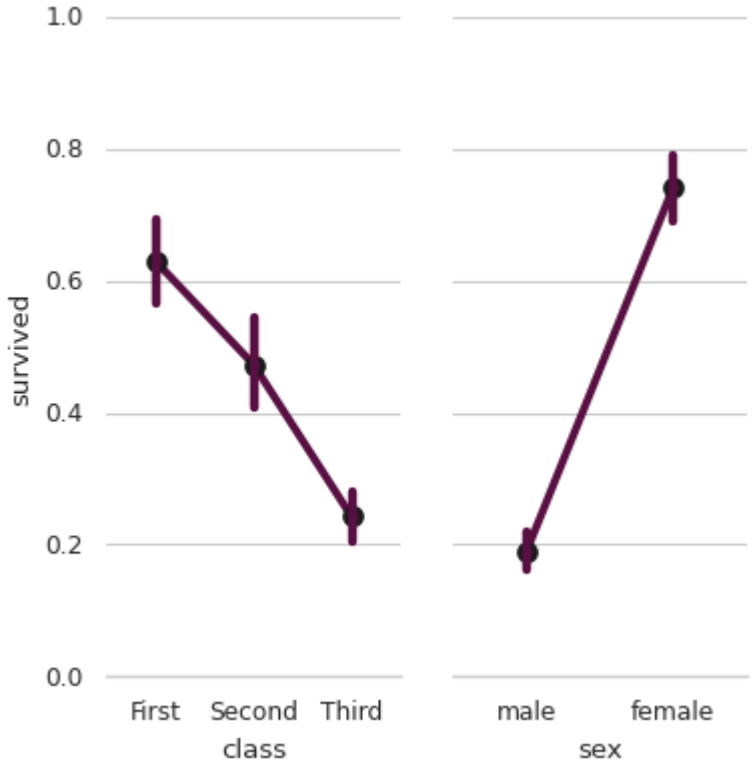
Out[193]: <Axes: xlabel='clarity', ylabel='carat'>



In [194…

```
#10. seaborn.pointplot()
import seaborn as sns
sns.set(style="whitegrid")

# Load the example Titanic dataset
titanic = sns.load_dataset("titanic")

# Set up a grid to plot survival probability against several variables
g = sns.PairGrid(titanic, y_vars="survived",
                 x_vars=["class", "sex"],
                 height=5, aspect=.5)

# Draw a seaborn pointplot onto each Axes
g.map(sns.pointplot, scale=1.3, errwidth=4, color="xkcd:plum")
g.set(ylim=(0, 1))
sns.despine(fig=g.fig, left=True)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1615: UserWarning:

The `scale` parameter is deprecated and will be removed in v0.15.0. You can now control the size of each plot element using matp
lotlib `Line2D` parameters (e.g., `linewidth`, `markersize`, etc.).

  func(x=x, y=y, **kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1615: FutureWarning:

The `errwidth` parameter is deprecated. And will be removed in v0.15.0. Pass `err_kws={'linewidth': 4}` instead.

  func(x=x, y=y, **kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1615: UserWarning:

The `scale` parameter is deprecated and will be removed in v0.15.0. You can now control the size of each plot element using matp
lotlib `Line2D` parameters (e.g., `linewidth`, `markersize`, etc.).

  func(x=x, y=y, **kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1615: FutureWarning:

The `errwidth` parameter is deprecated. And will be removed in v0.15.0. Pass `err_kws={'linewidth': 4}` instead.

  func(x=x, y=y, **kwargs)
```



In [195…

```
#11. seaborn.barplot()
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", context="talk")
rs = np.random.RandomState(8)

# Set up the matplotlib figure
f, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(7, 5), sharex=True)

# Generate some sequential data
x = np.array(list("ABCDEFGHIJ"))
y1 = np.arange(1, 11)
sns.barplot(x=x, y=y1, palette="rocket", ax=ax1)
ax1.axhline(0, color="k", clip_on=False)
ax1.set_ylabel("Sequential")

# Center the data to make it diverging
```

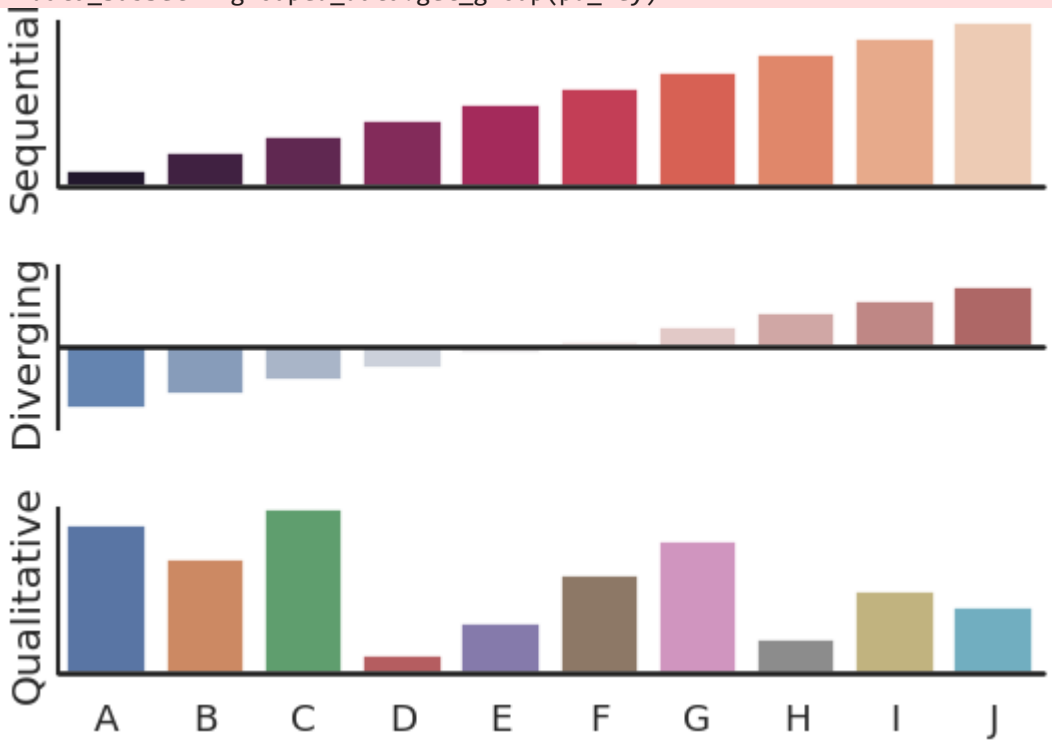
```
y2 = y1 - 5.5
sns.barplot(x=x, y=y2, palette="vlag", ax=ax2)
ax2.axhline(0, color="k", clip_on=False)
ax2.set_ylabel("Diverging")

# Randomly reorder the data to make it qualitative
y3 = rs.choice(y1, len(y1), replace=False)
sns.barplot(x=x, y=y3, palette="deep", ax=ax3)
ax3.axhline(0, color="k", clip_on=False)
ax3.set_ylabel("Qualitative")

# Finalize the plot
sns.despine(bottom=True)
plt.setp(f.axes, yticks=[])
plt.tight_layout(h_pad=2)
```

```
<ipython-input-195-aa420bdc67fc>:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=x, y=y1, palette="rocket", ax=ax1)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
```

```
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
<ipython-input-195-aa420bdc67fc>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=x, y=y2, palette="vlag", ax=ax2)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
```

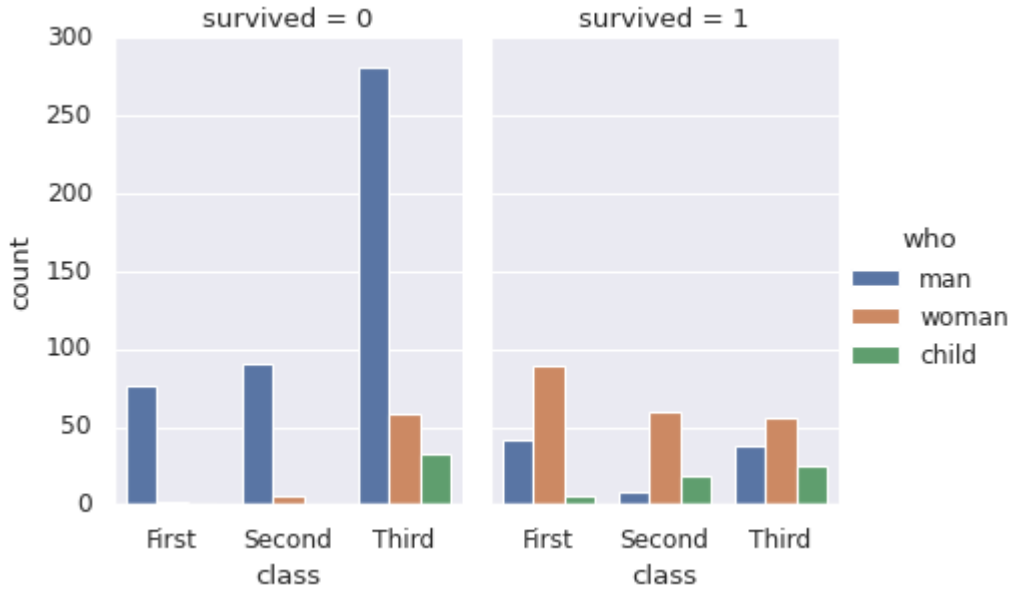
```
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
<ipython-input-195-aa420bdc67fc>:26: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=x, y=y3, palette="deep", ax=ax3)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
```

```
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
```



In [196…

```
#12. seaborn.countplot
import seaborn as sns
sns.set(style="darkgrid")
titanic = sns.load_dataset("titanic")
g = sns.catplot(x="class", hue="who", col="survived", data=titanic, kind="count", height=4, aspect=.7)
```



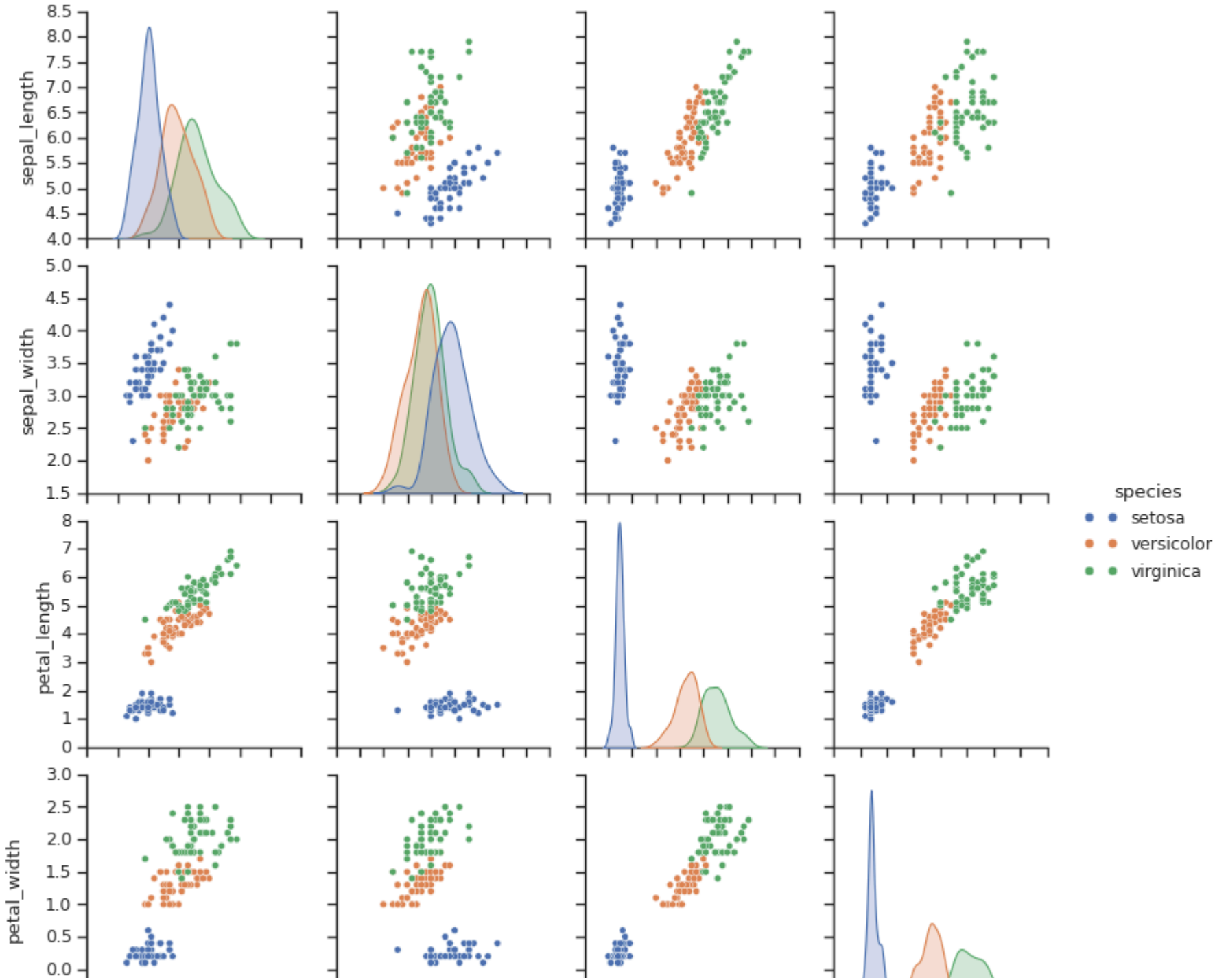
In [197…

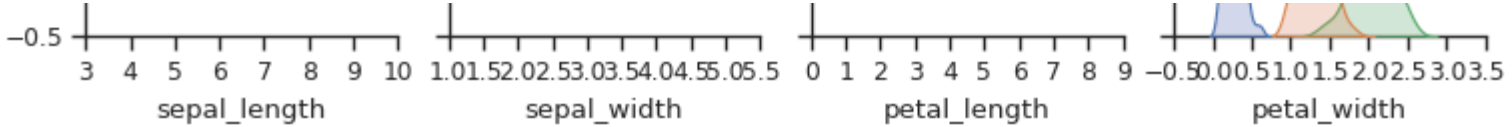
```
#13. seaborn.pairplot()
import seaborn as sns
sns.set(style="ticks")

df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
```

```
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```

Out[197]:
```
<seaborn.axisgrid.PairGrid at 0x79c69b7e0400>
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

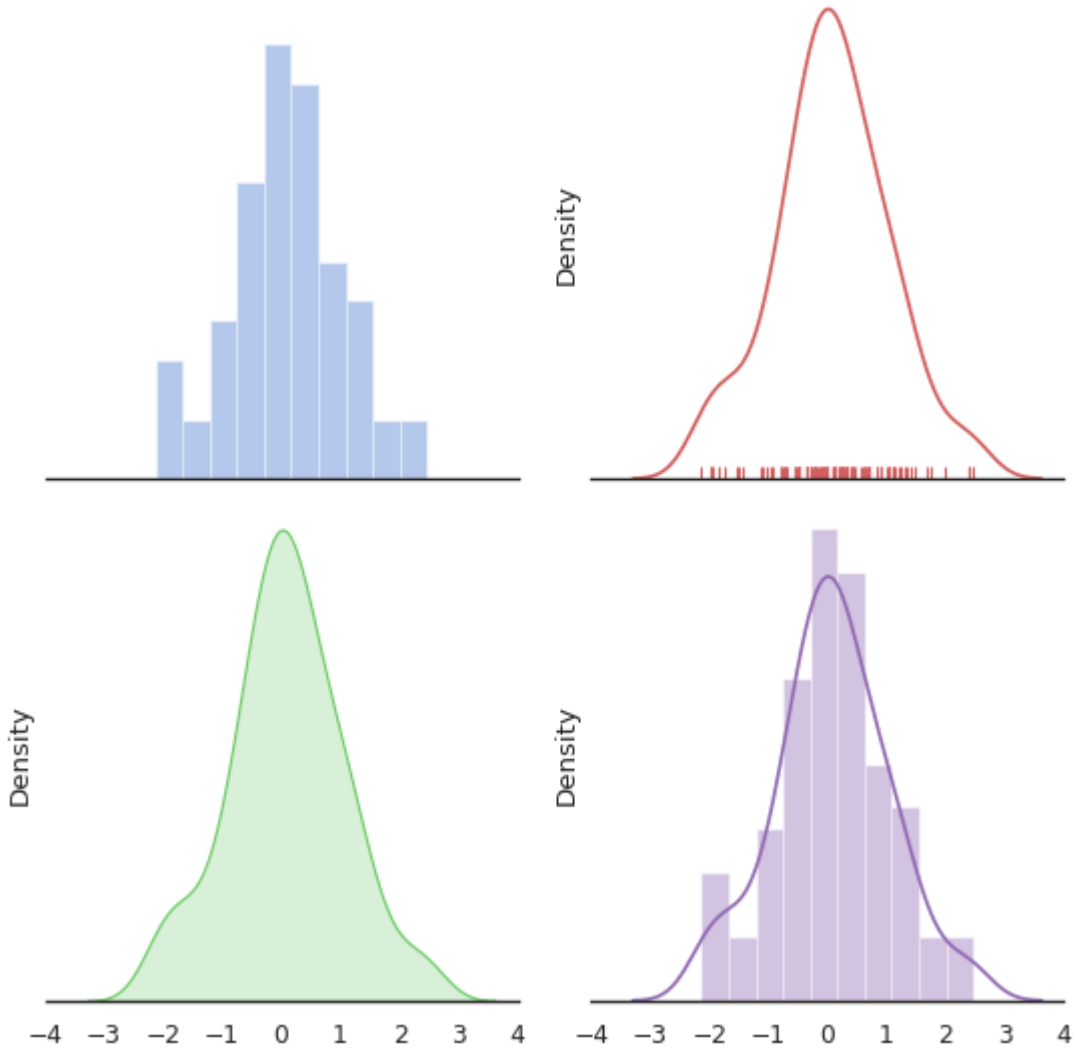


In [198…

```
#14. seaborn.distplot()

sns.set(style="white", palette="muted", color_codes=True)
rs = np.random.RandomState(10)

# Set up the matplotlib figure
f, axes = plt.subplots(2, 2, figsize=(7, 7), sharex=True)
sns.despine(left=True)

# Generate a random univariate dataset
d = rs.normal(size=100)

# Plot a simple histogram with binsize determined automatically
sns.distplot(d, kde=False, color="b", ax=axes[0, 0])

# Plot a kernel density estimate and rug plot
sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0, 1])

# Plot a filled kernel density estimate
sns.distplot(d, hist=False, color="g", kde_kws={"shade": True}, ax=axes[1, 0])

# Plot a historgram and kernel density estimate
sns.distplot(d, color="m", ax=axes[1, 1])

plt.setp(axes, yticks=[])
plt.tight_layout()
```

```
<ipython-input-198-d2e4fc2b8af1>:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(d, kde=False, color="b", ax=axes[0, 0])
<ipython-input-198-d2e4fc2b8af1>:17: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0, 1])
<ipython-input-198-d2e4fc2b8af1>:20: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(d, hist=False, color="g", kde_kws={"shade": True}, ax=axes[1, 0])
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2496: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
<ipython-input-198-d2e4fc2b8af1>:23: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(d, color="m", ax=axes[1, 1])
```

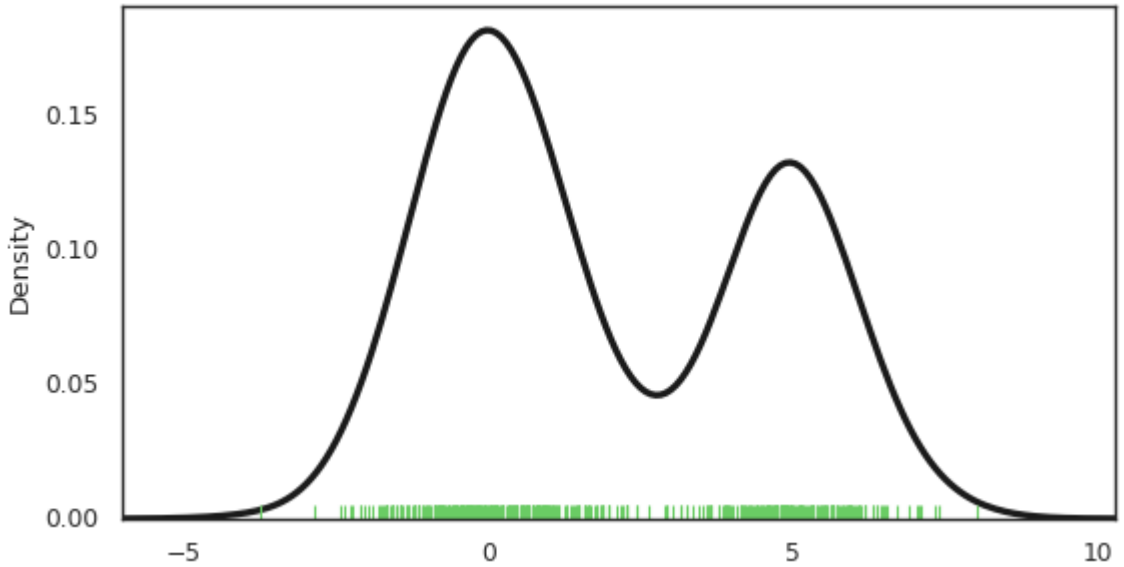


In [199…

```
#15 seaborn.rugplot()
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
sample = np.hstack((np.random.randn(300), np.random.randn(200)+5))
fig, ax = plt.subplots(figsize=(8,4))
sns.distplot(sample, rug=True, hist=False, rug_kws={"color": "g"},
    kde_kws={"color": "k", "lw": 3})
plt.show()
```

```
<ipython-input-199-8d2d9b7f2057>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(sample, rug=True, hist=False, rug_kws={"color": "g"},
```



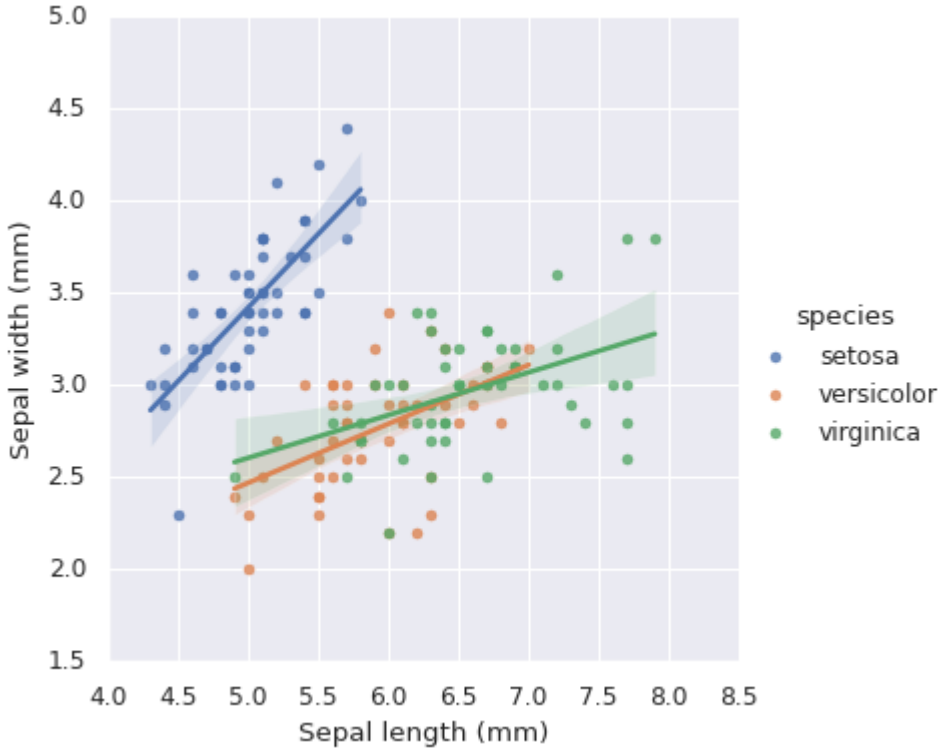
In [200…

```
#16. seaborn.lmplot()
import seaborn as sns
sns.set()

# Load the iris dataset
```

```
iris = sns.load_dataset("iris")

# Plot sepal with as a function of sepal_length across days
g = sns.lmplot(x="sepal_length", y="sepal_width", hue="species",
               truncate=True, height=5, data=iris)

# Use more informative axis labels than are provided by default
g.set_axis_labels("Sepal length (mm)", "Sepal width (mm)")
```

Out[200]:

```
<seaborn.axisgrid.FacetGrid at 0x79c699ee4f40>
```

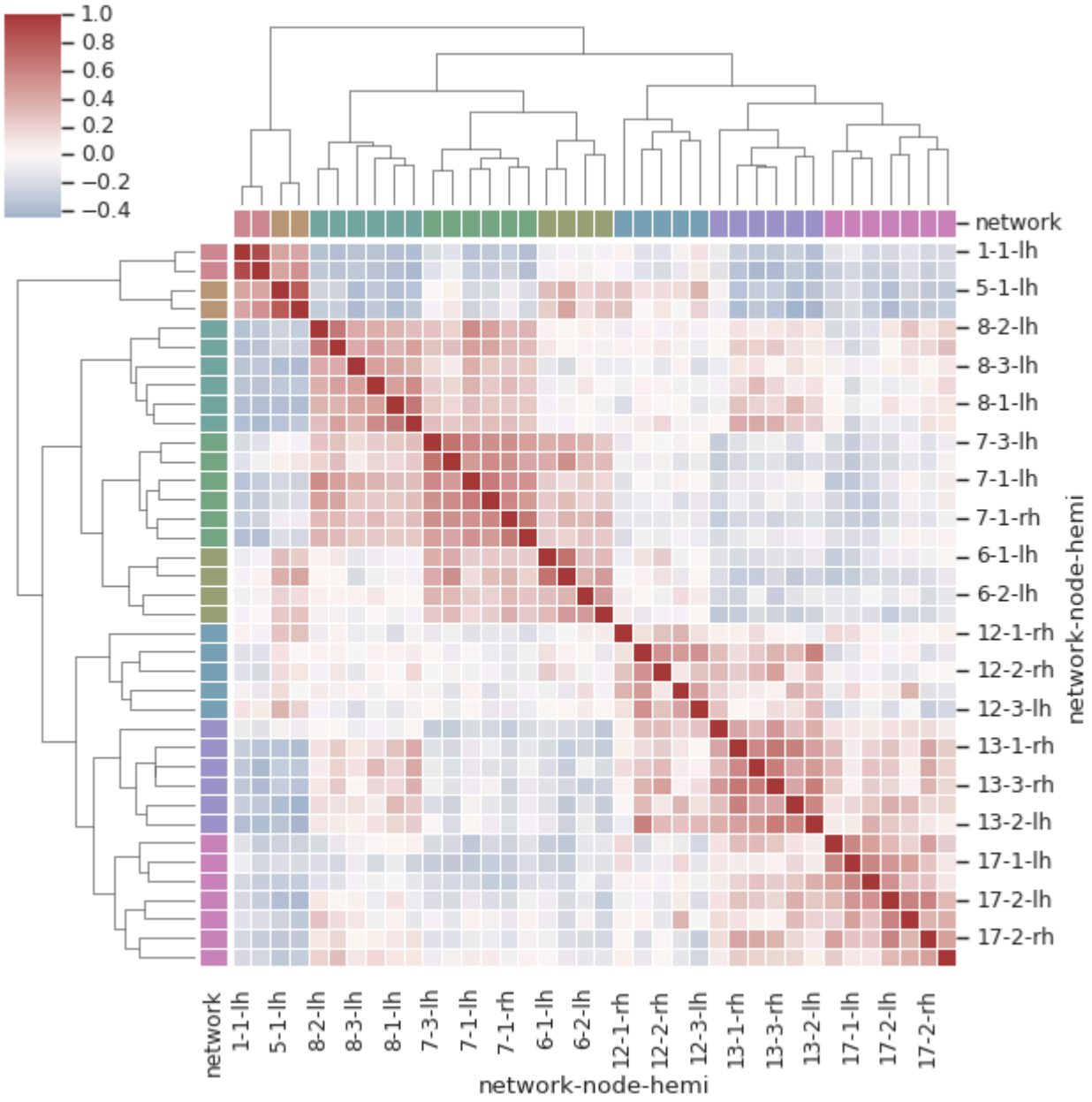


In [203…

```
#17 seaborn.clustermap()

# Load the brain networks example dataset
df = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)

# Select a subset of the networks
used_networks = [1, 5, 6, 7, 8, 12, 13, 17]
used_columns = (df.columns.get_level_values("network")
```

```
                          .astype(int)
                          .isin(used_networks))
df = df.loc[:, used_columns]

# Create a categorical palette to identify the networks
network_pal = sns.husl_palette(8, s=.45)
network_lut = dict(zip(map(str, used_networks), network_pal))

# Convert the palette to vectors that will be drawn on the side of the matrix
networks = df.columns.get_level_values("network")
network_colors = pd.Series(networks, index=df.columns).map(network_lut)

# Draw the full plot
sns.clustermap(df.corr(), center=0, cmap="vlag",
               row_colors=network_colors, col_colors=network_colors,
               linewidths=.75, figsize=(8, 8))
```

Out[203]: <seaborn.matrix.ClusterGrid at 0x79c6a125d2a0>



In [204…

```
# Consider Tips dataset from seaborn
```

```
tips=sns.load_dataset('tips')
tips
```

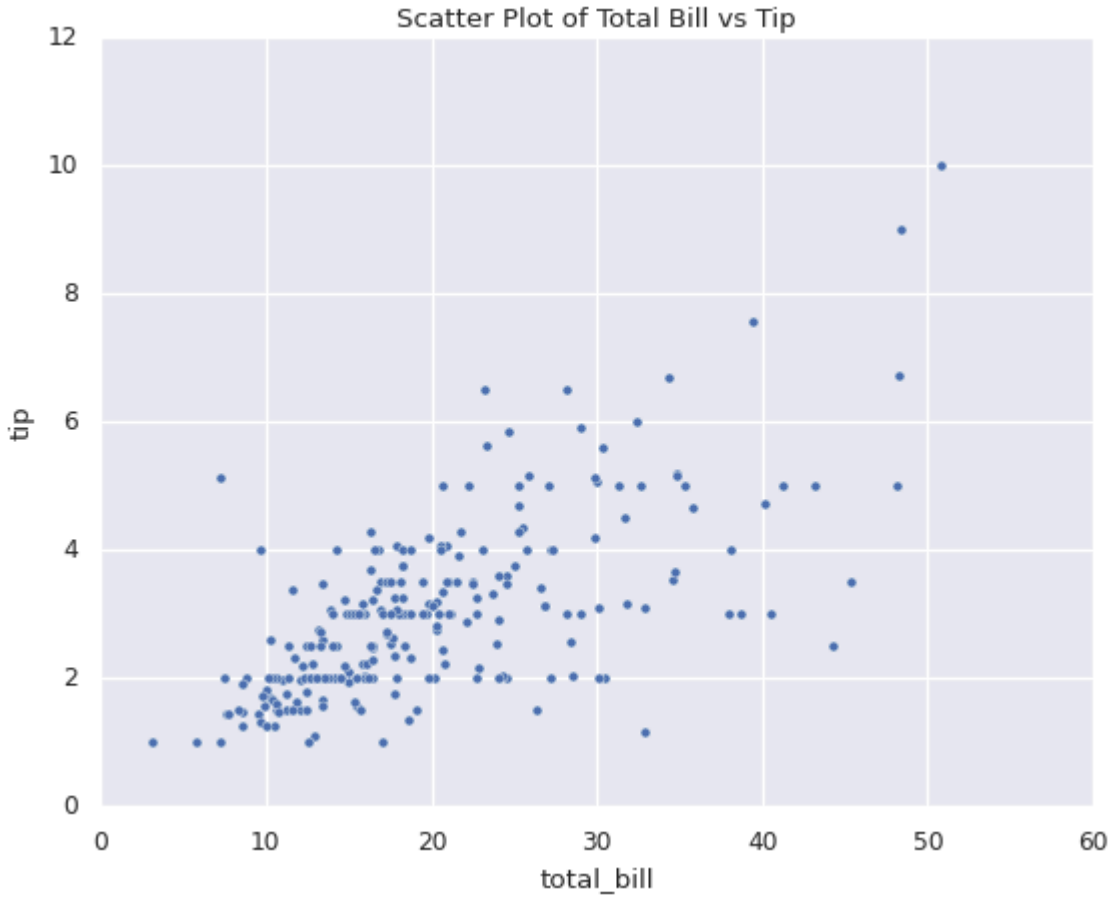
Out[204]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

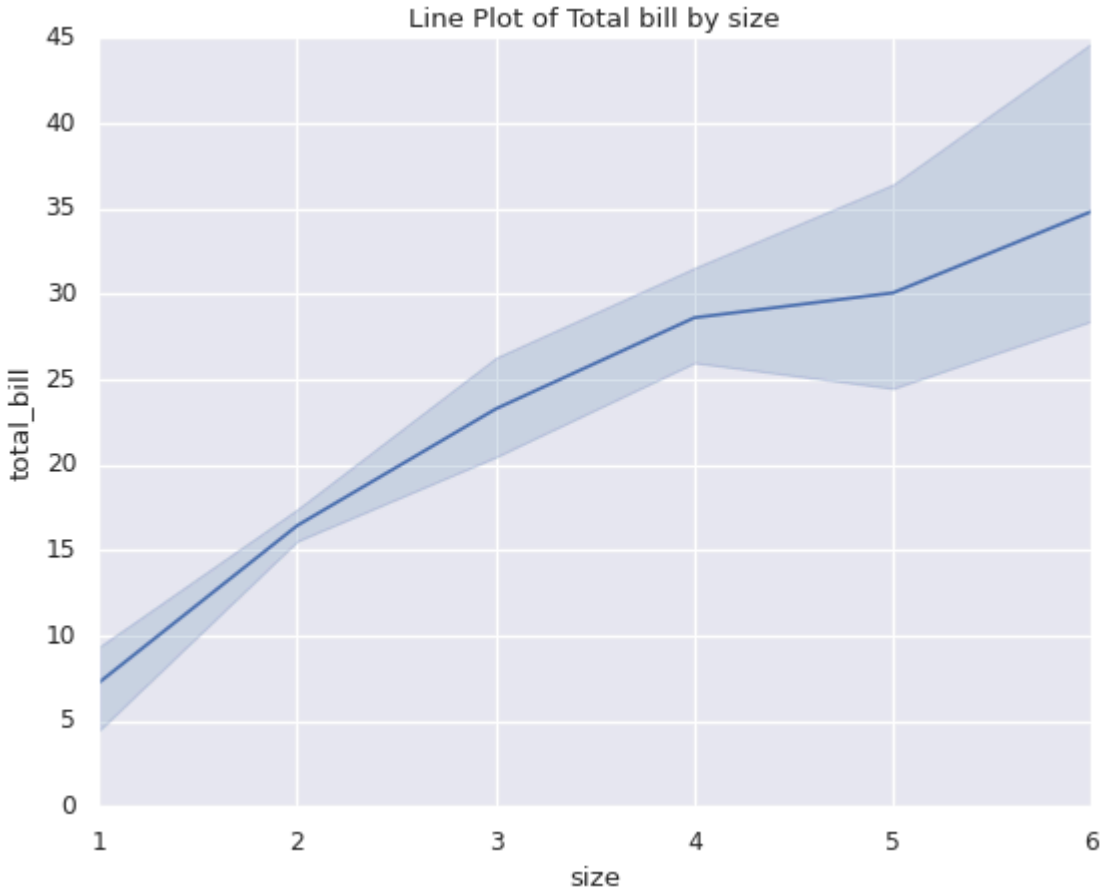
In [205…

```
sns.scatterplot(x='total_bill',y='tip',data=tips)
plt.title("Scatter Plot of Total Bill vs Tip")
plt.show()
```



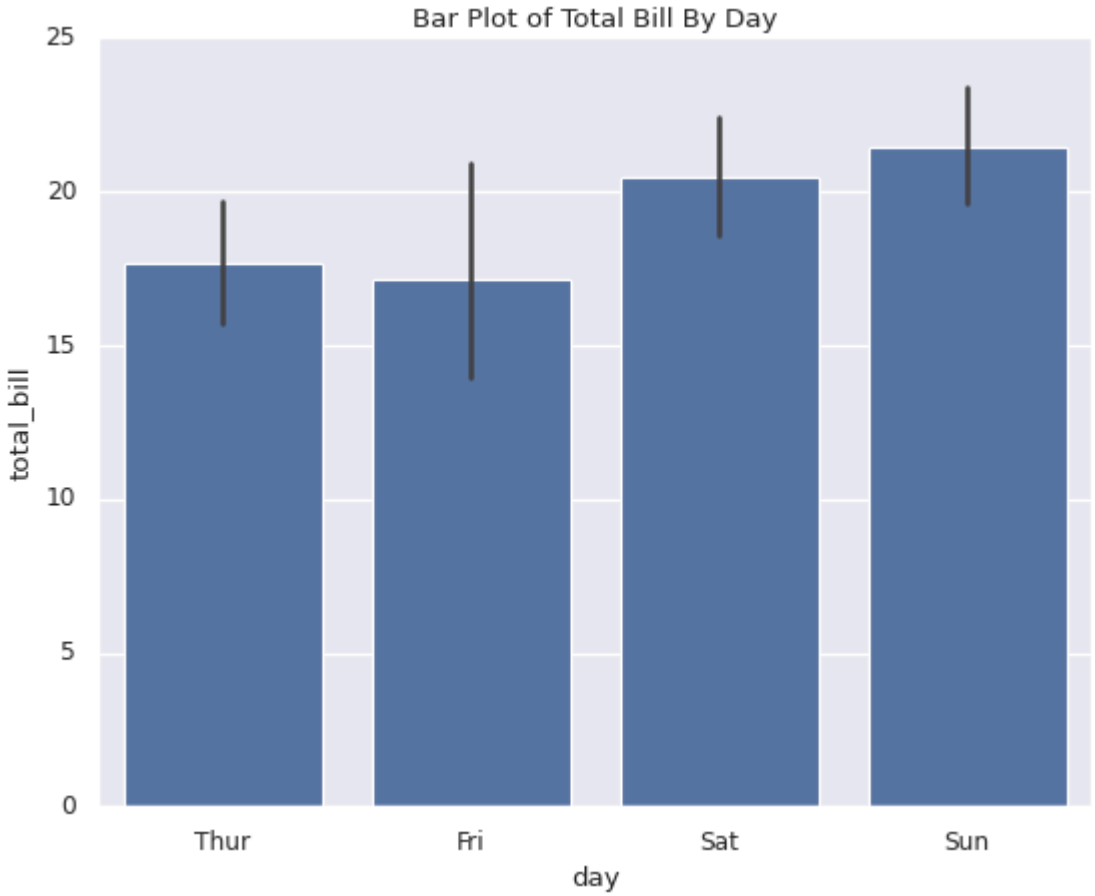
In [206…

```
## Line Plot

sns.lineplot(x='size',y='total_bill',data=tips)
plt.title("Line Plot of Total bill by size")
plt.show()
```



In [207…

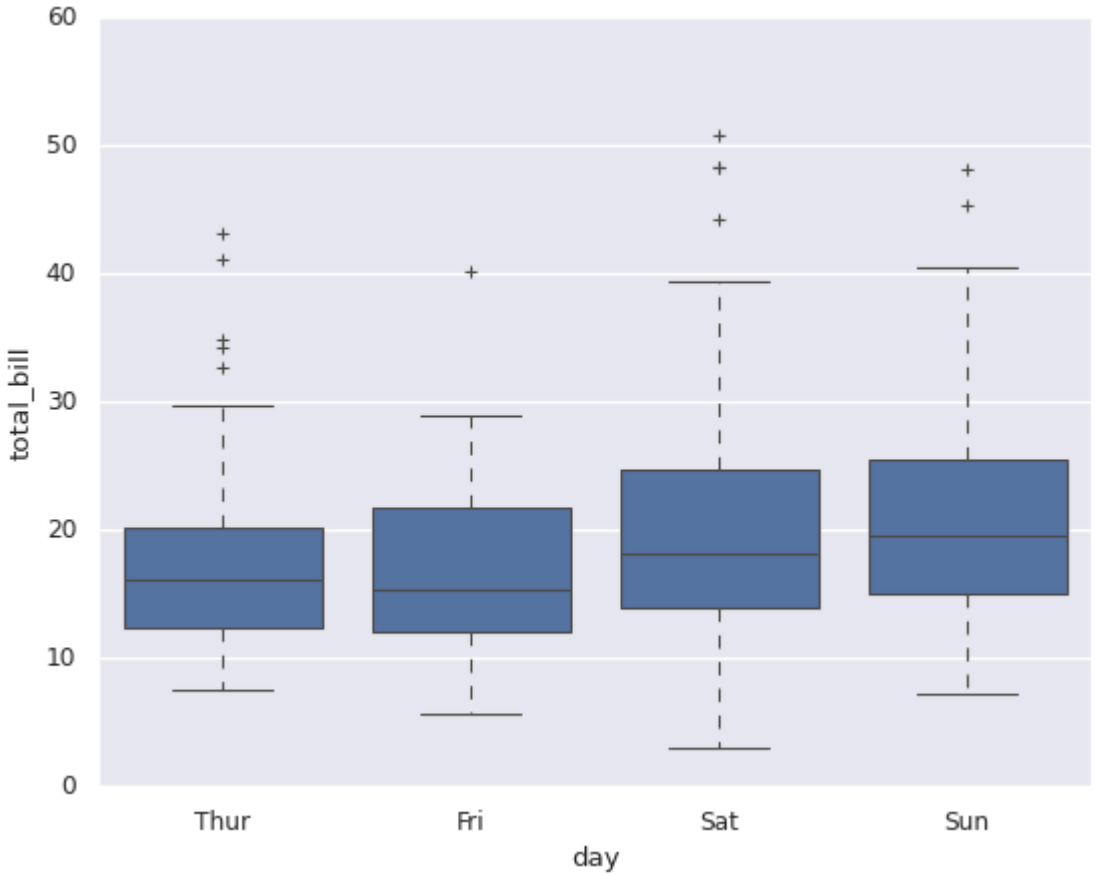
```
## Categorical Plots
## BAr Plot
sns.barplot(x='day',y='total_bill',data=tips)
plt.title('Bar Plot of Total Bill By Day')
plt.show()
```



In [208…
```
## Box Plot
sns.boxplot(x="day",y='total_bill',data=tips)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```

Out[208]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



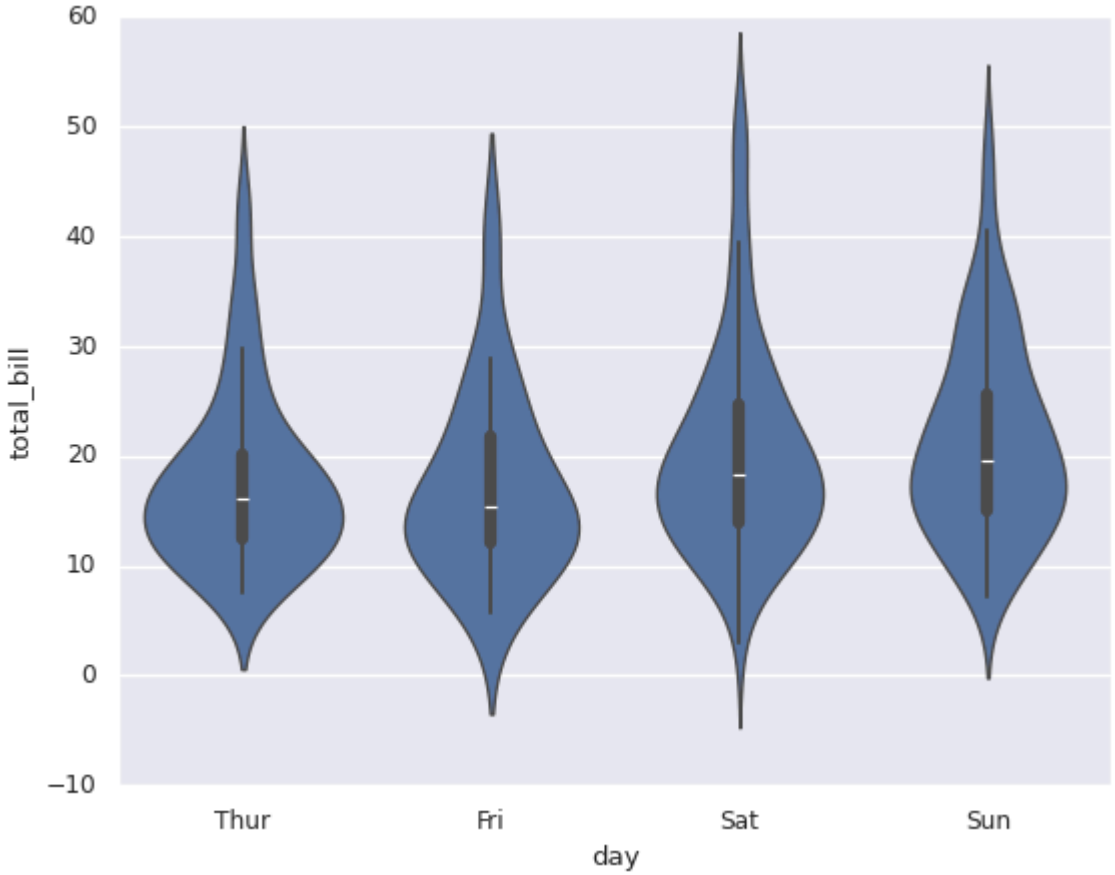
In [209…

```
## Violin Plot

sns.violinplot(x='day',y='total_bill',data=tips)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
eed to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warnin
g.
  data_subset = grouped_data.get_group(pd_key)
```

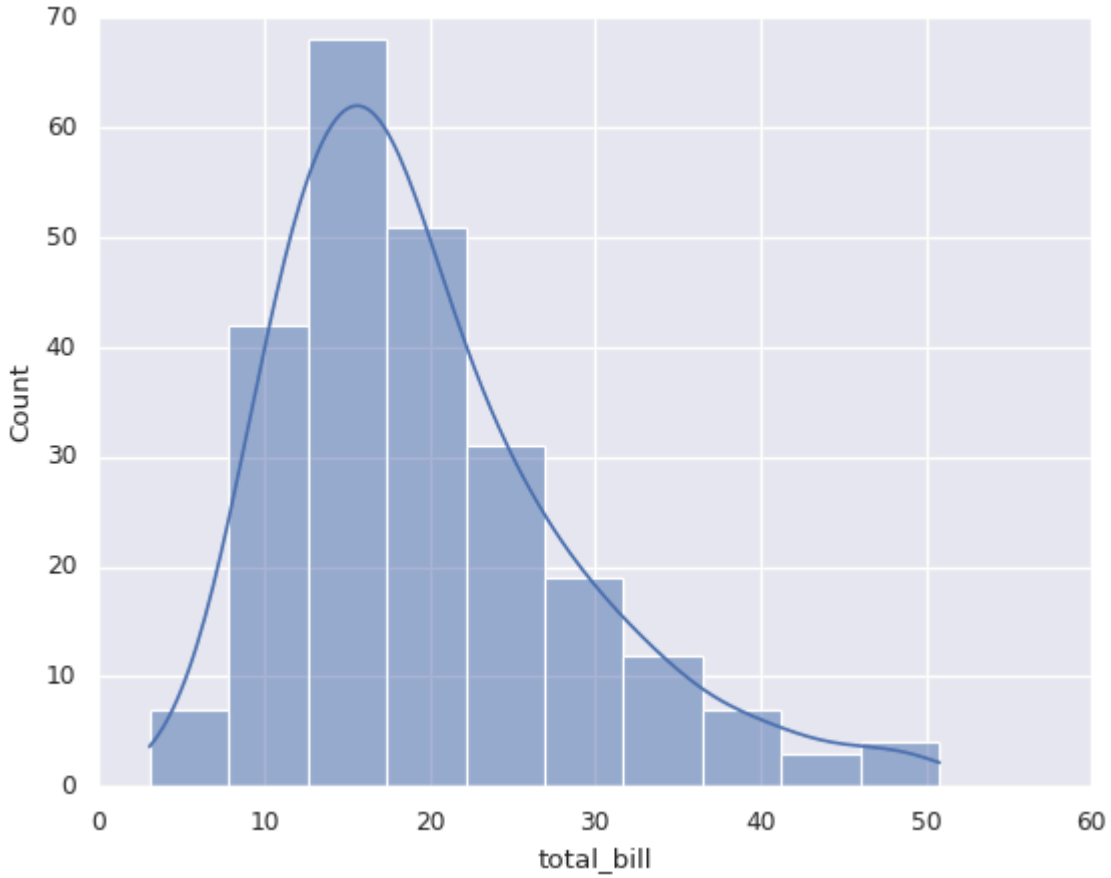
Out[209]:

```
<Axes: xlabel='day', ylabel='total_bill'>
```



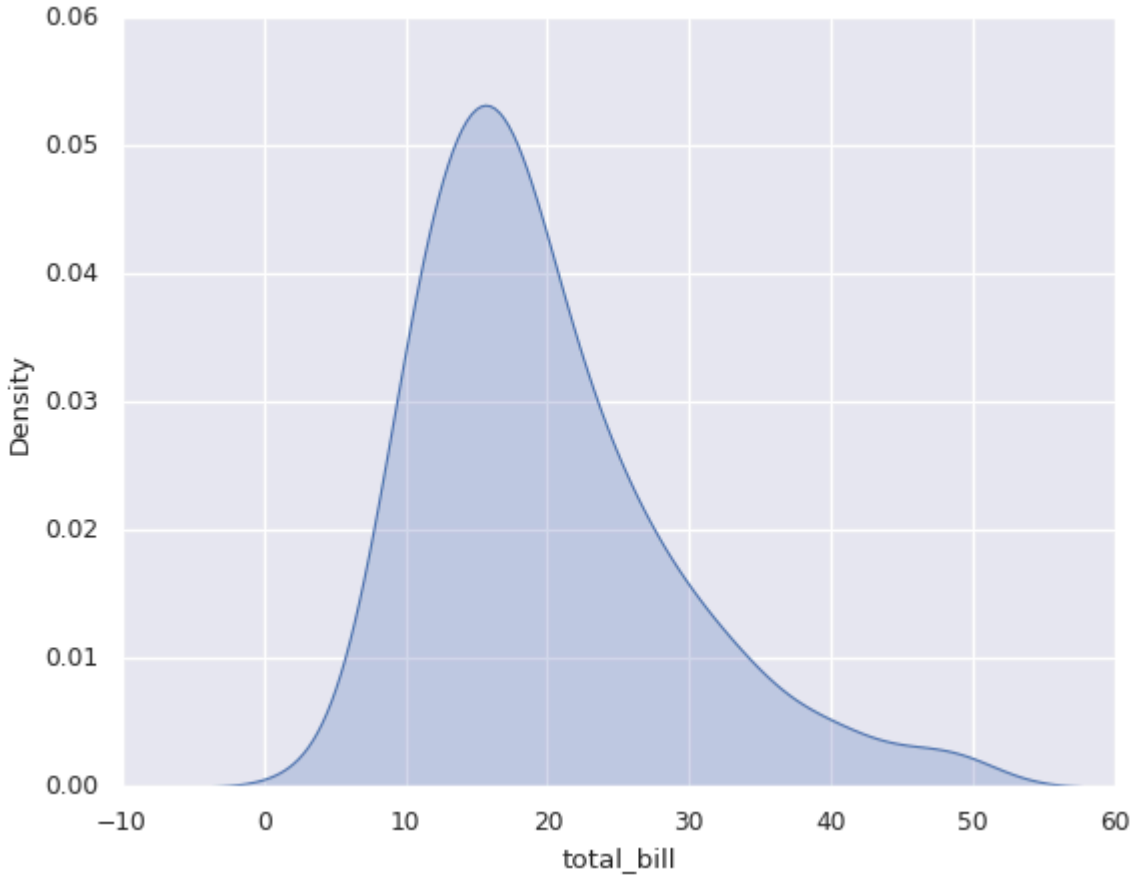
In [210…
```
### Histograms
sns.histplot(tips['total_bill'],bins=10,kde=True)
```

Out[210]:
```
<Axes: xlabel='total_bill', ylabel='Count'>
```



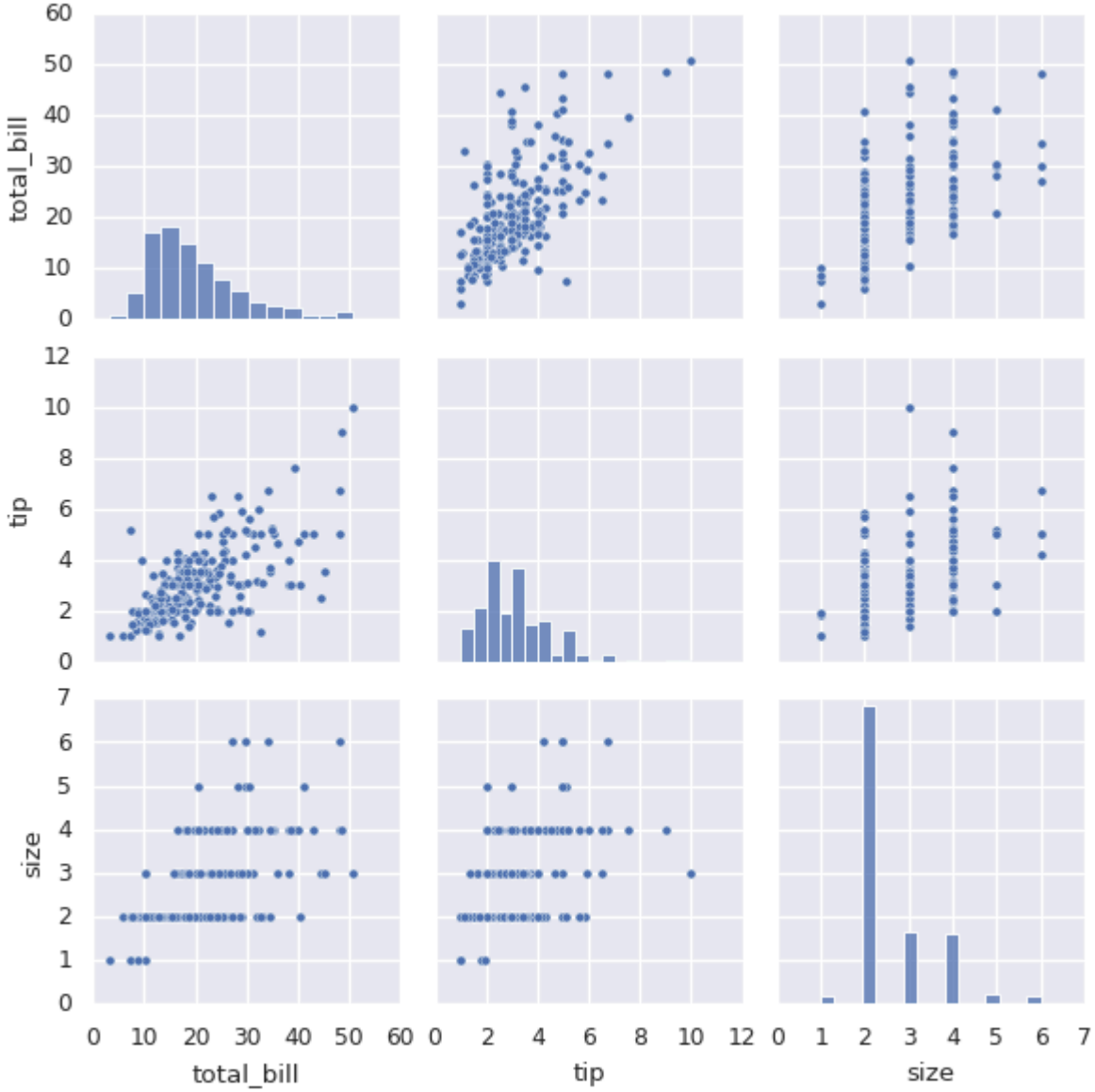
In [211…
```
## KDE Plot
sns.kdeplot(tips['total_bill'],fill=True)
```

Out[211]:
```
<Axes: xlabel='total_bill', ylabel='Density'>
```



In [212…
```
# Pairplot
sns.pairplot(tips)
```

Out[212]:
```
<seaborn.axisgrid.PairGrid at 0x79c69aead240>
```



In [213…

```
## HEatmap
corr=tips[['total_bill','tip','size']].corr()
corr
```

Out[213]:

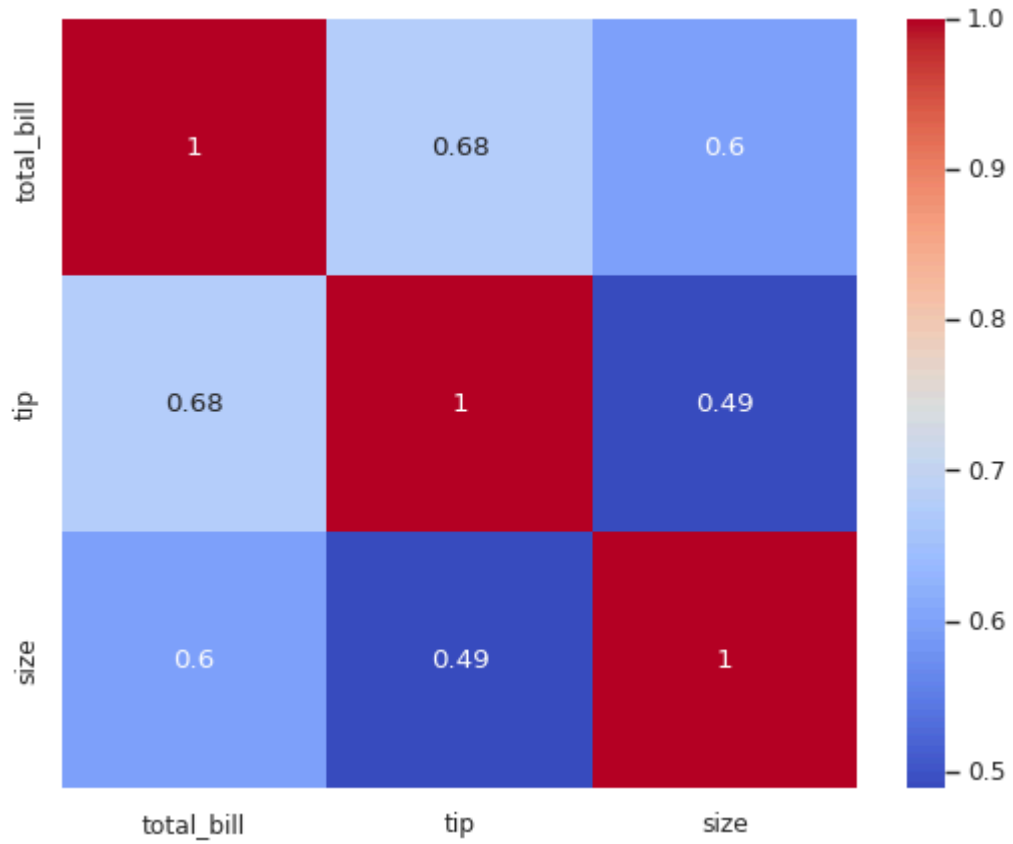
	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [214…

```
sns.heatmap(corr,annot=True,cmap='coolwarm')
```

Out[214]:

```
<Axes: >
```



In [ ]: