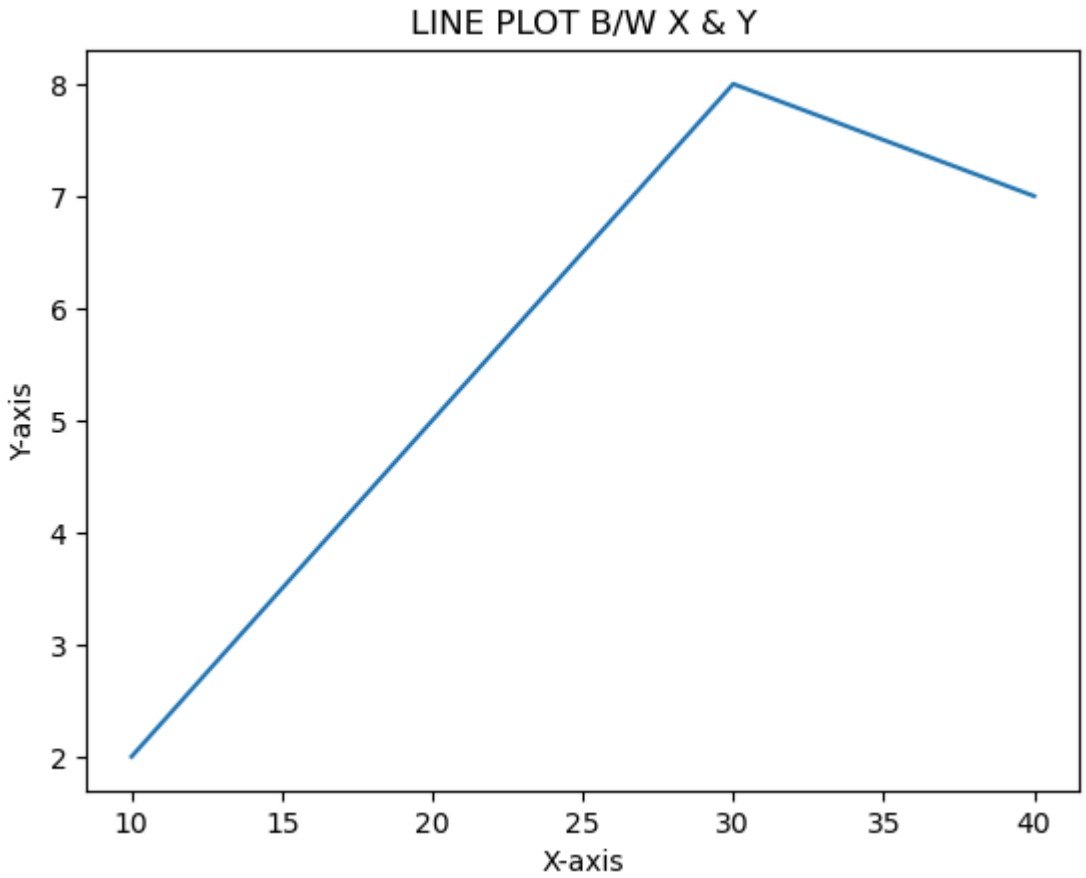
MATPLOTLIB TUTORIAL

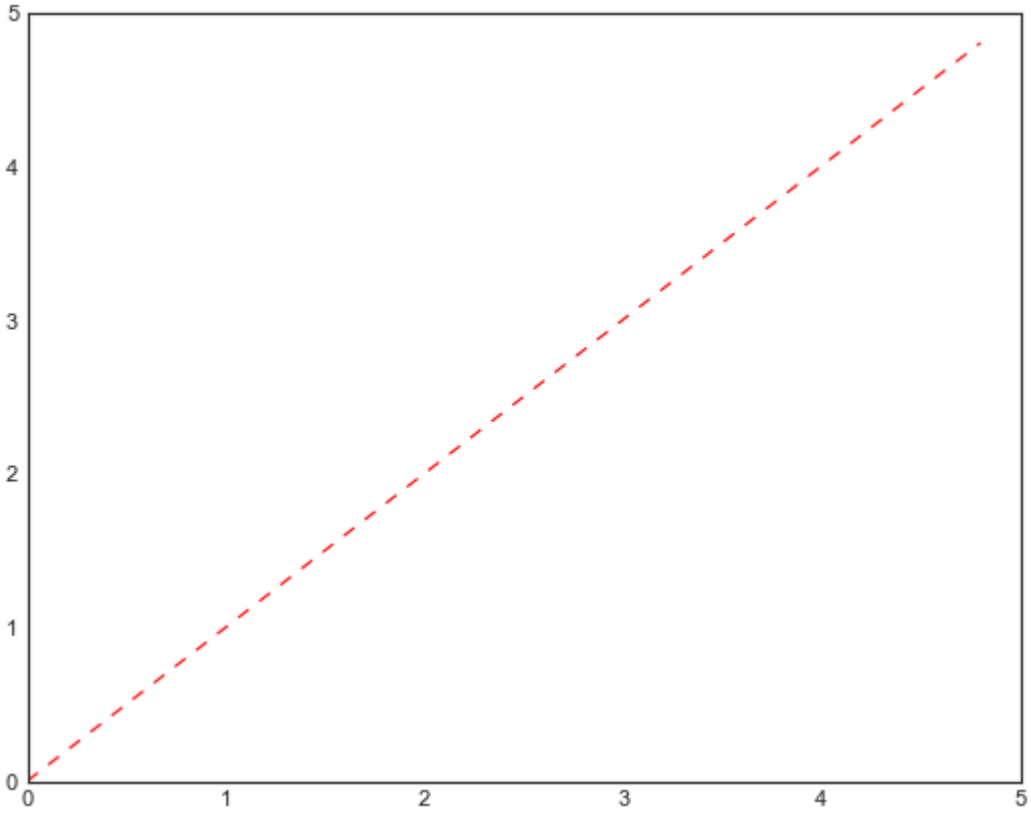
Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc.

In [55]:
```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

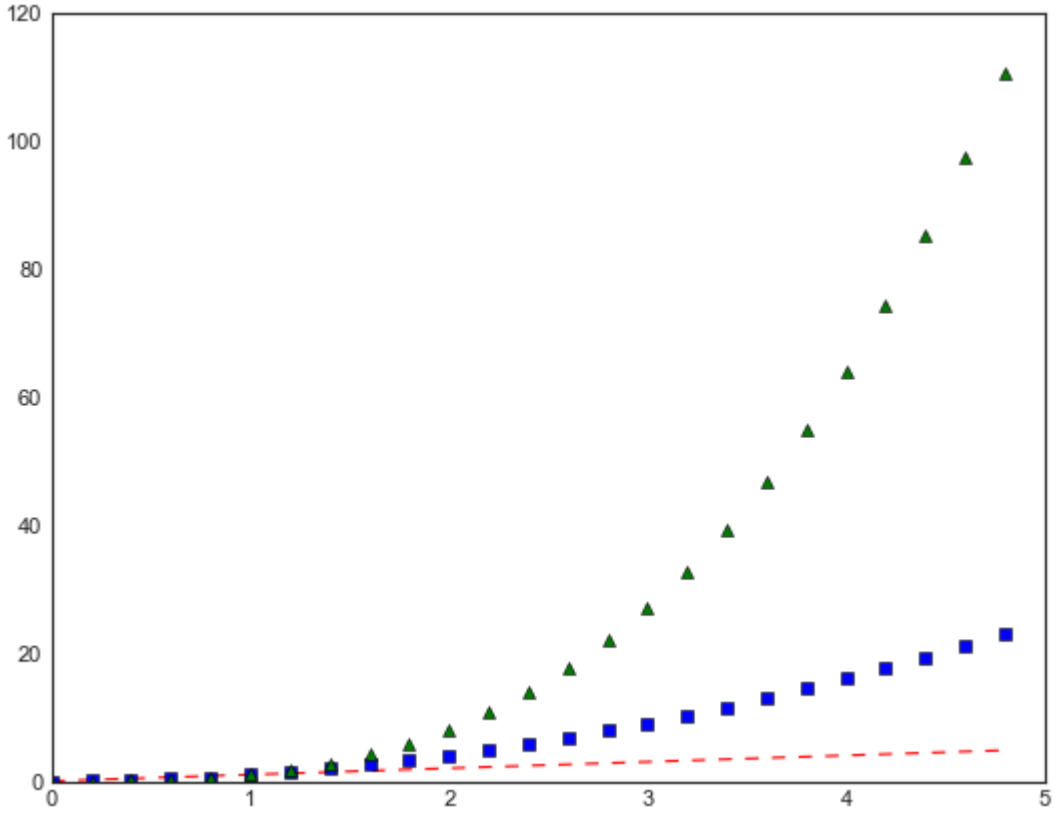
In [3]:
```
#initializing data
x=[10,20,30,40]
y=[2,5,8,7]

#plotting the data
plt.plot(x,y)

plt.title("LINE PLOT B/W X & Y") #adding titles

#adding labels
plt.ylabel("Y-axis")
plt.xlabel("X-axis")
plt.show()
```



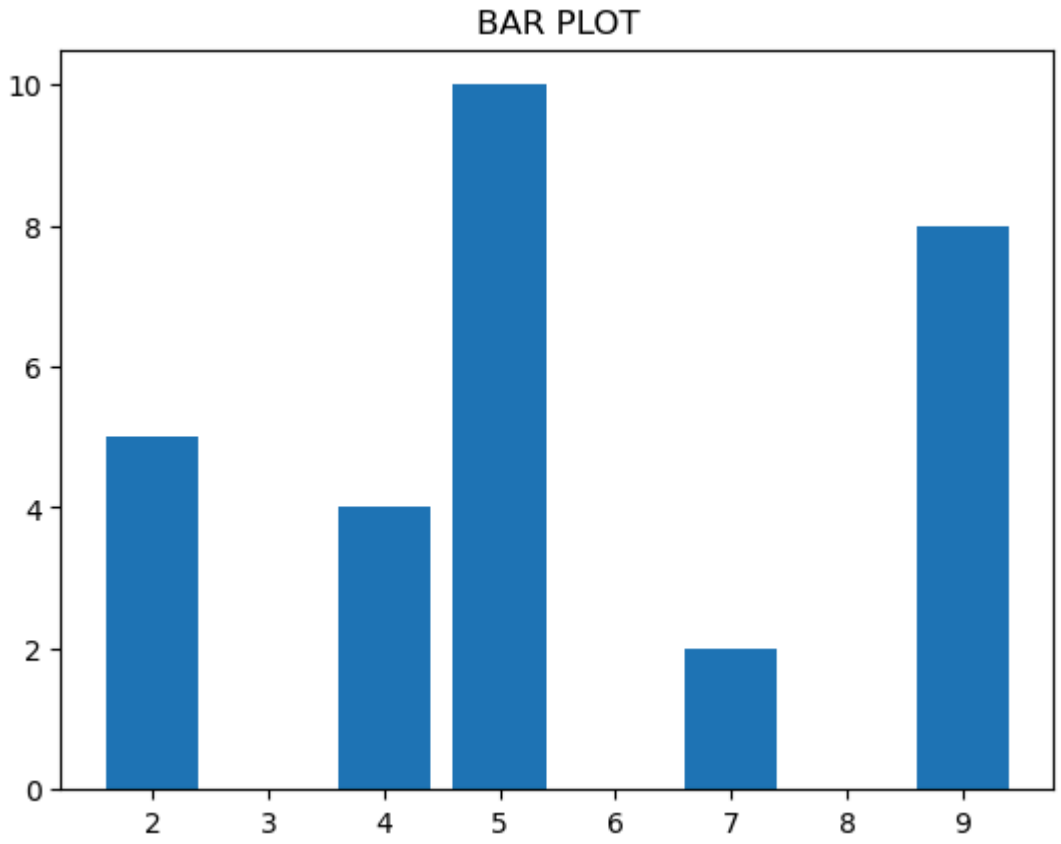
In [229]:
```
ar=np.arange(0,5,0.2)
plt.plot(ar,ar,'r--')
plt.show()
```



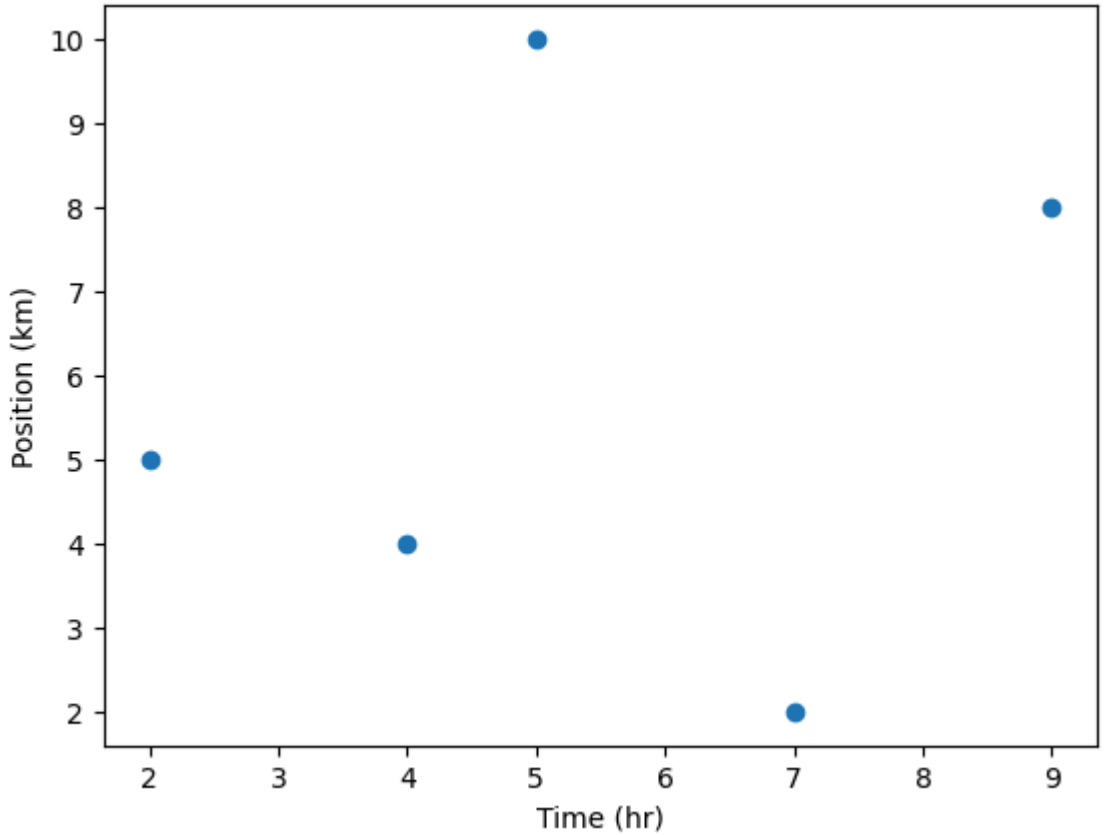
In [231]:
```
ar=np.arange(0,5,0.2)
plt.plot(ar,ar,'r--',ar,ar**2,'bs',ar,ar**3,'g^')
plt.show()
```



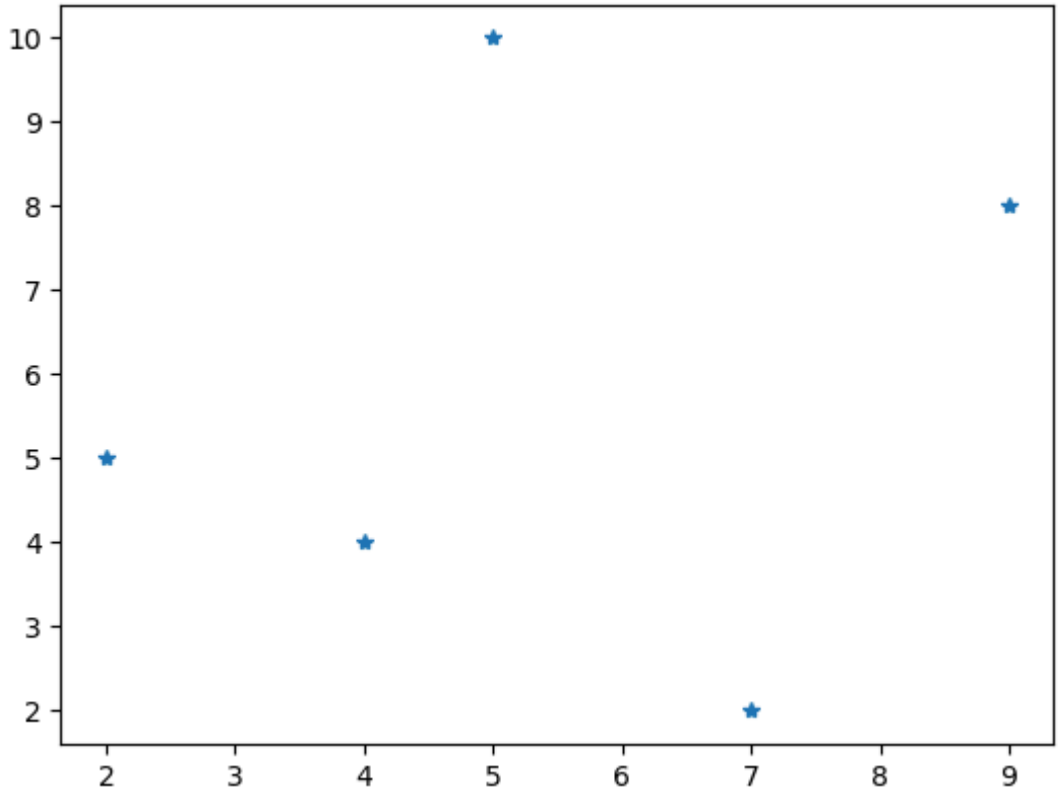
In [4]:
```
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]

# Function to plot the bar
plt.bar(x,y)

plt.title("BAR PLOT")
plt.show()
```



In [8]:
```
# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot the bar
plt.scatter(x,y)

plt.xlabel("Time (hr)")
plt.ylabel("Position (km)")

# function to show the plot
plt.show()
```

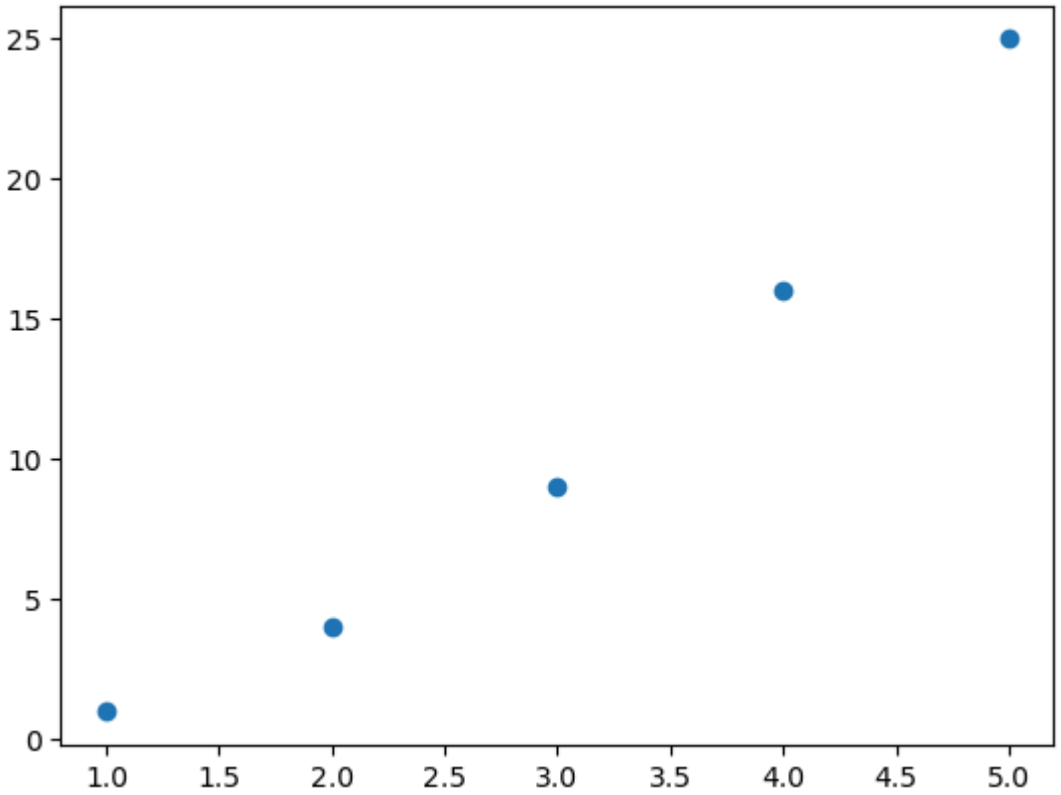


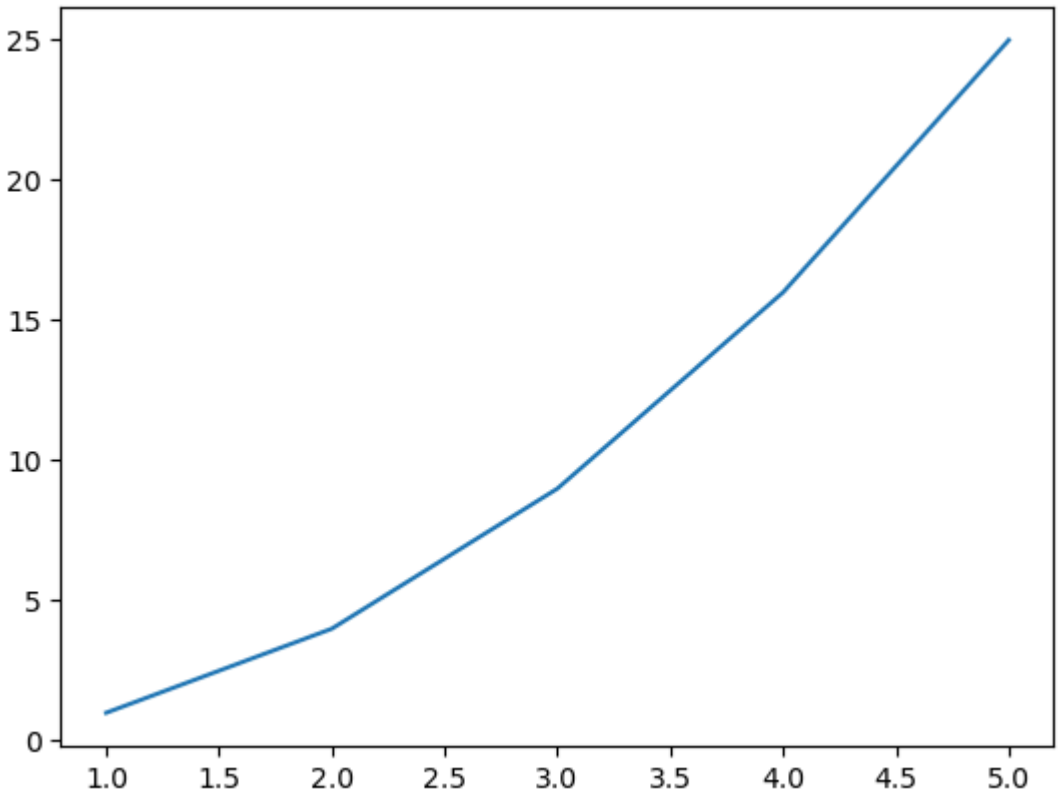
In [7]:
```
plt.plot(x,y,"*") # Function to plot scatter using plot() method
plt.show()
```



Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays.

In [9]:
```
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
plt.scatter(x, y)

plt.show()

plt.plot(x, y)

# function to show the plot
plt.show()
```

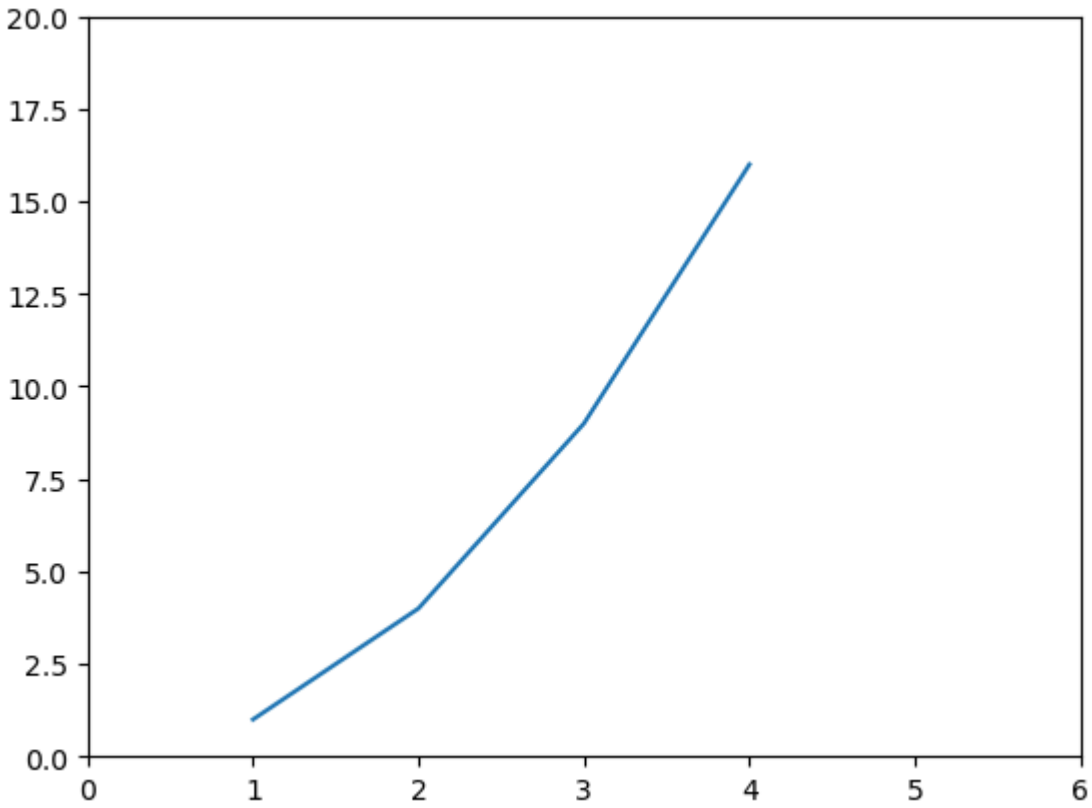




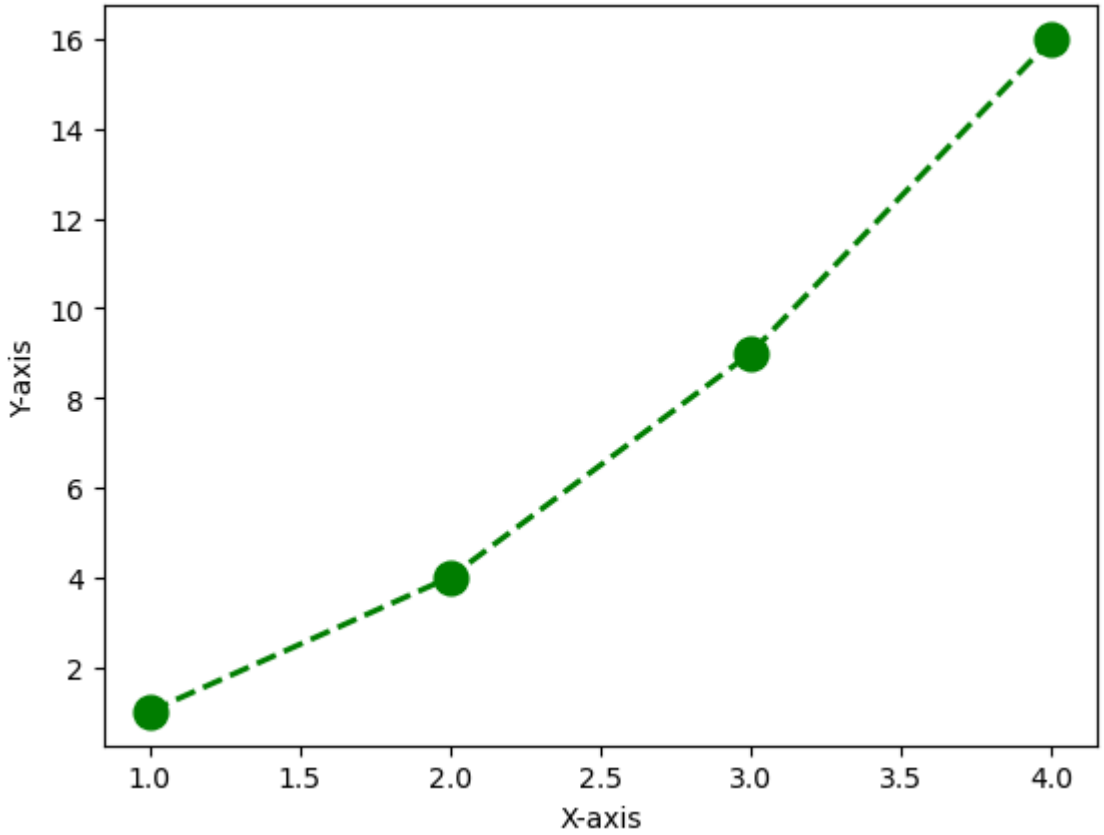
Pyplot in MatplotLib

Pyplot is a Matplotlib module which provides a MATLAB-like interface. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. The various plots we can utilize using Pyplot are Line Plot, Histogram, Scatter, 3D Plot, Image, Contour, and Polar.

In [10]:
```
#plot function
plt.plot([1,2,3,4],[1,4,9,16])
plt.axis([0,6,0,20])
plt.show()
```

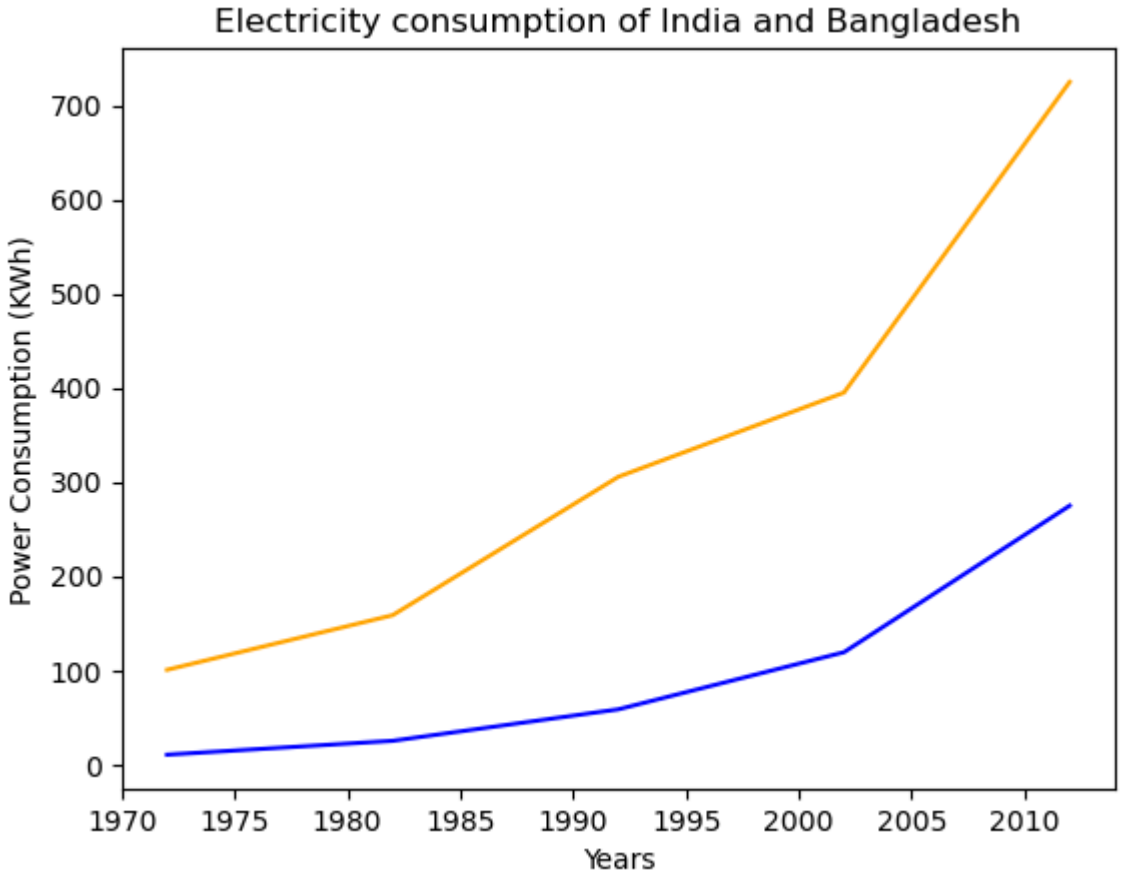


In [13]:
```
plt.plot([1,2,3,4],[1,4,9,16],color='green', marker='o', linestyle='dashed', linewidth=2, markersize=12)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



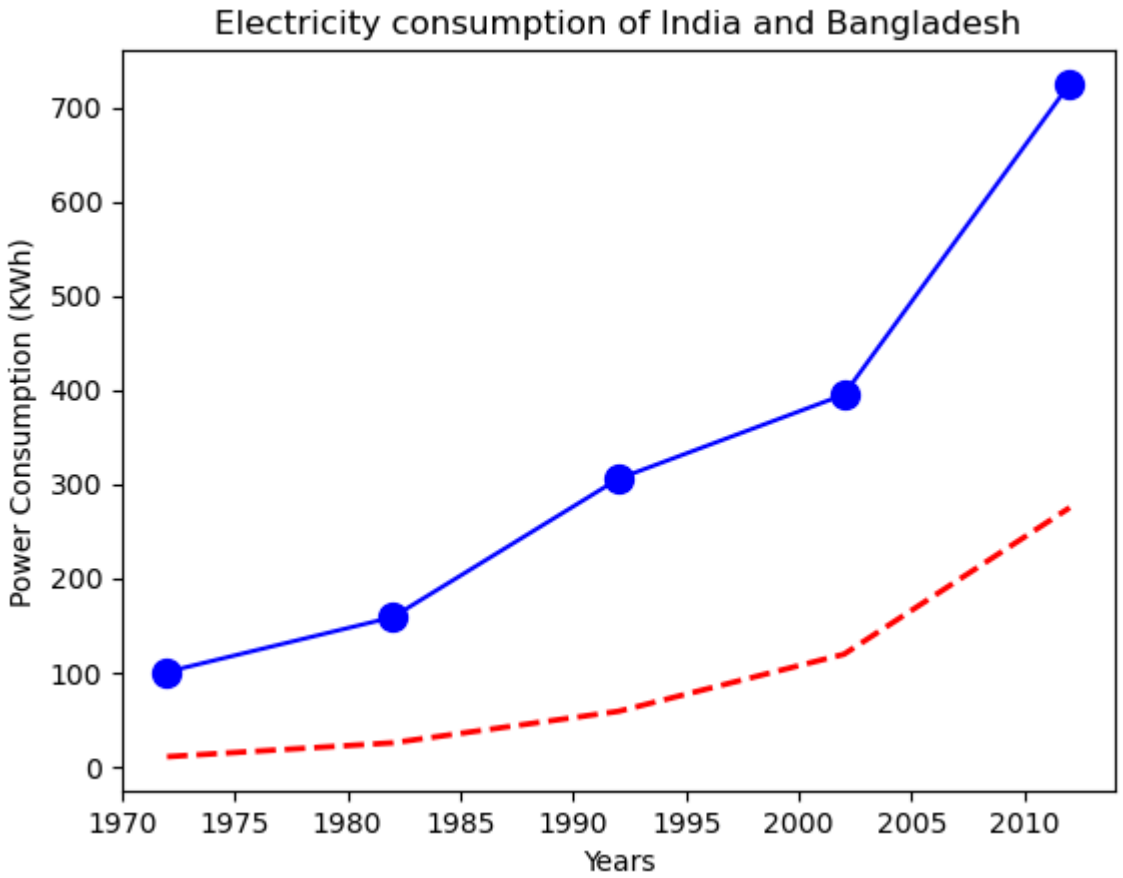
LINEAR PLOT

In [14]:
```
year = [1972, 1982, 1992, 2002, 2012]
india = [100.6, 158.61, 305.54, 394.96, 724.79]
bangladesh = [10.5, 25.21, 58.65, 119.27, 274.87]

plt.plot(year,india,color='orange',label='INDIA')
plt.plot(year,bangladesh,color='blue',label='BANGLADESH')

#naming of x-axis and y-axis
plt.xlabel('Years')
plt.ylabel('Power Consumption (KWh)')
plt.title('Electricity consumption of India and Bangladesh')

plt.show()
```



FORMATTING IN PLOTS

In [21]:
```
year = [1972, 1982, 1992, 2002, 2012]
india = [100.6, 158.61, 305.54, 394.96, 724.79]
bangladesh = [10.5, 25.21, 58.65, 119.27, 274.87]

plt.plot(year,india,color='blue',label='INDIA',marker='o',markersize=10)
plt.plot(year,bangladesh,color='red',label='BANGLADESH',linestyle='dashed',linewidth=2)

plt.xlabel('Years')
plt.ylabel('Power Consumption (KWh)')
plt.title('Electricity consumption of India and Bangladesh')

plt.show()
```

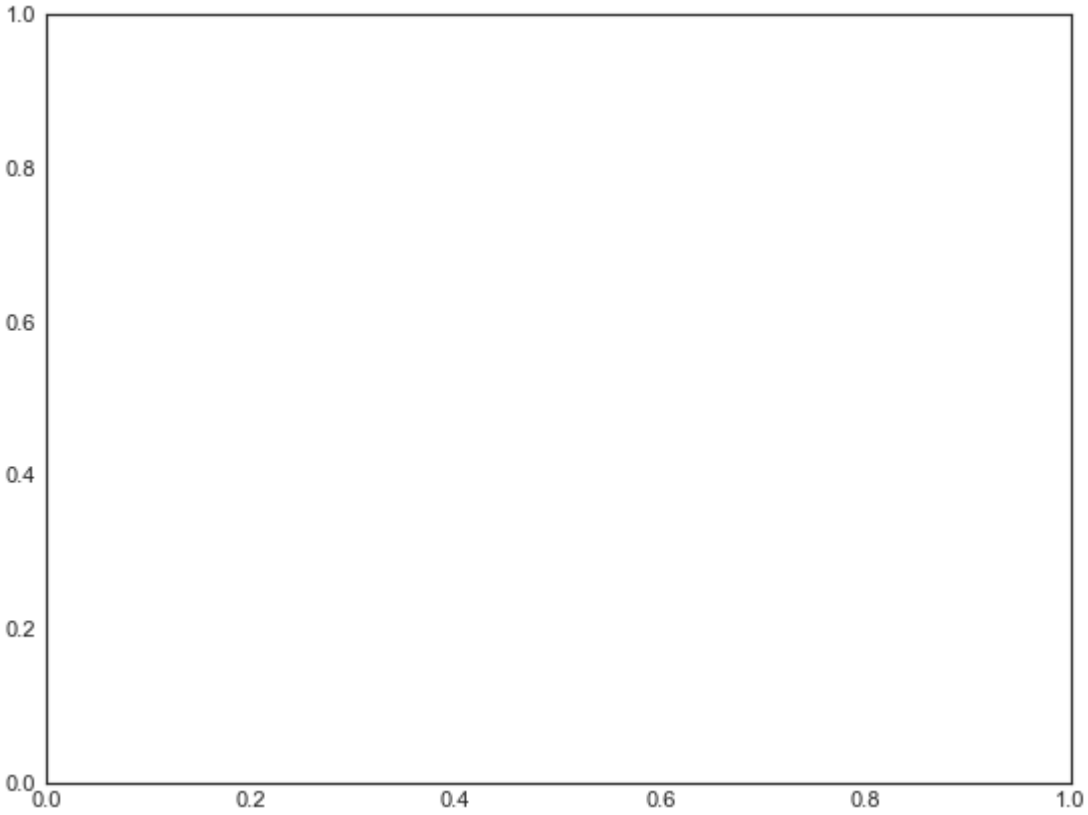


AXES CLASS

Axes is the most basic and flexible unit for creating sub-plots. Axes allow placement of plots at any location in the figure.

axes() function axes() function creates axes object with argument, where argument is a list of 4 elements [left, bottom, width, height]

In [232]:
```
fig = plt.figure()

#[left, bottom, width, height]
ax = plt.axes([0.1, 0.1, 0.8, 0.8])

#The first '0.1' refers to the distance between the left side axis and border of the figure window is 10% of the total
#The second '0.1' refers to the distance between the bottom side axis and the border of the figure window is 10%, of t
#The first '0.8' means the axes width from left to right is 80%
# next'0.8' means the axes height from the bottom to the top is 80%.
```



ax.plot() function

plot() function of the axes class plots the values of one array versus another as line or marker.

Syntax : plt.plot(X, Y, 'CLM')

Parameters: X is x-axis. Y is y-axis. 'CLM' stands for Color, Line and Marker.

Dotted line (':'), dashed line ('—'), solid line ('-')

'.' Point Marker 'o' Circle Marker '+' Plus Marker 's' Square Marker 'D' Diamond Marker 'H' Hexagon Marker

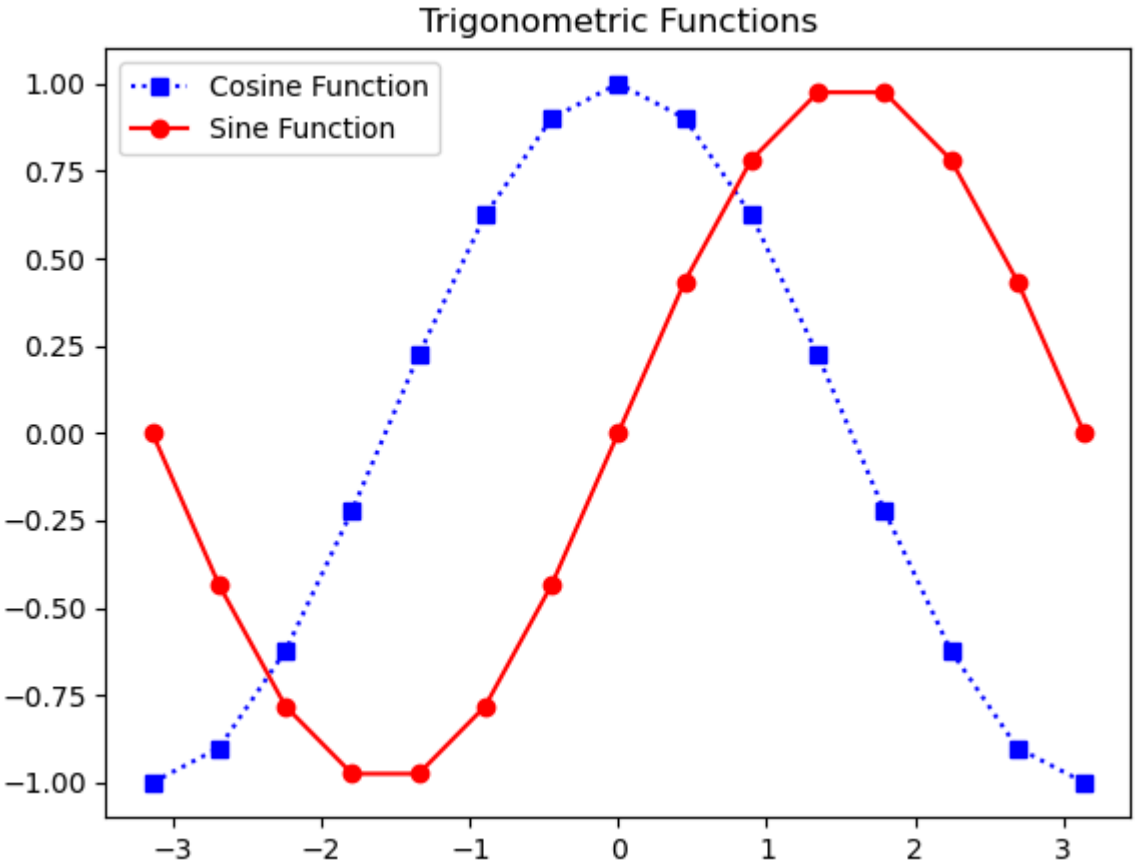
ax.legend() function

Adding legend to the plot figure can be done by calling the legend() function of the axes class. It consists of three arguments.

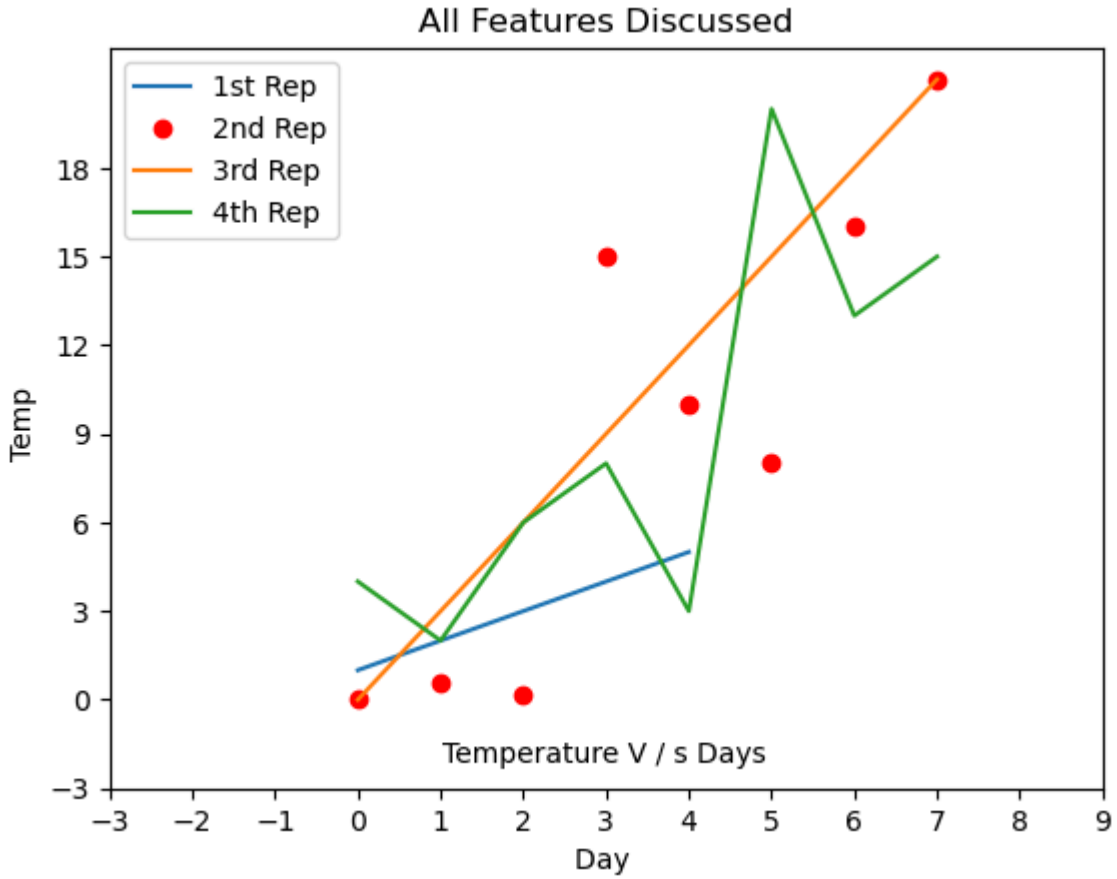
Syntax :ax.legend(handles, labels, loc)

Parameter: label- sequence of strings or handles loc - string or integer specifying the location of legend

In [25]:
```
X = np.linspace(-np.pi, np.pi, 15)
C = np.cos(X)
S = np.sin(X)

ax = plt.axes([0.1, 0.1, 0.8, 0.8])

# 'bs:' mentions blue color, square
# marker with dotted line.
ax1 = ax.plot(X, C, 'bs:')

#'ro-' mentions red color, circle
# marker with solid line.
ax2 = ax.plot(X, S, 'ro-')

ax.legend(labels = ('Cosine Function',
                    'Sine Function'),
          loc = 'upper left')

ax.set_title("Trigonometric Functions")

plt.show()
```


Trigonometric Functions
Cosine Function
Sine Function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−3
−2
−1
0
1
2
3

In [8]:
```
import matplotlib.pyplot as plt
a = [1, 2, 3, 4, 5]
b = [0, 0.6, 0.2, 15, 10, 8, 16, 21]
plt.plot(a)

plt.plot(b, "or") # o is for circles and r is for red
plt.plot(list(range(0, 22, 3)))

plt.xlabel('Day ')
plt.ylabel('Temp ')

c = [4, 2, 6, 8, 3, 20, 13, 15]
plt.plot(c, label = '4th Rep')

ax = plt.gca()  # get current axes command



# set the interval by which the x-axis set the marks
plt.xticks(list(range(-3, 10)))

# set the intervals by which y-axis set the marks
plt.yticks(list(range(-3, 20, 3)))

ax.legend(['1st Rep', '2nd Rep', '3rd Rep', '4th Rep'])


plt.annotate('Temperature V / s Days', xy = (1.01, -2.15))  # xy denotes the position on the graph

plt.title('All Features Discussed')
plt.show()
```



CREATING MULTIPLE SUBPLOTS IN MATPLOTLIB

To create multiple plots use matplotlib.pyplot.subplots method which returns the figure along with Axes object or array of Axes object. nrows, ncols attributes of subplots() method determine the number of rows and columns of the subplot grid.

By default, it returns a figure with a single plot. For each axes object i.e plot we can set title (set via set_title()), an x-label (set via set_xlabel()), and a y-label set via set_ylabel())

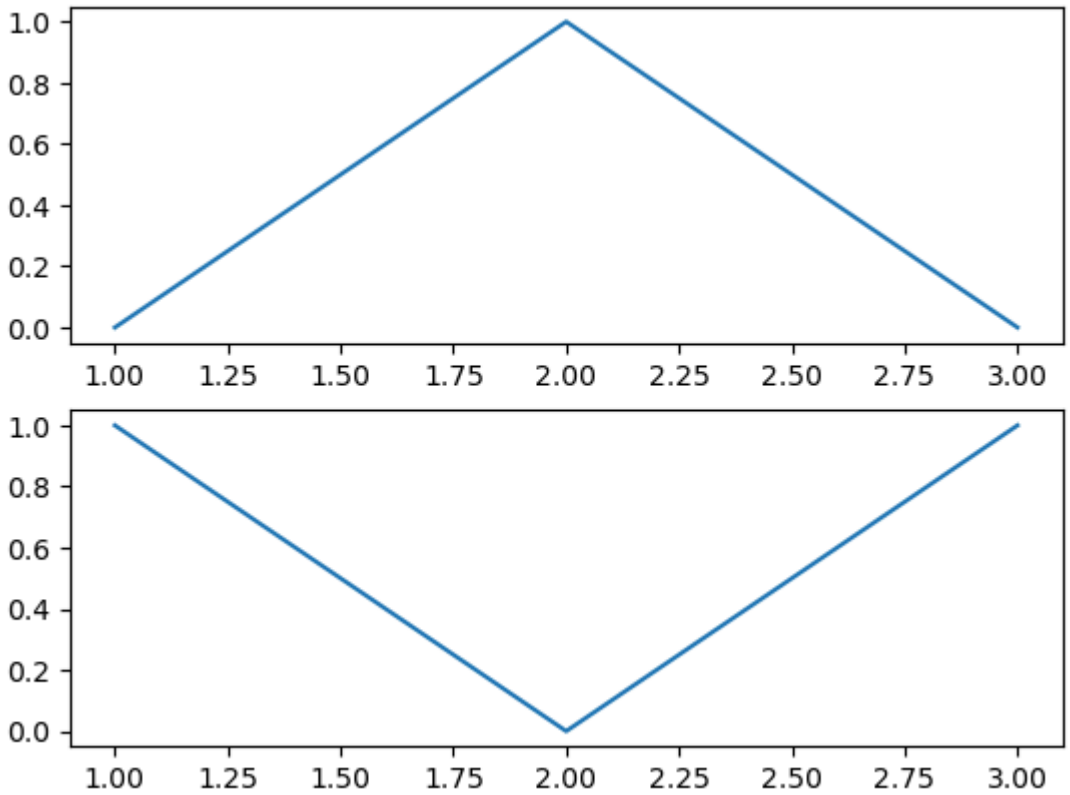
Subplots : The subplots() function in pyplot module of matplotlib library is used to create a figure and a set of subplots. Subplots are required when we want to show two or more plots in same figure.

1D ARRAY OF SUBPLOTS: stacking in only one direction

In [ ]:
```
fig=plt.figure()  # Create a figure
ax=fig.add_subplot()  # Add a subplot
plt.show()

# Creates just a figure and only one subplot
fig, ax = plt.subplots()
```

In [26]:
```
x = [1, 2, 3]
y = [0, 1, 0]
z = [1, 0, 1]

# Creating 2 subplots
fig, ax = plt.subplots(2)

#fig represents the Figure object, which is the top-level container for all plot elements.

# Accessing each axes object to plot the data through returned array
ax[0].plot(x, y)
ax[1].plot(x, z)
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x21583480190>]
```

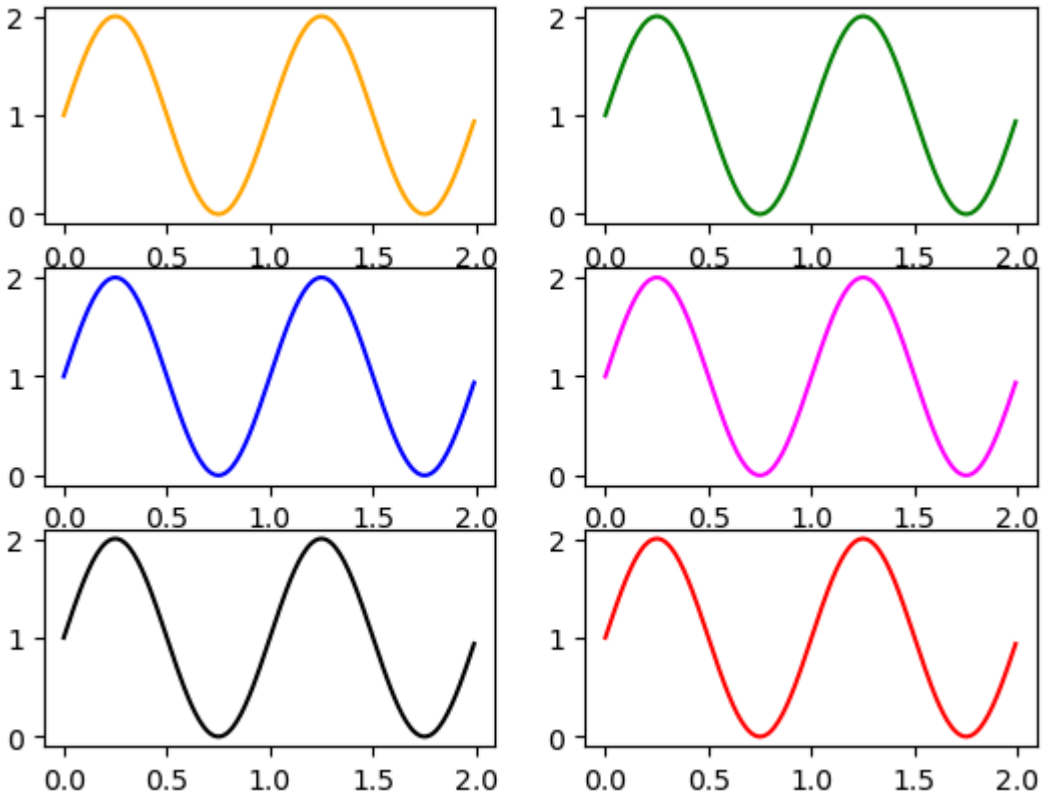


access these axes objects using indices just like we access elements of the array. To create specific subplots, call matplotlib.pyplot.plot() on the corresponding index of the axes

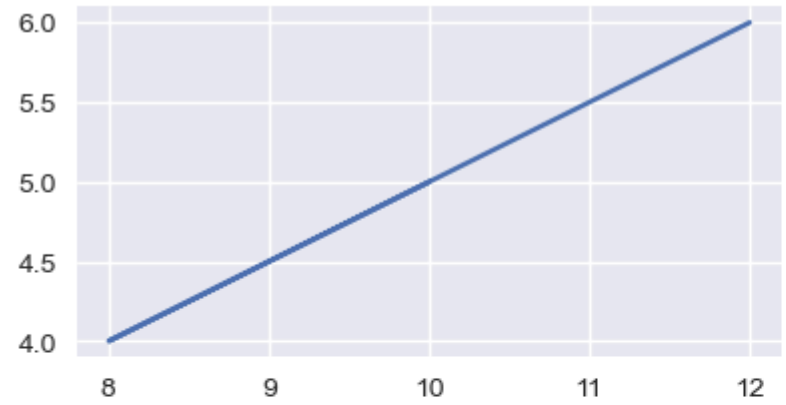
2D ARRAY OF AXES OBJECTS: stacking in two directions

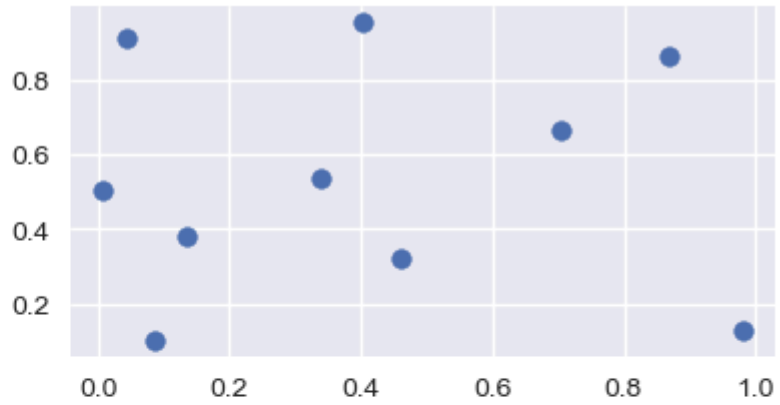
In [27]:
```
x = np.arange(0.0, 2.0, 0.01)
y = 1 + np.sin(2 * np.pi * x)

# Creating 6 subplots and unpacking the output array immediately
fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(3, 2)
ax1.plot(x, y, color="orange")
ax2.plot(x, y, color="green")
ax3.plot(x, y, color="blue")
ax4.plot(x, y, color="magenta")
ax5.plot(x, y, color="black")
ax6.plot(x, y, color="red")
```

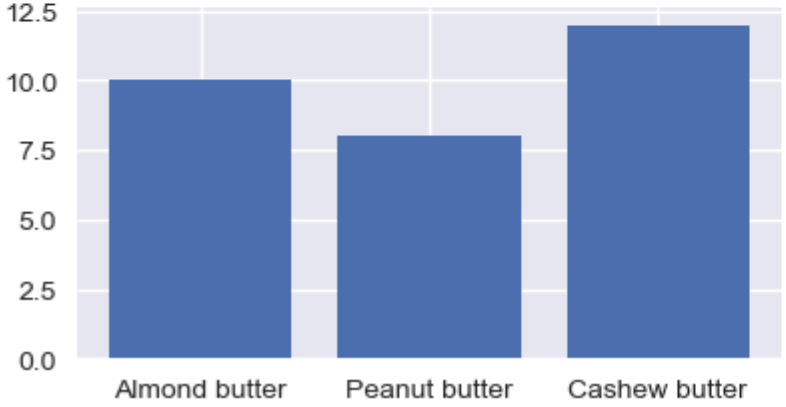
Out[27]:
```
[<matplotlib.lines.Line2D at 0x21581beafd0>]
```

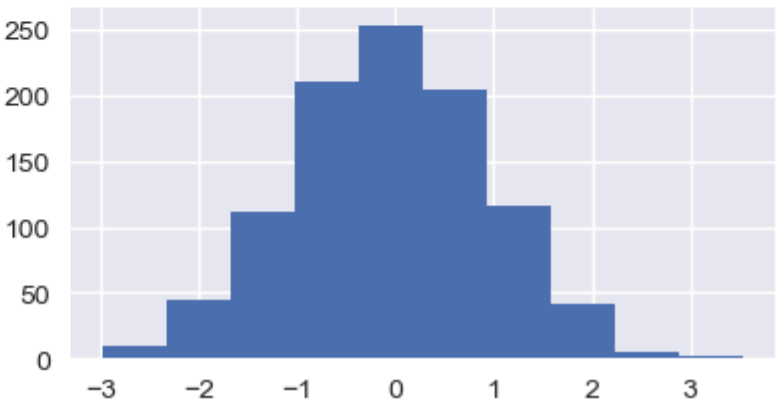

2
1
0
0.0
0.5
1.0
1.5
2.0

In [138]:
```
import matplotlib.pyplot as plt
import numpy as np

nut_butter_prices = {"Almond butter": 10, "Peanut butter": 8, "Cashew butter": 12}

# Create the figure and Axes objects
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize=(10, 5))

x=np.array([10,8,12])
# Plot data to each axis
ax1.plot(x, x/2)
ax2.scatter(np.random.random(10), np.random.random(10))
ax3.bar(nut_butter_prices.keys(), nut_butter_prices.values())
ax4.hist(np.random.randn(1000))

# Display the plot
plt.show()
```

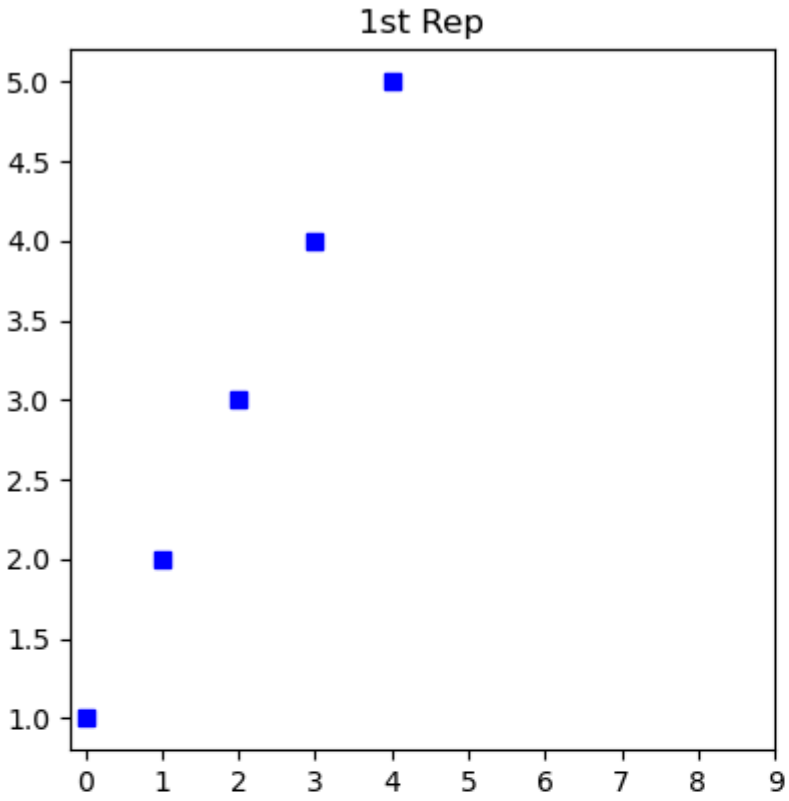

6.0
5.5
5.0
4.5
4.0
8
9
10
11
12

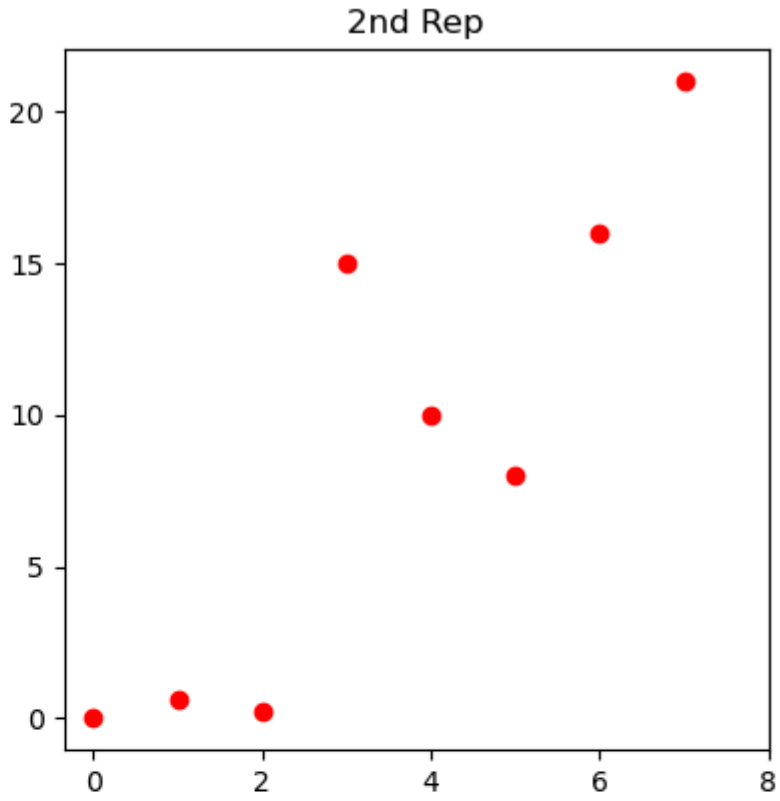

0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

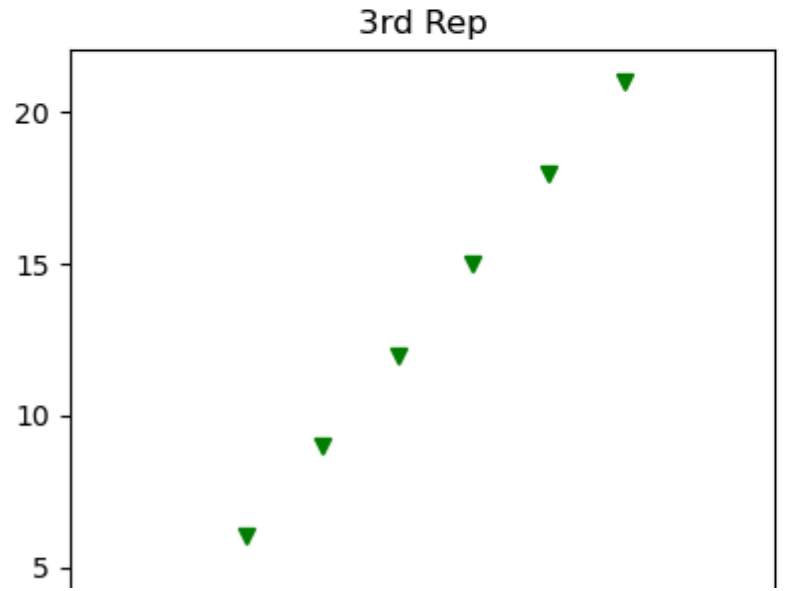

12.5
10.0
7.5
5.0
2.5
0.0
Almond butter
Peanut butter
Cashew butter

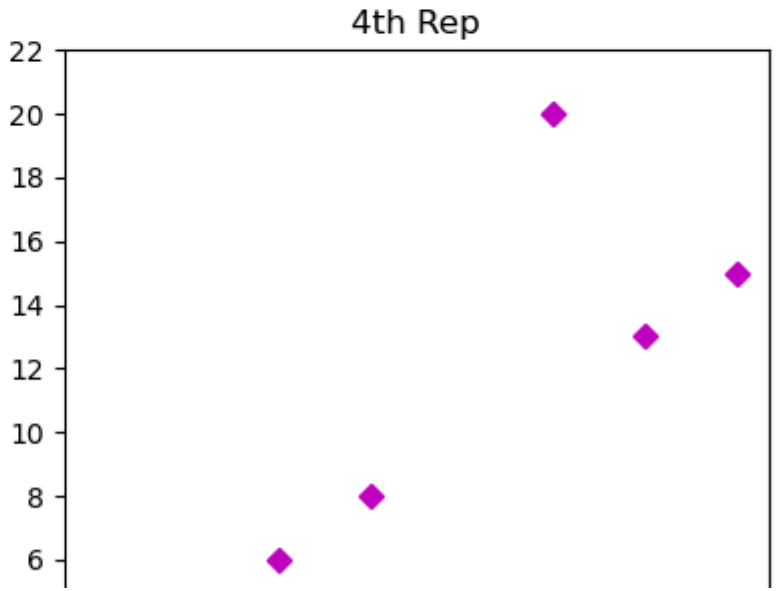

250
200
150
100
50
0
−3
−2
−1
0
1
2
3

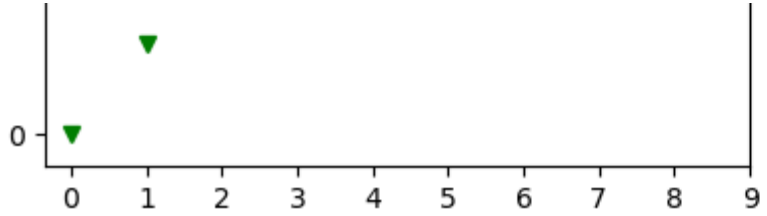
In [9]:
```
a = [1, 2, 3, 4, 5]
b = [0, 0.6, 0.2, 15, 10, 8, 16, 21]
c = [4, 2, 6, 8, 3, 20, 13, 15]

fig = plt.figure(figsize =(10, 10))
# creating multiple plots in a single plot

sub1 = plt.subplot(2, 2, 1)
sub2 = plt.subplot(2, 2, 2)
sub3 = plt.subplot(2, 2, 3)
sub4 = plt.subplot(2, 2, 4)
sub1.plot(a, 'sb')

sub1.set_xticks(list(range(0, 10, 1)))
sub1.set_title('1st Rep')

sub2.plot(b, 'or')
sub2.set_xticks(list(range(0, 10, 2)))
sub2.set_title('2nd Rep')


sub3.plot(list(range(0, 22, 3)), 'vg')
sub3.set_xticks(list(range(0, 10, 1)))
sub3.set_title('3rd Rep')

sub4.plot(c, 'Dm')
sub4.set_yticks(list(range(0, 24, 2)))
sub4.set_title('4th Rep')

plt.show()
```

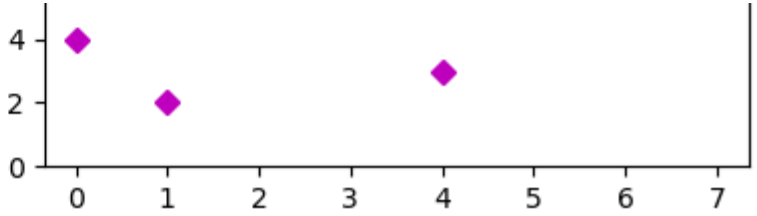

1st Rep
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
2
3
4
5
6
7
8
9


2nd Rep
20
15
10
5
0
0
2
4
6
8

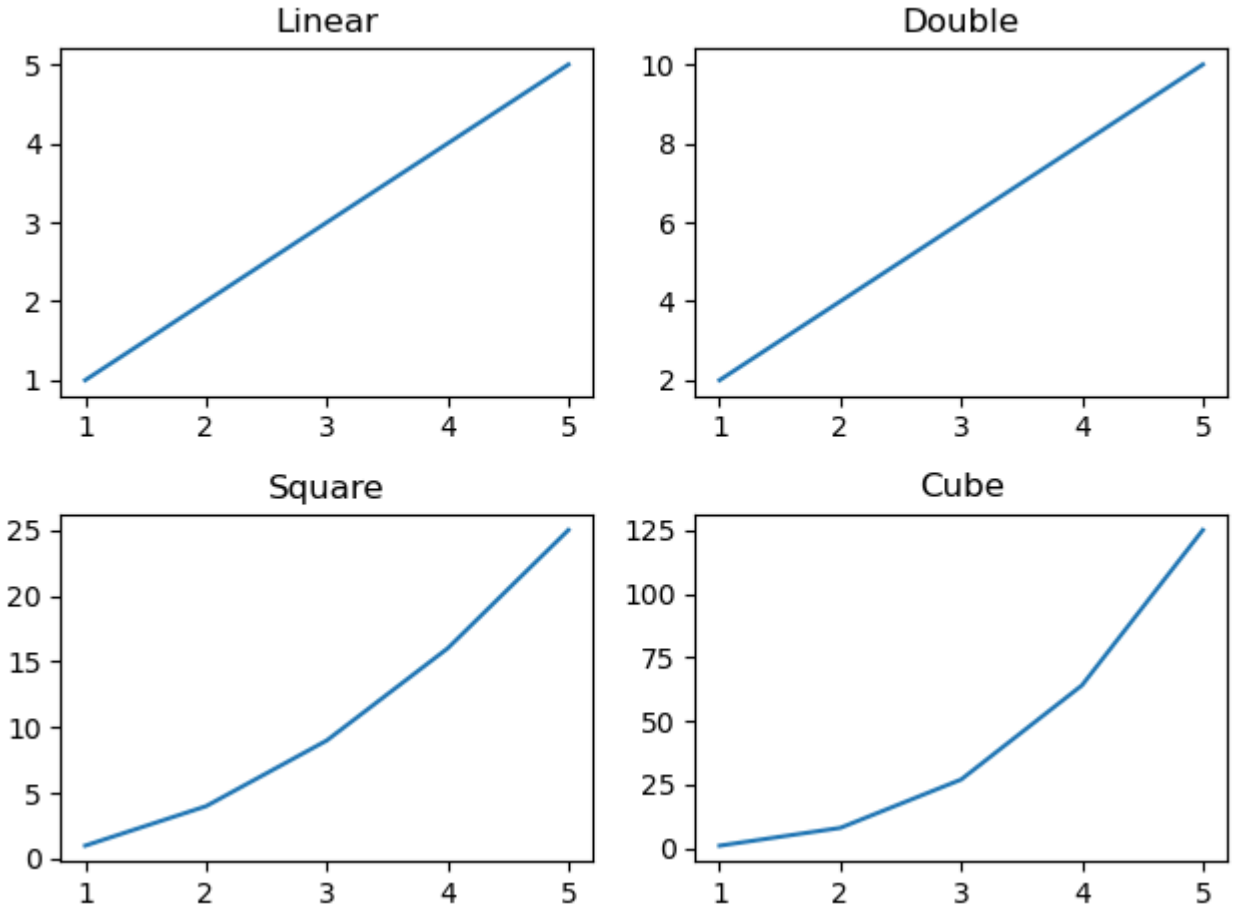

3rd Rep
20
15
10
5


4th Rep
22
20
18
16
14
12
10
8
6

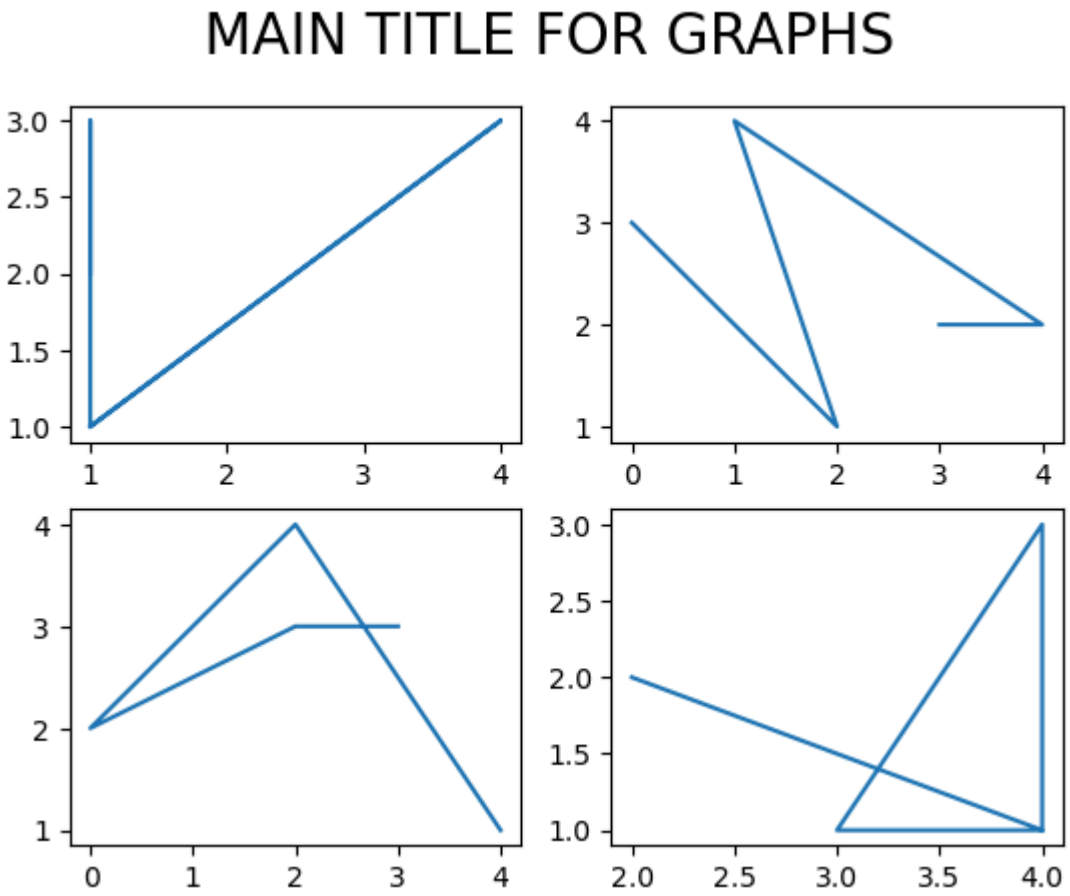

0
0 1 2 3 4 5 6 7 8 9


4
2
0
0 1 2 3 4 5 6 7

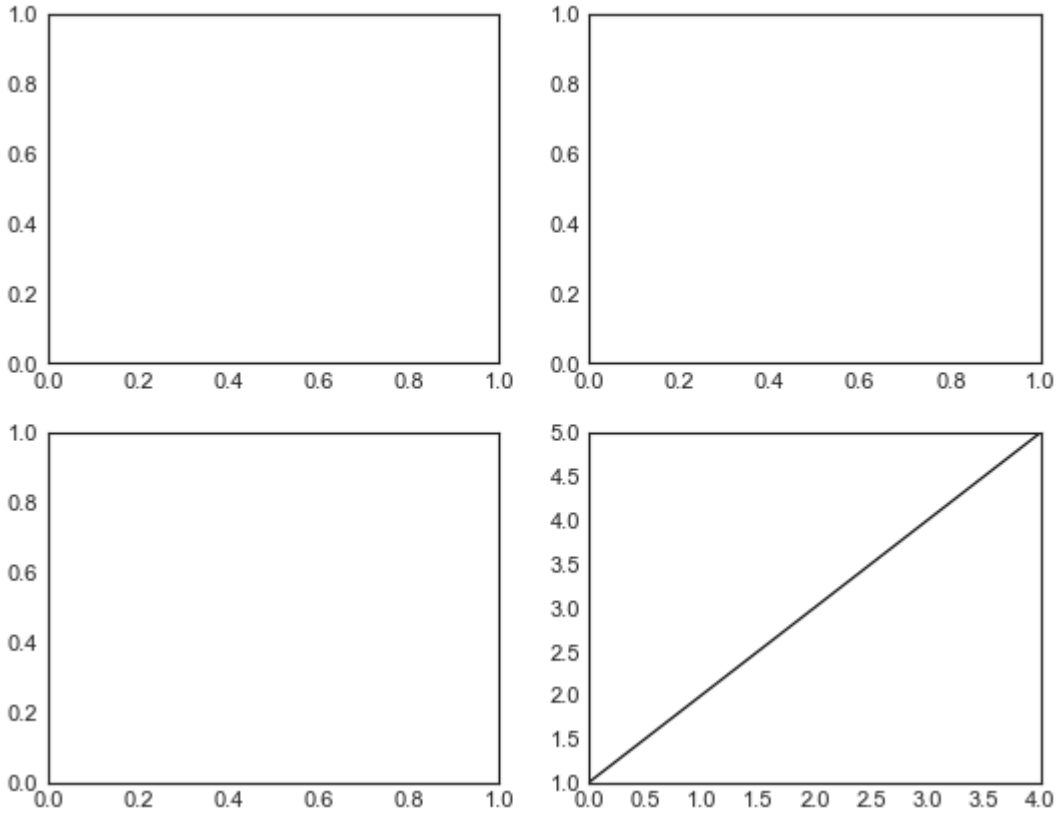
In [3]:
```
x=np.array([1, 2, 3, 4, 5])

# making subplots
fig, ax = plt.subplots(2, 2)

# set data with subplots and plot
ax[0, 0].plot(x, x)
ax[0, 1].plot(x, x*2)
ax[1, 0].plot(x, x*x)
ax[1, 1].plot(x, x*x*x)

# set the title to subplots using set_title()
ax[0, 0].set_title("Linear")
ax[0, 1].set_title("Double")
ax[1, 0].set_title("Square")
ax[1, 1].set_title("Cube")

#or ax[0, 0].title.set_text("Linear")

# set spacing
fig.tight_layout()
plt.show()
```


Linear
Double
Square
Cube

In [7]:
```
# making subplots objects
fig, ax = plt.subplots(2, 2)

# draw graph
ax[0][0].plot(np.random.randint(0, 5, 5), np.random.randint(0, 5, 5))
#np.random.randint(0, 5, 5) generates an array of 5 random integers between 0 and 5,
ax[0][1].plot(np.random.randint(0, 5, 5), np.random.randint(0, 5, 5))
ax[1][0].plot(np.random.randint(0, 5, 5), np.random.randint(0, 5, 5))
ax[1][1].plot(np.random.randint(0, 5, 5), np.random.randint(0, 5, 5))

fig.suptitle(' MAIN TITLE FOR GRAPHS ', fontsize=20)

plt.show()
```

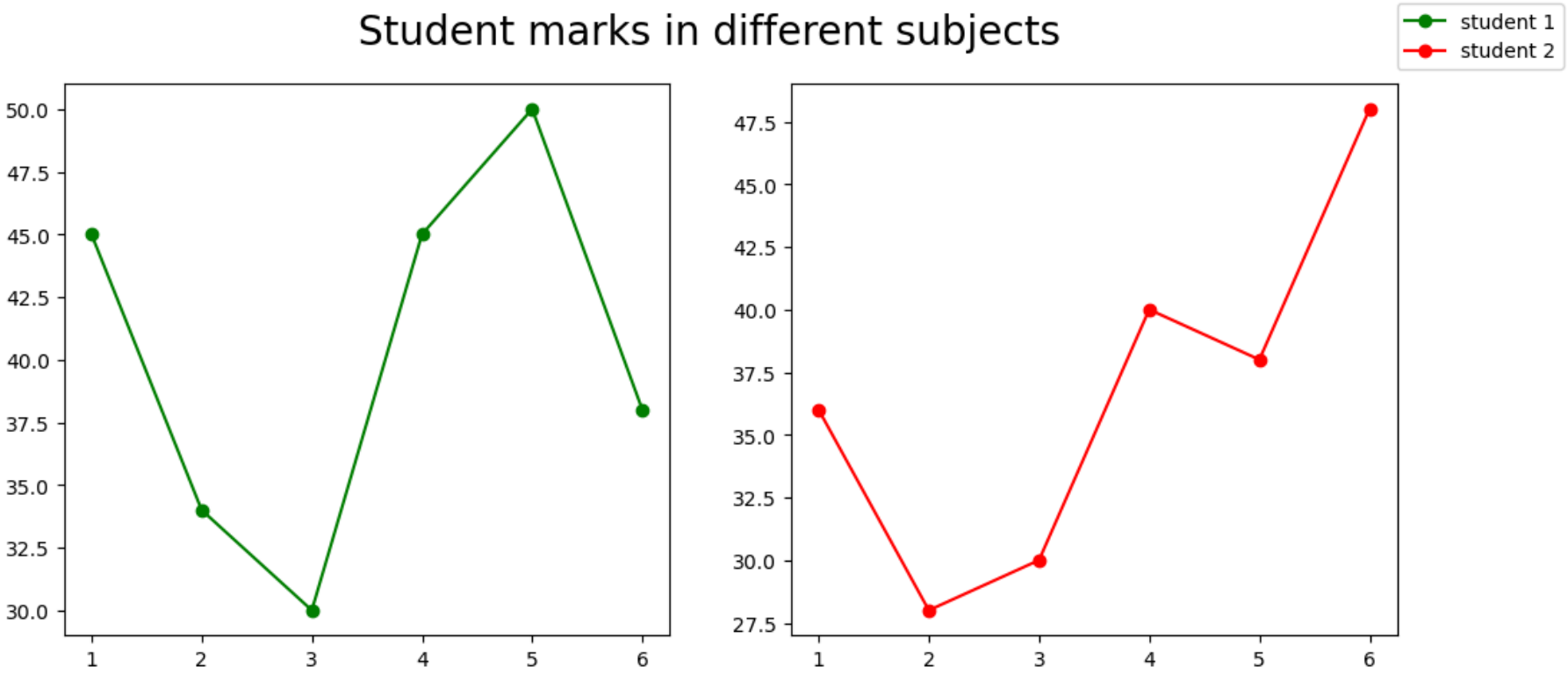

MAIN TITLE FOR GRAPHS
3.0
2.5
2.0
1.5
1.0
1
2
3
4
4
3
2
1
0
1
2
3
4
4
3
2
1
0
1
2
3
4
3.0
2.5
2.0
1.5
1.0
2.0
2.5
3.0
3.5
4.0

In [234]:
```
fig=plt.figure()
a=fig.add_subplot(2,2,1)
b=fig.add_subplot(2,2,2)
c=fig.add_subplot(2,2,3)
d=fig.add_subplot(2,2,4)
plt.plot([1,2,3,4,5])
plt.show()
```

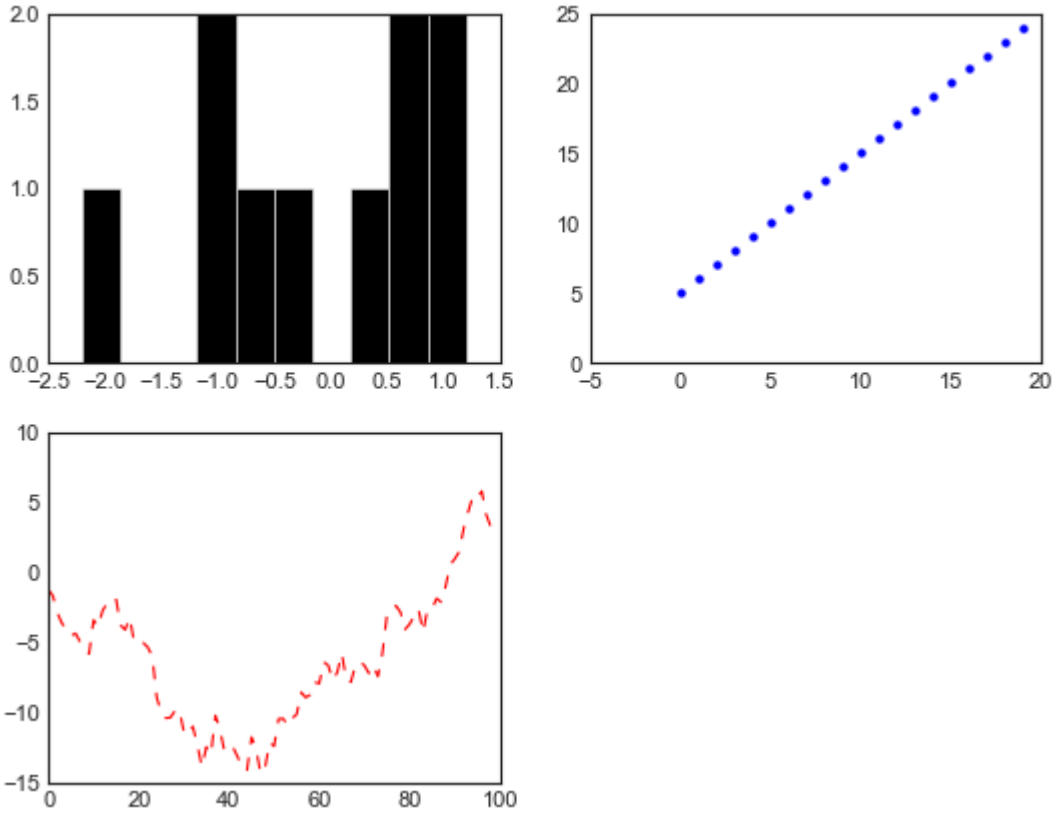


In [11]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
#1 specifies the number of rows in the subplot grid.
#2 specifies the number of columns in the subplot grid.
#figsize=(12, 5) specifies the width and height of the figure in inches.

x1 = [1, 2, 3, 4, 5, 6]
y1 = [45, 34, 30, 45, 50, 38]
y2 = [36, 28, 30, 40, 38, 48]

labels = ["student 1", "student 2"]

fig.suptitle(' Student marks in different subjects ', fontsize=20)

# Creating the sub-plots.
l1 = ax1.plot(x1, y1, 'go-')
l2 = ax2.plot(x1, y2, 'ro-')

fig.legend([l1, l2], labels=labels, loc="upper right")
plt.subplots_adjust(right=0.9)

plt.show()
```

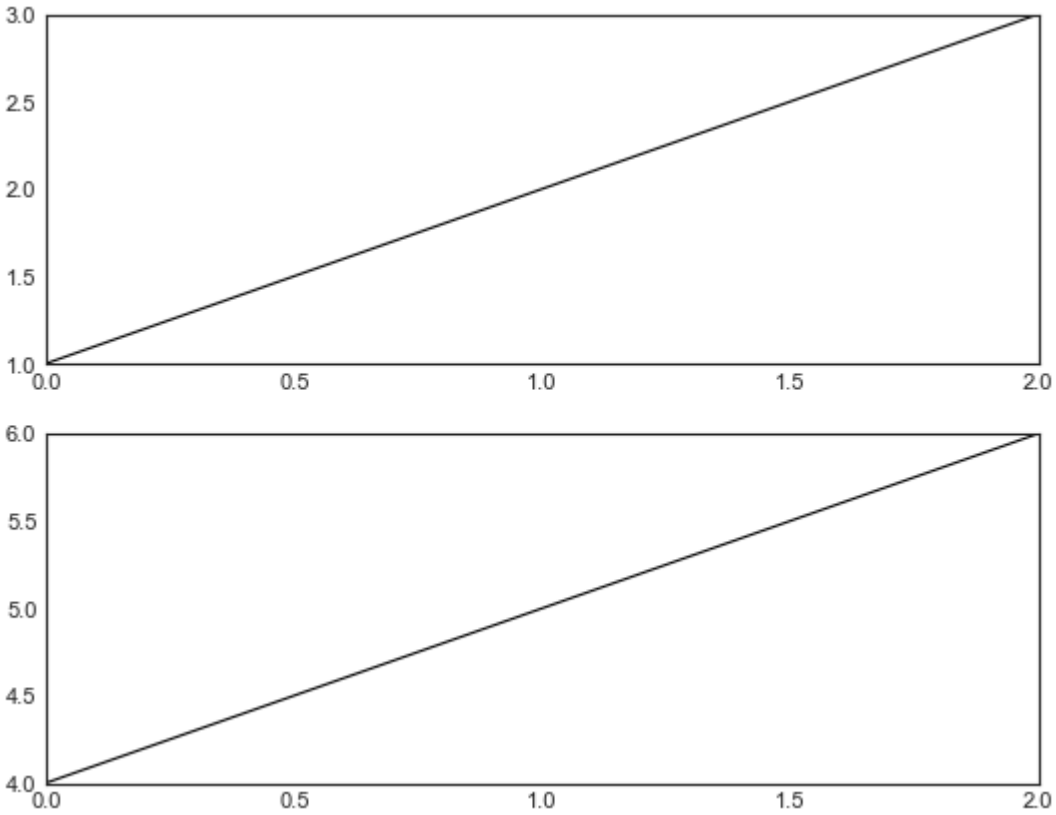
```
C:\Users\tanis\AppData\Local\Temp\ipykernel_59200\2018406077.py:19: UserWarning: You have mixed positional and keywor
d arguments, some input may be discarded.
  fig.legend([l1, l2], labels=labels, loc="upper right")
```

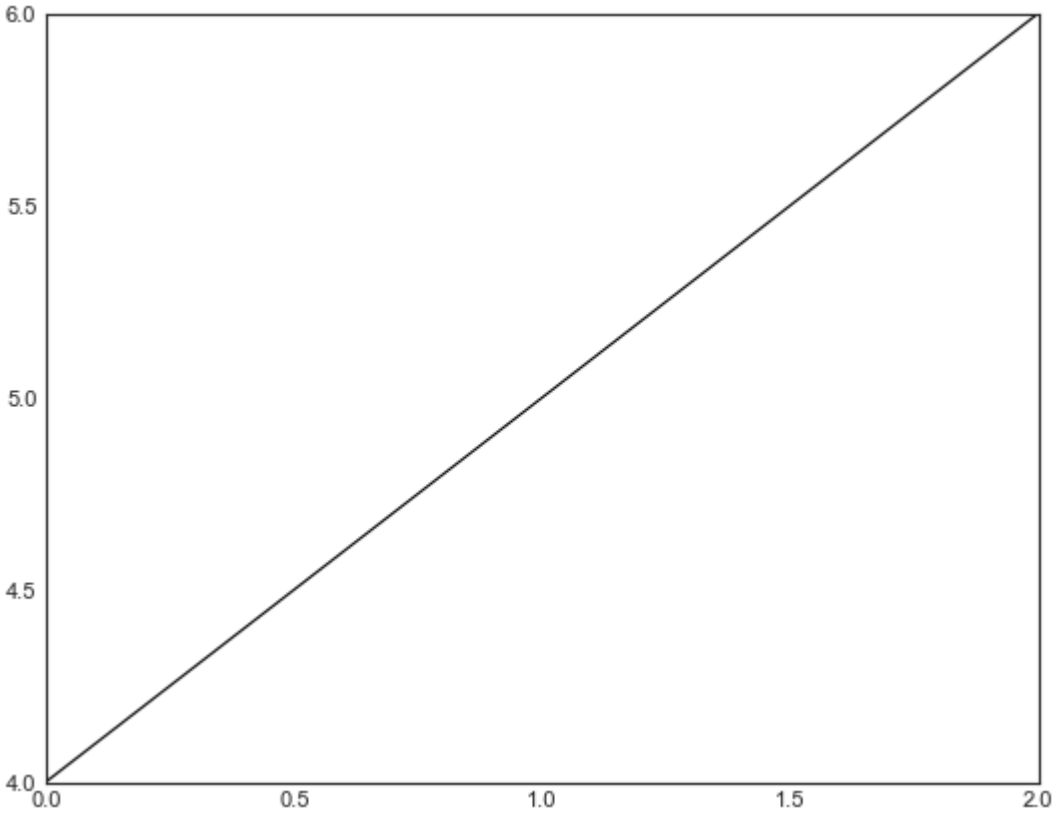

Student marks in different subjects
student 1
student 2
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
1
2
3
4
5
6
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
27.5
1
2
3
4
5
6

In [237]:
```
from numpy.random import randn
fig=plt.figure()
a=fig.add_subplot(2,2,1)
b=fig.add_subplot(2,2,2)
c=fig.add_subplot(2,2,3)
plt.plot(randn(100).cumsum(),'r--')
a.hist(randn(10))
b.scatter(np.arange(20),np.arange(20)+5)
plt.show()
```

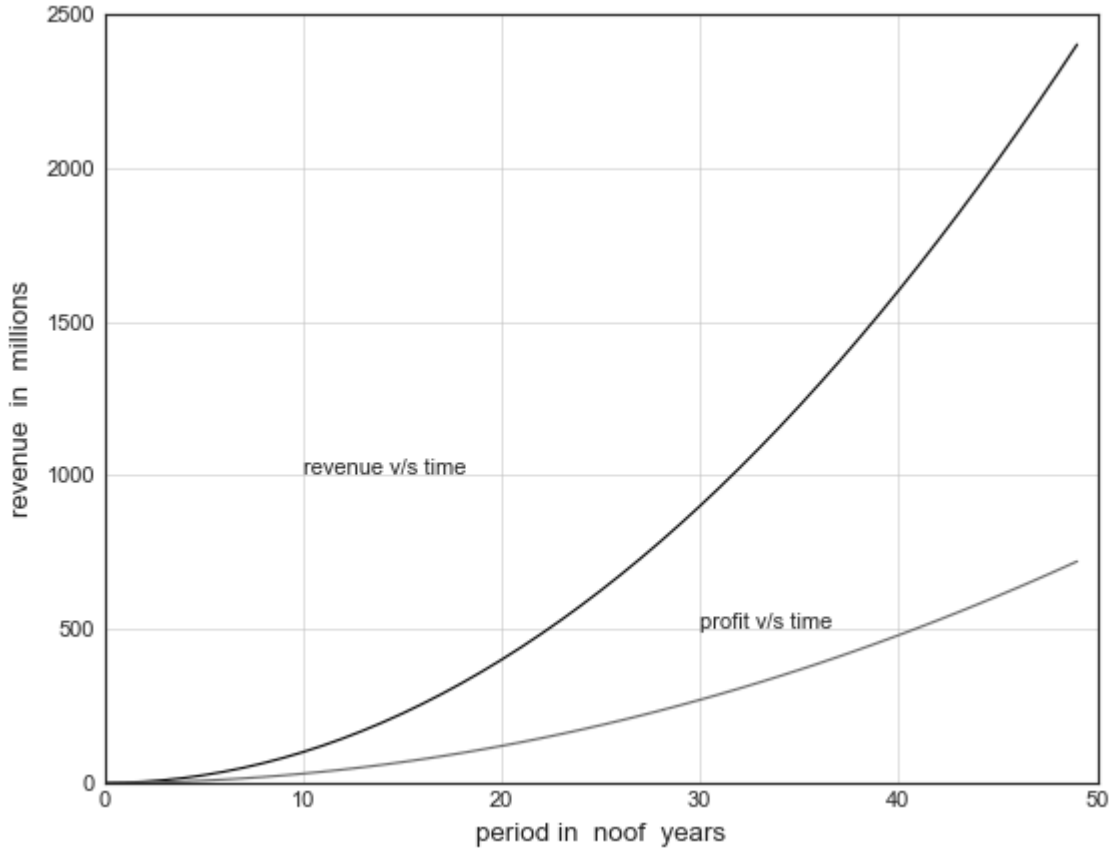


In [238]:
```
plt.figure(1)
plt.subplot(2,1,1)
plt.plot([1,2,3])

plt.subplot(212)
plt.plot([4,5,6])

plt.figure(2)
plt.plot([4,5,6])
plt.show()
```




6.0
5.5
5.0
4.5
4.0
0.0
0.5
1.0
1.5
2.0

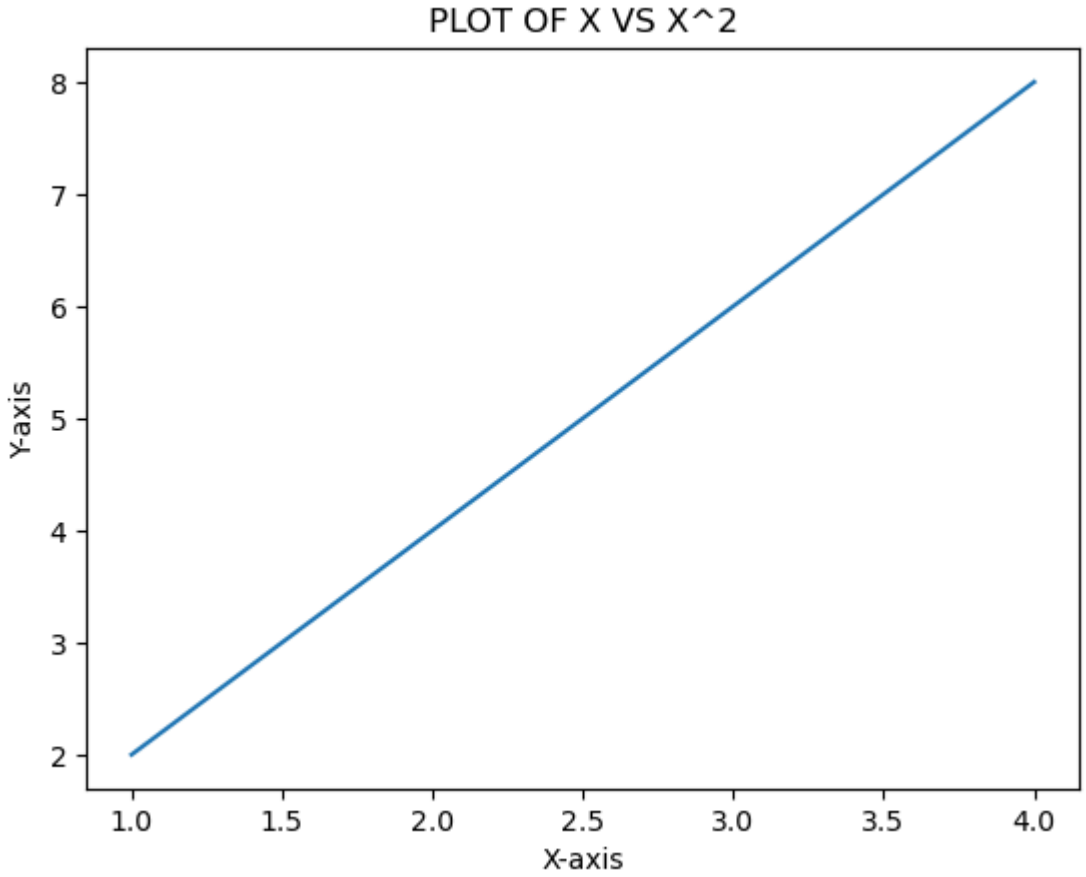
In [239]:
```
plt.plot(np.arange(50),np.arange(50)**2)
plt.plot(np.arange(50),0.3*np.arange(50)**2)
plt.ylabel('revenue  in  millions')
plt.xlabel('period in  noof  years')
plt.text(10,1000,'revenue v/s time')
plt.text(30,500,'profit v/s time')
plt.grid(True)
plt.show()
```



LINE CHART IN MATPLOTLIB

Line charts are used to represent the relation between two data X and Y on a different axis.

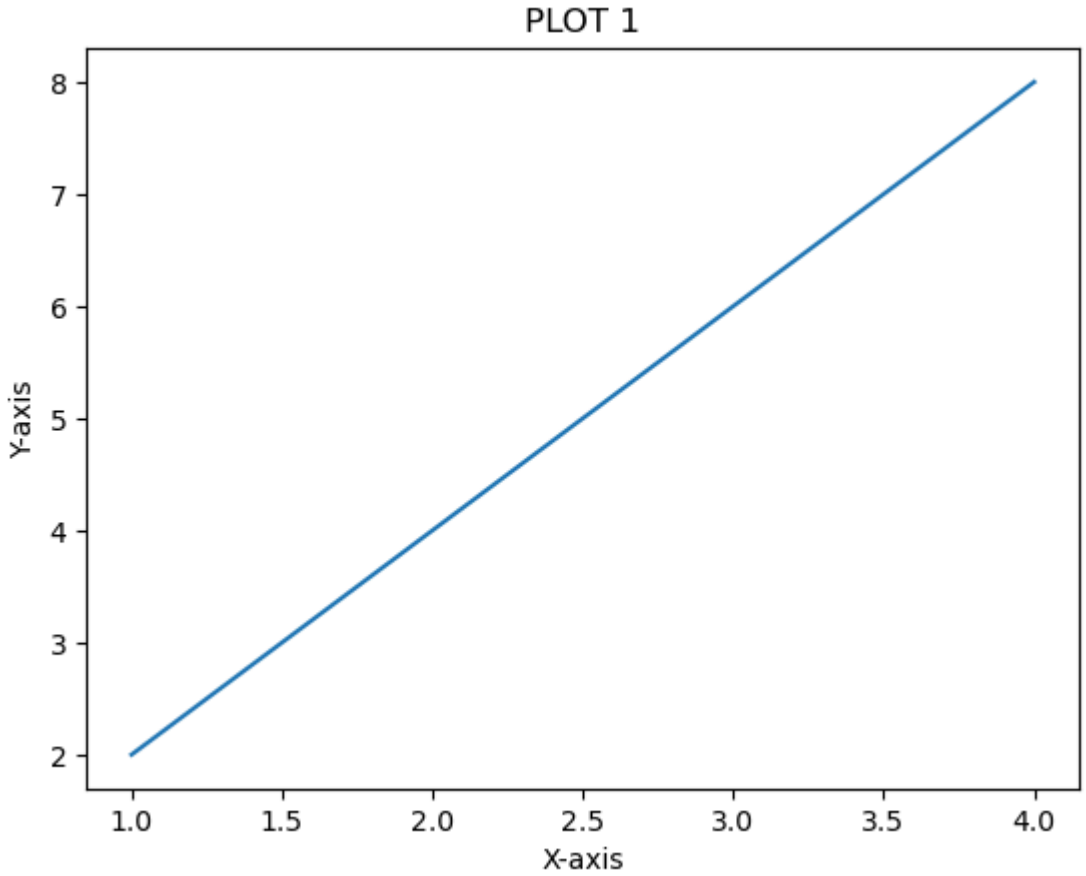
In [13]:
```
x = np.array([1, 2, 3, 4])  # X-axis points
y = x*2  # Y-axis points

plt.plot(x, y)  # Plot the chart
plt.xlabel("X-axis")  # add X-axis label
plt.ylabel("Y-axis")  # add Y-axis label
plt.title("PLOT OF X VS X^2")  # add title
plt.show()  # display
```

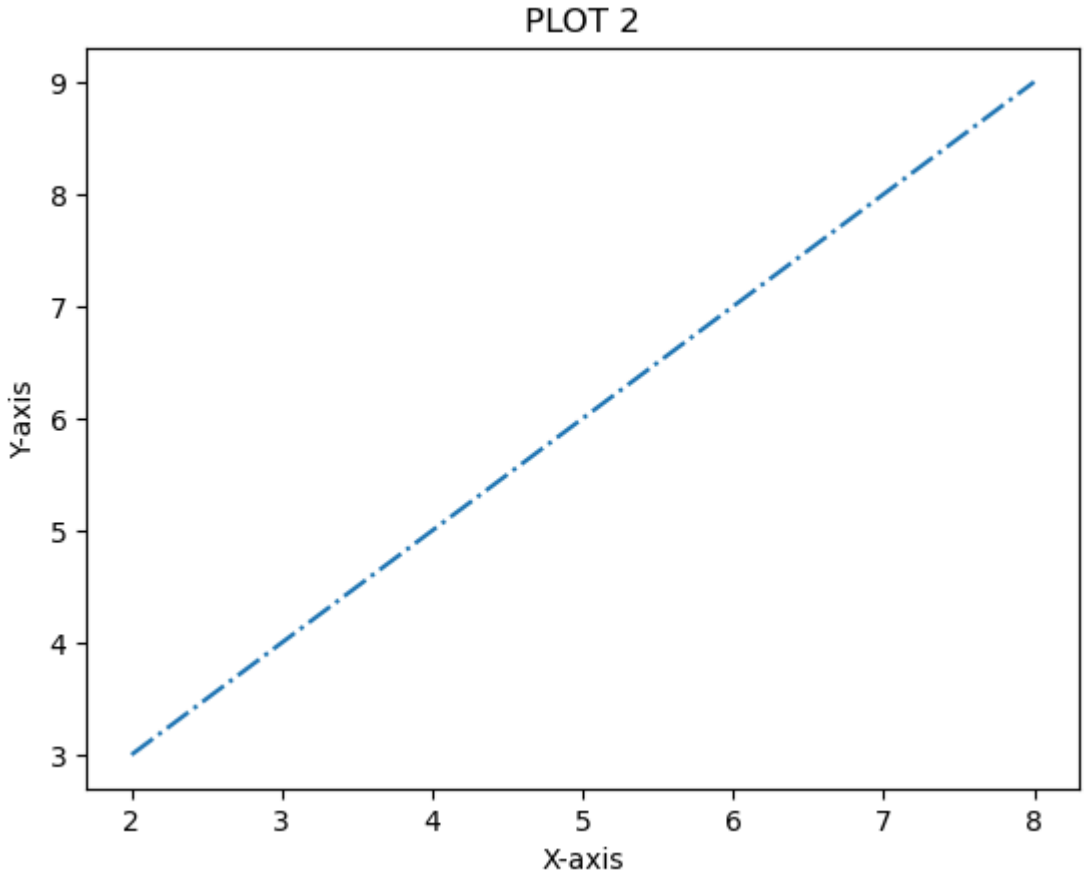


MULTIPLE CHART

Display more than one chart in the same container by using pyplot.figure() function. This will help us in comparing the different charts.

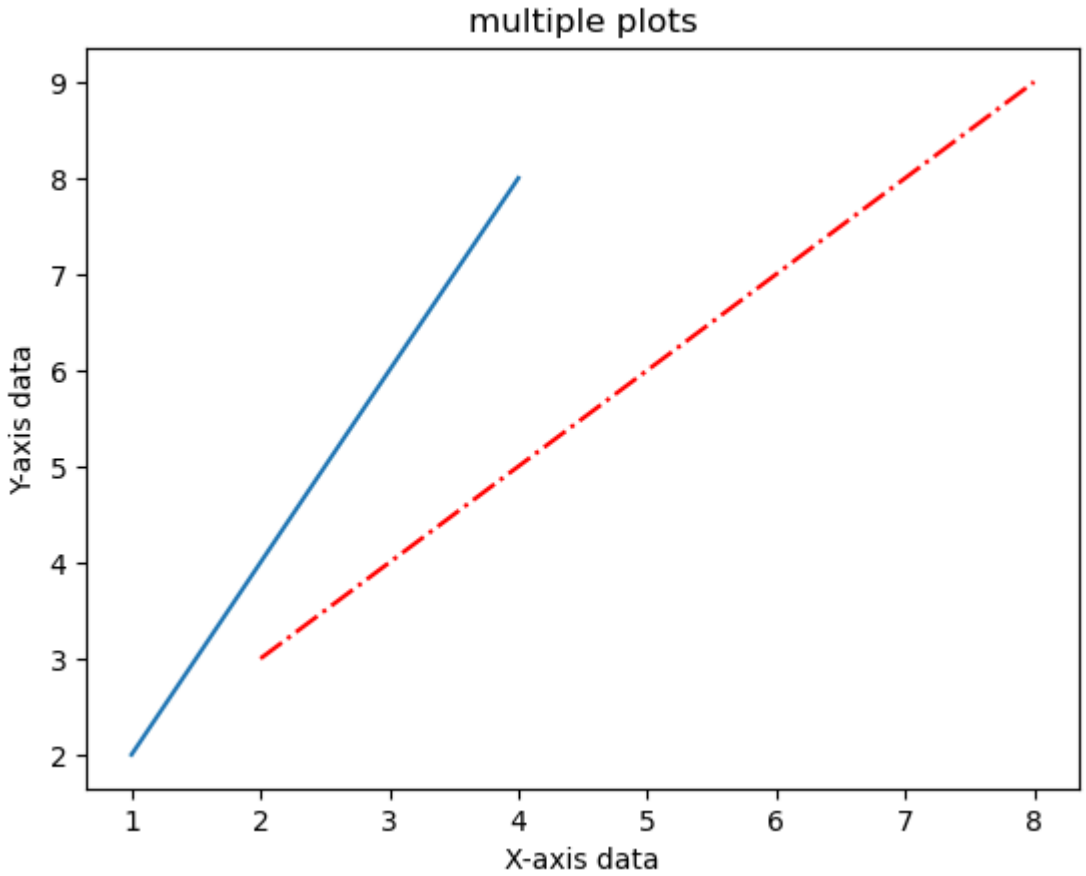
In [15]:
```
x = np.array([1, 2, 3, 4])
y = x*2
plt.plot(x, y)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("PLOT 1")
plt.show()  # show first chart

# The figure() function helps in creating a new figure that can hold a new chart in it.
plt.figure()
x1 = [2, 4, 6, 8]
y1 = [3, 5, 7, 9]
plt.plot(x1, y1, '-.') # Show another chart with '-' dotted line
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("PLOT 2")

plt.show()
```


PLOT 1
Y-axis
X-axis
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0


PLOT 2
Y-axis
X-axis
9
8
7
6
5
4
3
2
3
4
5
6
7
8

MULTIPLE PLOTS ON SAME AXIS

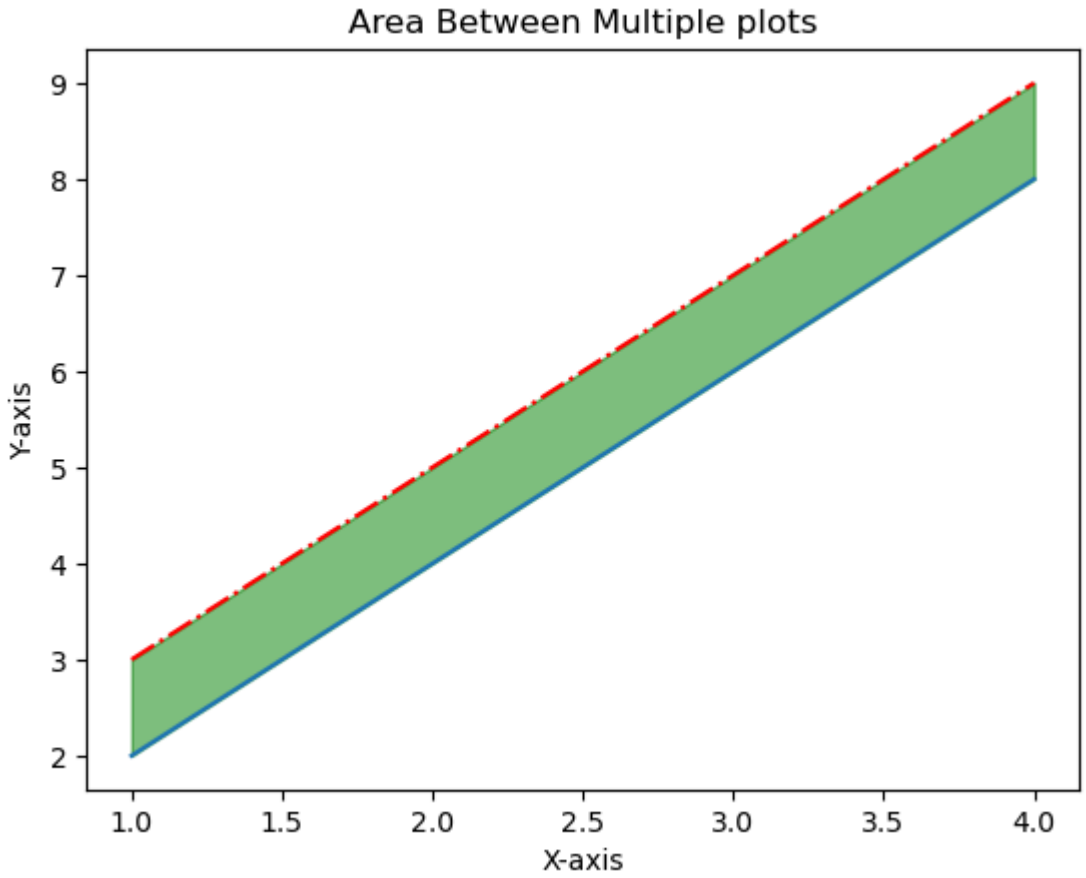
In [16]:
```
x = np.array([1, 2, 3, 4])
y = x*2

# first plot with X and Y data
plt.plot(x, y)

x1 = [2, 4, 6, 8]
y1 = [3, 5, 7, 9]

# second plot with x1 and y1 data
plt.plot(x1, y1, 'r-.')

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title('Multiple plots')
plt.show()
```



Area between two plots

Using the pyplot.fill_between() function we can fill in the region between two line plots in the same graph

In [21]:
```
x = np.array([1, 2, 3, 4])
y = x*2
plt.plot(x, y)

y1 = [3, 5, 7, 9]
plt.plot(x, y1, 'r-.')

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title('Area Between Multiple plots')
plt.fill_between(x, y, y1, color='green', alpha=0.5)
plt.show()
```



LINE PLOT FORMATTING '-' Solid line

-- Dashed line

-. Dashed dot line : Dotted line

. Point marker

o circle marker

, pixel marker

s square marker

p pentagon marker

'*' star marker

h hexagon marker

'+' plus marker

x X marker

D diamond marker

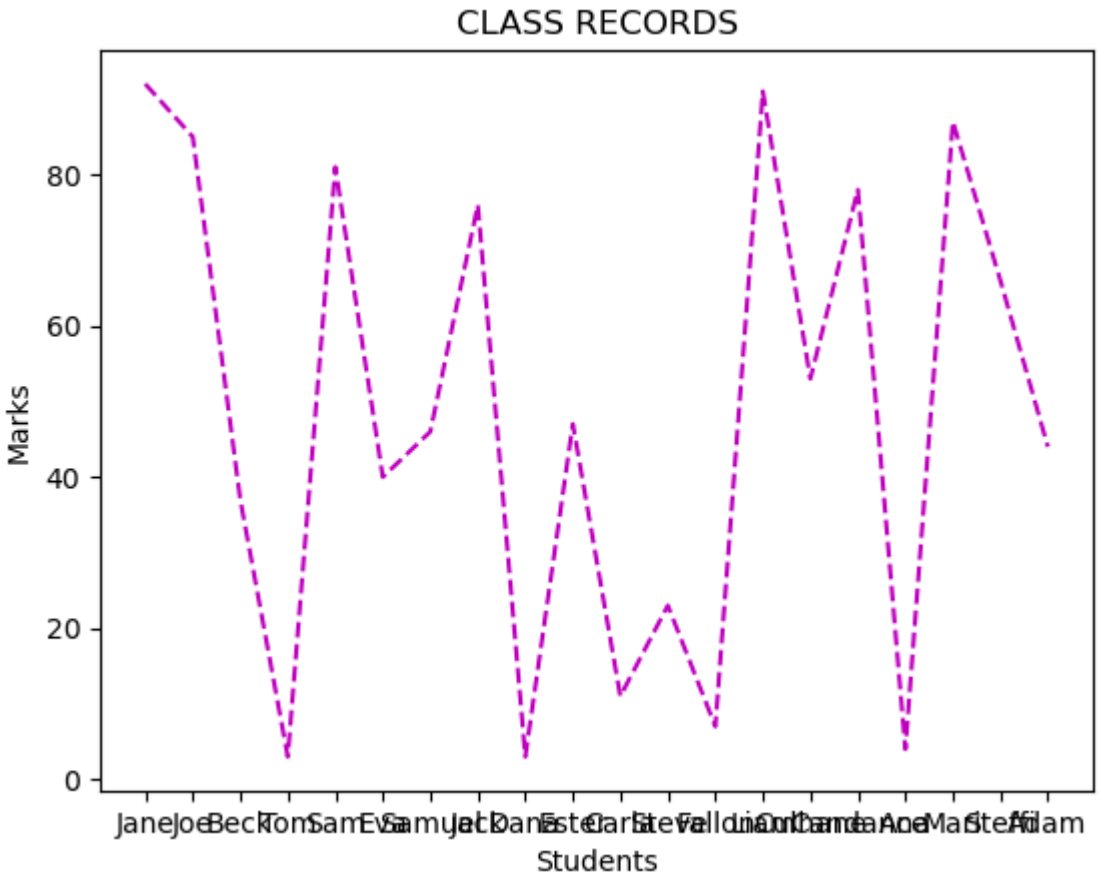
| vline marker

'-' hline marker

Colour Code Description b - blue g - green r - red c - cyan m - magenta y - yellow k - black w - white

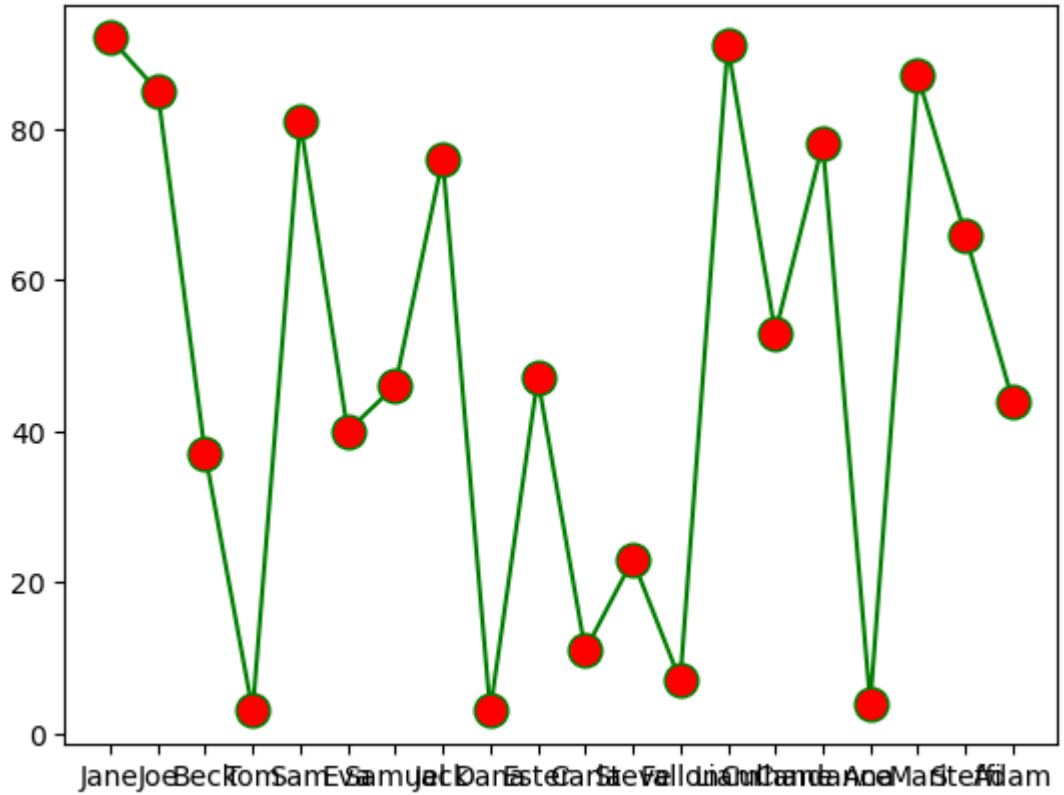
In [23]:
```
import random as random
students = ["Jane","Joe","Beck","Tom","Sam","Eva","Samuel","Jack",
            "Dana","Ester","Carla","Steve","Fallon","Liam","Culhane","Candance",
            "Ana","Mari","Steffi","Adam"]

marks=[]
for i in range(0,len(students)):
     marks.append(random.randint(0, 101))

plt.xlabel("Students")
plt.ylabel("Marks")
plt.title("CLASS RECORDS")
plt.plot(students,marks,'m--')
#m: magenta, --: dashed line
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x206649902b0>]
```


CLASS RECORDS
Marks
0
20
40
60
80
Students

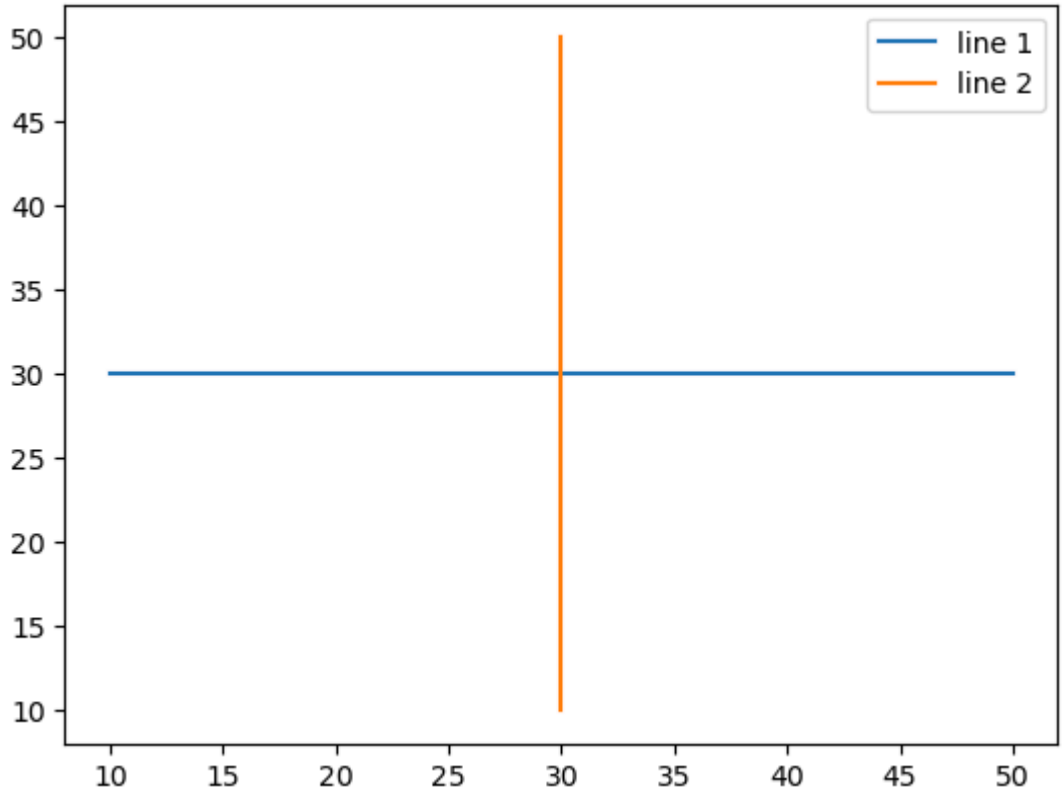
In [24]:
```
plt.plot(students, marks, color = 'green',linestyle = 'solid',
         marker = 'o', markerfacecolor = 'red', markersize = 12)
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x206631747c0>]
```

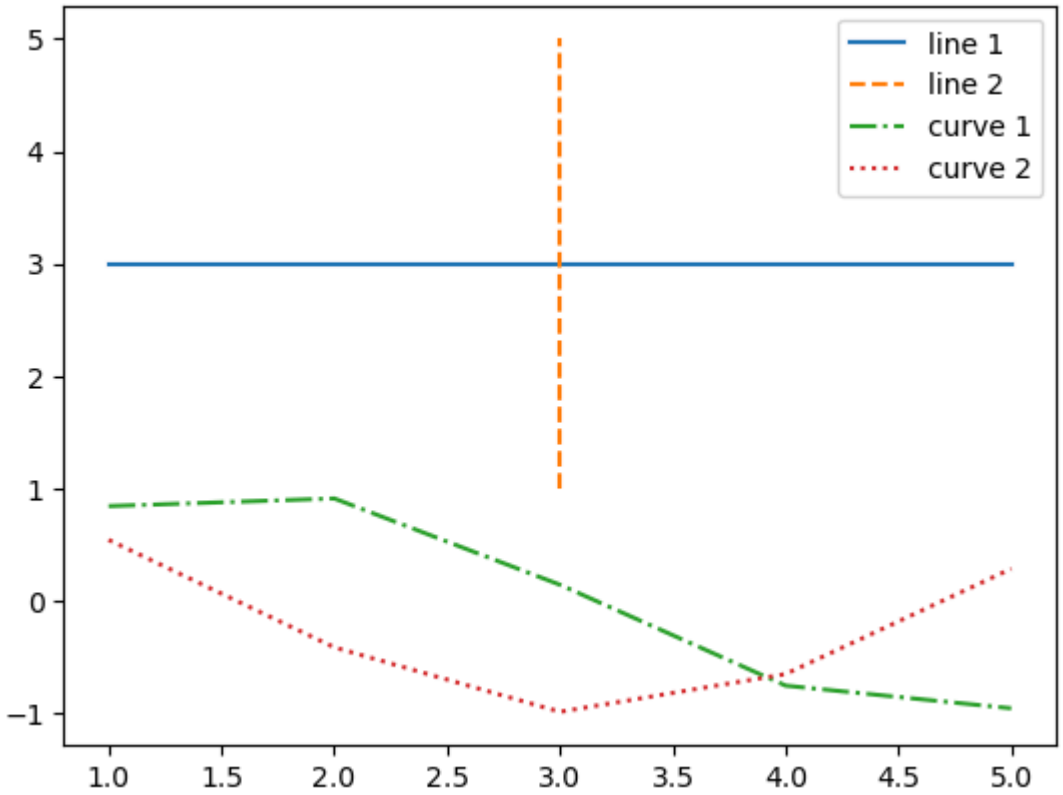


PLOTTING MULTIPLE LINES

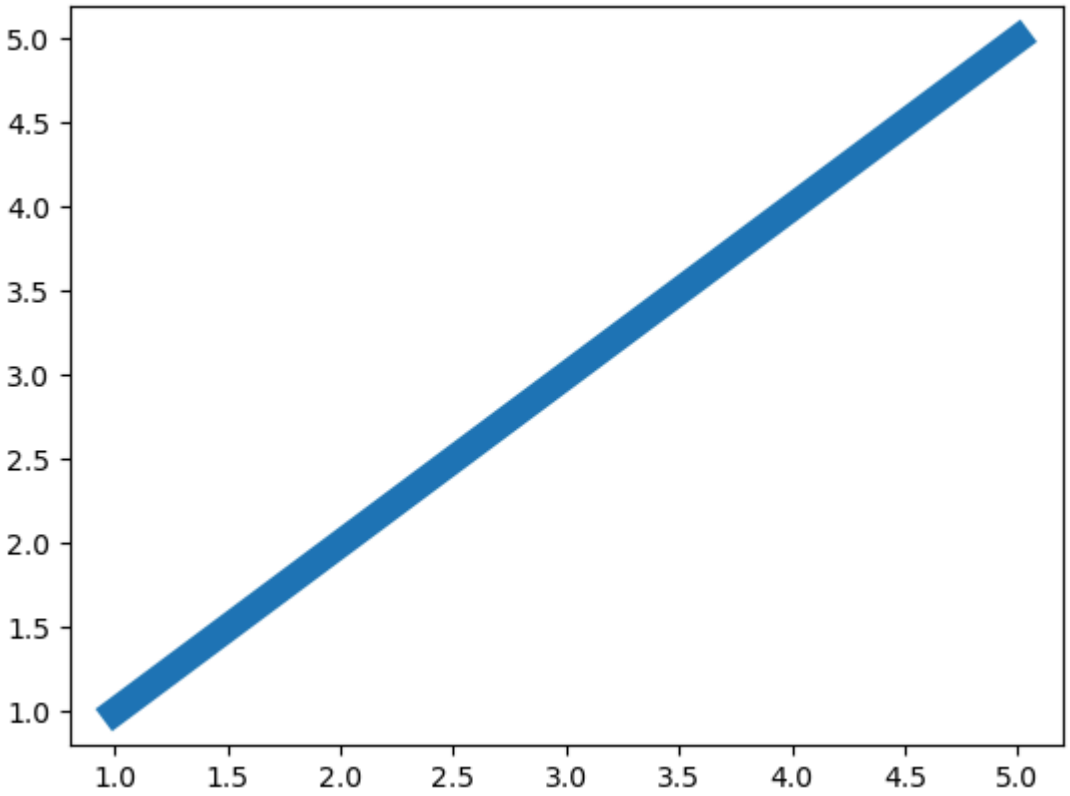
In [25]:
```
x = [10,20,30,40,50]
y = [30,30,30,30,30]

# plot lines
plt.plot(x, y, label = "line 1")
plt.plot(y, x, label = "line 2")
plt.legend()
plt.show()
```



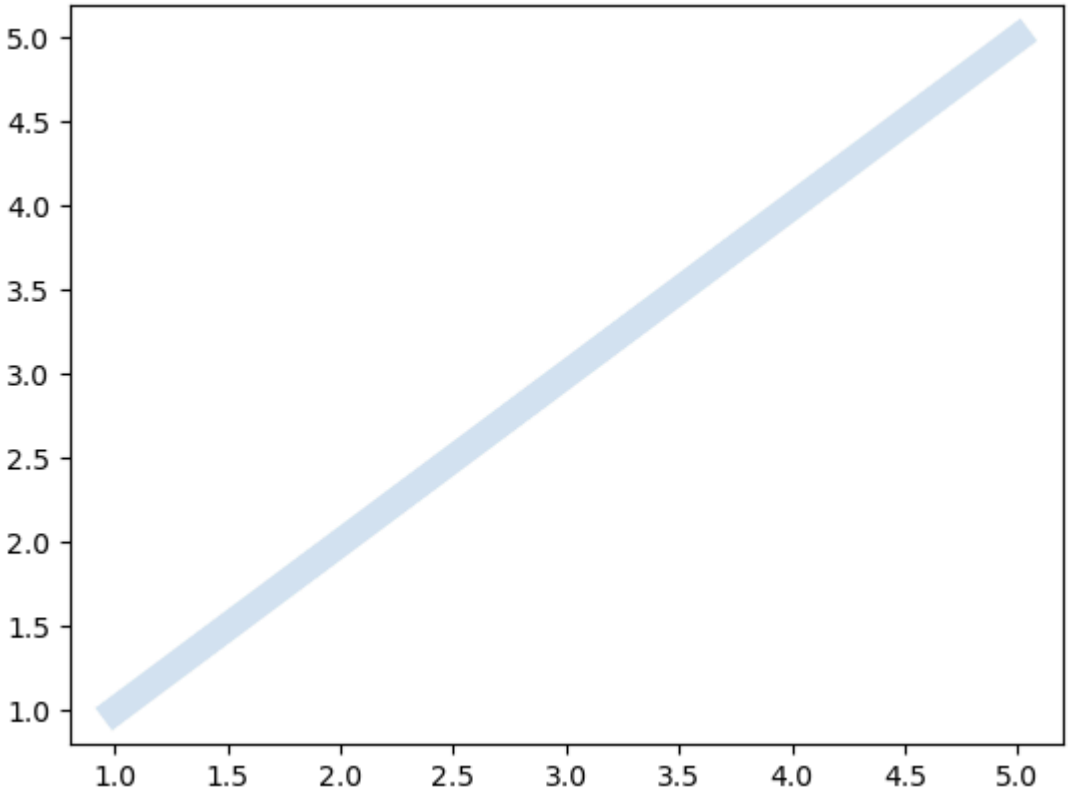
In [27]:
```
x = [1,2,3,4,5]
y = [3,3,3,3,3]

# plot lines
plt.plot(x, y, label = "line 1", linestyle="-")
plt.plot(y, x, label = "line 2", linestyle="--")
plt.plot(x, np.sin(x), label = "curve 1", linestyle="-.")
plt.plot(x, np.cos(x), label = "curve 2", linestyle=":")
plt.legend()
plt.show()
```



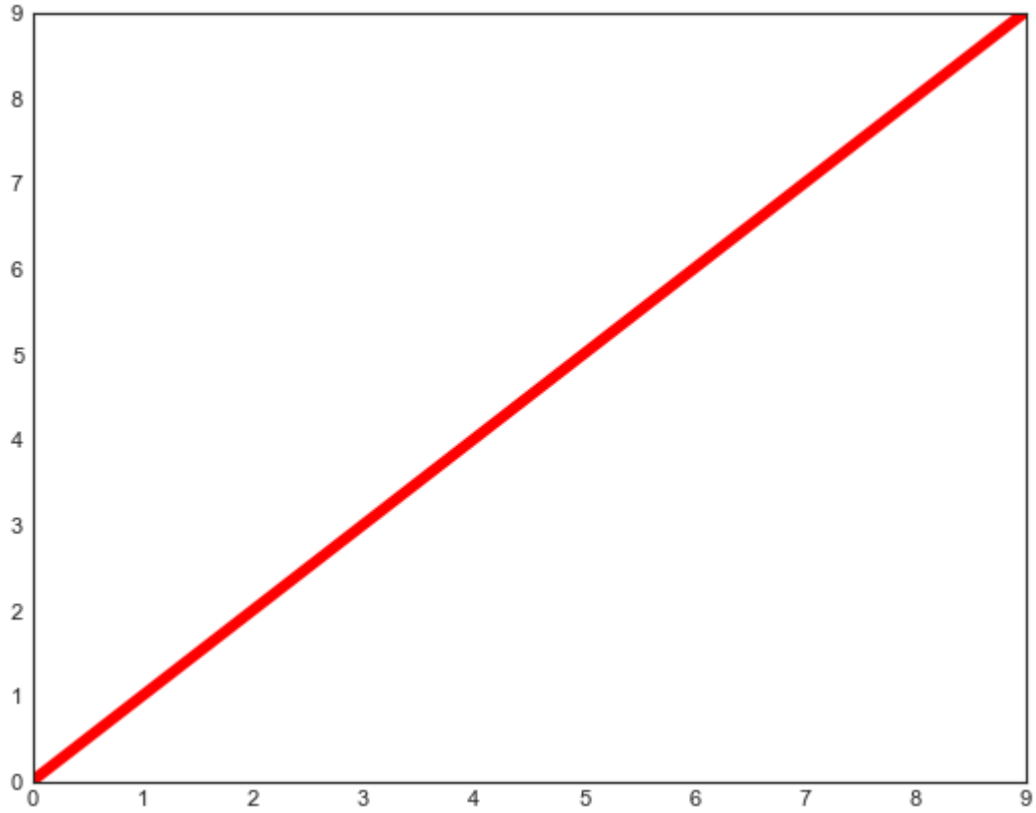
In [28]:
```
#Changing opacity
x = [1, 2, 3, 4, 5]
y = x

# plot the graph
plt.plot(x, y, linewidth=10, alpha=1)
plt.show()
```



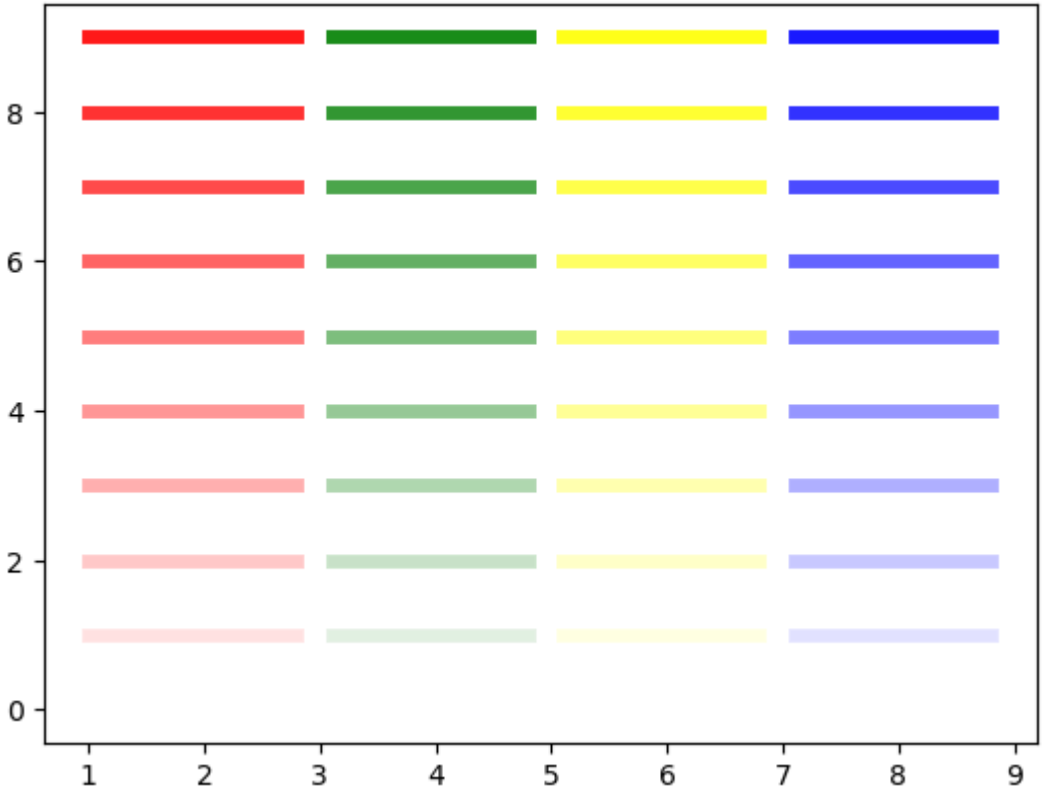
In [29]:
```
plt.plot(x, y, linewidth=10, alpha=0.2)
plt.show()
```



In [233]:
```
line =plt.plot(np.arange(10),np.arange(10))
plt.setp(line,color='r',linewidth=5)
plt.show()
```



In [30]:
```
for i in range(10):
    plt.plot([1, 2.8], [i]*2, linewidth=5, color='red', alpha=0.1*i)
    plt.plot([3.1, 4.8], [i]*2, linewidth=5, color='green', alpha=0.1*i)
    plt.plot([5.1, 6.8], [i]*2, linewidth=5, color='yellow', alpha=0.1*i)
    plt.plot([7.1, 8.8], [i]*2, linewidth=5, color='blue', alpha=0.1*i)

plt.show()
```



xlabel() function: function is used to set labels for x-axis. Syntax: plt.xlabel(xlabel, fontdict=None, labelpad=None, **kwargs)

xlabel: accepts string type value and is used to label the X axis.

fontdict: used to override default font properties of the label. Its default value is None and is optional.

labelpad: default value is None. It is used to specify the spacing of the label from the axes. This is optional.

kwargs: used to specify other properties that can be used to modify the appearance of the label.

ylabel()- this function is used to set labels for y-axis

Syntax: plt.ylabel(ylabel, fontdict=None, labelpad=None, kwargs)

Parameter: ylabel: accepts string type value and is used to label the Y axis.

fontdict: used to override default font properties of the label. Its default value is None and is optional.

labelpad: default value is None. It is used to specify the spacing of the label from the axes. This is optional.

kwargs: used to specify other properties that can be used to modify the appearance of the label.

plot()- It is used to make a 2D hexagonal binning plot of points x, y.

Syntax: plt.plot(x,y, data=None, kwargs)

Parameter x,y : used to specify the data to be plotted along the x and y axis. data: default value is None. It is an object with labelled data and can be passed instead of the x,y values. If data object is passed then the xand y label should be specified. kwargs: used to specify line properties like linewidth, color, antialiasing, marker, markercolor etc.

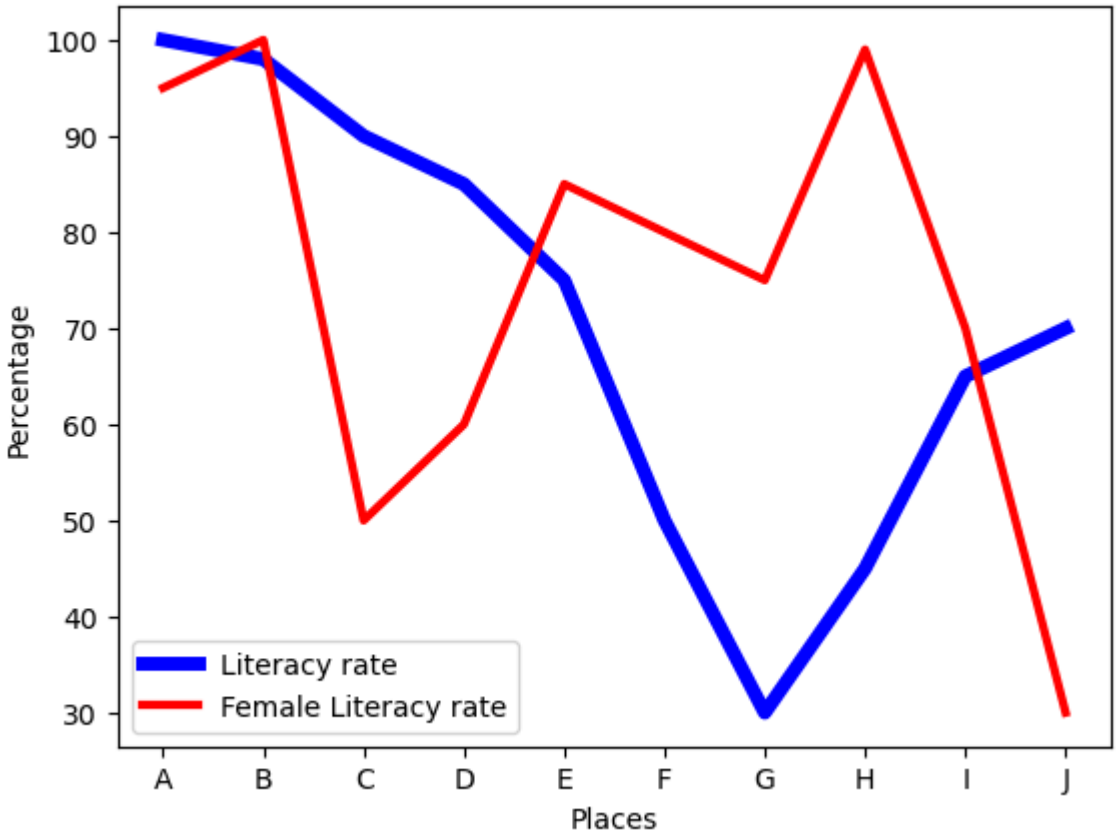
legend()- A legend is an area describing the elements of the graph. There's a function called legend() which is used to Place a legend on the axes.

Syntax: plt.legend(options)

Parameter options: used to specify the properties of the legend, its size, its location, edgecolor, facecolor etc

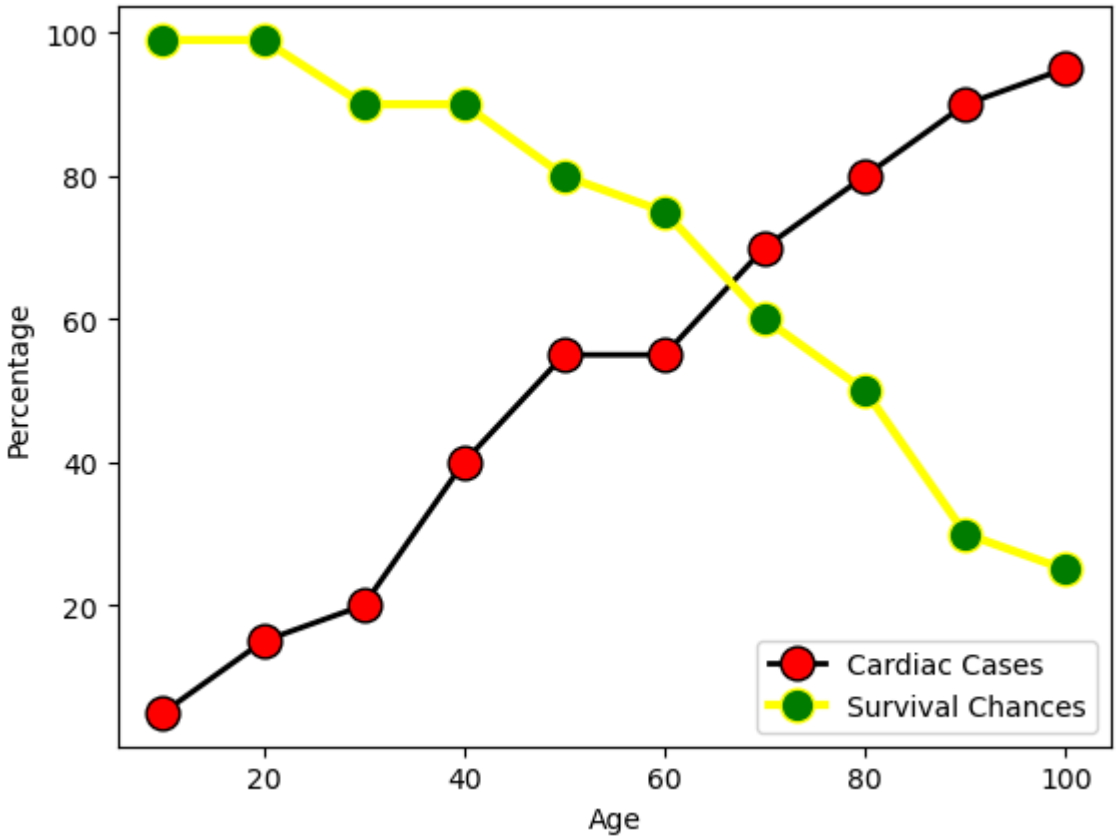
In [33]:
```
places = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J"]
literacy_rate = [100, 98, 90, 85, 75, 50, 30, 45, 65, 70]
female_literacy = [95, 100, 50, 60, 85, 80, 75, 99, 70, 30]

plt.xlabel("Places")
plt.ylabel("Percentage")

plt.plot(places, literacy_rate, color='blue',
         linewidth=5, label="Literacy rate")

plt.plot(places, female_literacy, color='red',
         linewidth=3, label="Female Literacy rate")

plt.legend(loc='lower left', ncol=1)
```

Out[33]:
```
<matplotlib.legend.Legend at 0x2066280ce20>
```


100
90
80
70
60
50
40
30
Percentage
Literacy rate
Female Literacy rate
A
B
C
D
E
F
G
H
I
J
Places

In [34]:
```
age = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
cardiac_cases = [5, 15, 20, 40, 55, 55, 70, 80, 90, 95]
survival_chances = [99, 99, 90, 90, 80, 75, 60, 50, 30, 25]

plt.xlabel("Age")
plt.ylabel("Percentage")

plt.plot(age, cardiac_cases, color='black', linewidth=2,
         label="Cardiac Cases", marker='o', markerfacecolor='red', markersize=12)

plt.plot(age, survival_chances, color='yellow', linewidth=3,
         label="Survival Chances", marker='o', markerfacecolor='green', markersize=12)

plt.legend(loc='lower right', ncol=1)
```

Out[34]:
```
<matplotlib.legend.Legend at 0x206635aaf40>
```

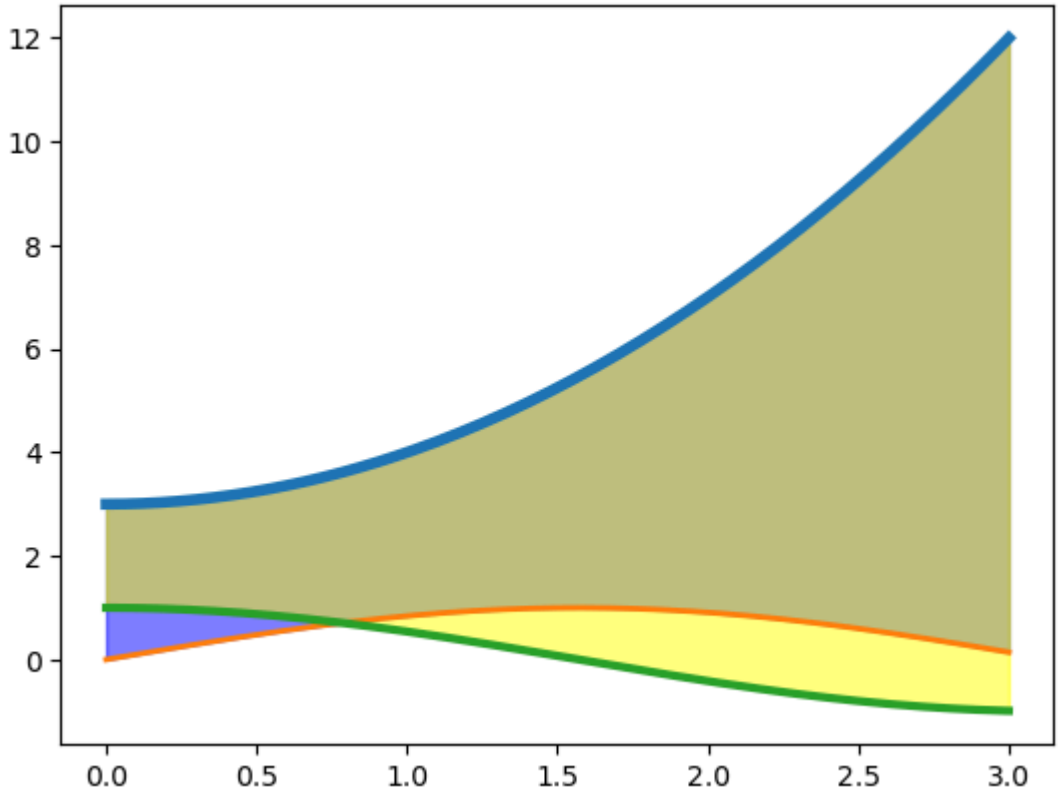


Colouring Between the lines

fill_between() function: fill the color between any multiple lines or any two horizontal curves on a 2D plane.

Syntax: matplotlib.pyplot.fill_between(x, y1, y2=0, where=None, step=None, interpolate=False, *, data=None, kwargs)

In [41]:
```
X = np.linspace(0, 3, 200)
Y1 = X**2 + 3
Y2 = np.sin(X)
Y3 = np.cos(X)

plt.plot(X, Y1, linewidth=4)
plt.plot(X, Y2, lw=2)
plt.plot(X, Y3, lw=3)

plt.fill_between(X, Y1, Y2, color='blue', alpha=.5)
plt.fill_between(X, Y1, Y3, color='yellow', alpha=.5)

plt.show()
```

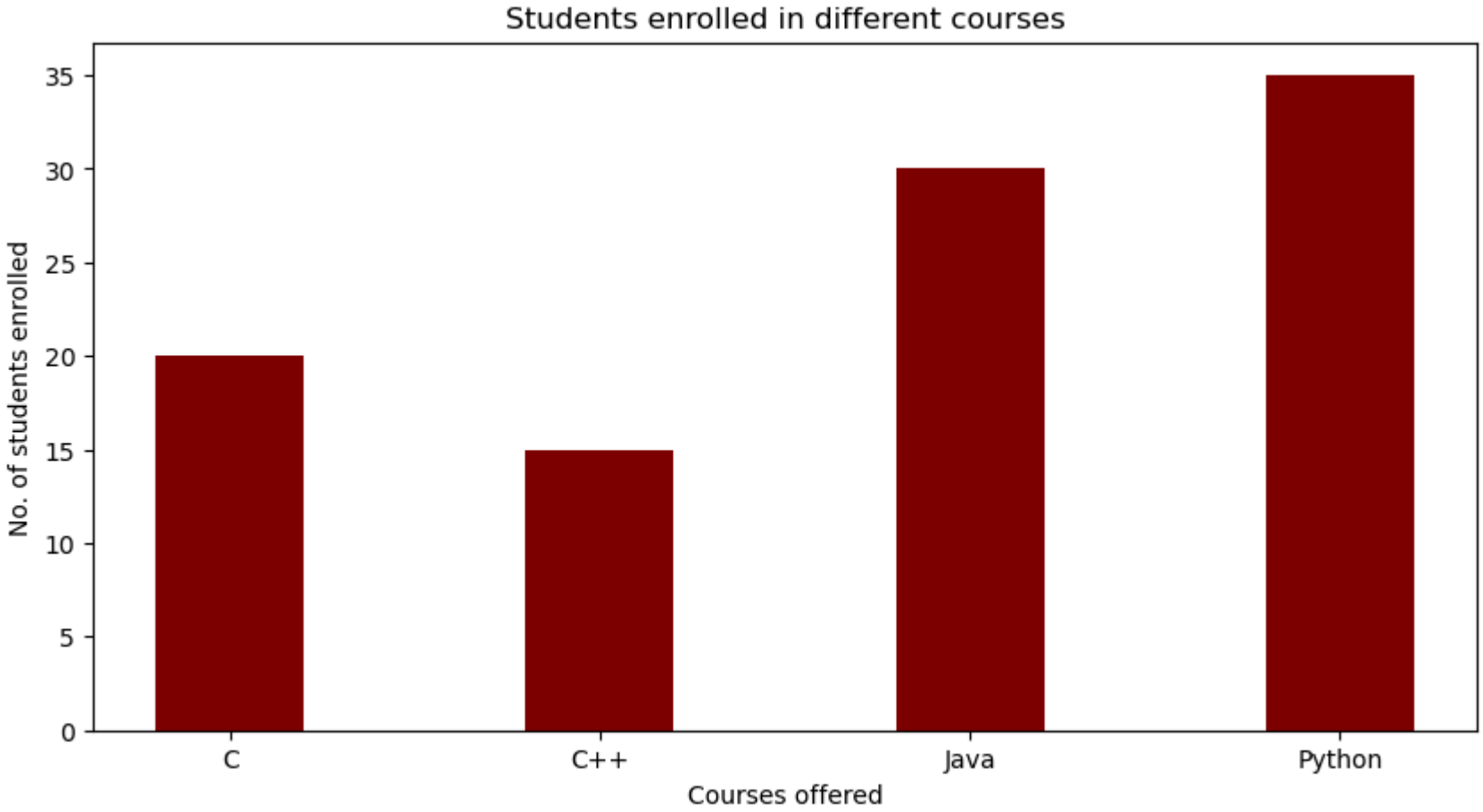


BAR CHART IN MATPLOTLIB

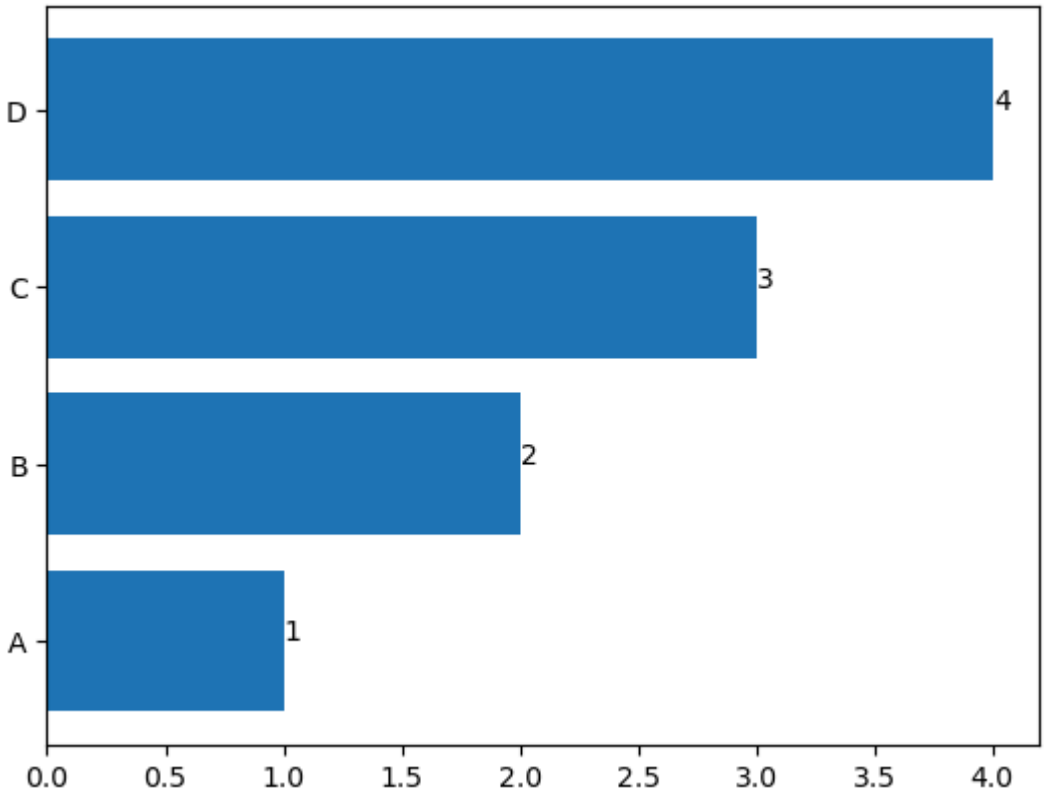
A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories.

matplotlib API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API. **SYNTAX: plt.bar(x, height, width, bottom, align)**

In [43]:
```
data = {'C':20, 'C++':15, 'Java':30,'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',width = 0.4)

plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
plt.show()
```

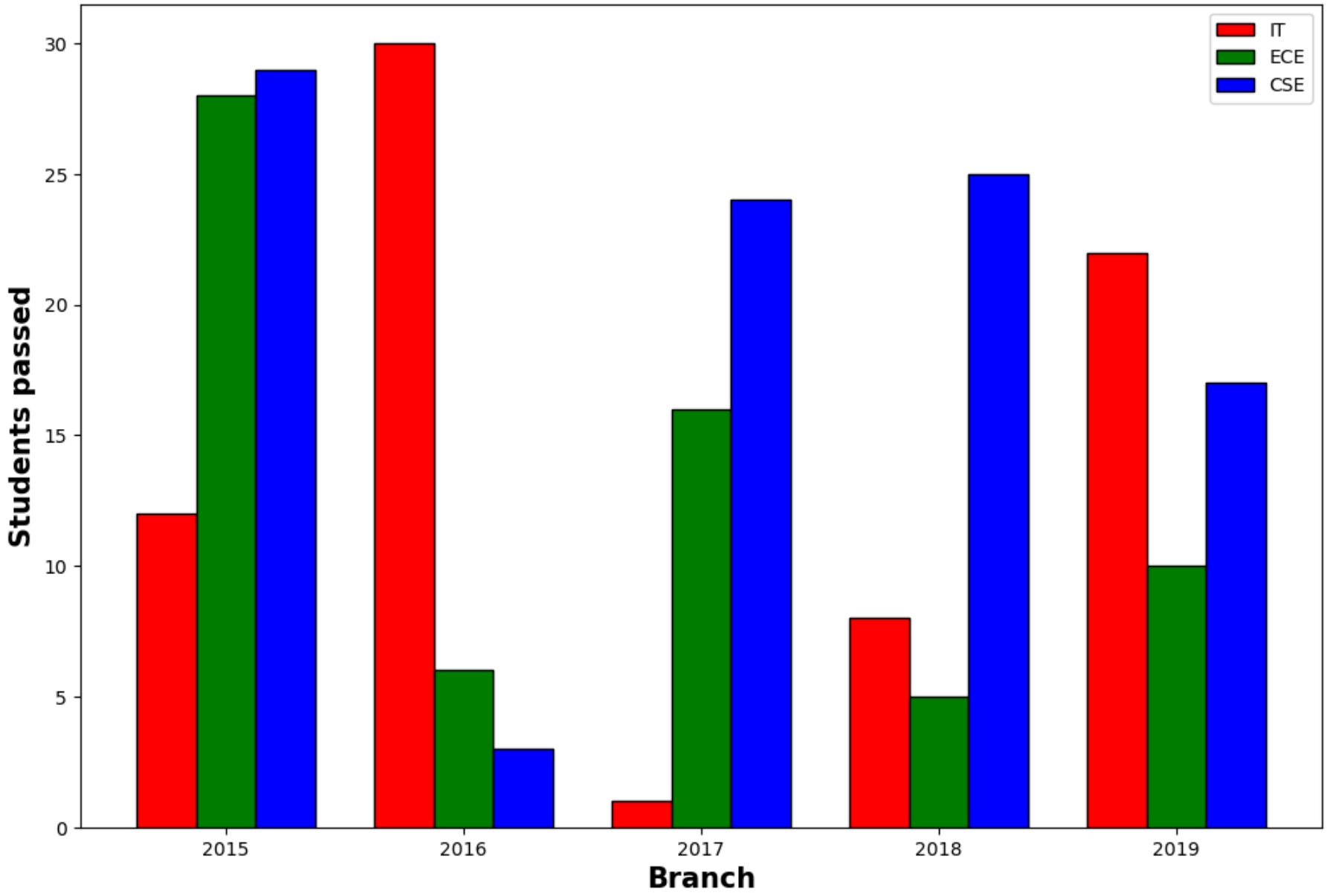

Students enrolled in different courses
No. of students enrolled
0
5
10
15
20
25
30
35
C
C++
Java
Python
Courses offered

In [58]:
```
x = ["A", "B", "C", "D"]
y = [1, 2, 3, 4]
plt.barh(x, y) #horizontal bar plot

for index, value in enumerate(y):
    plt.text(value, index,str(value))

plt.show()
```



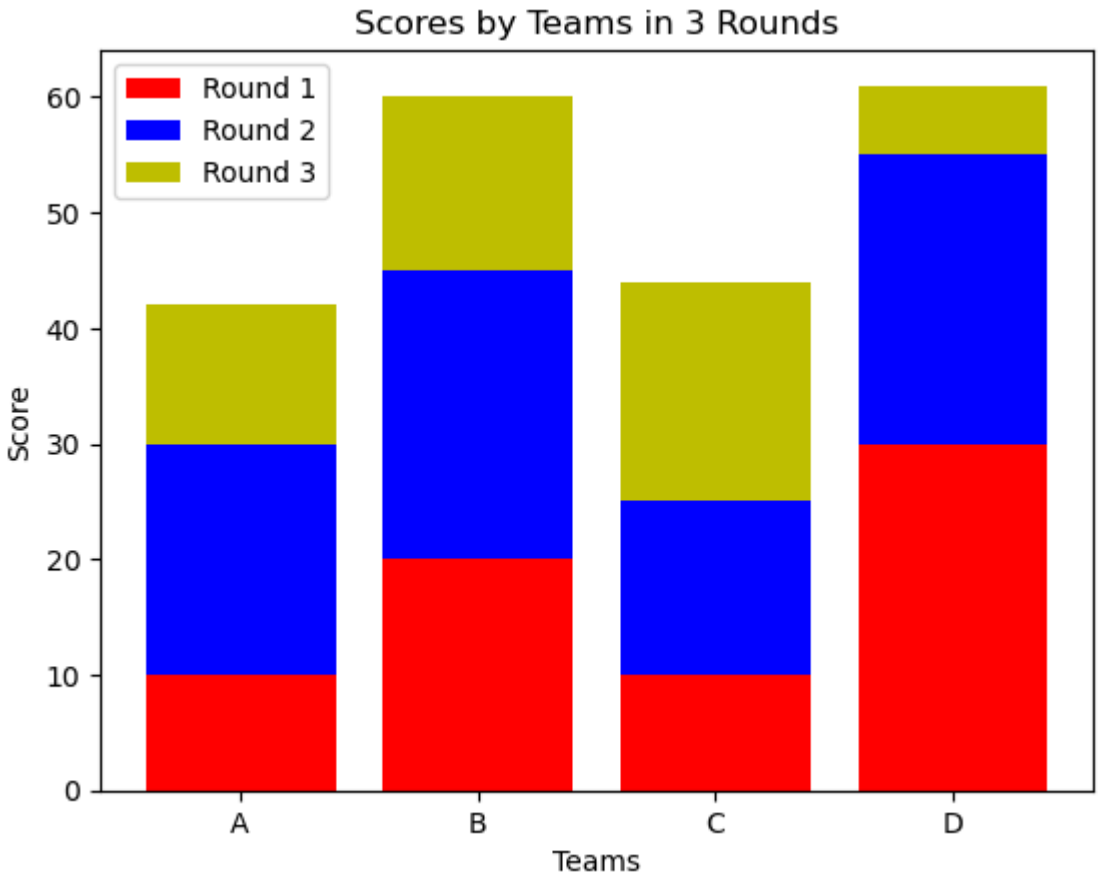
MULTIPLE BAR PLOT

In [45]:
```
barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

# set height of bar
IT = [12, 30, 1, 8, 22]
ECE = [28, 6, 16, 5, 10]
CSE = [29, 3, 24, 25, 17]

# Set position of bar on X axis
br1 = np.arange(len(IT))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

# Make the plot
plt.bar(br1, IT, color ='r', width = barWidth, edgecolor ='black', label ='IT')
plt.bar(br2, ECE, color ='g', width = barWidth,edgecolor ='black', label ='ECE')
plt.bar(br3, CSE, color ='b', width = barWidth,edgecolor ='black', label ='CSE')

# Adding Xticks
plt.xlabel('Branch', fontweight ='bold', fontsize = 15)
plt.ylabel('Students passed', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(IT))],
        ['2015', '2016', '2017', '2018', '2019'])

plt.legend()
plt.show()
```


IT
ECE
CSE
Students passed
0
5
10
15
20
25
30
2015
2016
2017
2018
2019
Branch

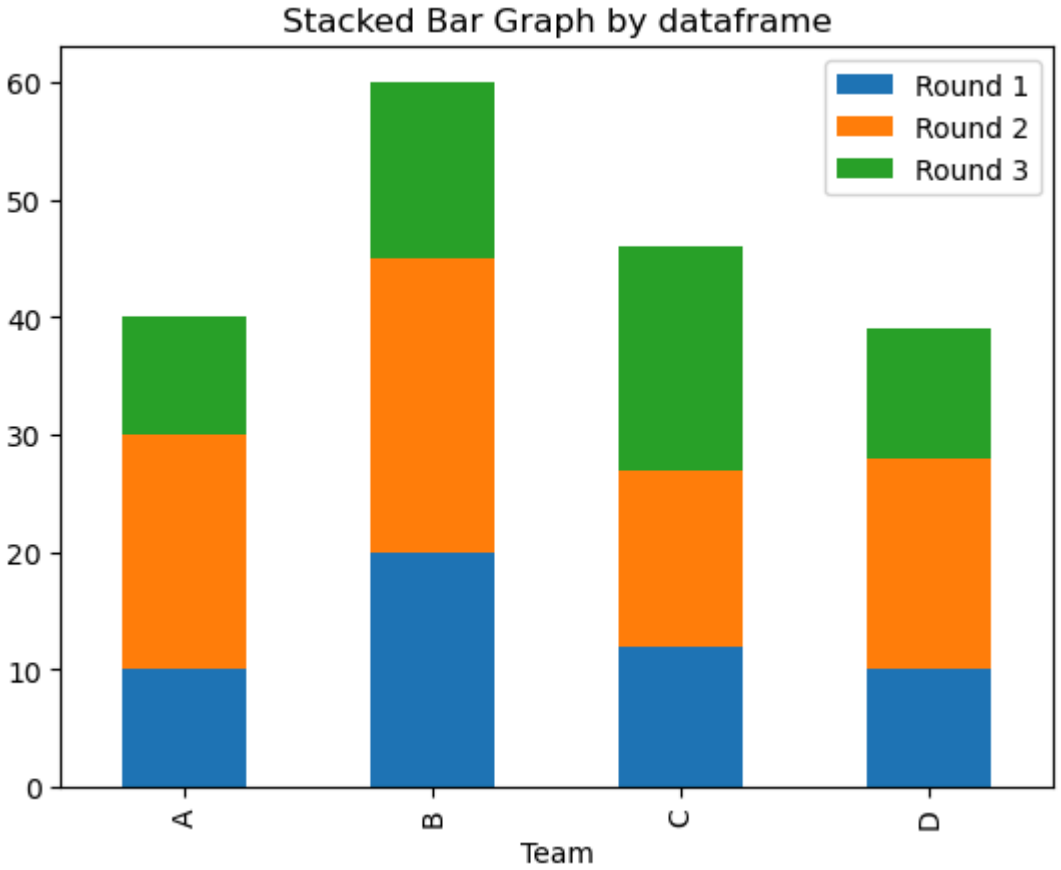
STACKED BAR PLOT

In [53]:
```
x = ['A', 'B', 'C', 'D']
y1 = np.array([10, 20, 10, 30])
y2 = np.array([20, 25, 15, 25])
y3 = np.array([12, 15, 19, 6])

# plot bars in stack manner
plt.bar(x, y1, color='r')
plt.bar(x, y2, bottom=y1, color='b')
plt.bar(x, y3, bottom=y1+y2, color='y')

plt.xlabel("Teams")
plt.ylabel("Score")
plt.legend(["Round 1", "Round 2", "Round 3"])
plt.title("Scores by Teams in 3 Rounds")
plt.show()
```

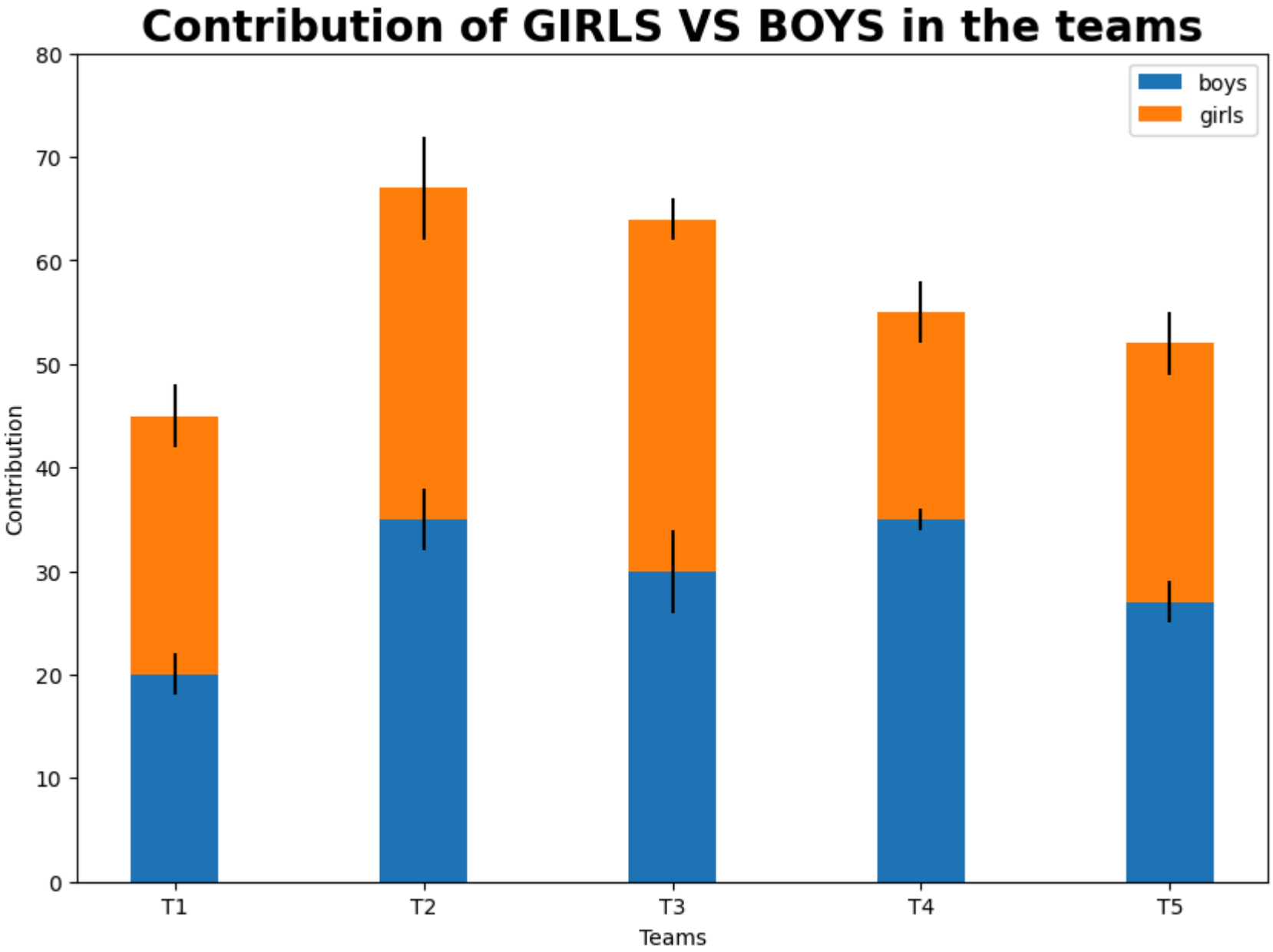

Scores by Teams in 3 Rounds
Round 1
Round 2
Round 3
Score
0
10
20
30
40
50
60
A
B
C
D
Teams

In [57]:
```
df = pd.DataFrame([['A', 10, 20, 10], ['B', 20, 25, 15],
                   ['C', 12, 15, 19],['D', 10, 18, 11]],
                  columns=['Team', 'Round 1', 'Round 2', 'Round 3'])
print(df)
df.plot(x='Team', kind='bar', stacked=True,
        title='Stacked Bar Graph by dataframe')
plt.show()
```

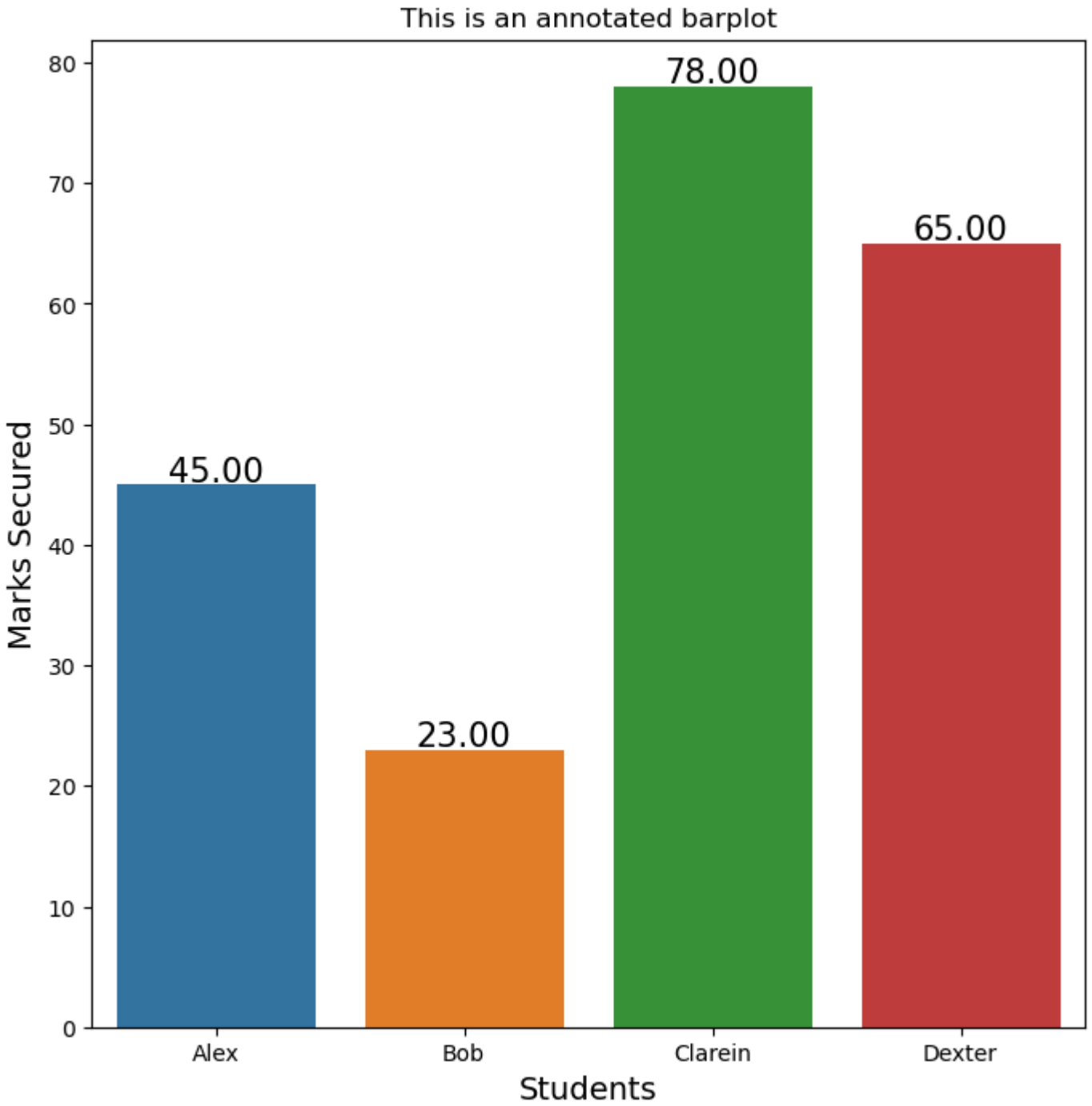
```
  Team  Round 1  Round 2  Round 3
0    A       10       20       10
1    B       20       25       15
2    C       12       15       19
3    D       10       18       11
```



In [49]:
```
N = 5
boys = (20, 35, 30, 35, 27)
girls = (25, 32, 34, 20, 25)
boyStd = (2, 3, 4, 1, 2)
girlStd = (3, 5, 2, 3, 3)
ind = np.arange(N)
width = 0.35

fig = plt.subplots(figsize =(10, 7))
p1 = plt.bar(ind, boys, width, yerr = boyStd)
p2 = plt.bar(ind, girls, width,
             bottom = boys, yerr = girlStd)

plt.ylabel('Contribution')
plt.xlabel('Teams')
plt.title('Contribution of GIRLS VS BOYS in the teams',fontweight ='bold', fontsize = 20)
plt.xticks(ind, ('T1', 'T2', 'T3', 'T4', 'T5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('boys', 'girls'))

plt.show()
```


Contribution of GIRLS VS BOYS in the teams
boys
girls
Contribution
0
10
20
30
40
50
60
70
80
T1
T2
T3
T4
T5
Teams

In [64]:
```
import seaborn as sns
data = {"Name": ["Alex", "Bob", "Clarein", "Dexter"],
        "Marks": [45, 23, 78, 65]}

df = pd.DataFrame(data, columns=['Name', 'Marks'])

plt.figure(figsize=(8, 8))

# Defining the values for x-axis, y-axis
# and from which dataframe the values are to be picked
plots = sns.barplot(x="Name", y="Marks", data=df)

# Iterating over the bars one-by-one
for bar in plots.patches:

  # Using Matplotlib's annotate function and
  # passing the coordinates where the annotation shall be done
  # x-coordinate: bar.get_x() + bar.get_width() / 2
  # y-coordinate: bar.get_height()
  # free space to be left to make graph pleasing: (0, 8)
  # ha and va stand for the horizontal and vertical alignment

  plots.annotate(format(bar.get_height(), '.2f'),
                 (bar.get_x() + bar.get_width() / 2,
                  bar.get_height()), ha='center', va='center',
                 size=15, xytext=(0, 6),
                 textcoords='offset points')

plt.xlabel("Students", size=14)
plt.ylabel("Marks Secured", size=14)
plt.title("This is an annotated barplot")
plt.show()
```

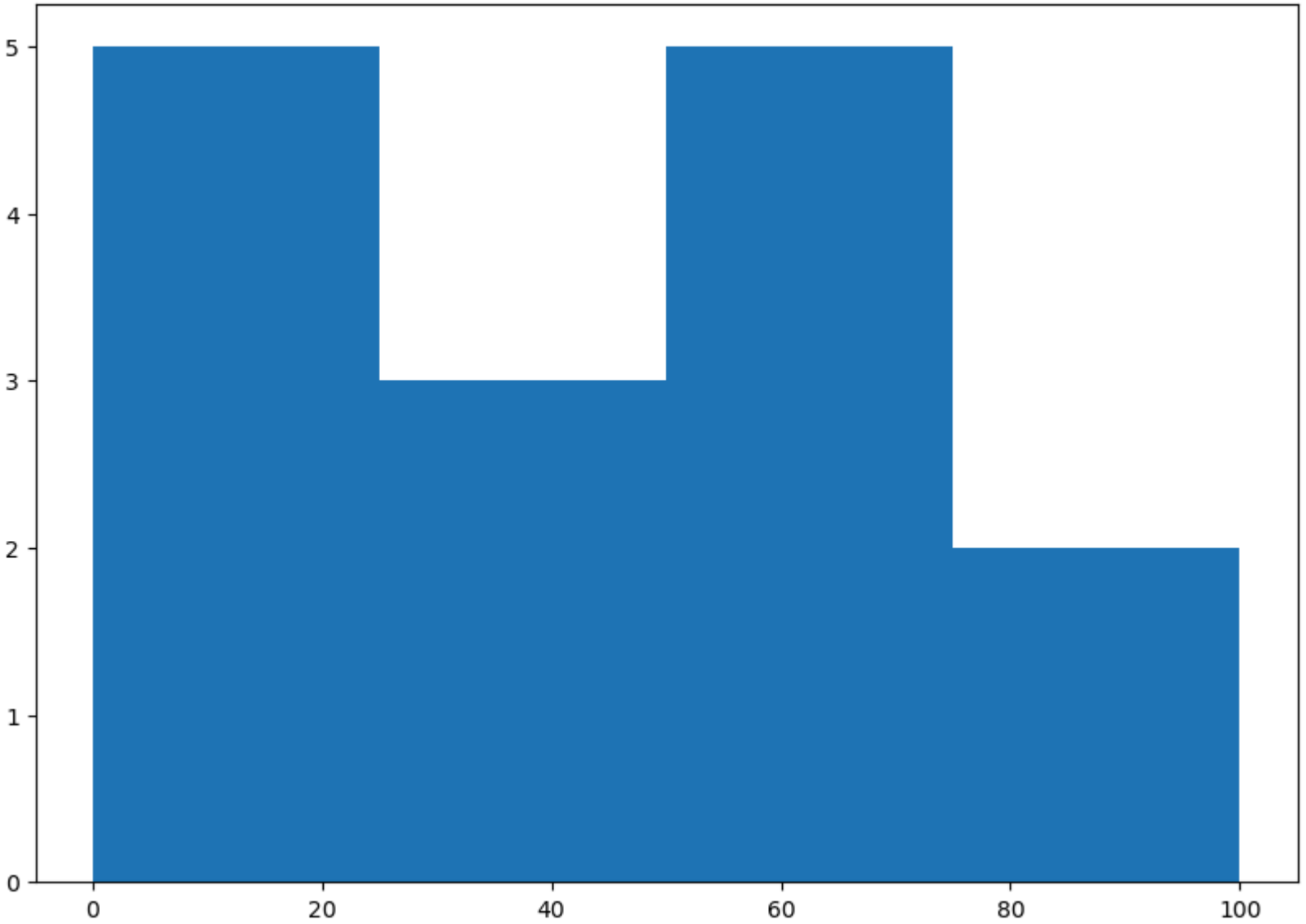

This is an annotated barplot
Marks Secured
Students
Alex
Bob
Clarein
Dexter
45.00
23.00
78.00
65.00
0
10
20
30
40
50
60
70
80

HISTOGRAM IN MATPLOTLIB

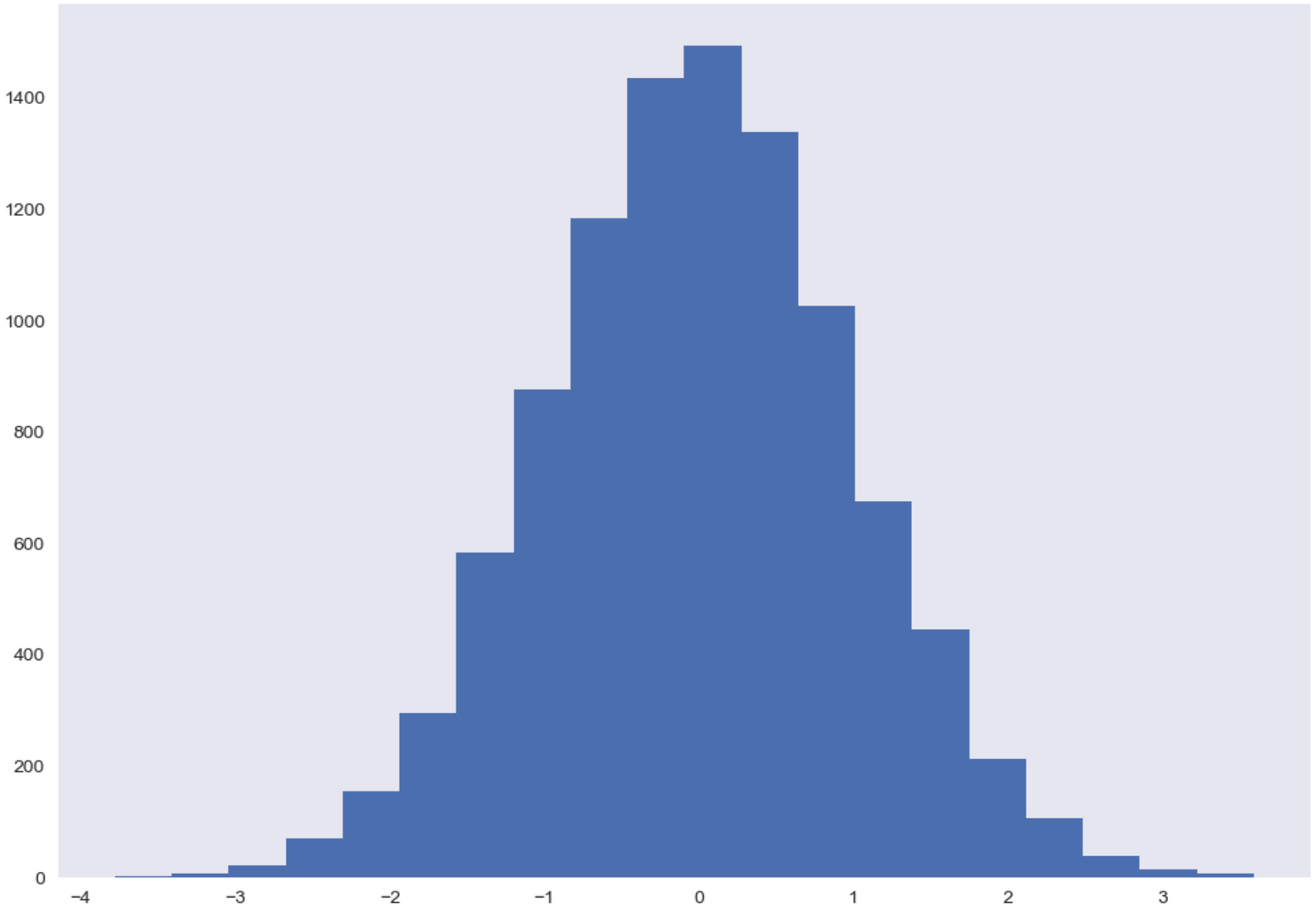
A histogram is basically used to represent data provided in a form of some groups. It is accurate method for the graphical representation of numerical data distribution. It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency.

matplotlib.pyplot.hist() function is used to compute and create histogram

In [65]:
```
a = np.array([22, 87, 5, 43, 56,73, 55, 54, 11, 20, 51, 5, 79, 31,27])

fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(a, bins = [0, 25, 50, 75, 100])

plt.show()
```

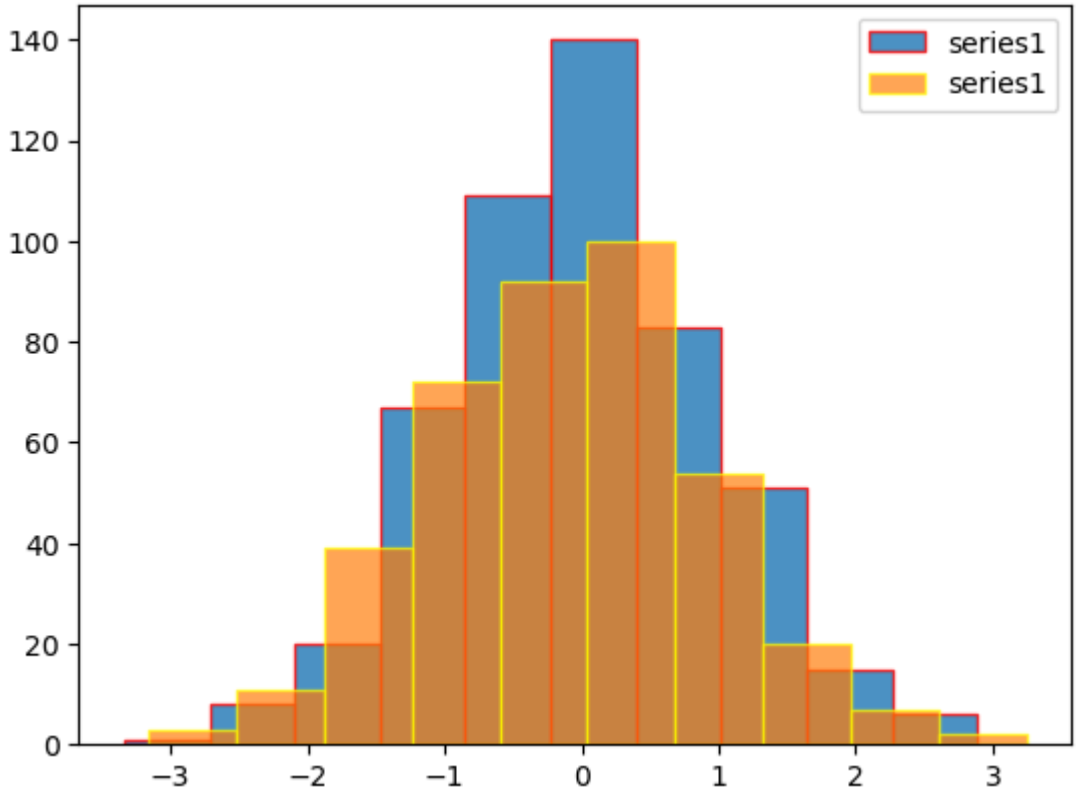

5
4
3
2
1
0
0
20
40
60
80
100

In [125]:
```
np.random.seed(23685752)
N_points = 10000
n_bins = 20

# Creating distribution
x = np.random.randn(N_points)

# Creating histogram
fig, ax = plt.subplots(1, 1,figsize =(10, 7),tight_layout = True)
plt.grid(visible=False)
ax.hist(x, bins = n_bins)
plt.show()
```


1400
1200
1000
800
600
400
200
0
−4
−3
−2
−1
0
1
2
3

PLOTTING TWO HISTOGRAMS TOGETHER

In [69]:
```
series1 = np.random.randn(500, 1)
series2 = np.random.randn(400, 1)

plt.hist(series1,label='series1', alpha=0.8, edgecolor='red')

plt.hist(series2,label='series1', alpha=0.7, edgecolor='yellow')
plt.legend()
plt.show()
```



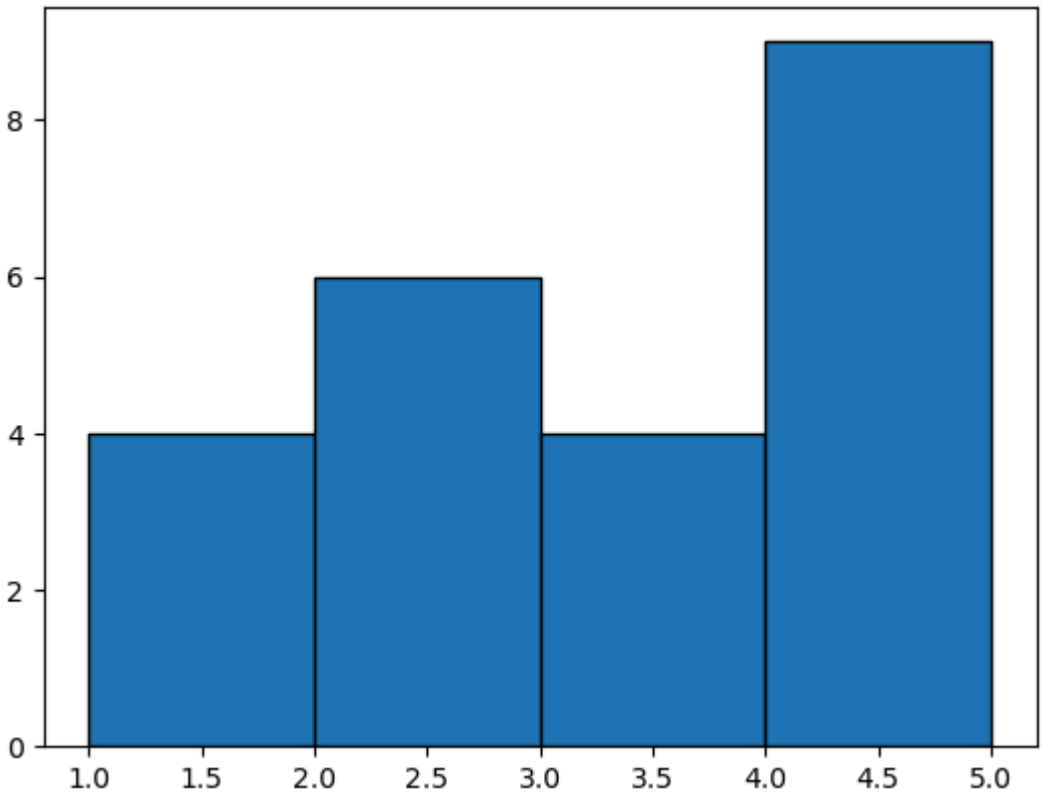
Bin size in Matplotlib Histogram A histogram is a graphical representation of the distribution of data given by the user. Its appearance is similar to Bar-Graph except it is continuous.

The towers or bars of a histogram are called bins. The height of each bin shows how many values from that data fall into that range.

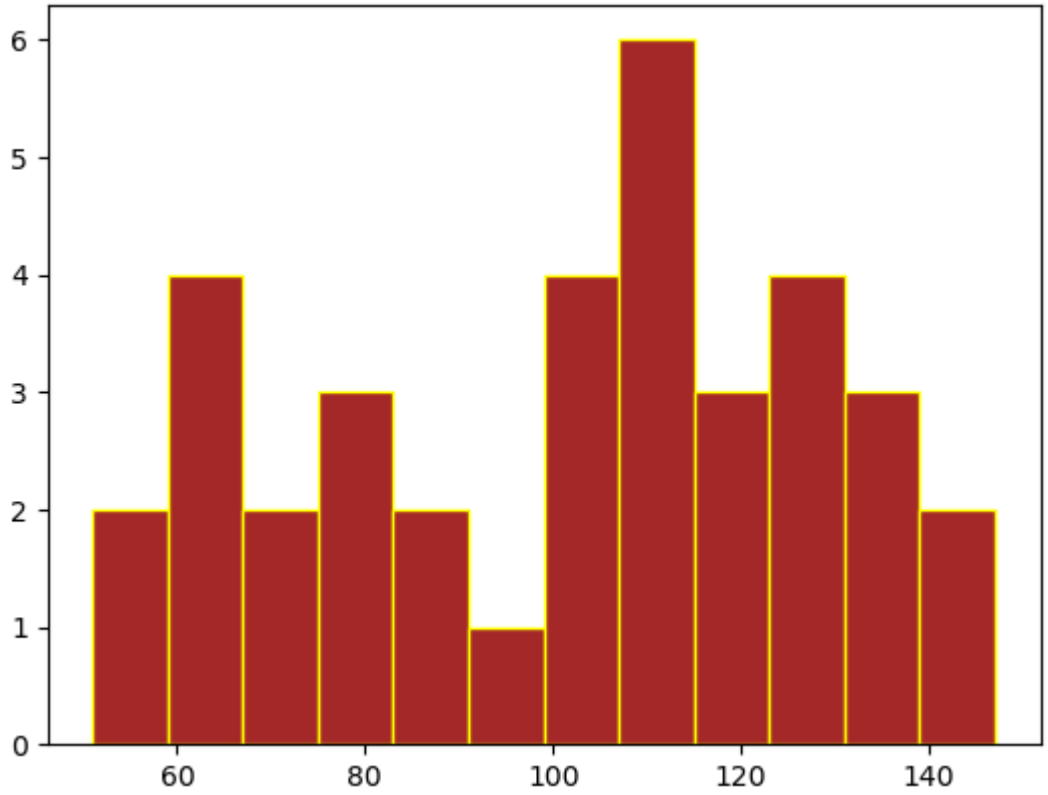
Width of each bin is = (max value of data – min value of data) / total number of bins

The default value of the number of bins to be created in a histogram is 10.

In [70]:
```
marks = [1, 2, 3, 2, 1, 2, 3, 2,1, 4, 5, 4, 3, 2, 5, 4,5, 4, 5, 3, 2, 1, 5]

plt.hist(marks, bins=[1, 2, 3, 4, 5], edgecolor="black")
#plt.hist(marks, edgecolor="red", bins=5)
plt.show()
```



In [71]:
```
import matplotlib.pyplot as plt

data = [87, 53, 66, 61, 67, 68, 62, 110,104, 61, 111, 123, 117, 119, 116,
        104, 92, 111, 90, 103, 81, 80, 101,51, 79, 107, 110, 129, 145, 128,
        132, 135, 131, 126, 139, 110]

binwidth = 8
plt.hist(data, bins=range(min(data), max(data) + binwidth, binwidth),
         edgecolor="yellow", color="brown")

plt.show()
```



SCATTER PLOT IN MATPLOTLIB

Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot. Scatter plots are widely used to represent relation among variables and how change in one affects the other.

**Syntax matplotlib.pyplot.scatter(x_axis_data, y_axis_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None) **

x_axis_data- An array containing x-axis data

y_axis_data- An array containing y-axis data

s- marker size (can be scalar or array of size equal to size of x or y)

c- color of sequence of colors for markers

marker- marker style

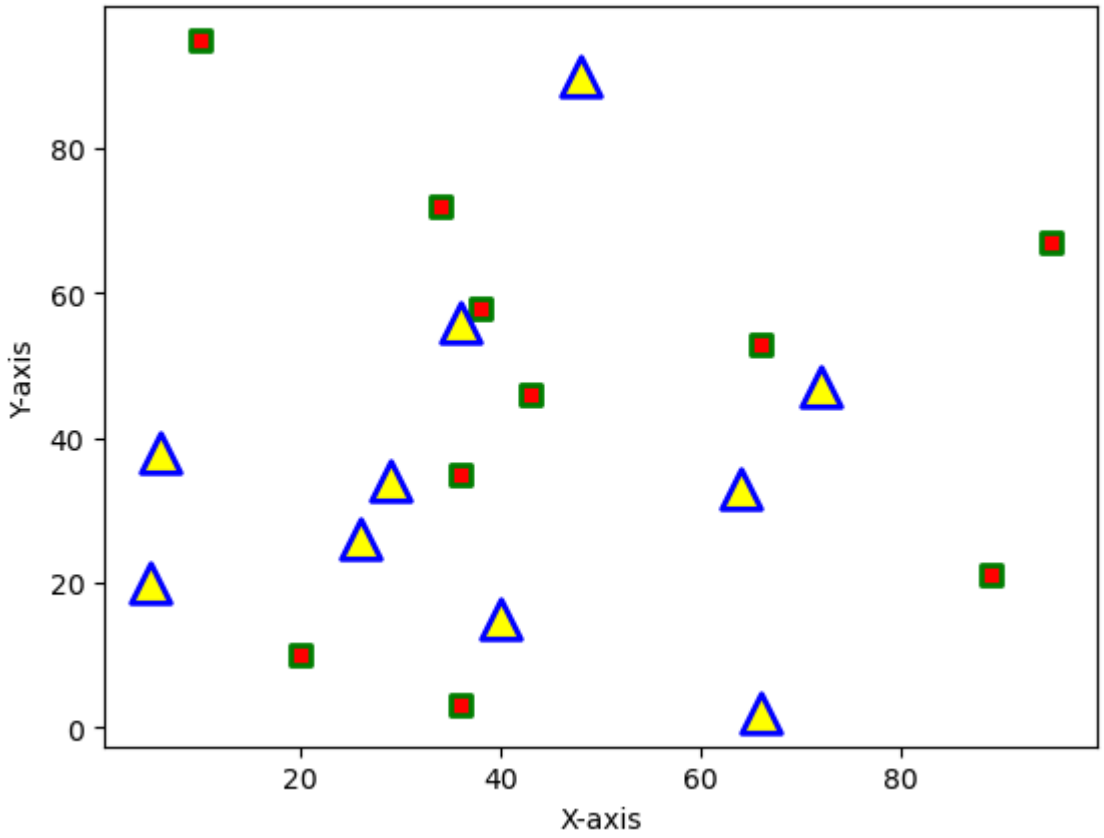
cmap- cmap name

linewidths- width of marker border

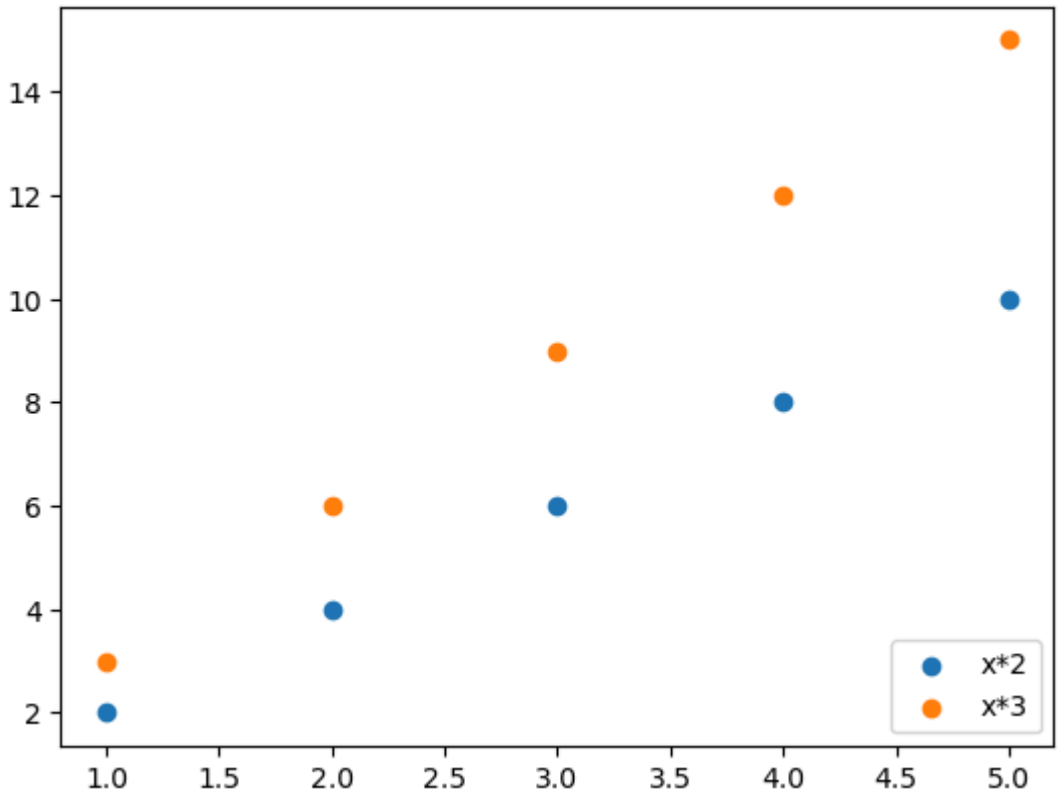
edgecolor- marker border color

alpha- blending value, between 0 (transparent) and 1 (opaque)

In [75]:
```
x1 = [89, 43, 36, 36, 95, 10,66, 34, 38, 20]
y1 = [21, 46, 3, 35, 67, 95,53, 72, 58, 10]

x2 = [26, 29, 48, 64, 6, 5,36, 66, 72, 40]

y2 = [26, 34, 90, 33, 38,20, 56, 2, 47, 15]

plt.scatter(x1, y1, c ="red",linewidths = 2, marker ="s",edgecolor ="green",
            s = 50)

plt.scatter(x2, y2, c ="yellow",linewidths = 2,marker ="^", edgecolor ="blue",
            s = 200)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt.show()
```

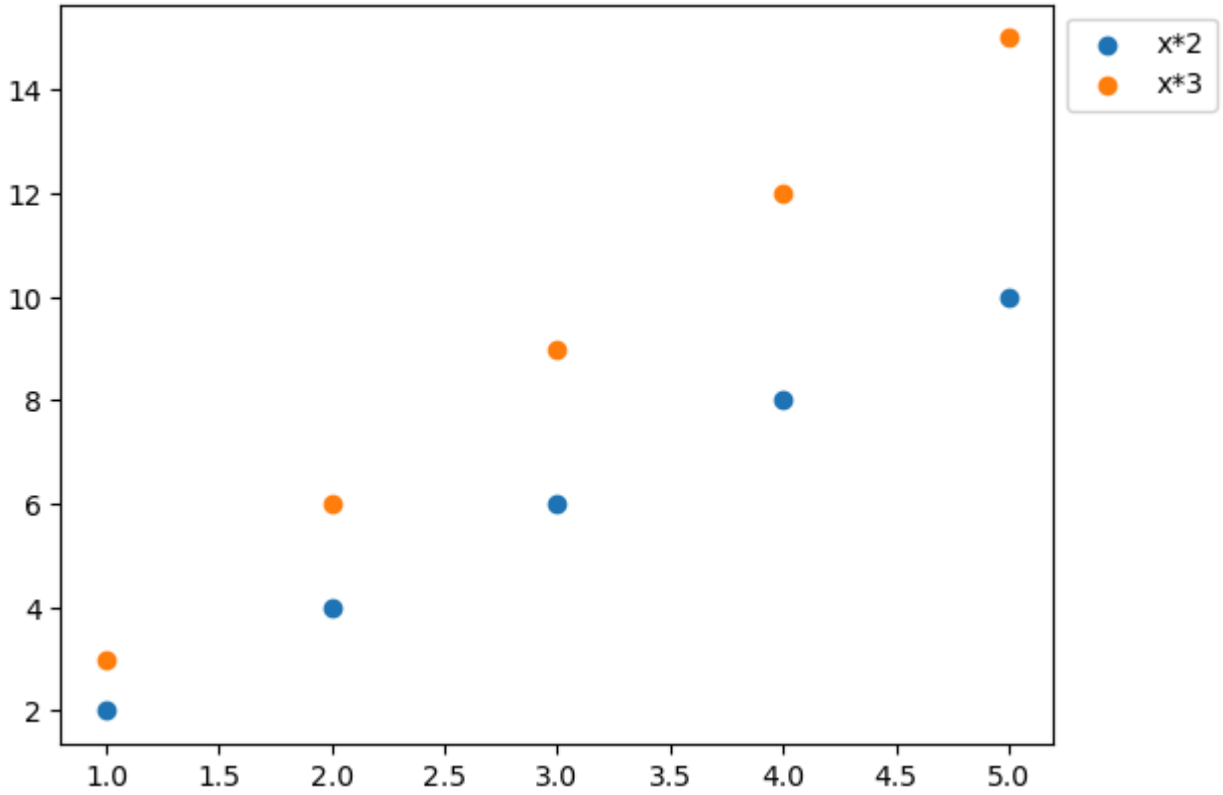

Y-axis
X-axis
0
20
40
60
80
20
40
60
80

In [80]:
```
x =  [1,2,3,4,5]
y1 = [2,4,6,8,10]
y2 = [3,6,9,12,15]

plt.scatter(x, y1)
plt.scatter(x,y2)

plt.legend(["x*2" , "x*3"], ncol = 1 , loc = "lower right")
plt.show()
```

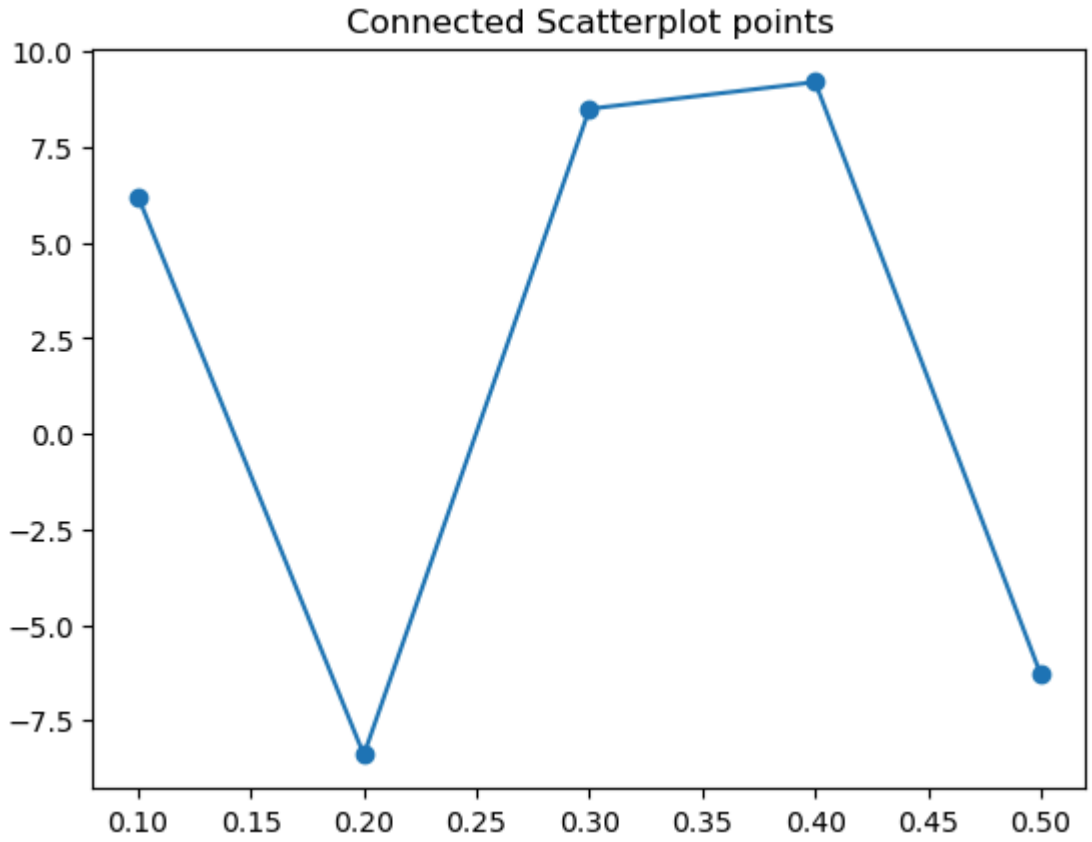


In [81]:

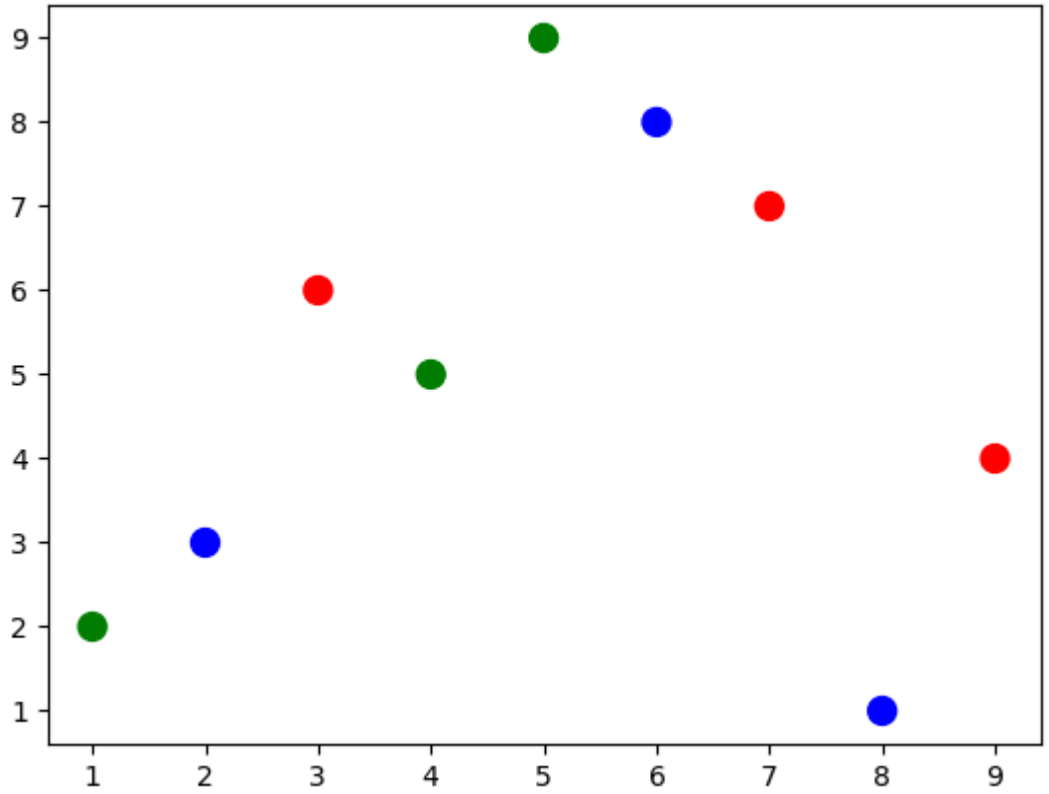
```
x =  [1,2,3,4,5]
y1 = [2,4,6,8,10]
y2 = [3,6,9,12,15]

plt.scatter(x, y1)
plt.scatter(x,y2)

# apply legend()
plt.legend(["x*2" , "x*3"], bbox_to_anchor = (1 , 1))
plt.show()
```



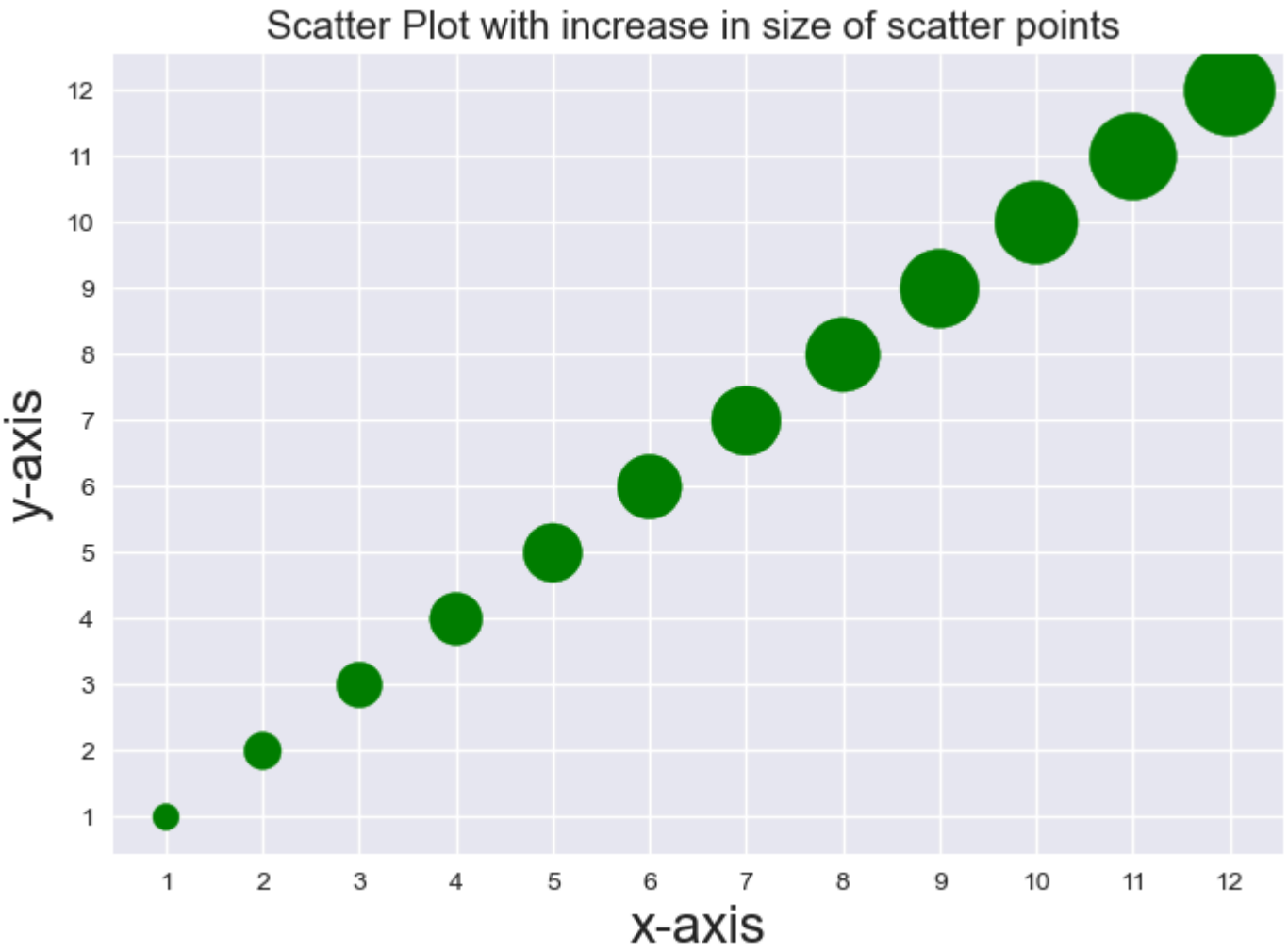
In [83]:
```
x = [0.1, 0.2, 0.3, 0.4, 0.5]
y = [6.2, -8.4, 8.5, 9.2, -6.3]

plt.title("Connected Scatterplot points")

plt.scatter(x, y)

plt.plot(x, y,marker="*")
plt.show()
```



In [87]:
```
a = np.array([[9, 1, 2, 7, 5, 8, 3, 4, 6],
              [4, 2, 3, 7, 9, 1, 6, 5, 8]])

categories = np.array([0, 1, 2, 0, 1, 2, 0, 1, 2])

colormap = np.array(['r', 'g', 'b'])

plt.scatter(a[0], a[1], s=100, c=colormap[categories])
plt.show()
```



In [186]:
```
plt.style.use('seaborn')  #background grid

x = [1,2,3,4,5,6,7,8,9,10,11,12]
y = [1,2,3,4,5,6,7,8,9,10,11,12]
points_size = [100,200,300,400,500,600,700,800,900,1000,1100,1200]

plt.xticks(np.arange(13))
plt.yticks(np.arange(13))

#plt.xticks(x, fontsize=12)
#plt.yticks(y, fontsize=12)

plt.scatter(x,y,s=points_size,c='g')

plt.title("Scatter Plot with increase in size of scatter points ", fontsize=22)

plt.xlabel('x-axis',fontsize=20)
plt.ylabel('y-axis',fontsize=20)

plt.show()
```



In [ ]:

PIE CHART IN MATPLOTLIB

A Pie Chart is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges.The area of a wedge represents the relative percentage of that part with respect to whole data. Pie charts are commonly used in business

presentations like sales, operations, survey results, resources, etc

Syntax: matplotlib.pyplot.pie(data, explode=None, labels=None, colors=None, autopct=None, shadow=False) Parameters:

data:represents the array of data values to be plotted, the fractional area of each slice is represented by data/sum(data).

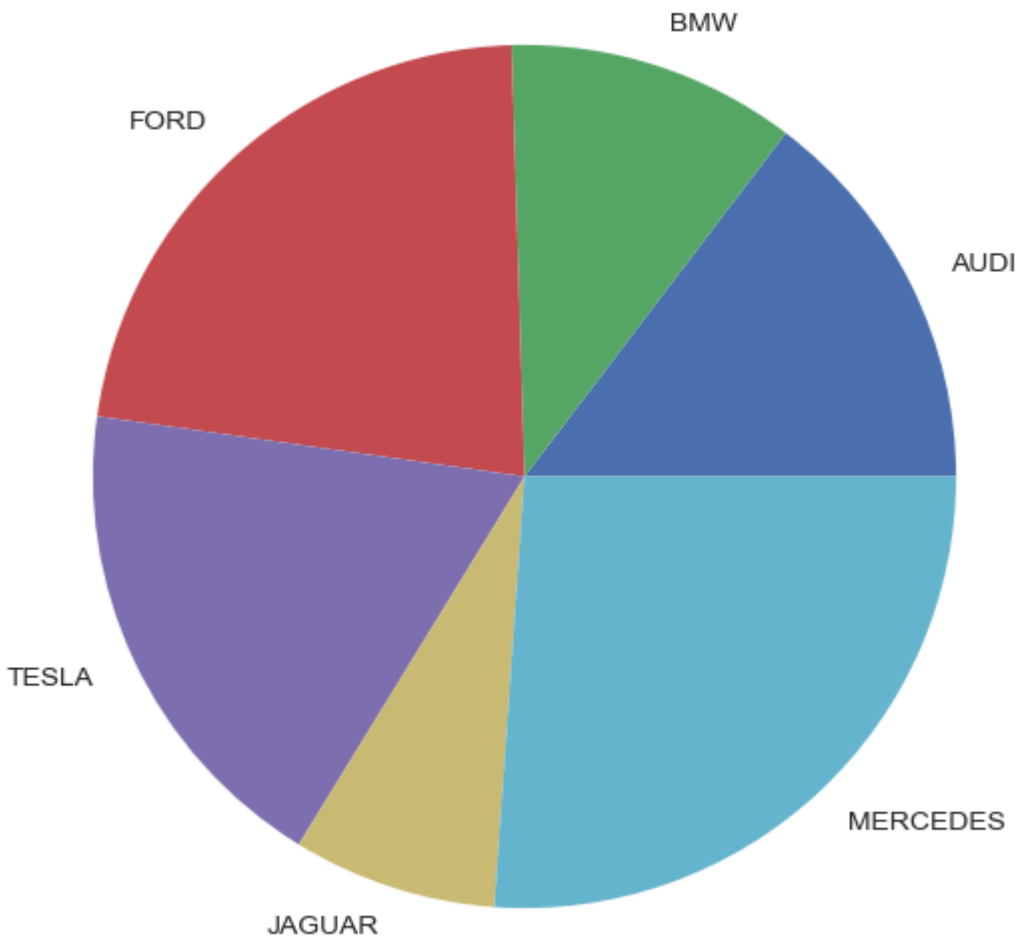
labels: is a list of sequence of strings which sets the label of each wedge.

color: attribute is used to provide color to the wedges.

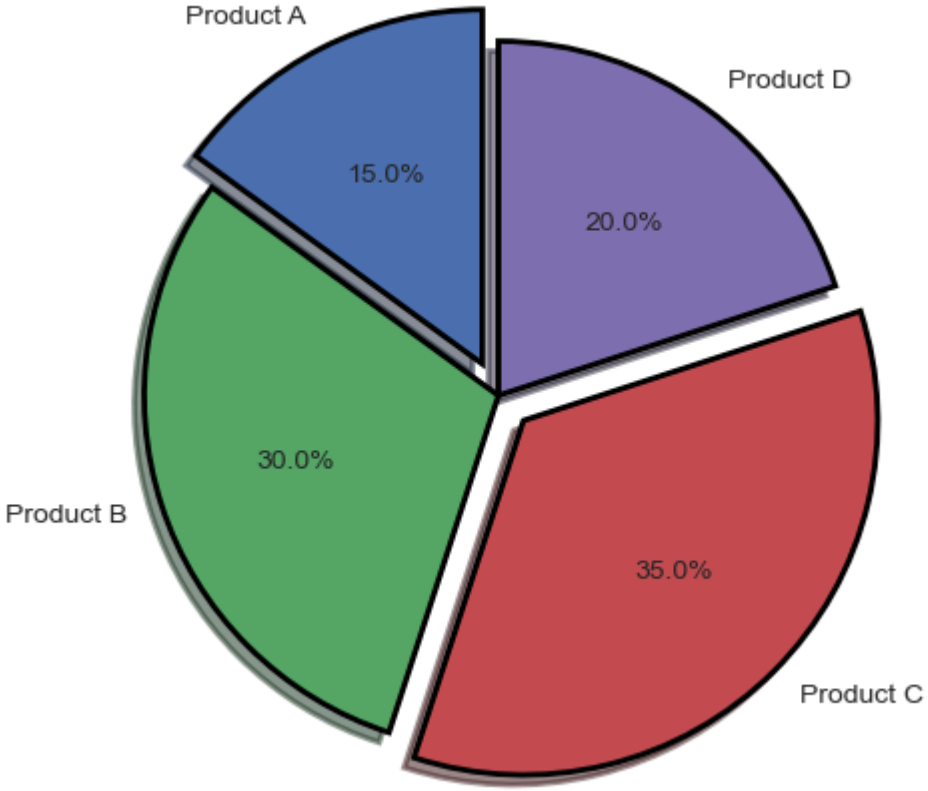
autopct: is a string used to label the wedge with their numerical value.

shadow: is used to create shadow of wedge.

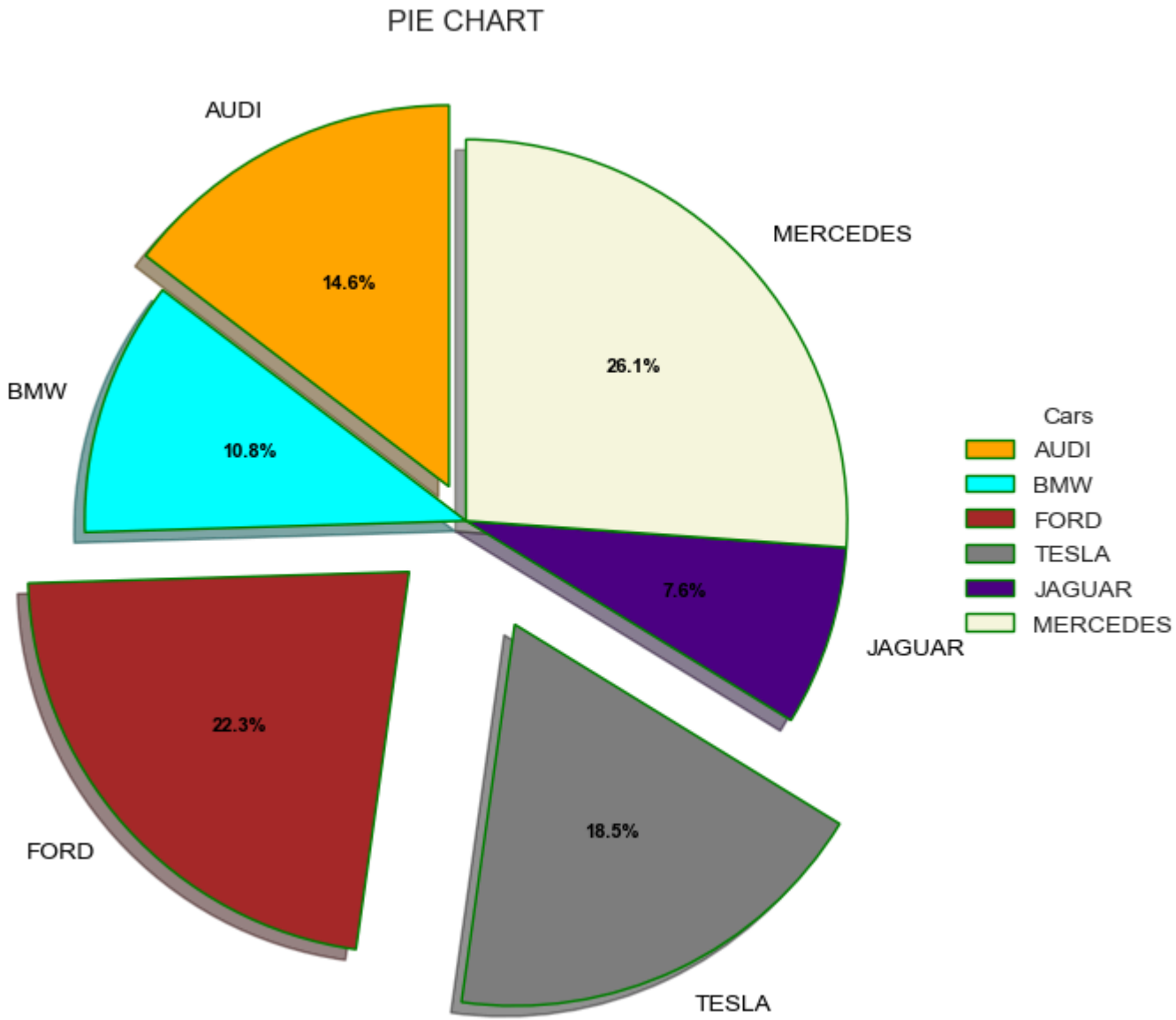
In [91]:
```
cars = ['AUDI', 'BMW', 'FORD','TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = cars)
plt.show()
```


BMW
FORD
AUDI
TESLA
MERCEDES
JAGUAR

In [112]:
```
product = ['Product A', 'Product B',
           'Product C', 'Product D']

stock = [15, 30, 35, 20]
explode = (0.1, 0, 0.1, 0)


plt.pie(stock, explode=explode, labels = product, autopct = '%1.1f%%',
        startangle = 90, shadow=True,
        wedgeprops = {"edgecolor" : "black",
                      'linewidth': 2,
                      'antialiased': True})

plt.axis('equal')
plt.show()
```


Product A
15.0%
Product D
20.0%
30.0%
Product B
35.0%
Product C

In [106]:
```
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

# Creating explode data
explode = (0.1, 0.0, 0.2, 0.3, 0.0, 0.0)

# Creating color parameters
colors = ( "orange", "cyan", "brown","grey", "indigo", "beige")

# Wedge properties
wp = { 'linewidth' : 1, 'edgecolor' : "green" }

# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(data,autopct ='%1.1f%%',
                                  explode = explode,labels = cars,
                                  shadow = True,colors = colors,
                                  startangle = 90,wedgeprops = wp,
                                  textprops = dict(color ="black"))

# Adding legend
ax.legend(wedges, cars,title ="Cars",loc ="center left",
          bbox_to_anchor =(1, 0, 0.5, 1))

plt.setp(autotexts, size = 8, weight ="bold")
ax.set_title("PIE CHART")

# show plot
plt.show()
```


PIE CHART
AUDI
MERCEDES
BMW
JAGUAR
FORD
TESLA
14.6%
26.1%
10.8%
7.6%
22.3%
18.5%
Cars
AUDI
BMW
FORD
TESLA
JAGUAR
MERCEDES

PLOTTING DATA DIRECTLY WITH PANDAS

In [167]:
```
df=pd.read_csv("car-sales.csv")
df
```

Out[167]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

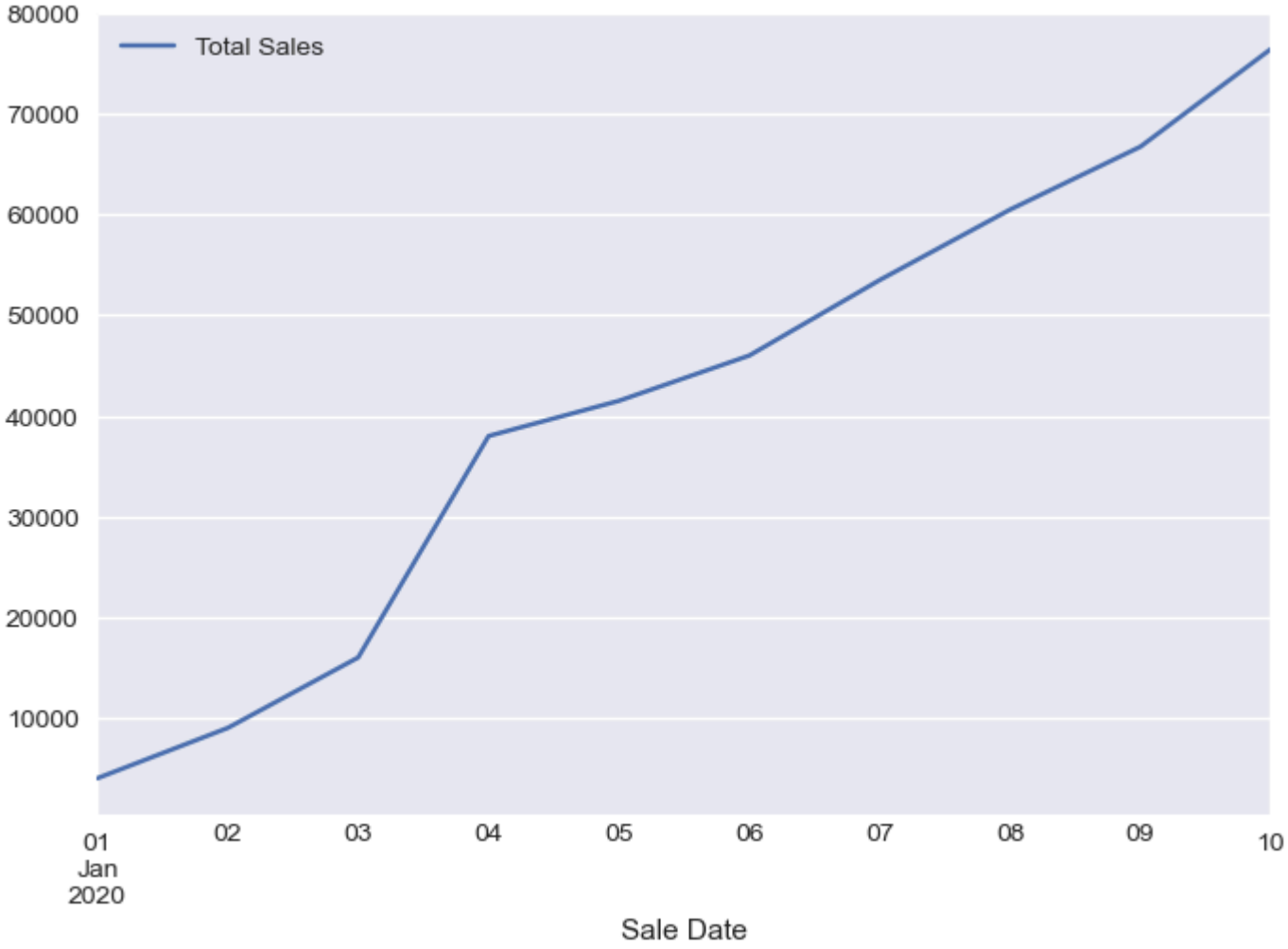
In [197]:
```
df["Price"] = df["Price"].str.replace('[\$\,\.]', '')
df['Price']=df['Price'].str[:-2]
df["Sale Date"] = pd.date_range("1/1/2020", periods=len(df))
df["Total Sales"] = df["Price"].astype(int).cumsum()
df
```

```
C:\Users\tanis\AppData\Local\Temp\ipykernel_59200\2301023276.py:1: FutureWarning: The default value of regex will cha
nge from True to False in a future version.
  df["Price"] = df["Price"].str.replace('[\$\,\.]', '')
```

Out[197]:

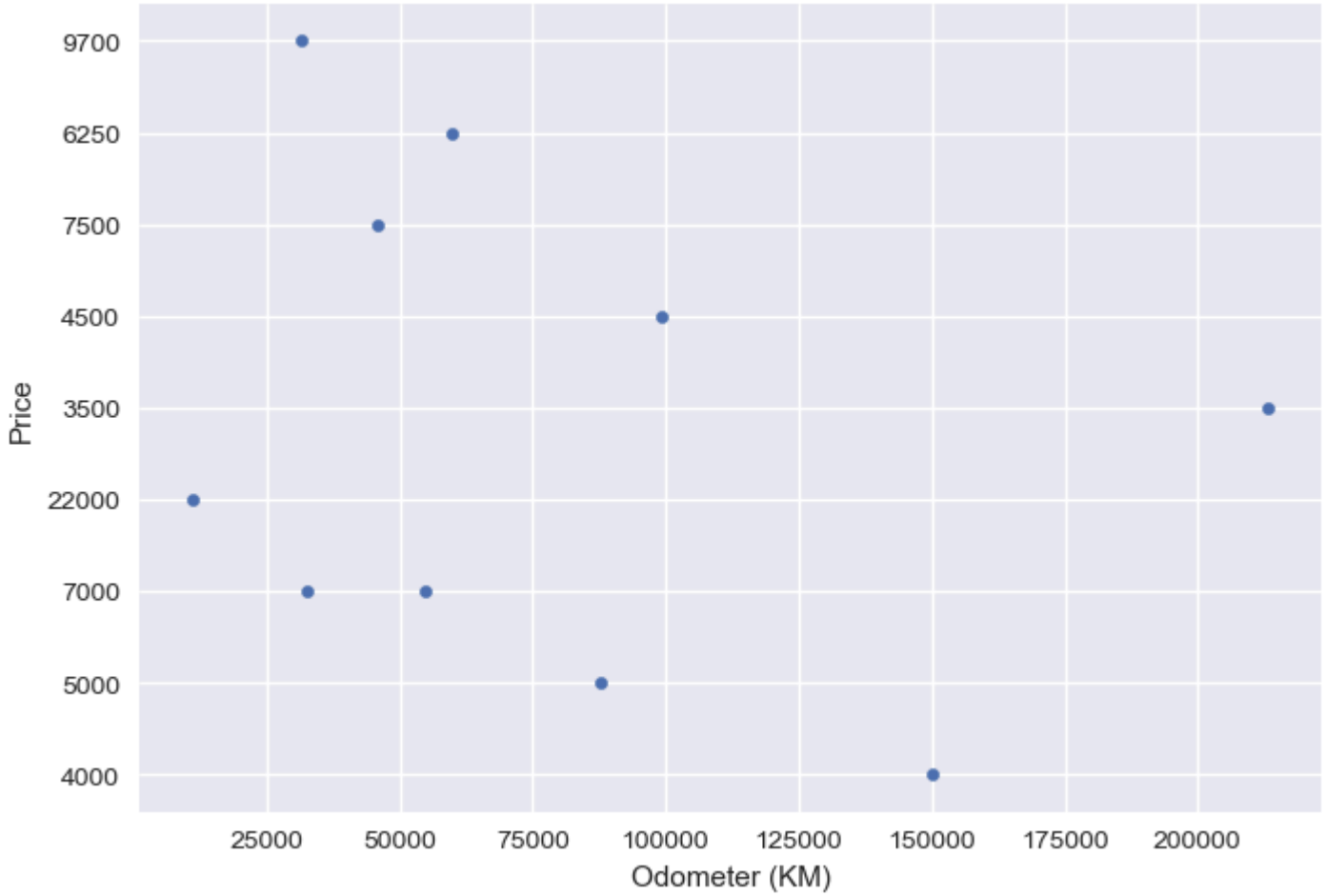
	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	40	2020-01-01	40
1	Honda	Red	87899	4	50	2020-01-02	90
2	Toyota	Blue	32549	3	70	2020-01-03	160
3	BMW	Black	11179	5	220	2020-01-04	380
4	Nissan	White	213095	4	35	2020-01-05	415
5	Toyota	Green	99213	4	45	2020-01-06	460
6	Honda	Blue	45698	4	75	2020-01-07	535
7	Honda	Blue	54738	4	70	2020-01-08	605
8	Toyota	White	60000	4	62	2020-01-09	667
9	Nissan	White	31600	4	97	2020-01-10	764

In [169]:
```
df.plot(x='Sale Date',y='Total Sales');
```

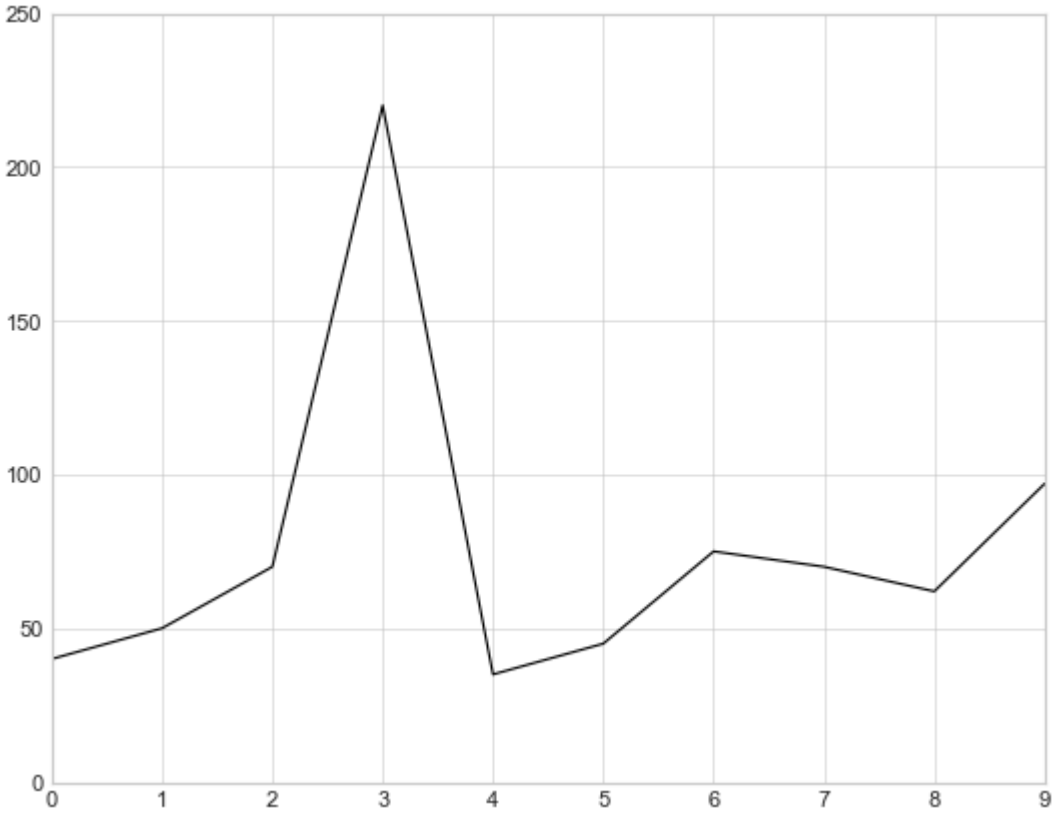


In [170]:
```
df.plot(x="Odometer (KM)", y="Price", kind="scatter")
```

Out[170]:
```
<AxesSubplot:xlabel='Odometer (KM)', ylabel='Price'>
```



In [203]:
```
df['Price']=df['Price'].astype(int)
df["Price"].plot();
```

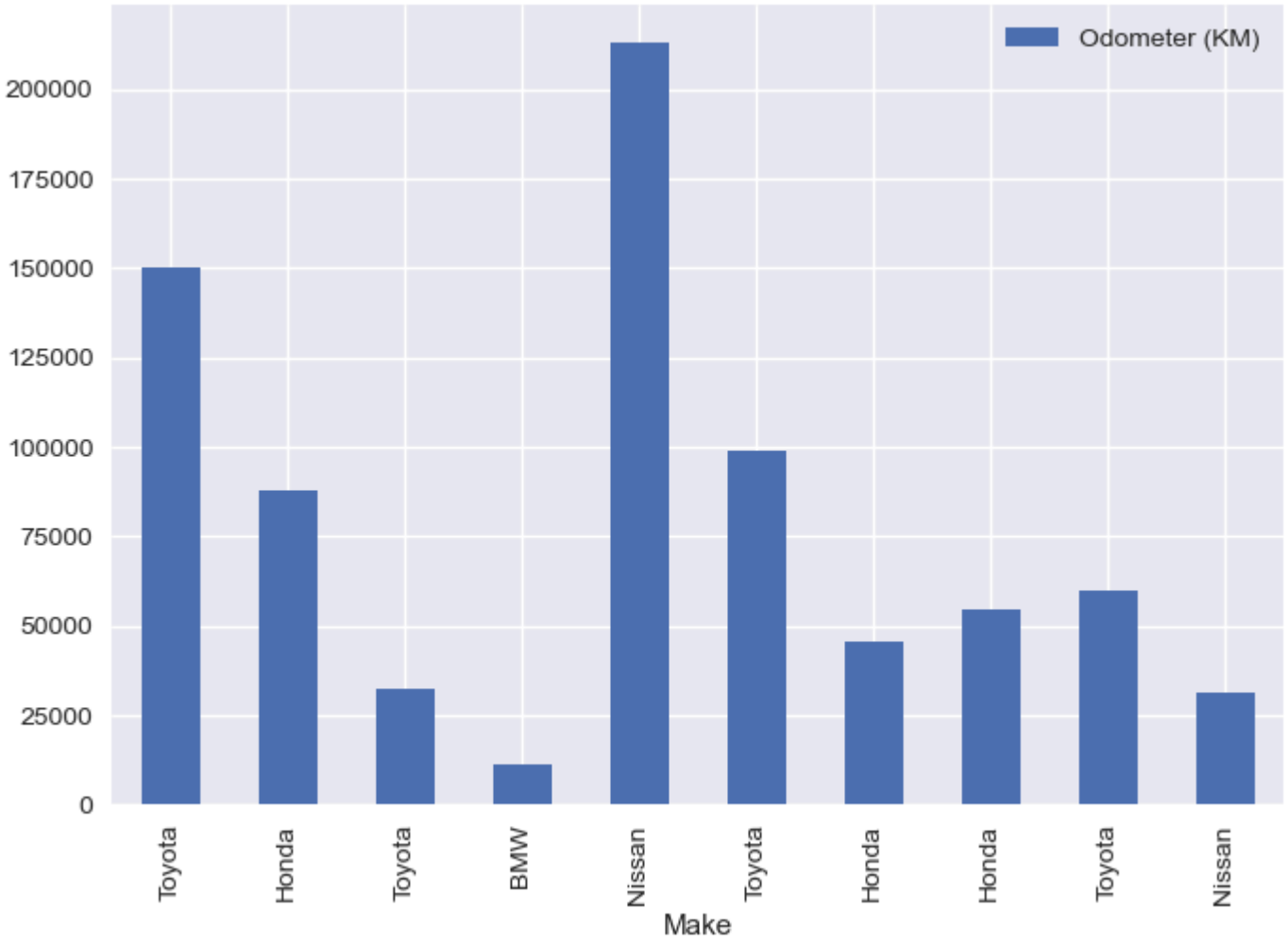


In [171]:
```
x = np.random.rand(10, 4)
data = pd.DataFrame(x, columns=['a', 'b', 'c', 'd'])
data.plot.bar()
#df.plot(kind="bar")
```

Out[171]: <AxesSubplot:>

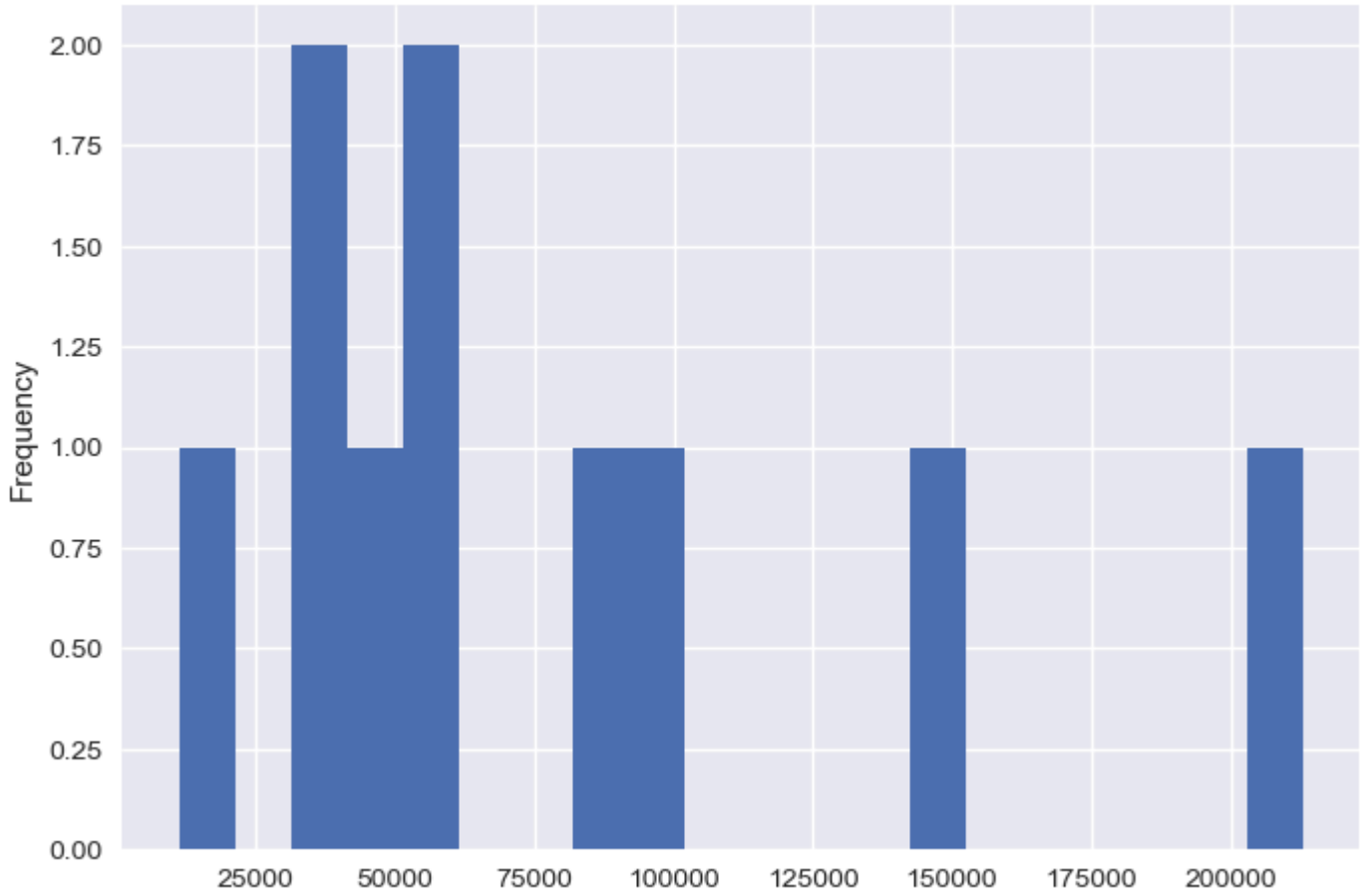
In [172]:
```
df.plot(x='Make',y='Odometer (KM)',kind='bar')
```

Out[172]:
```
<AxesSubplot:xlabel='Make'>
```



In [175]:
```
df['Odometer (KM)'].plot.hist(bins=20)
```

Out[175]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [176]:
```
hd=pd.read_csv("heart_disease.csv")
hd
```

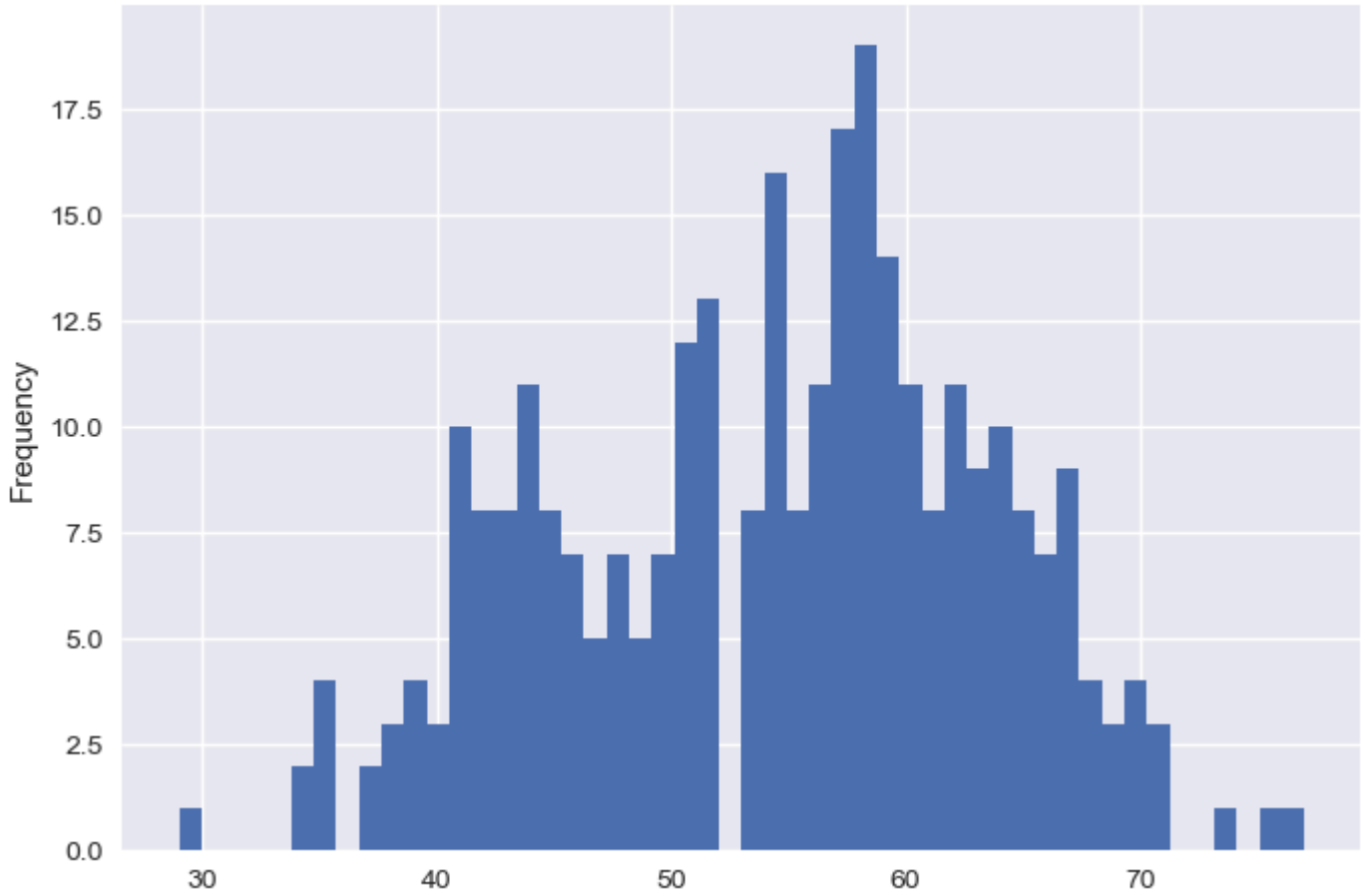
Out[176]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

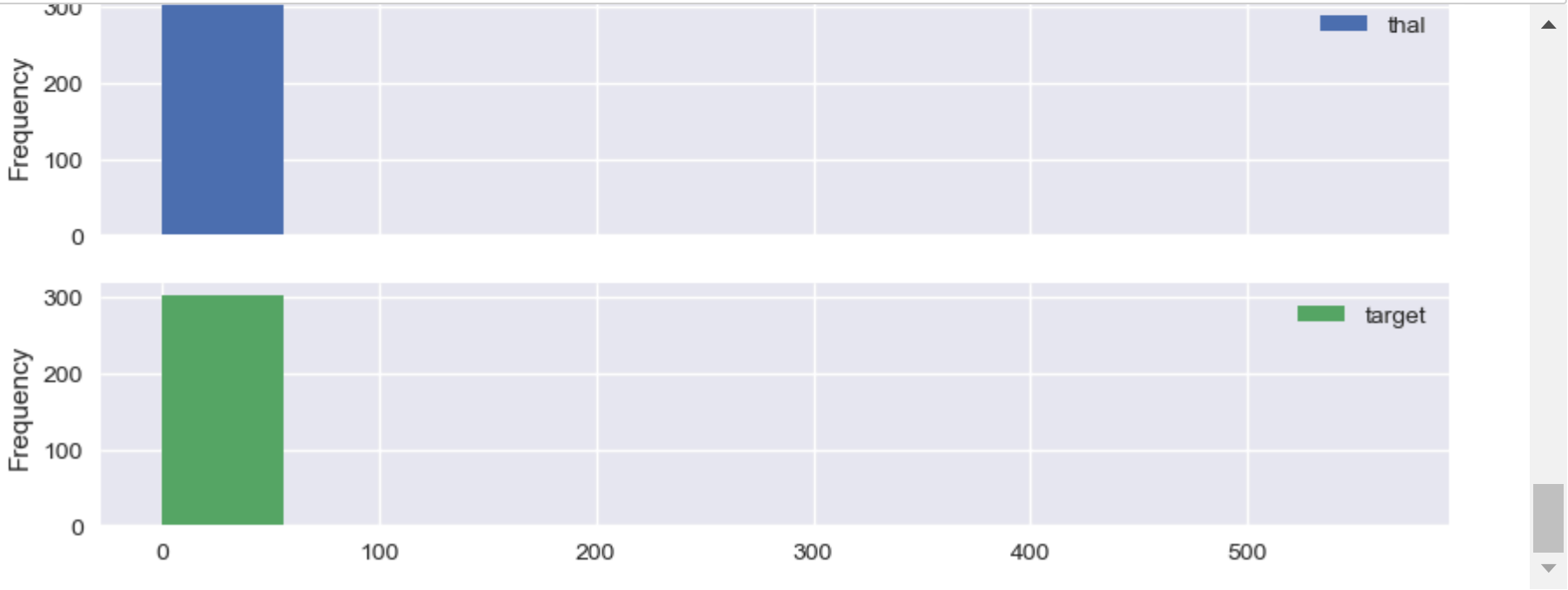
303 rows × 14 columns

In [177]:
```
hd['age'].plot.hist(bins=50)
```

Out[177]: <AxesSubplot:ylabel='Frequency'>

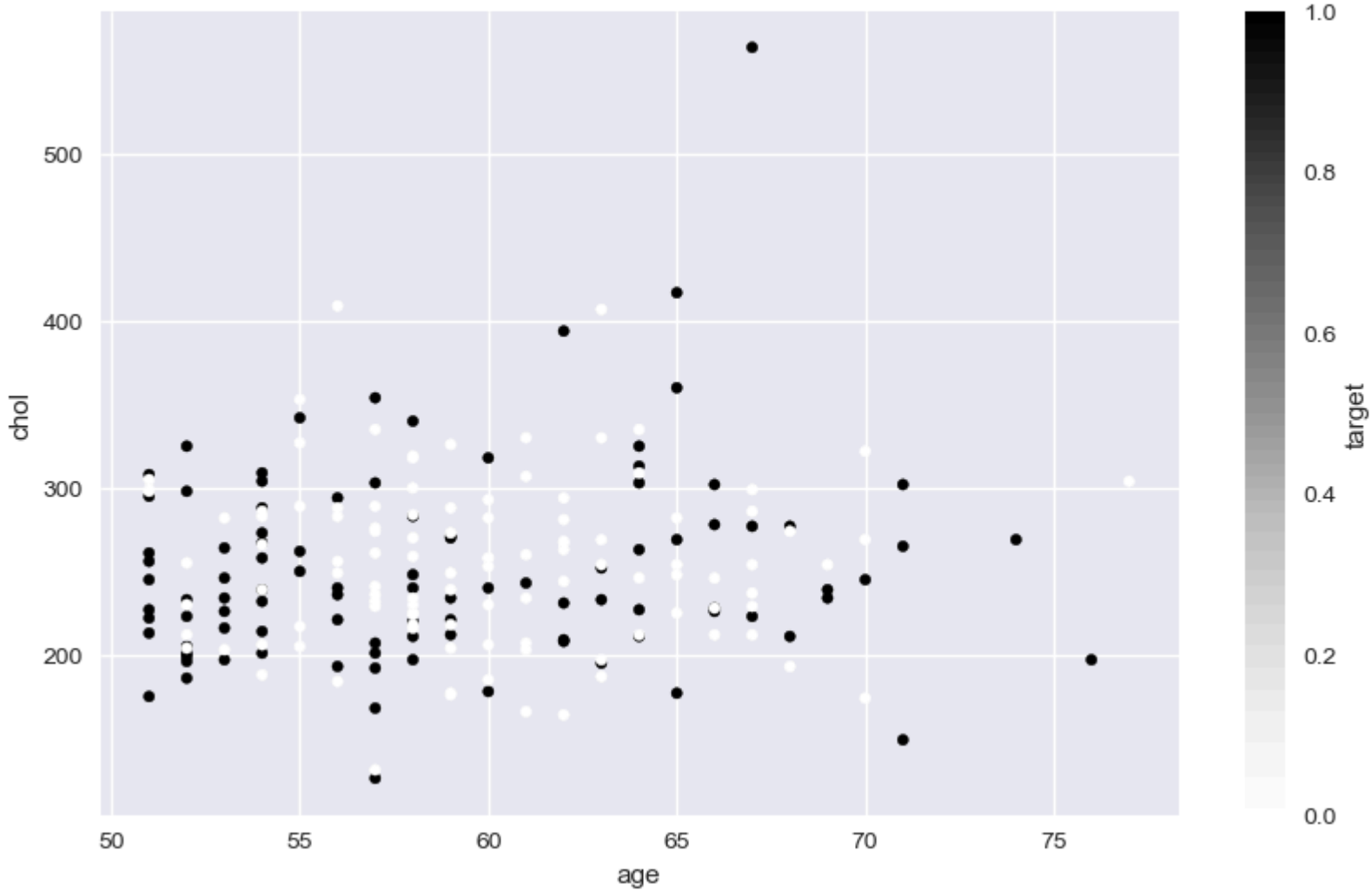


In [178]:
```
hd.plot.hist(figsize=(10, 30), subplots=True)
```

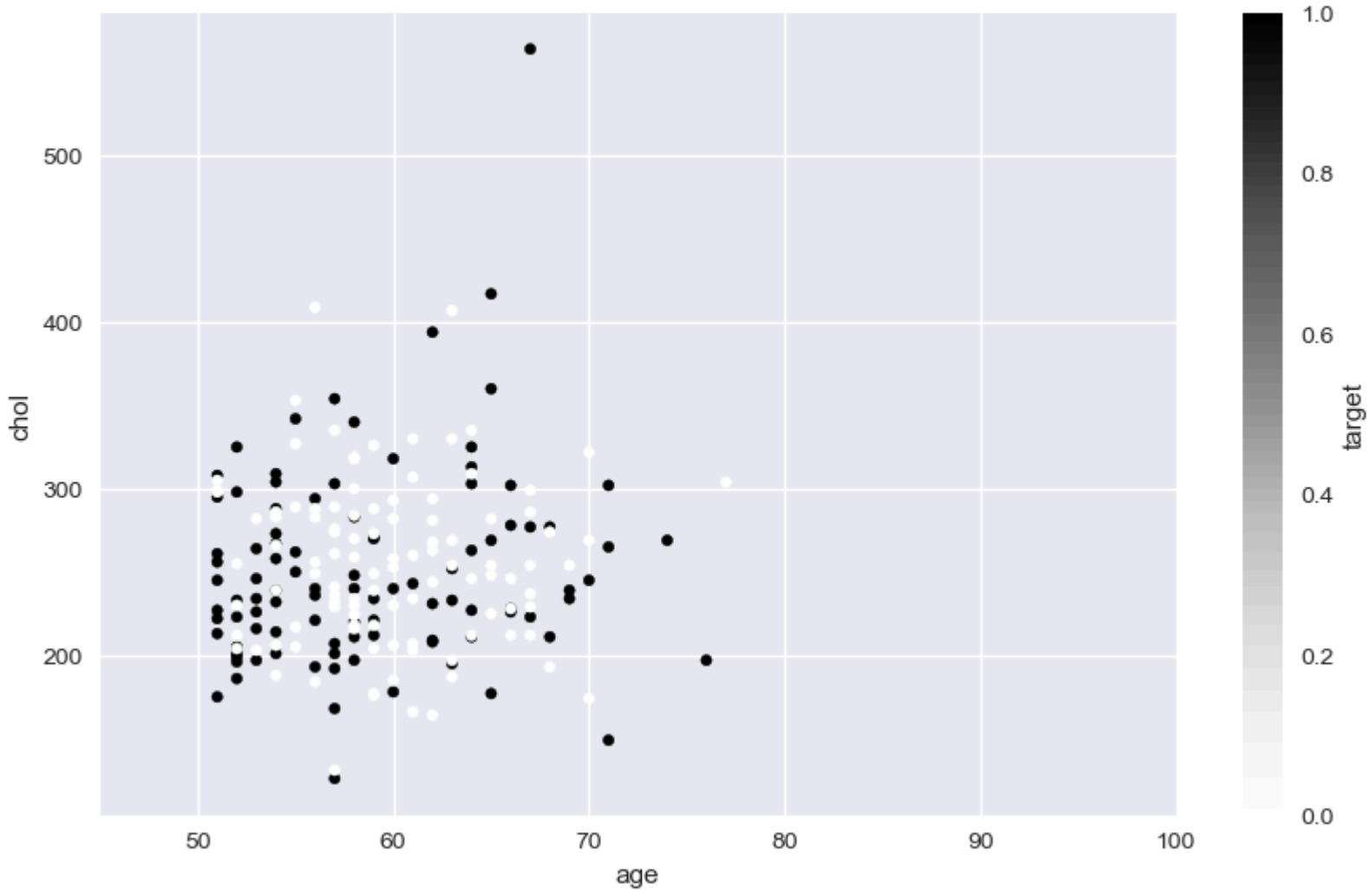


In [181]:
```
over_50 = hd[hd["age"] > 50]
over_50.plot(kind='scatter',x='age',y='chol',c='target',figsize=(10,6))
```

Out[181]: <AxesSubplot:xlabel='age', ylabel='chol'>

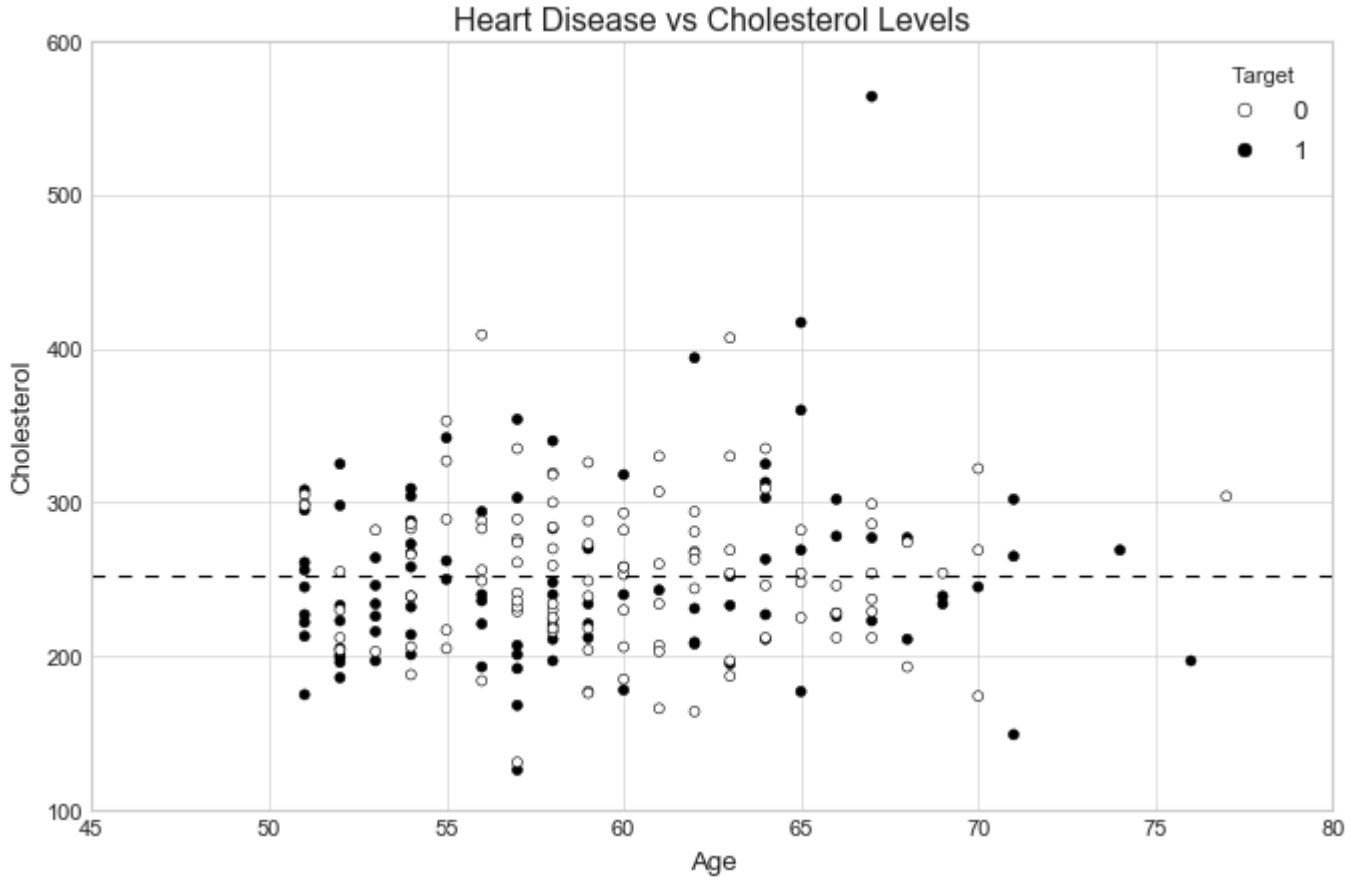


In [182]:
```
fig, ax1 = plt.subplots(figsize=(10, 6))
over_50.plot(kind='scatter', x="age", y="chol",
             c='target', ax=ax1);
ax1.set_xlim([45, 100]);
```

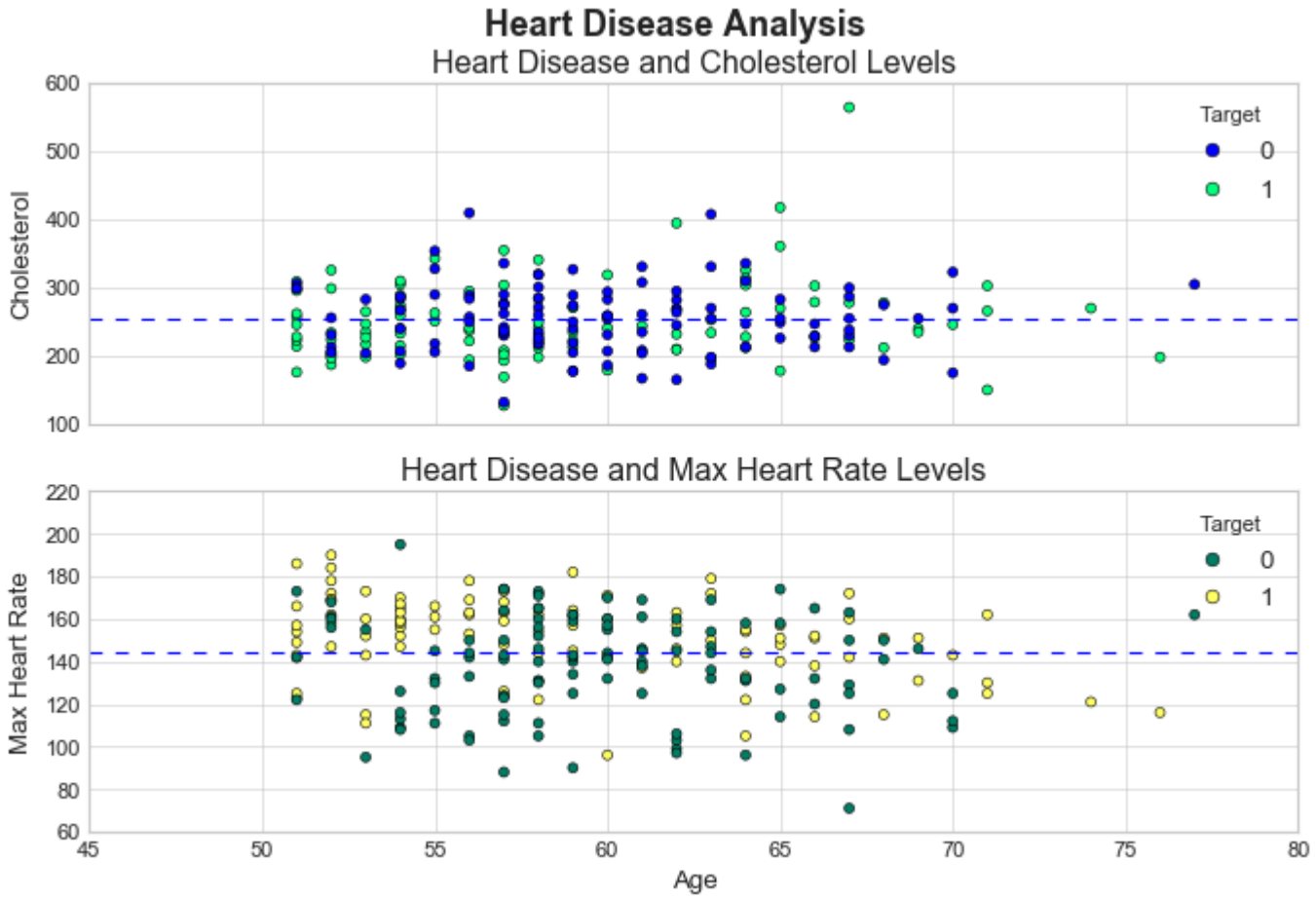


In [194]:
```
fig, ax = plt.subplots(figsize=(10, 6))
#fig: plotting object

scatter = ax.scatter(over_50["age"], over_50["chol"], c=over_50["target"])

# Customize the plot
ax.set(title="Heart Disease vs Cholesterol Levels",
       xlabel="Age",
       ylabel="Cholesterol");
ax.legend(*scatter.legend_elements(), title="Target");
ax.axhline(over_50["chol"].mean(),linestyle='--')
```

Out[194]: <matplotlib.lines.Line2D at 0x2066e87e910>



In [211]:
```
plt.style.use('seaborn-whitegrid')
fig, (ax0, ax1) = plt.subplots(nrows=2, # 2 rows
                               ncols=1, sharex=True,
                               figsize=(10,6))

# Add data for ax0
scatter = ax0.scatter(over_50["age"], over_50["chol"],c=over_50["target"]
                      ,cmap='winter')

ax0.set(title="Heart Disease and Cholesterol Levels",
        ylabel="Cholesterol")
ax0.legend(*scatter.legend_elements(), title="Target")

# Setup a mean line
ax0.axhline(y=over_50["chol"].mean(), color='b', linestyle='--',
            label="Average")


# Add data for ax1
scatter = ax1.scatter(over_50["age"], over_50["thalach"],
                      c=over_50["target"],cmap="summer")

ax1.set(title="Heart Disease and Max Heart Rate Levels",
        xlabel="Age",ylabel="Max Heart Rate")
ax1.legend(*scatter.legend_elements(), title="Target")

# Setup a mean line
ax1.axhline(y=over_50["thalach"].mean(),
            color='b',
            linestyle='--',
            label="Average")

# Title the figure
fig.suptitle('Heart Disease Analysis', fontsize=16, fontweight='bold');
```


Heart Disease Analysis
Heart Disease and Cholesterol Levels
Cholesterol
Target
0
1
Heart Disease and Max Heart Rate Levels
Max Heart Rate
Target
0
1
Age

In [195]:
```
#Style
plt.style.available
```

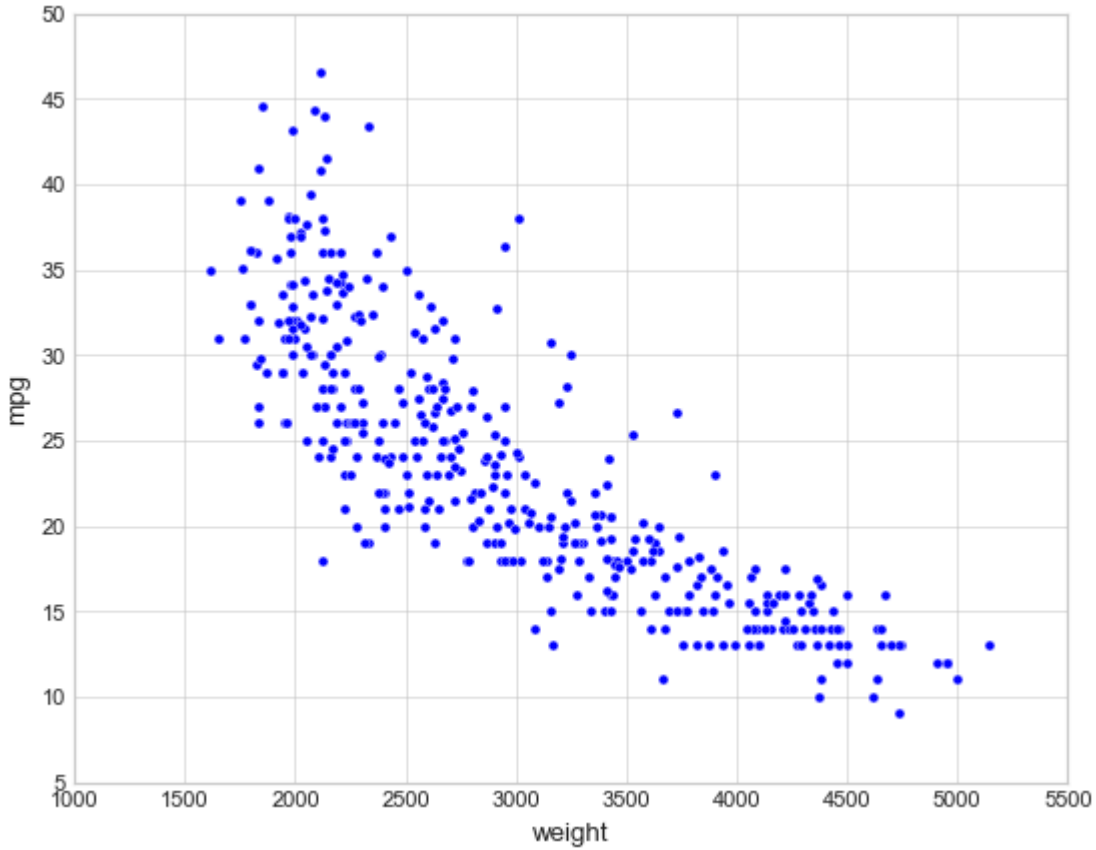
Out[195]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

In [214]:
```
import seaborn as sns
# load a seaborn dataset
mpg_df = sns.load_dataset("mpg")
mpg_df.head()
```

Out[214]:

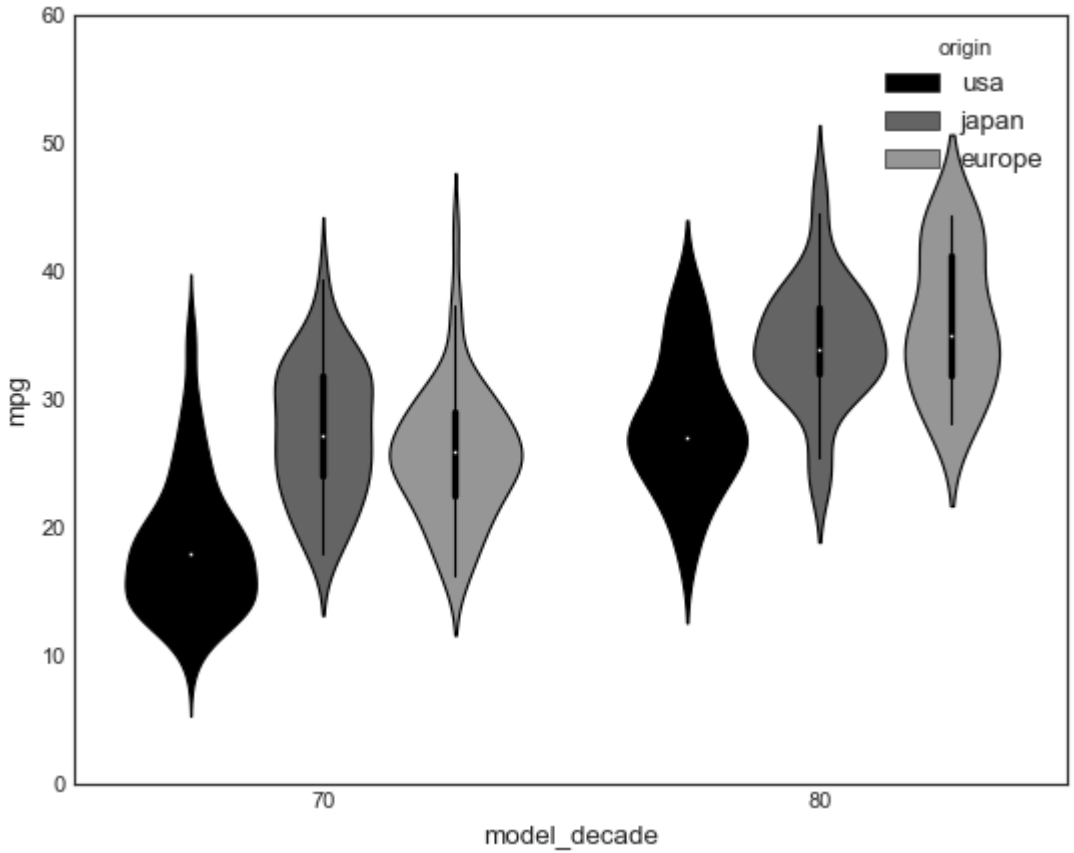
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

In [215]:
```
ax=sns.scatterplot(x="weight",y="mpg",data=mpg_df)
```



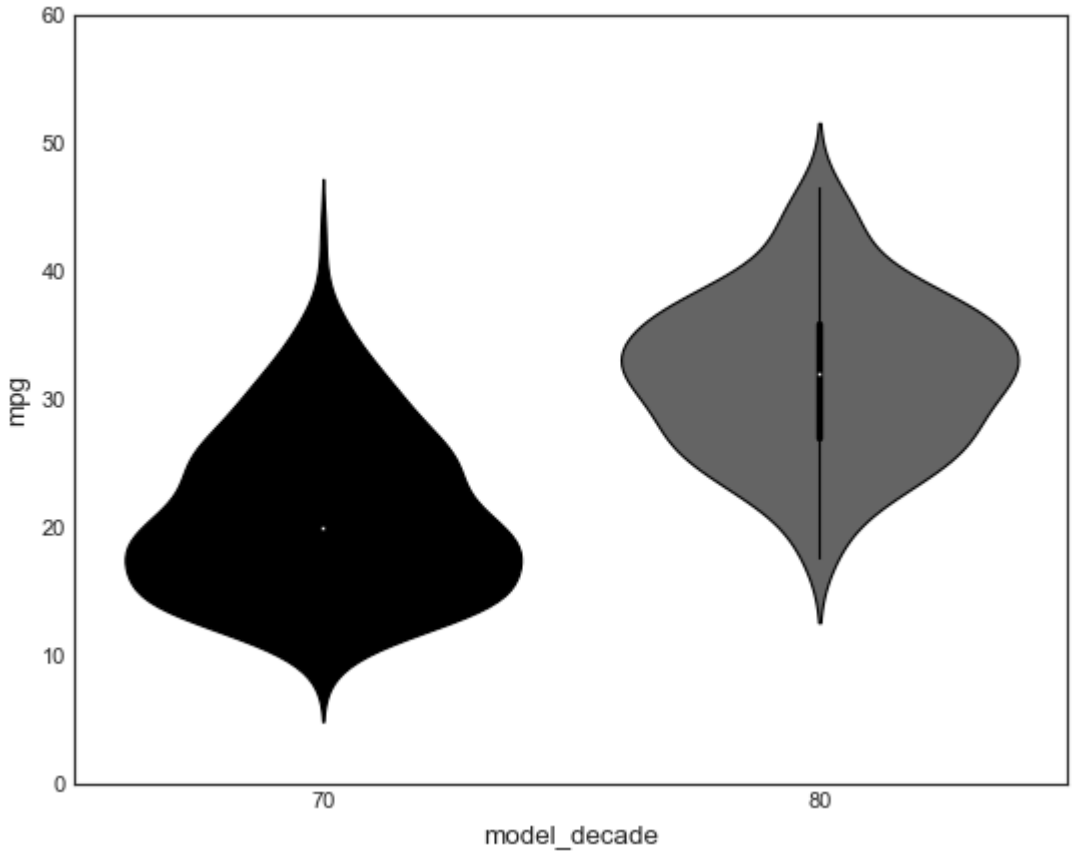
In [228]:
```
mpg_df['model_decade'] = np.floor(mpg_df.model_year/10)*10
mpg_df['model_decade'] = mpg_df['model_decade'].astype(int)

# code for violinplots
# parameter hue is used to group by a specific feature, in this case 'origin',
#while x represents the model year and y represent mileage
sns.violinplot(x='model_decade', y='mpg', data=mpg_df, hue='origin')
```

Out[228]: <AxesSubplot:xlabel='model_decade', ylabel='mpg'>



In [225]:
```
sns.violinplot(x='model_decade', y='mpg', data=mpg_df)
```

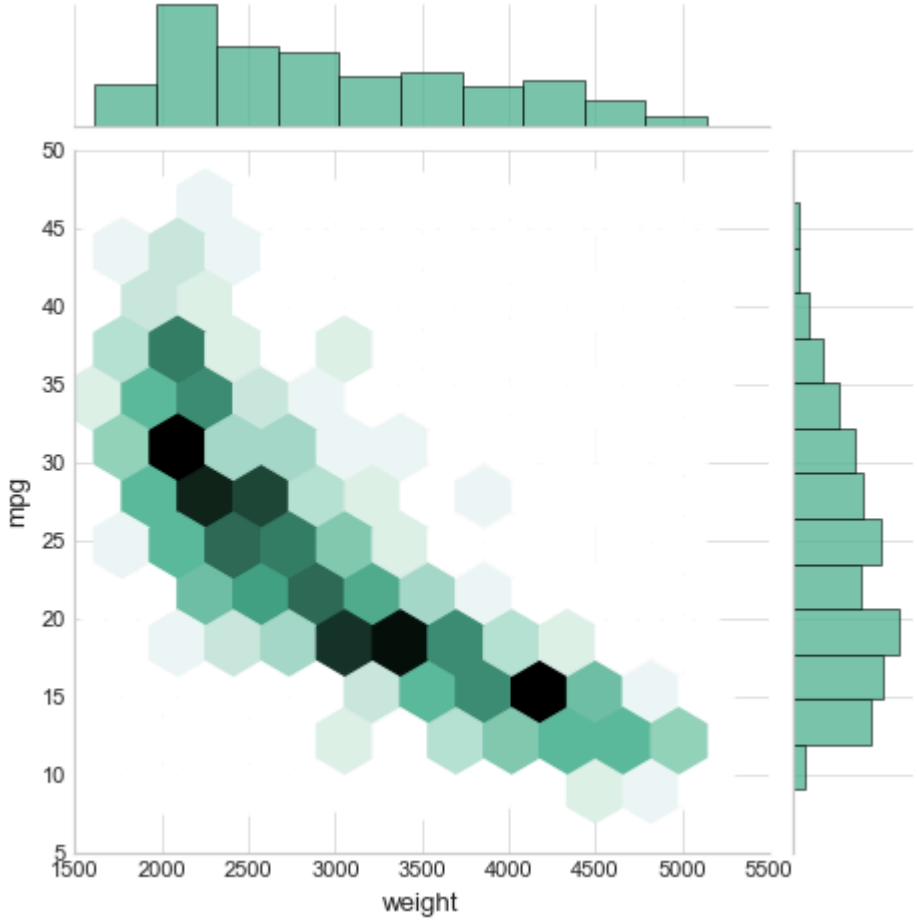
Out[225]:
```
<AxesSubplot:xlabel='model_decade', ylabel='mpg'>
```



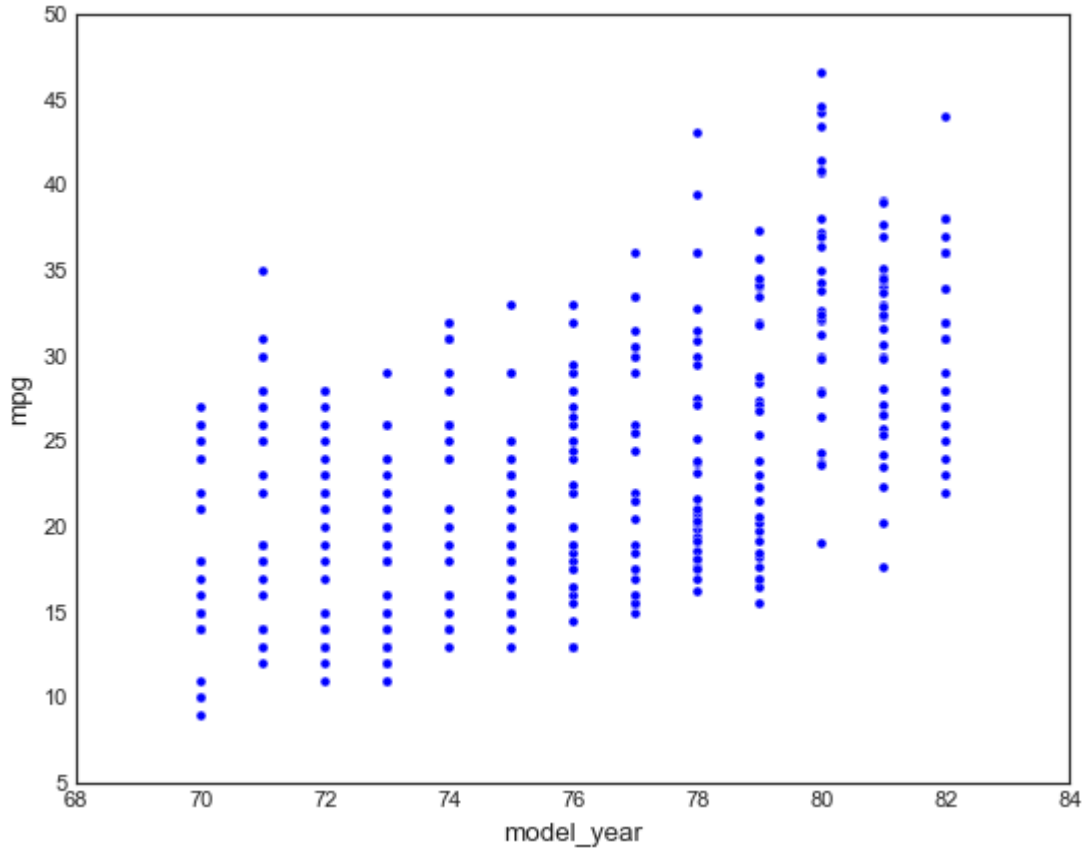
In [216]:
```
sns.jointplot(mpg_df.weight, mpg_df.mpg, kind="hex", color="#4CB391")
#kind : { "scatter" | "kde" | "hist" | "hex" | "scatter" | "resid" }
```

```
C:\Users\tanis\python\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as key
word args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[216]: <seaborn.axisgrid.JointGrid at 0x2066b78b310>

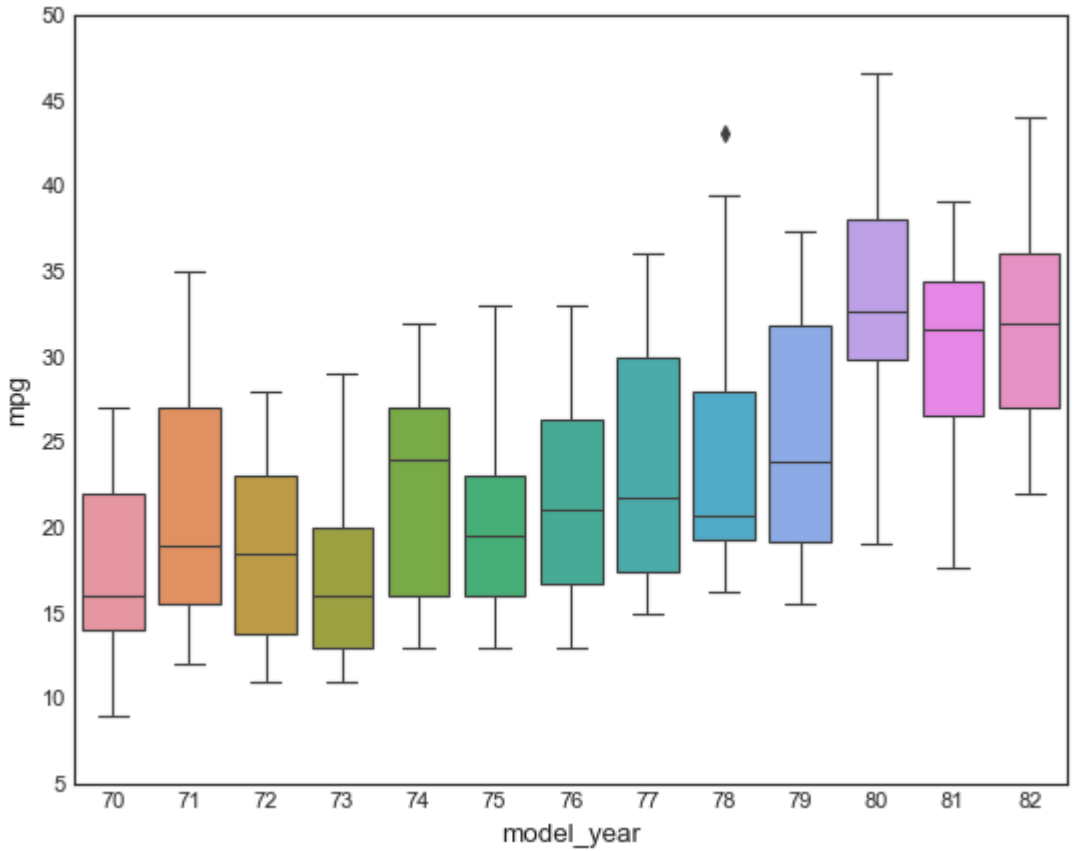


In [217]:
```
sns.set_style("white")
ax1 = sns.scatterplot(x="model_year", y="mpg", data=mpg_df)
```

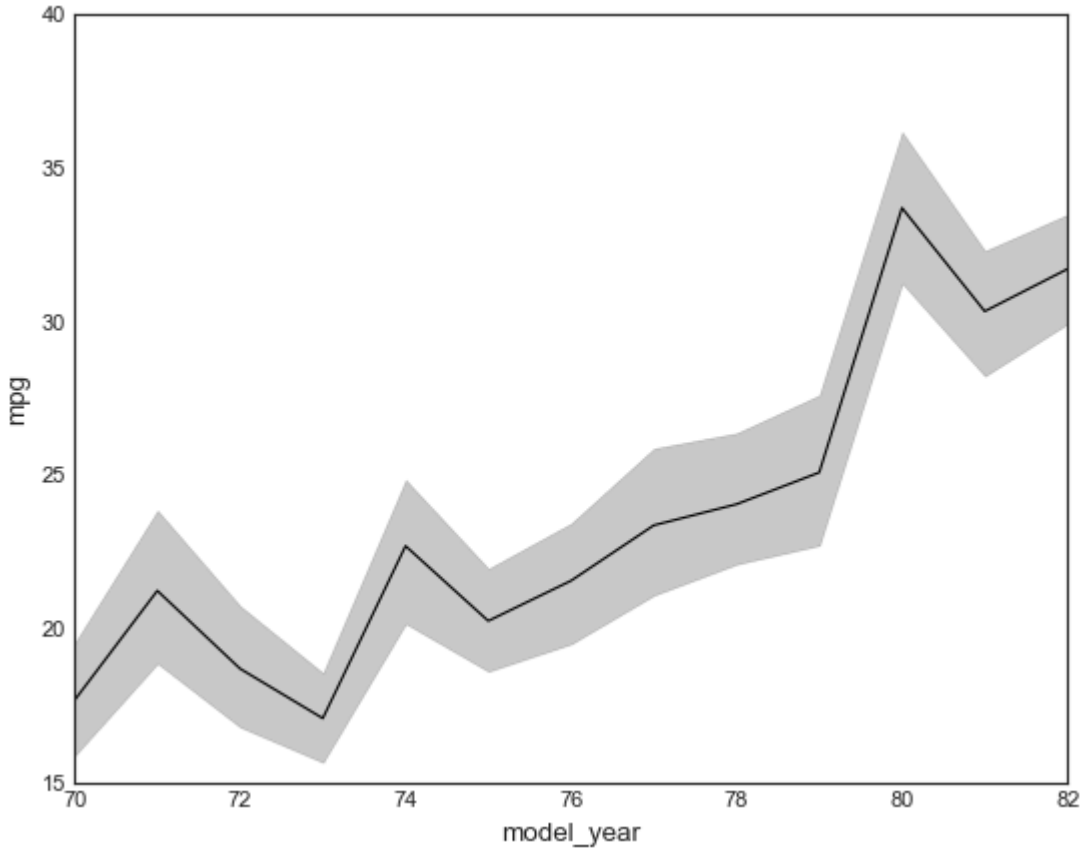


In [223]:
```
sns.boxplot(x='model_year', y='mpg', data=mpg_df)
```

Out[223]:
```
<AxesSubplot:xlabel='model_year', ylabel='mpg'>
```



In [218]:
```
ax2=sns.lineplot(x="model_year",y="mpg",data=mpg_df)
```



In [219]:
```
flights_df=sns.load_dataset("flights")
flights_df.head()
```

Out[219]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

In [221]:
```
pivot_df=flights_df.pivot("month","year","passengers")
pivot_df
```

Out[221]:

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

In [222]:
```
ax4=sns.heatmap(pivot_df)
```