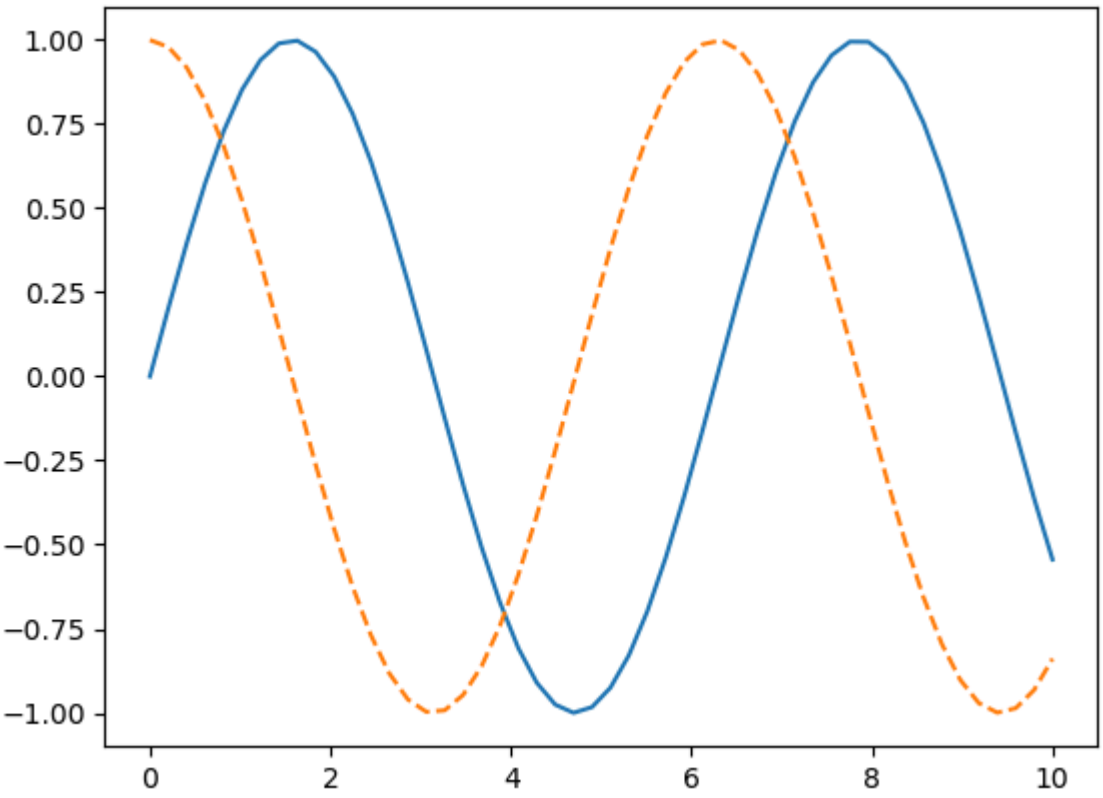
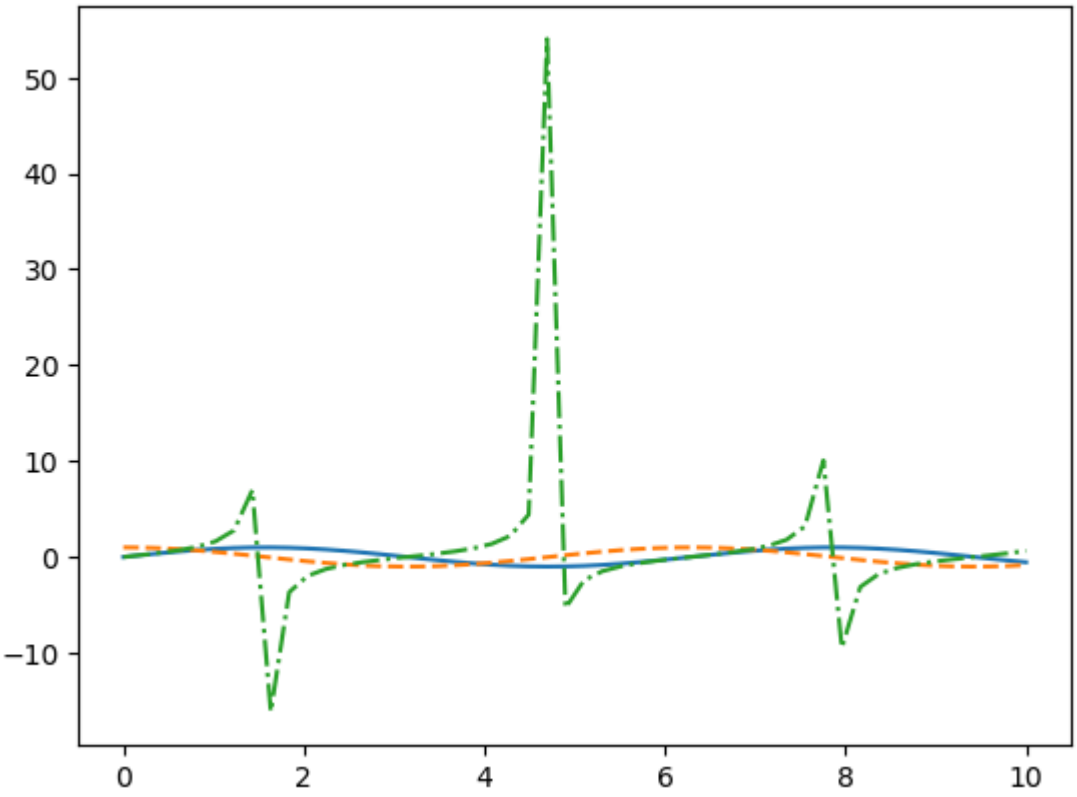
In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
```

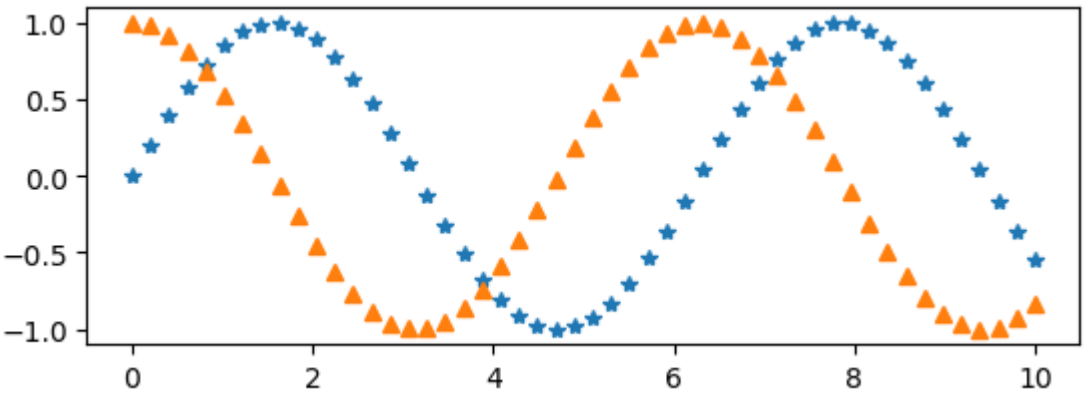
In [3]:
```
%matplotlib inline
x1=np.linspace(0,10,50)
plt.plot(x1,np.sin(x1),'-')
plt.plot(x1,np.cos(x1),'--')
#plt.plot(x1,np.tan(x1),'-.')
plt.show()
```



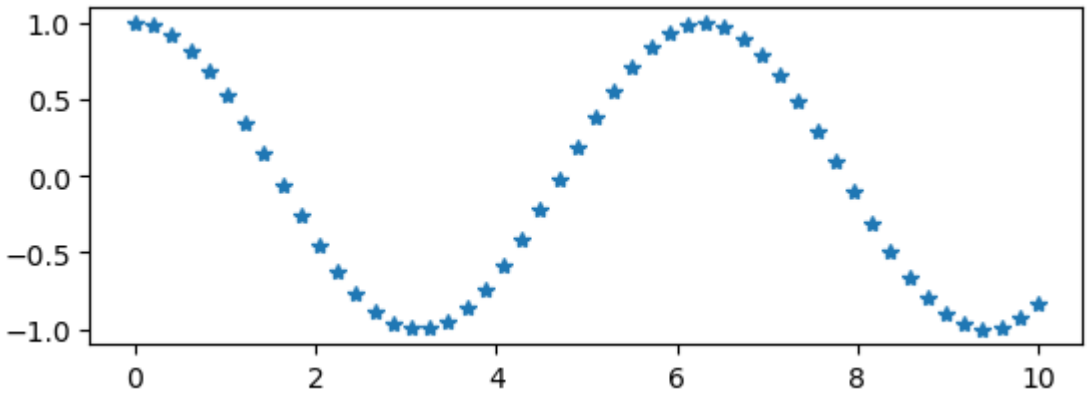
In [4]:
```
%matplotlib inline
x1=np.linspace(0,10,50)
plt.plot(x1,np.sin(x1),'-')
plt.plot(x1,np.cos(x1),'--')
plt.plot(x1,np.tan(x1),'-.')
plt.show()
```



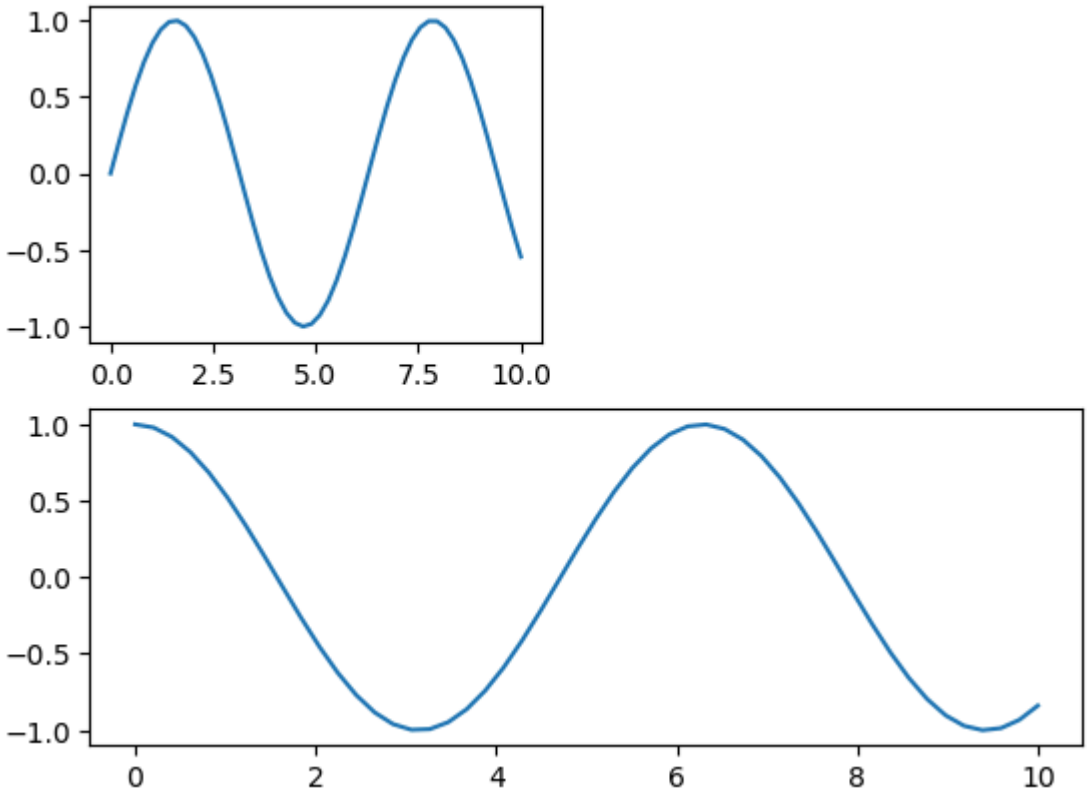
In [5]:
```
# create the first of two panels and set current axis
plt.subplot(2,1,1)  #(rows,columns,panel number)
plt.plot(x1,np.sin(x1),'*')
plt.plot(x1,np.cos(x1),'^')
plt.show()
```



In [6]:
```
plt.subplot(2,1,1)  #(rows,columns,panel number)
plt.plot(x1,np.cos(x1),'*')
plt.show()
```



In [7]:
```
plt.figure()  # create a plot figure


plt.subplot(2,2,1)        #these is the first of two panels and set current axis
# (rows, columns, panel number)
plt.plot(x1,np.sin(x1))
plt.subplot(2,1,2)        #these is the second of two panels and set current axi
# (rows, columns, panel number)
plt.plot(x1,np.cos(x1))
plt.show()
```



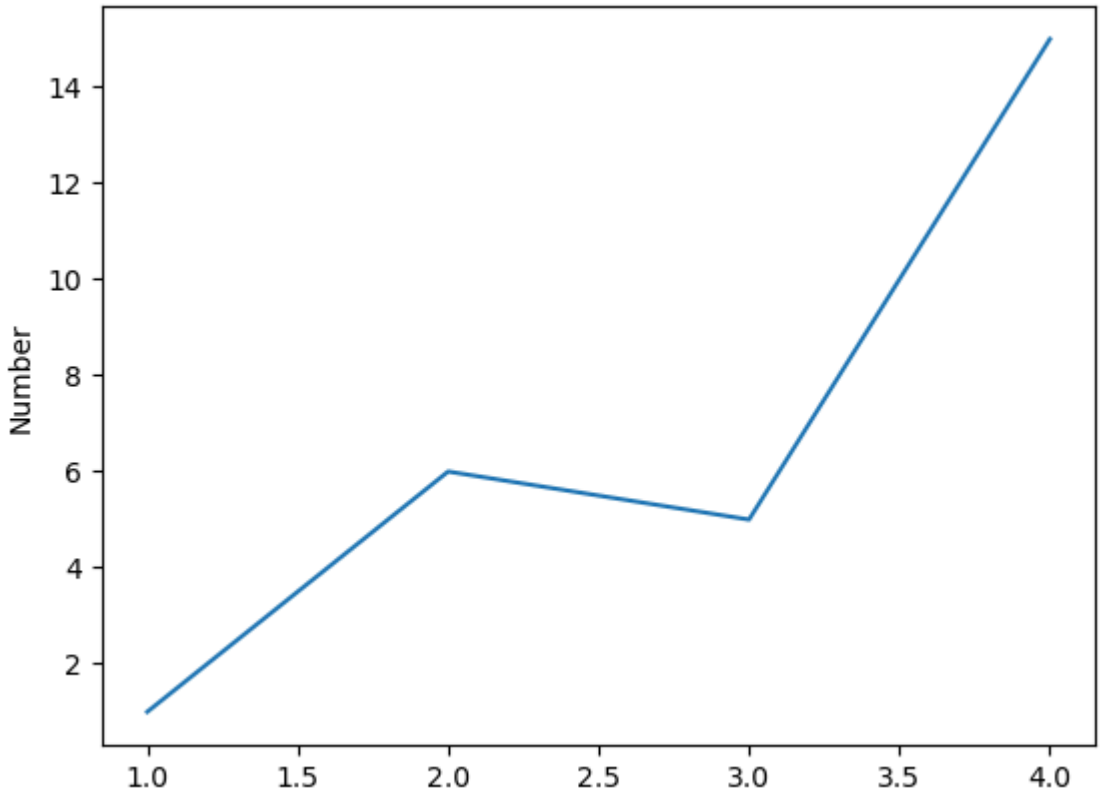
In [8]:
```
print(plt.gcf()) # here the plt.gcf()-it is used for current fig information
```

```
Figure(640x480)
```

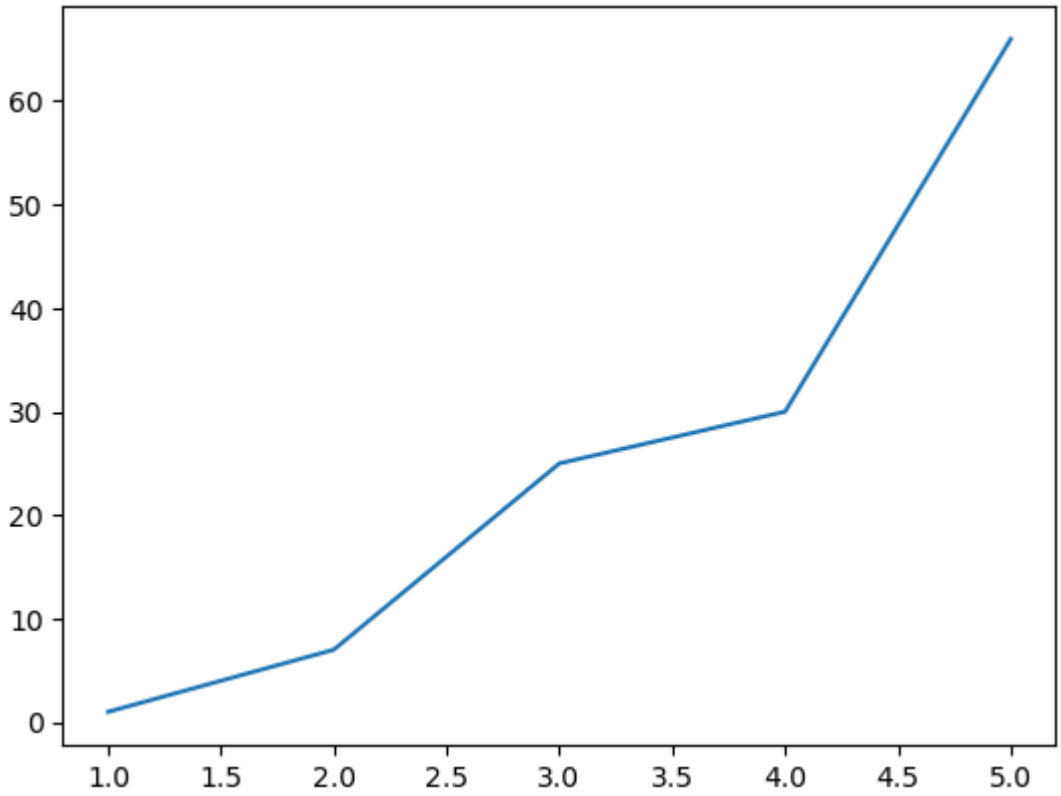
In [9]:
```
print(plt.gca()) #here the plt.gca()- it is used for get current axis infor
```

```
Axes(0.125,0.11;0.775x0.77)
```

In [10]:
```
plt.plot([1,2,3,4],[1,6,5,15])   # here we are given the values to print the gra
plt.ylabel('Number')
plt.show()
```

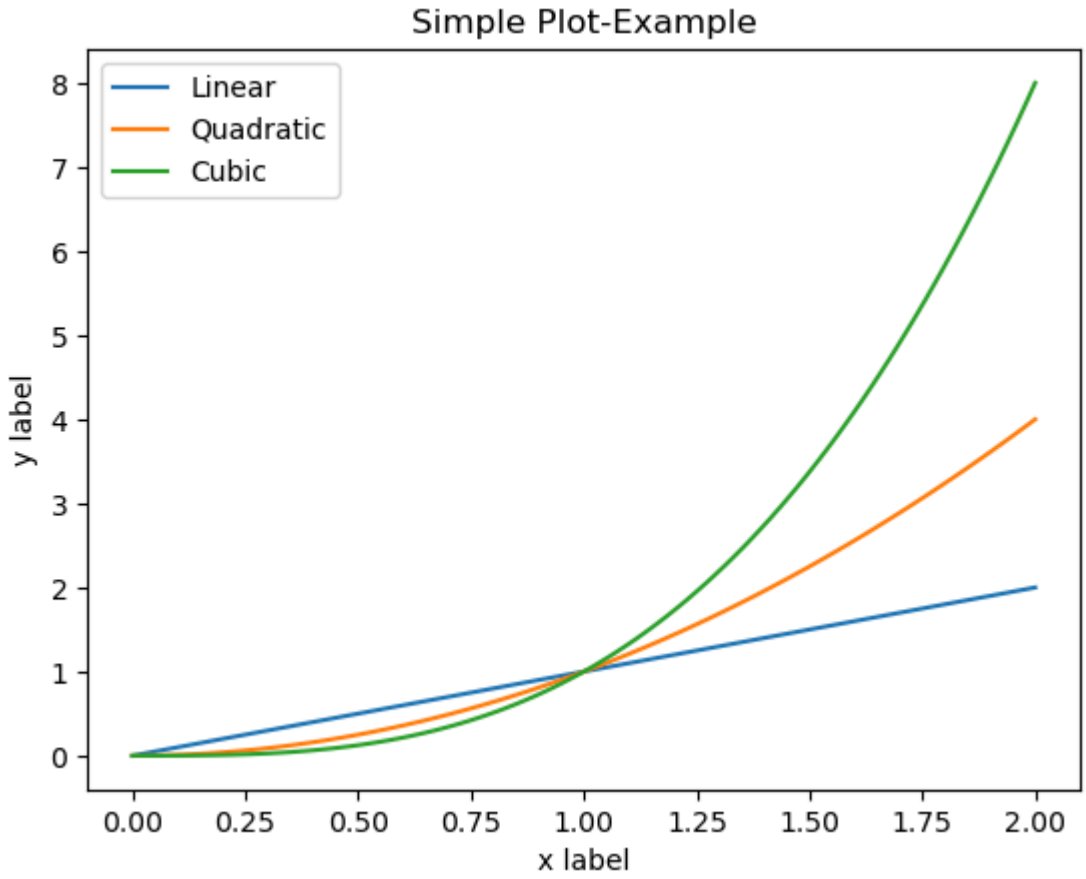


In [11]:
```
import matplotlib.pyplot as plt    # here it is the Verstile command(plot())
plt.plot([1,2,3,4,5],[1,7,25,30,66])
plt.show()
```

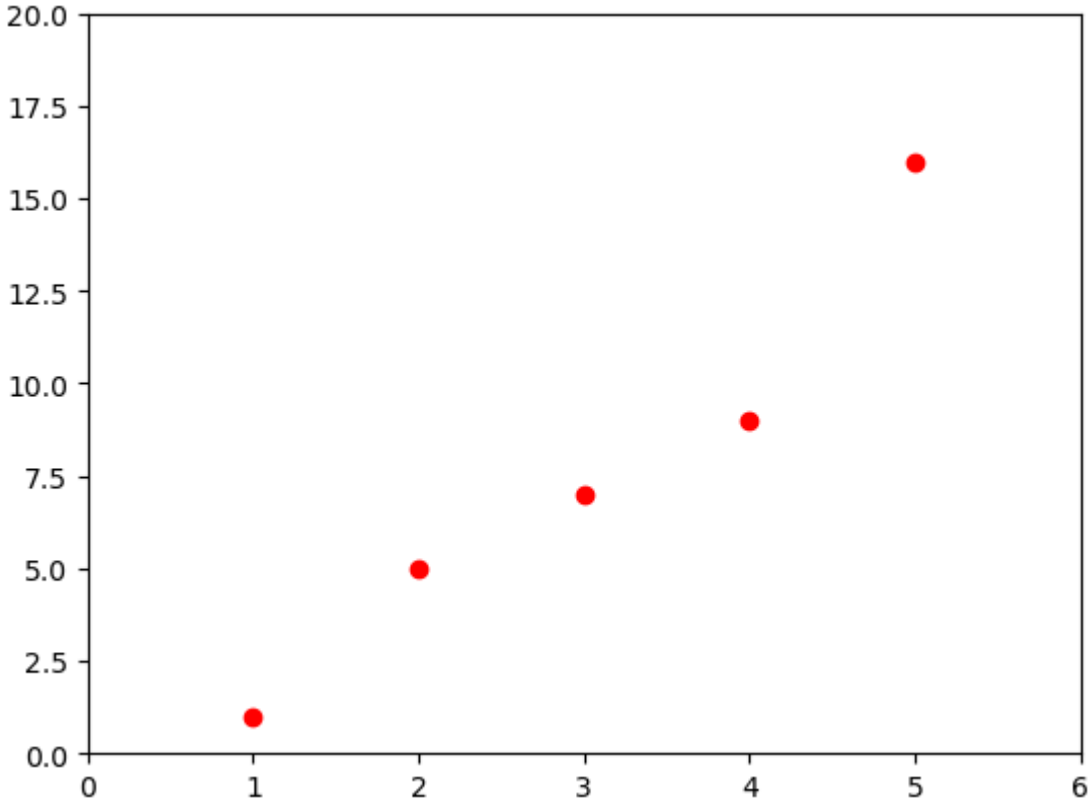


In [12]:
```
# State-Machine interface
x=np.linspace(0,2,100)
plt.plot(x,x,label='Linear')
plt.plot(x,x**2,label='Quadratic')
plt.plot(x,x**3,label='Cubic')
```

```
plt.xlabel('x label')
plt.ylabel('y label')

plt.title("Simple Plot-Example")  # this is the title of the graph

plt.legend()   #here it will print the box which content for ex:liner, quadraic

plt.show()
```

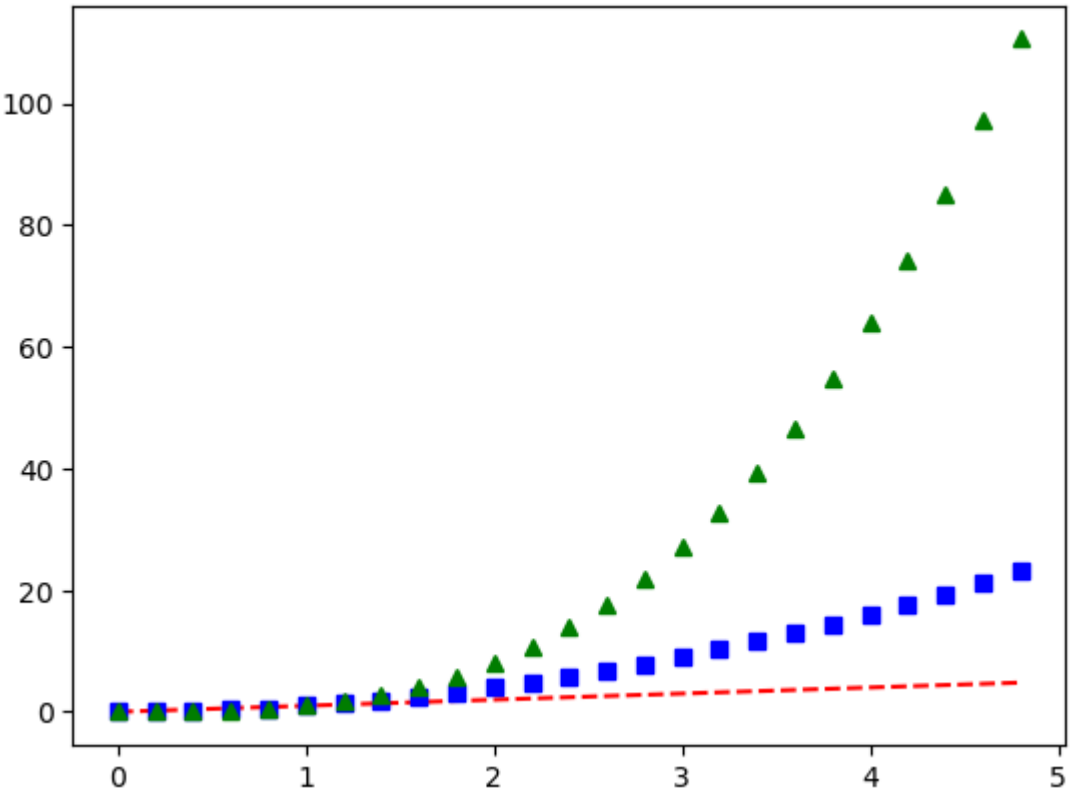


In [13]:
```
plt.plot([1,2,3,4,5],[1,5,7,9,16],'ro')
plt.axis([0,6,0,20])
plt.show()
```



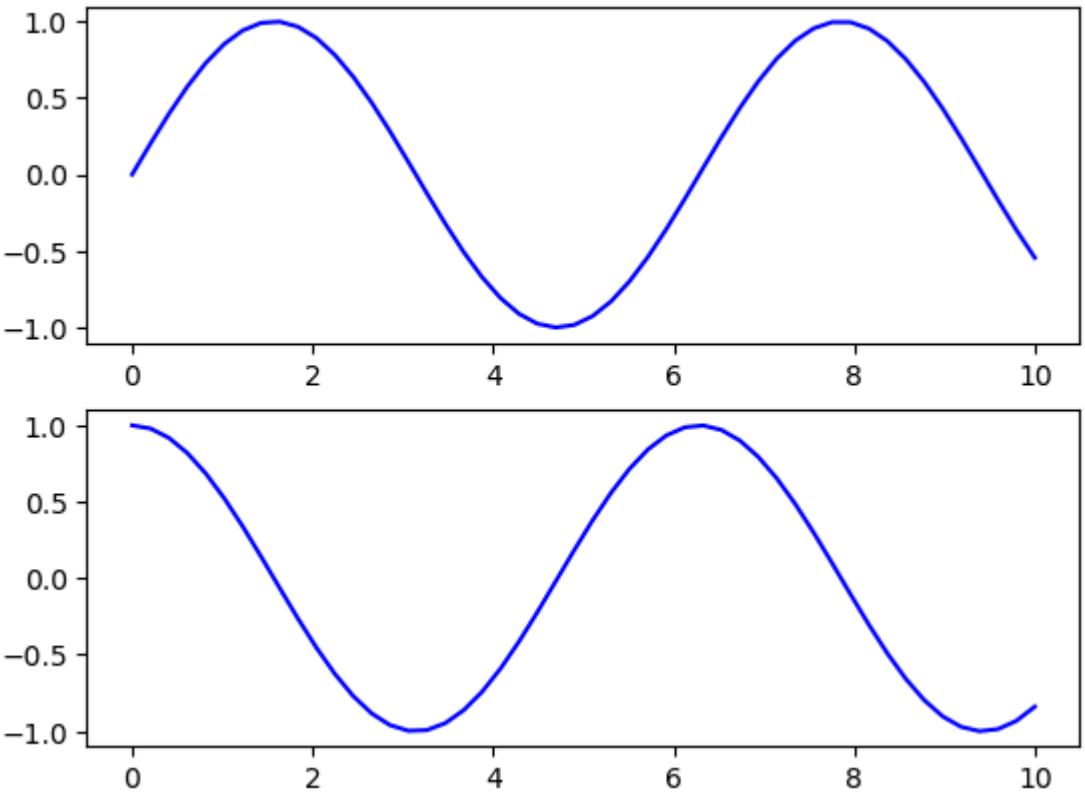
Working with NUMPY Arrays

In [14]:
```
t=np.arange(0.,5.,0.2)  #evenly Sampled time at 200ms  intervels
plt.plot(t,t,'r--',t,t**2,'bs',t,t**3,'g^')
plt.show()
```



In [15]:
```
# First create a grid of plots
# ax will be an array of two Axes objects
import matplotlib.pyplot as plt
```

```
import numpy as np
fig, ax = plt.subplots(2)
x1=np.linspace(0,10,50)


# Call plot() method on the appropriate object
ax[0].plot(x1, np.sin(x1), 'b-')
ax[1].plot(x1, np.cos(x1), 'b-');
plt.show()
```



In [16]:
```
fig =plt.figure()

x2=np.linspace(0, 5 , 10)
y2=x2 ** 2

axes =fig.add_axes([0.1,0.1,0.8,0.8])

axes.plot(x2, y2, 'r')

axes.set_xlabel('x2')
axes.set_ylabel('y2')
axes.set_title('title');
```

Figures and Axes

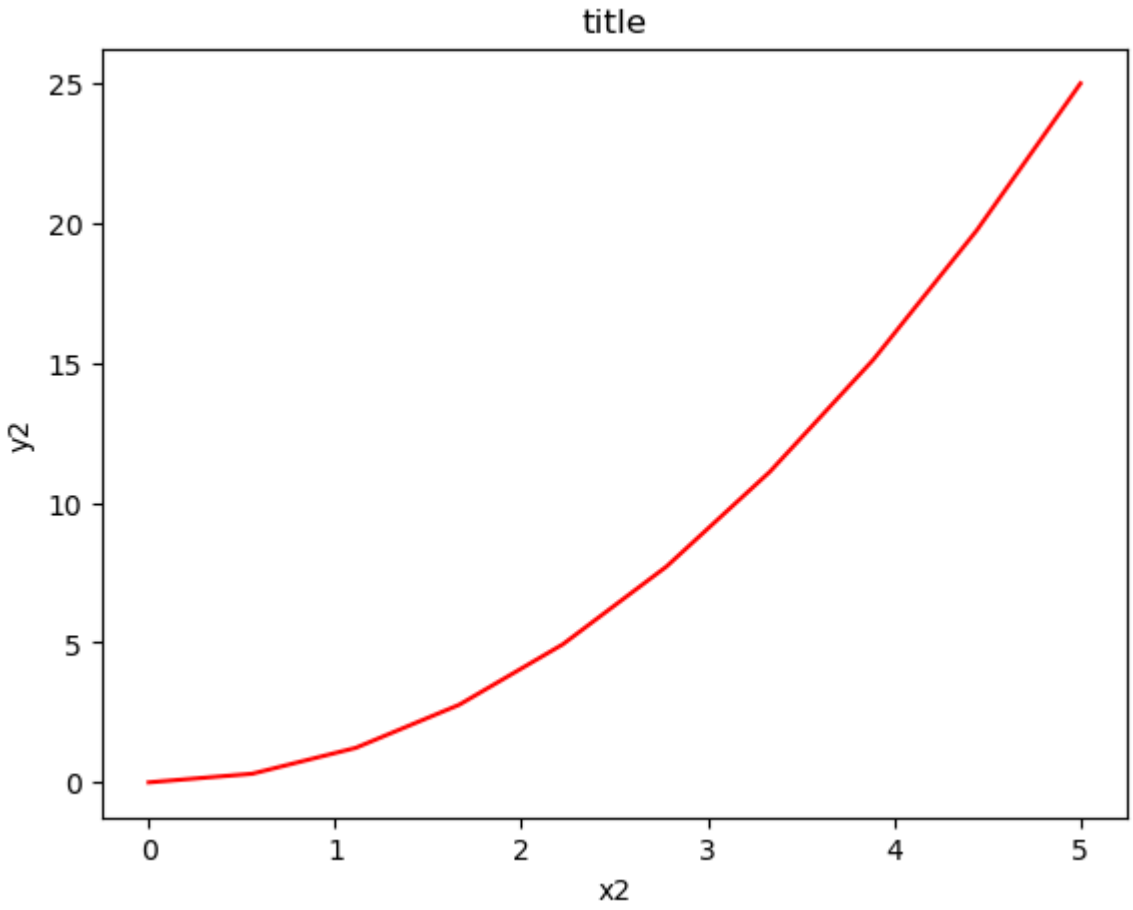
In [17]:
```
fig=plt.figure()
```

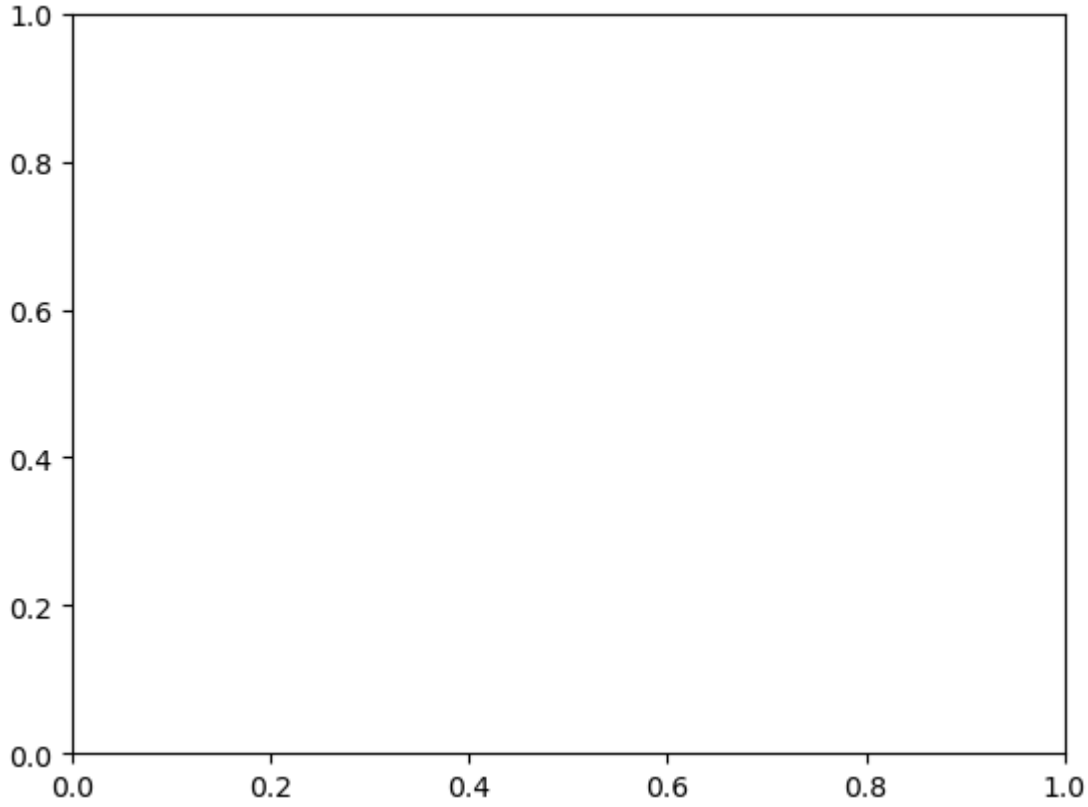
In [18]:
```
ax=plt.axes()
```

In [19]:
```
fig=plt.figure()
```

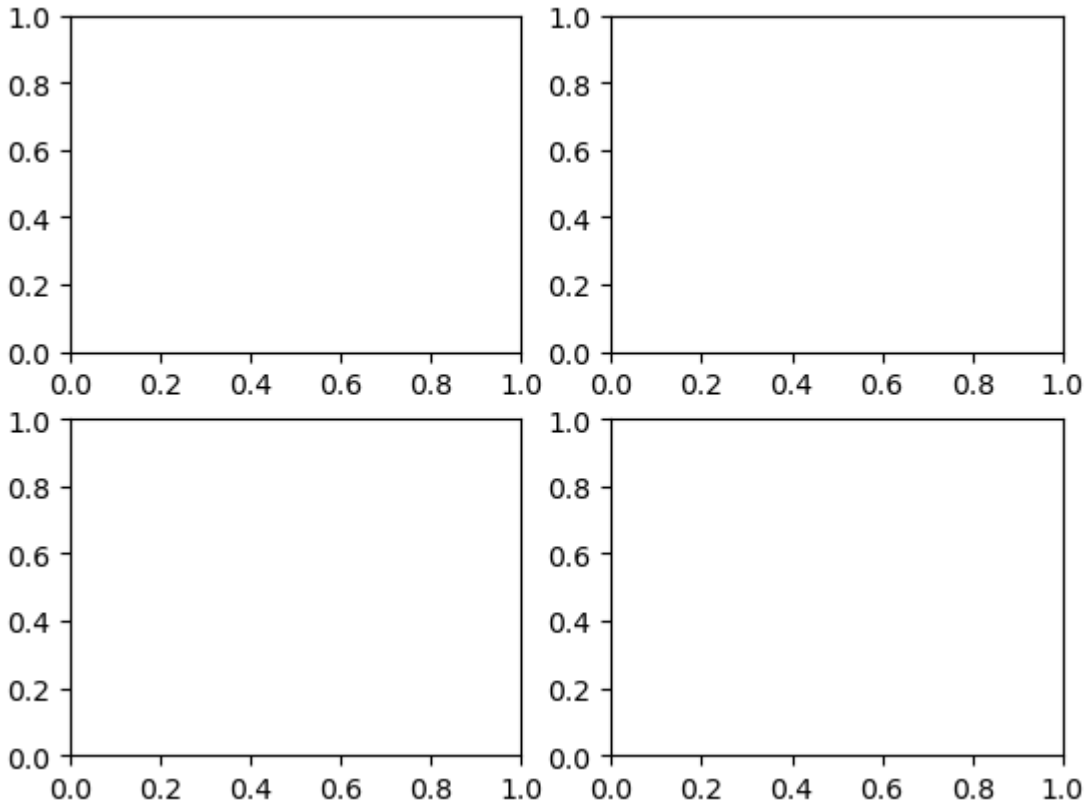
In [20]:
```
fig=plt.figure()
ax1=fig.add_subplot(2,2,1)
```

```
ax2=fig.add_subplot(2,2,2)
ax3=fig.add_subplot(2,2,3)
ax4=fig.add_subplot(2,2,4)
plt.show()
```



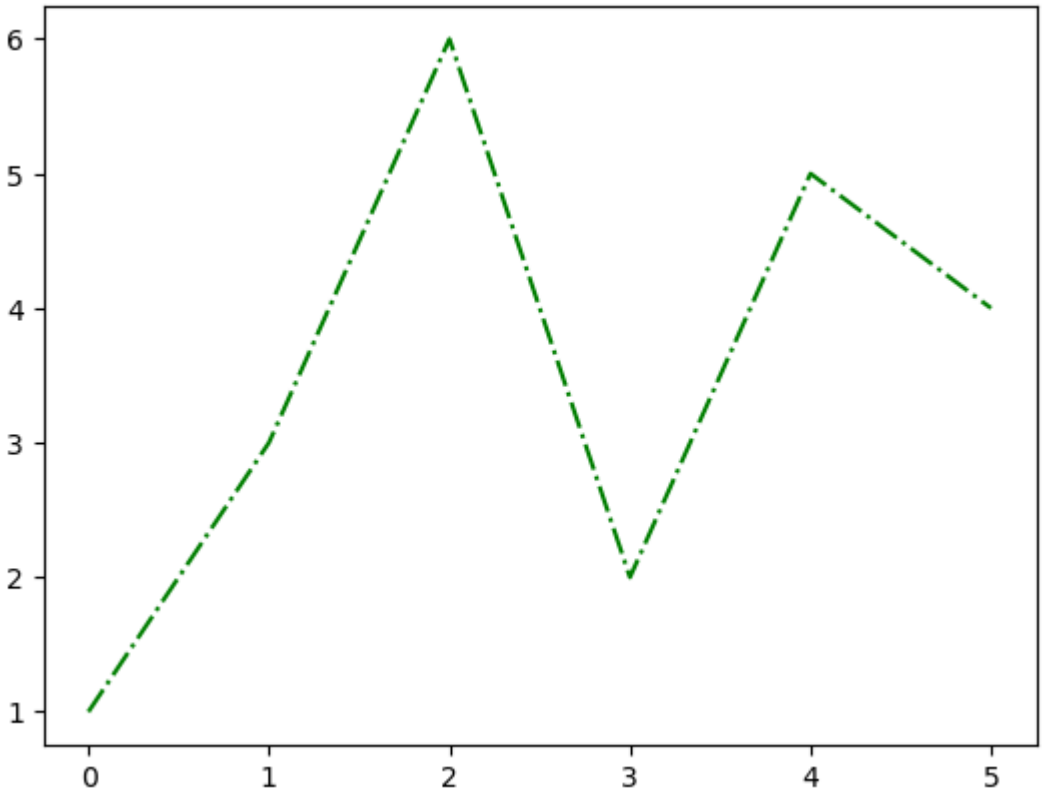



```
<Figure size 640x480 with 0 Axes>
```

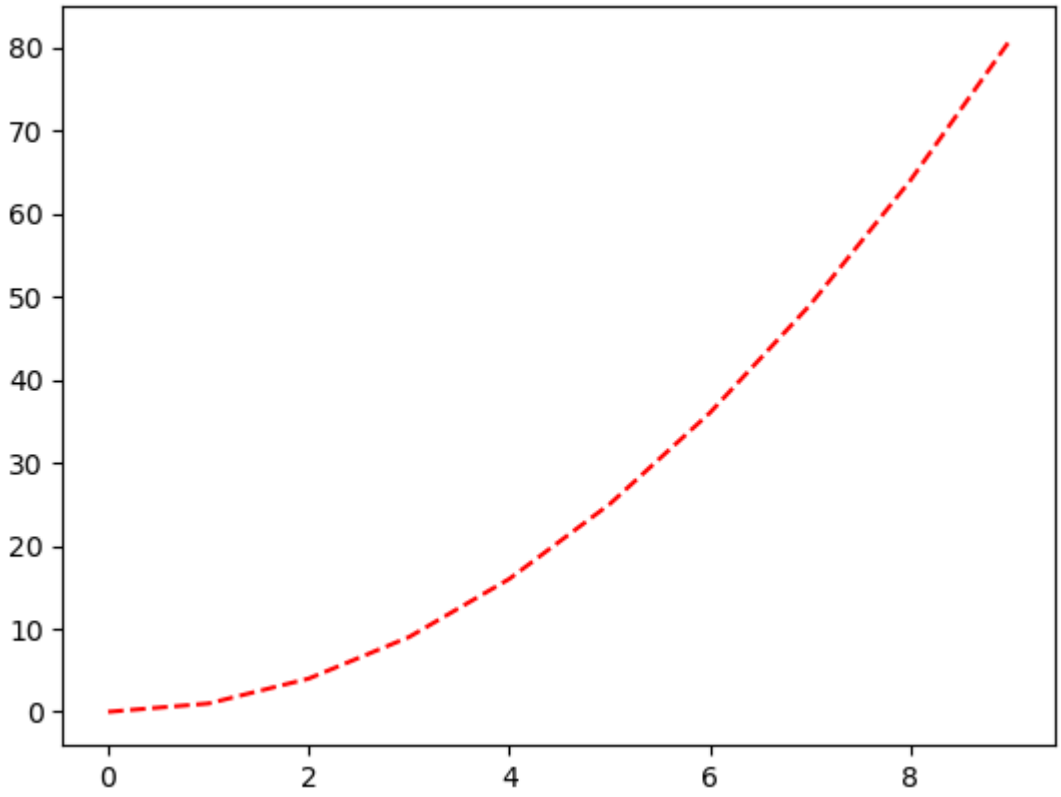


FIRST plot with MATPLOTLIB

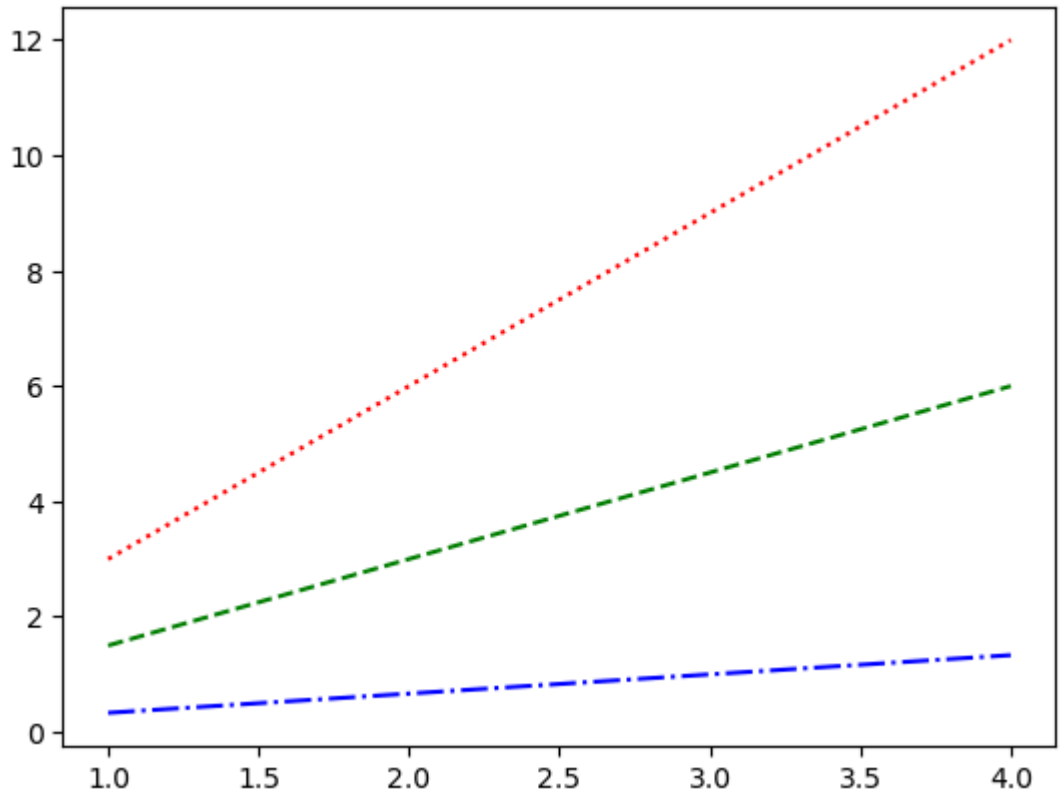
In [21]:
```
plt.plot([1,3,6,2,5,4],'g-.')
plt.show()
```



In [22]:
```
x3=range(10)
plt.plot(x3,[xi**2 for xi in x3],'r--')
plt.show()
```



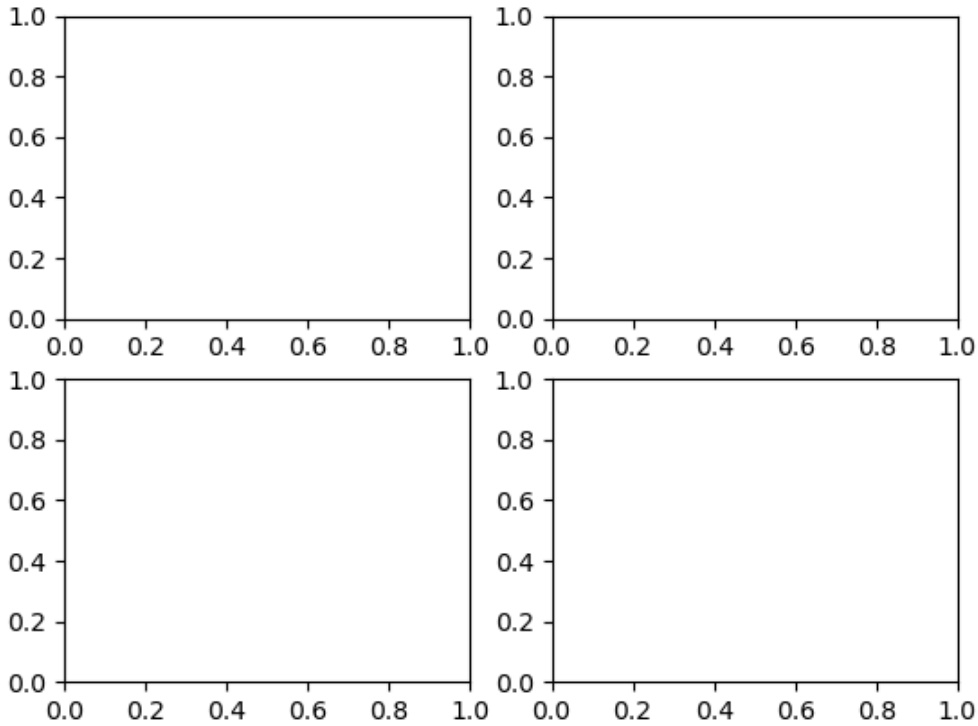
In [23]:
```
x4=range(1,5)
plt.plot(x4,[xi*1.5 for xi in x4],'g--')
plt.plot(x4,[xi*3 for xi in x4],'r:')
plt.plot(x4,[xi/3.0 for xi in x4],'b-.')
plt.show()
```



In [24]:
```
fig.savefig('fig1.png')
```

In [25]:
```
from IPython.display import Image
Image('fig1.png')
```

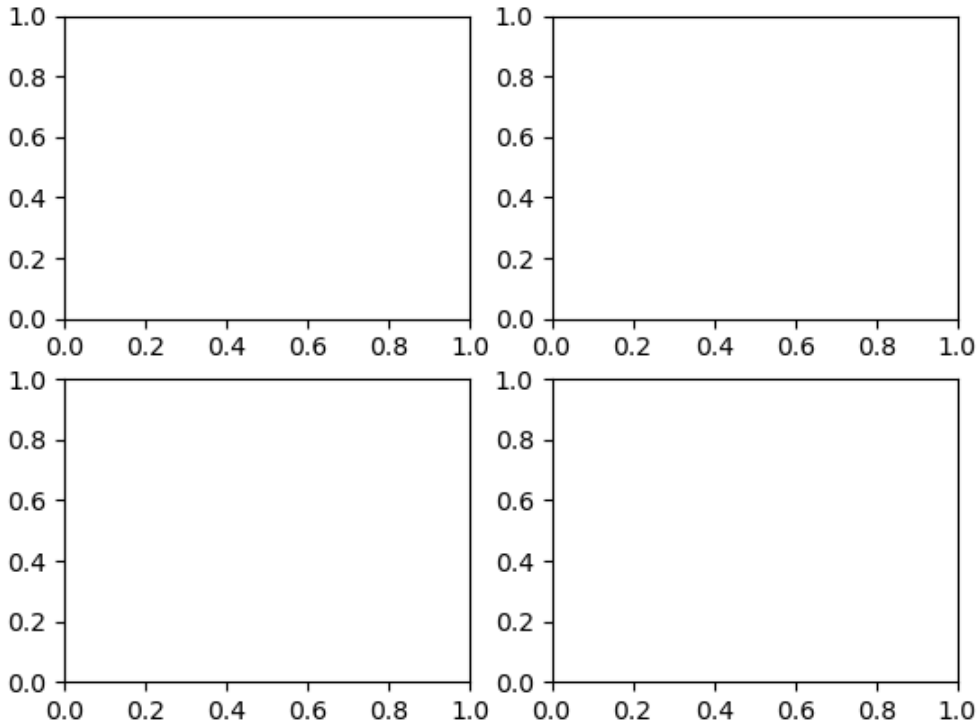
Out[25]:



In [26]:
```
# Saving the figure

fig.savefig('plot1.png')
```

In [27]:
```
# Explore the contents of figure

from IPython.display import Image

Image('plot1.png')
```

Out[27]:

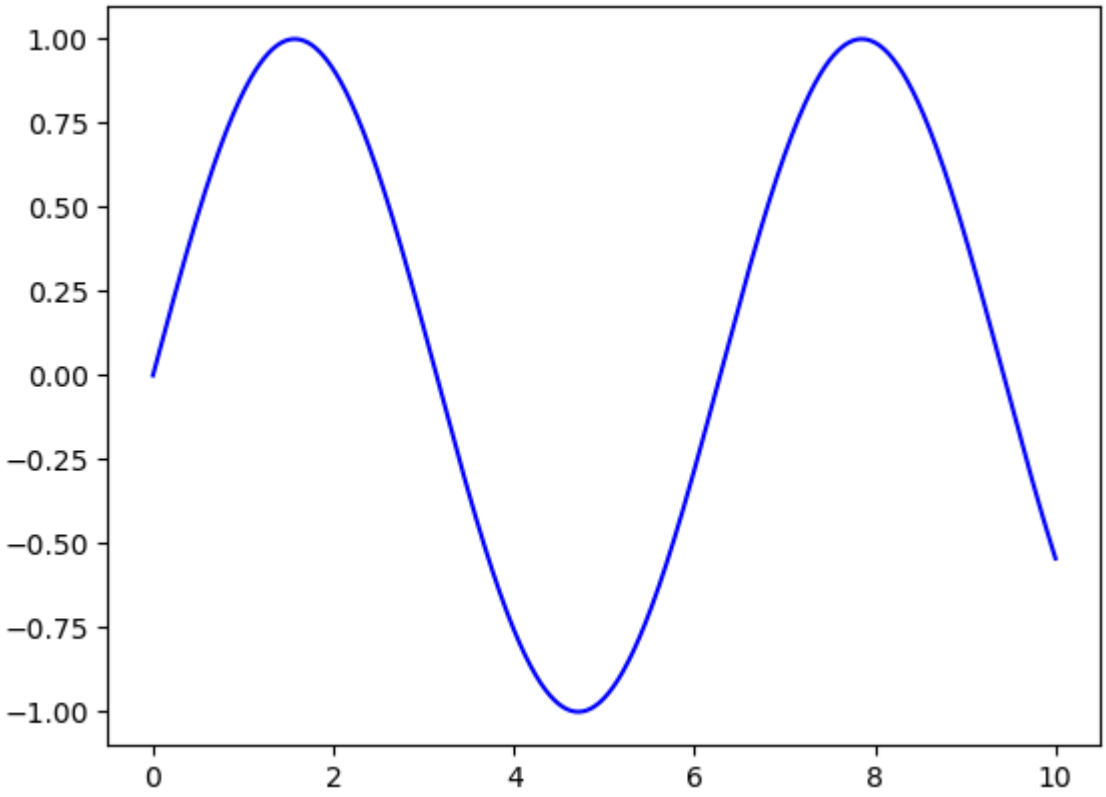


In [28]:
```
fig.canvas.get_supported_filetypes()
```

Out[28]:
```
{'eps': 'Encapsulated Postscript',
 'jpg': 'Joint Photographic Experts Group',
 'jpeg': 'Joint Photographic Experts Group',
 'pdf': 'Portable Document Format',
 'pgf': 'PGF code for LaTeX',
 'png': 'Portable Network Graphics',
 'ps': 'Postscript',
 'raw': 'Raw RGBA bitmap',
 'rgba': 'Raw RGBA bitmap',
 'svg': 'Scalable Vector Graphics',
 'svgz': 'Scalable Vector Graphics',
 'tif': 'Tagged Image File Format',
 'tiff': 'Tagged Image File Format',
 'webp': 'WebP Image Format'}
```

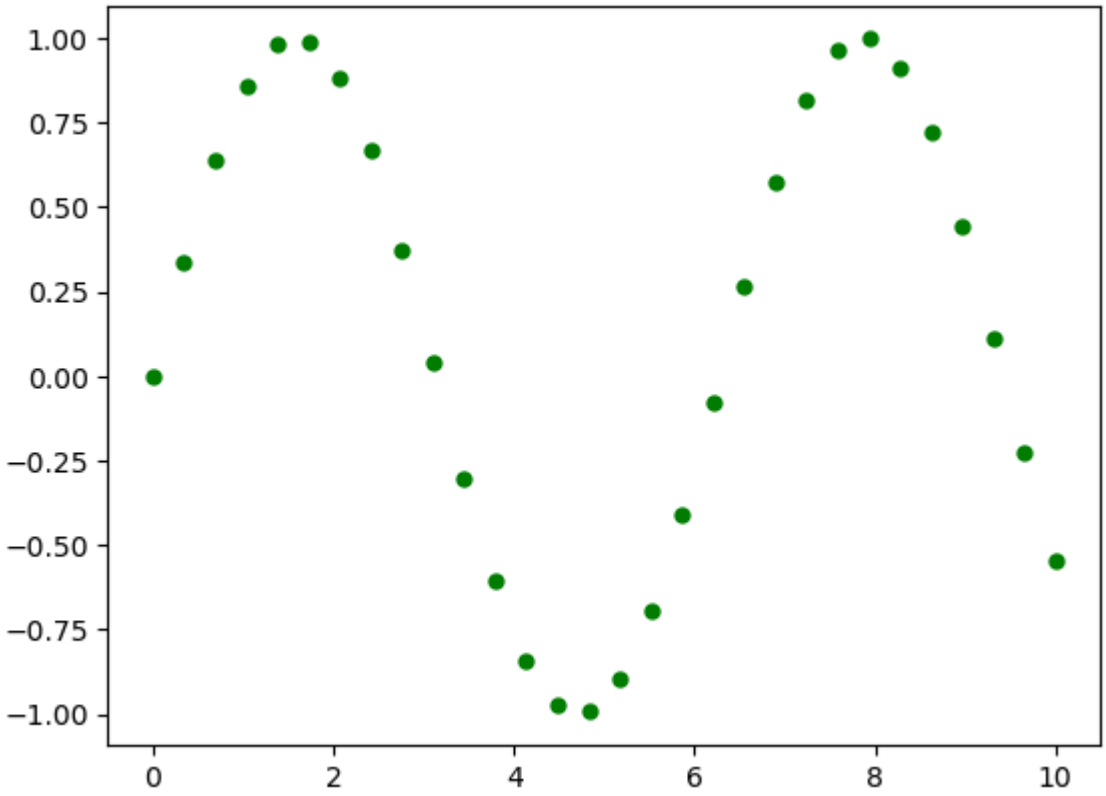
Line Plot Graph

In [29]:
```
# Create figure and axes first
fig = plt.figure()

ax = plt.axes()

# Declare a variable x5
x5 = np.linspace(0, 10, 1000)


# Plot the sinusoid function
ax.plot(x5, np.sin(x5), 'b-')
plt.show()
```



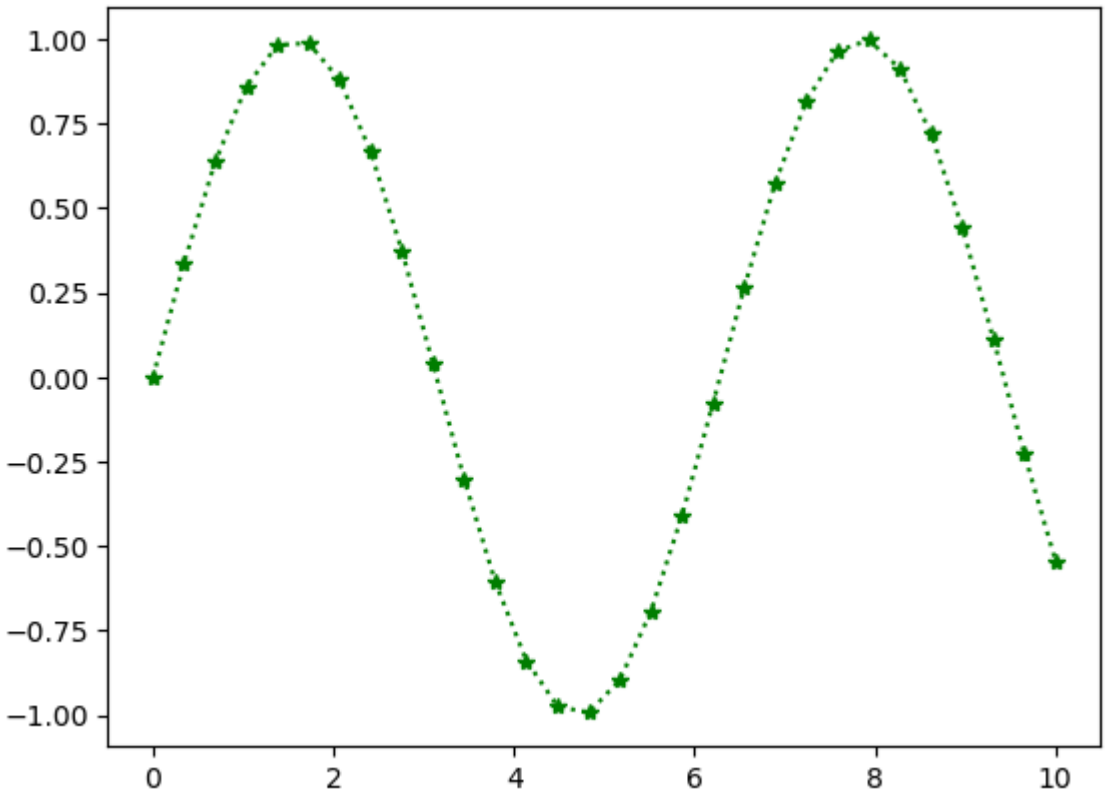
Scatter Plot Graph

In [30]:
```
x7=np.linspace(0,10,30)
y7=np.sin(x7)
plt.plot(x7,y7,'o',color='g',ms=5)
plt.show()
```



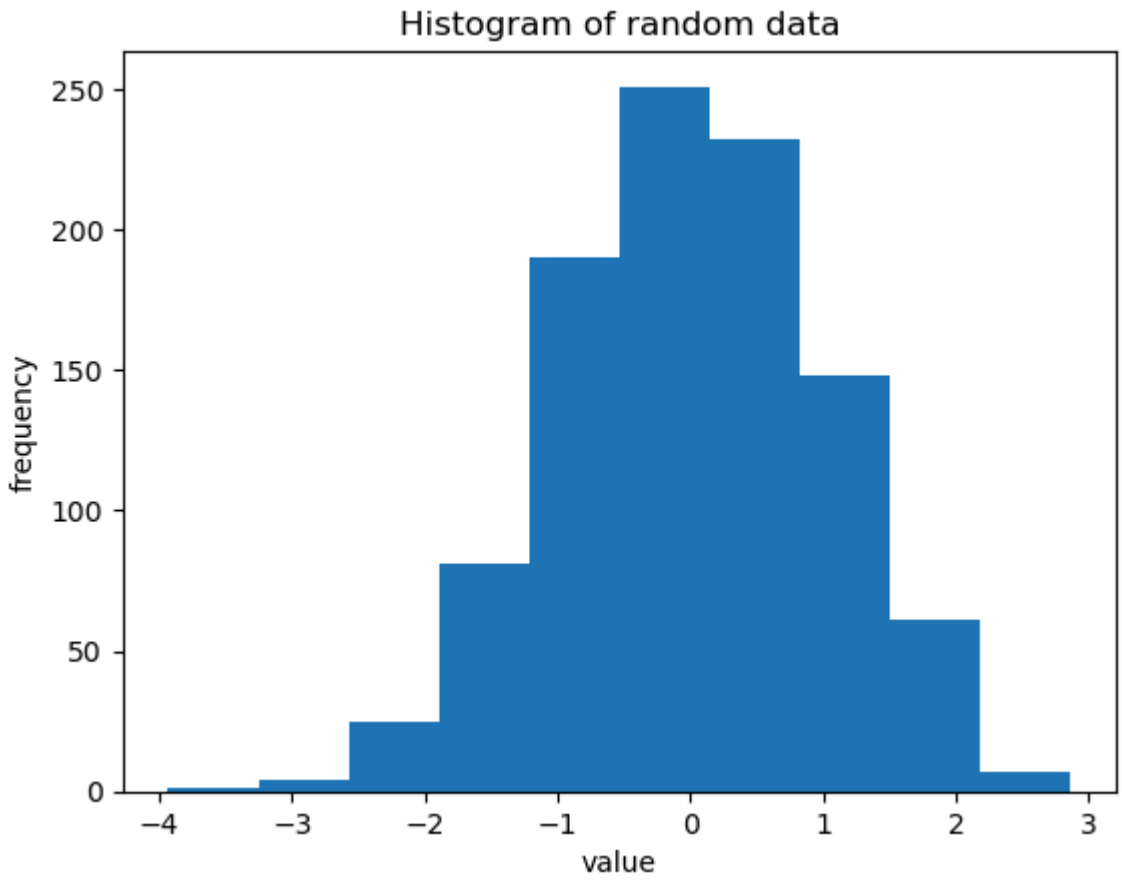
In [31]:
```
x7=np.linspace(0,10,30)   # 30 x-values from 0 to 10
y7=np.sin(x7)               # Calculate sine for each x
```

```
plt.plot(x7,y7,'*:',color='g') # Plot points as green star
plt.show()
```



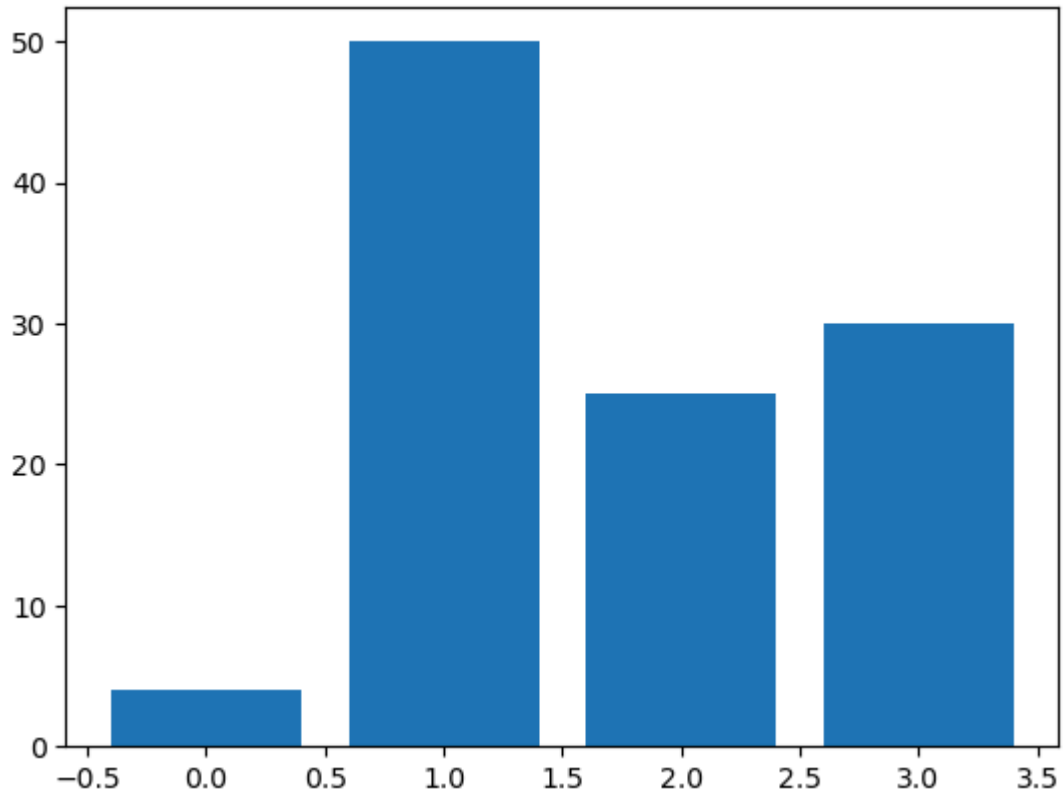
Histogram plot Graph

In [32]:
```
data1= np.random.randn(1000)   # this line generates 1000 random numbers from no
plt.hist(data1)
plt.title('Histogram of random data')
plt.xlabel("value")
plt.ylabel("frequency")
plt.show()
```



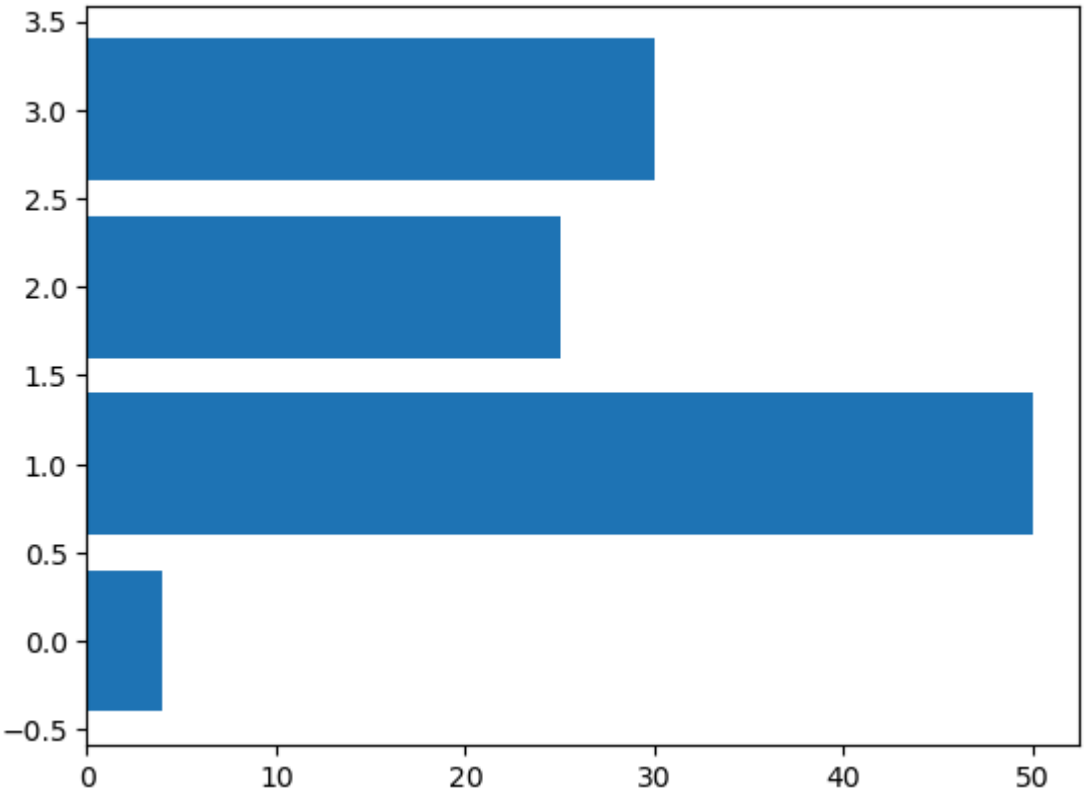
Vertical Bar Chart

In [33]:
```
data2=[4.,50.,25,30]      # here we give th list  values of height of the bar
plt.bar(range(len(data2)),data2)  # here bar position heightof the bar (x,height
plt.show()                # here x= bar position (0,1,2,3,4) and height=data2
```



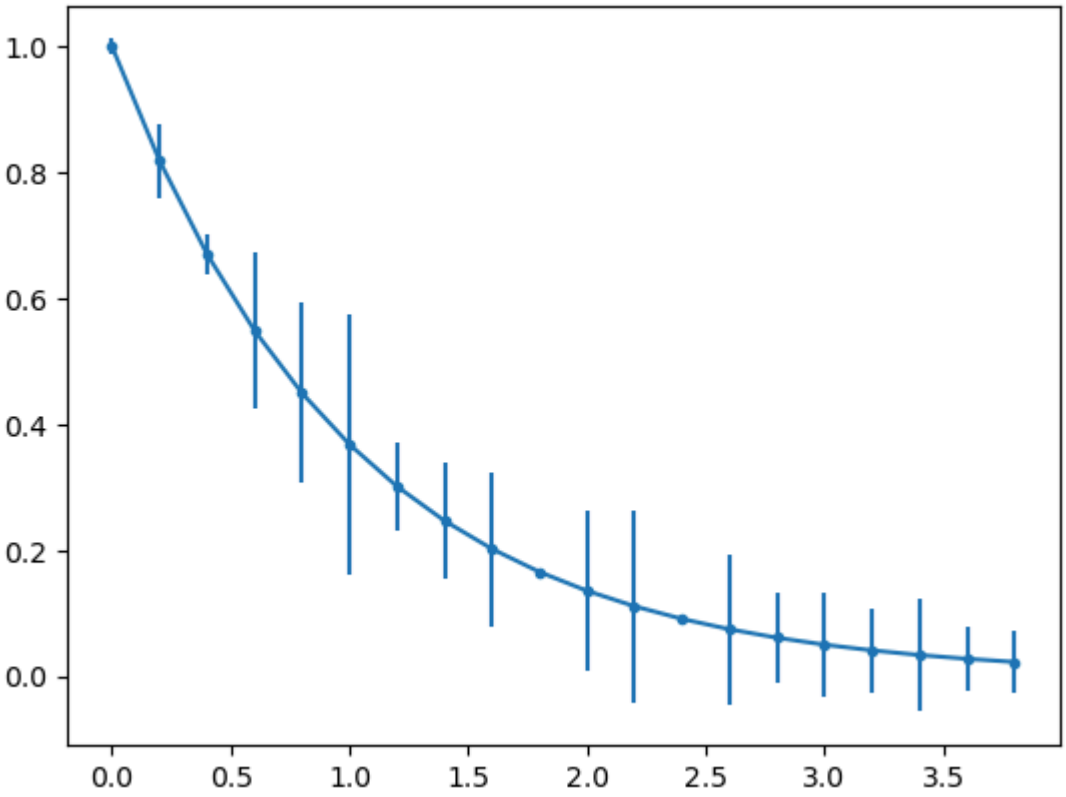
Horizontal Bar Chart

In [34]:
```
plt.barh(range(len(data2)),data2)
plt.show()
```



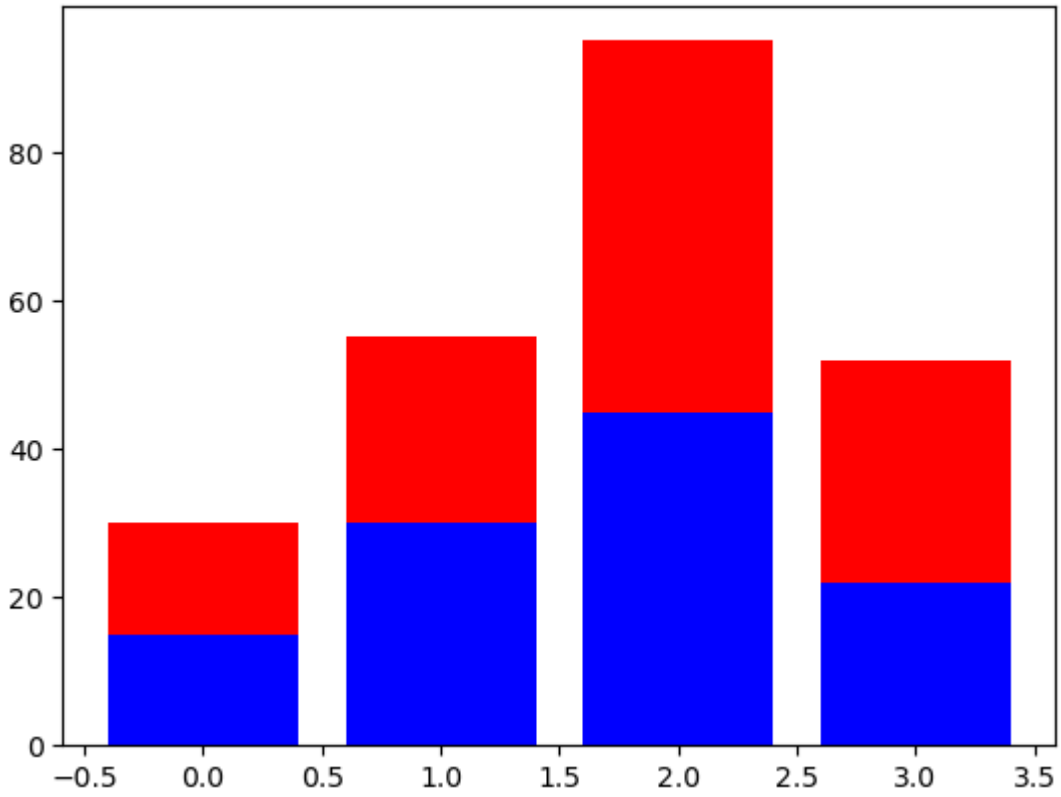
Error Bar Chart

In [35]:
```
x9=np.arange(0,4,0.2)
y9=np.exp(-x9)
e1=0.1*np.abs(np.random.randn(len(y9)))

plt.errorbar(x9,y9,yerr=e1,fmt='.-')
plt.show()
```

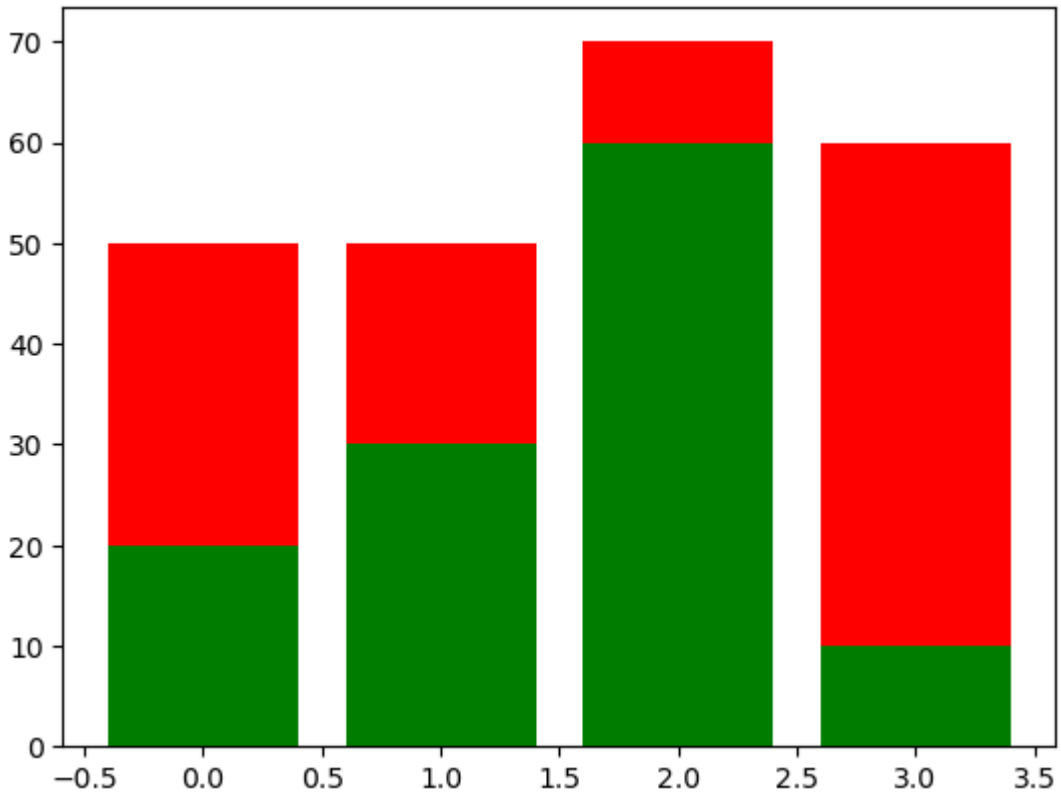


Stacked Bar Chart

In [36]:
```
#two sets bars Stacked on the of each other to compare
a=[15.,30.,45.,22.]
b=[15.,25.,50.,30.]
z2=range(4)


plt.bar(z2,a,color='b')
plt.bar(z2,b,color='r',bottom=a)

plt.show()
```

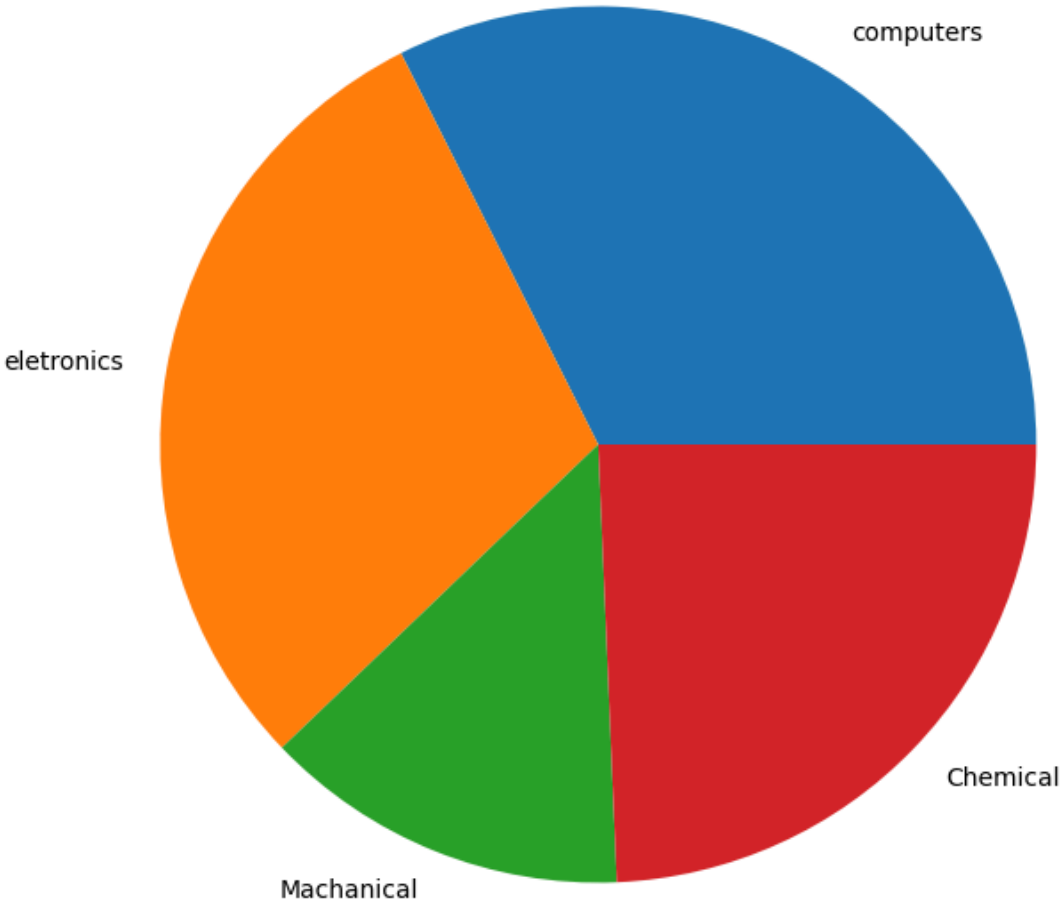


In [37]:
```
A=[20,30,60,10]
B=[30,20,10,50]
Z2=[0,1,2,3]
plt.bar(Z2,A,color='g')
plt.bar(Z2,B,color='r',bottom=A)
plt.show()
```



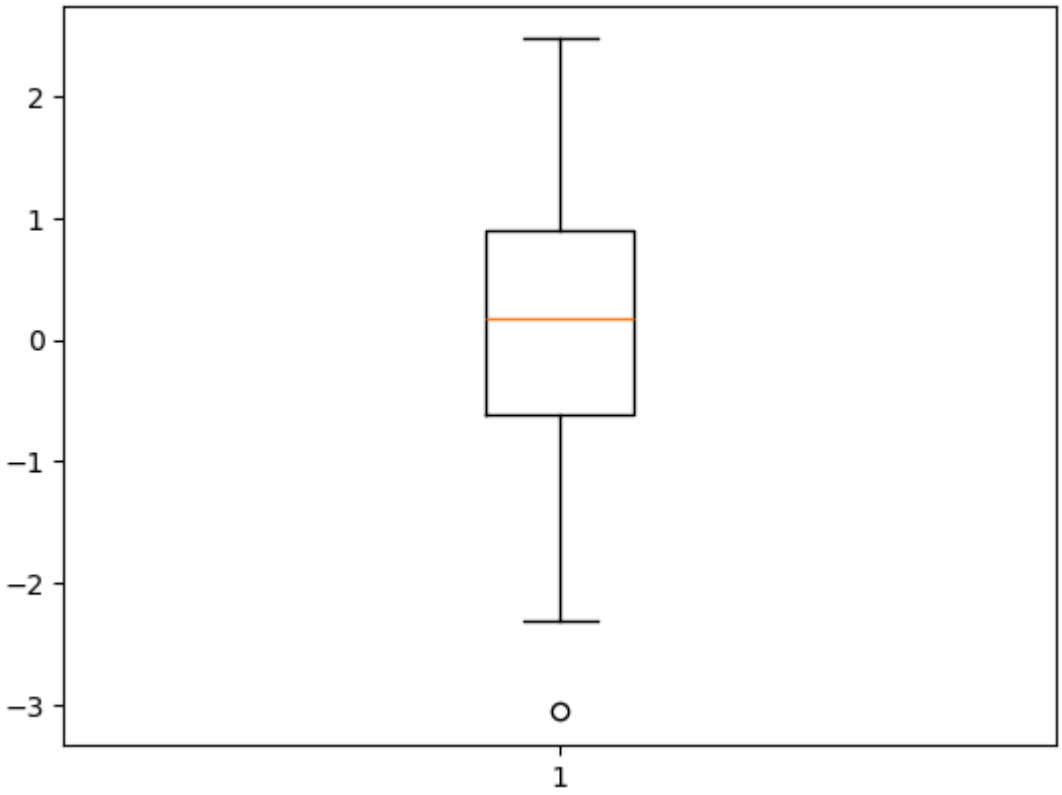
Pie Chart

In [38]:
```
plt.figure(figsize=(8,8))   # here it is crates a squre figure for better circul
x10=[60,55,25,45]
labels=['computers','eletronics','Machanical','Chemical'] # her we give the the
plt.pie(x10,labels=labels)
plt.show()
```



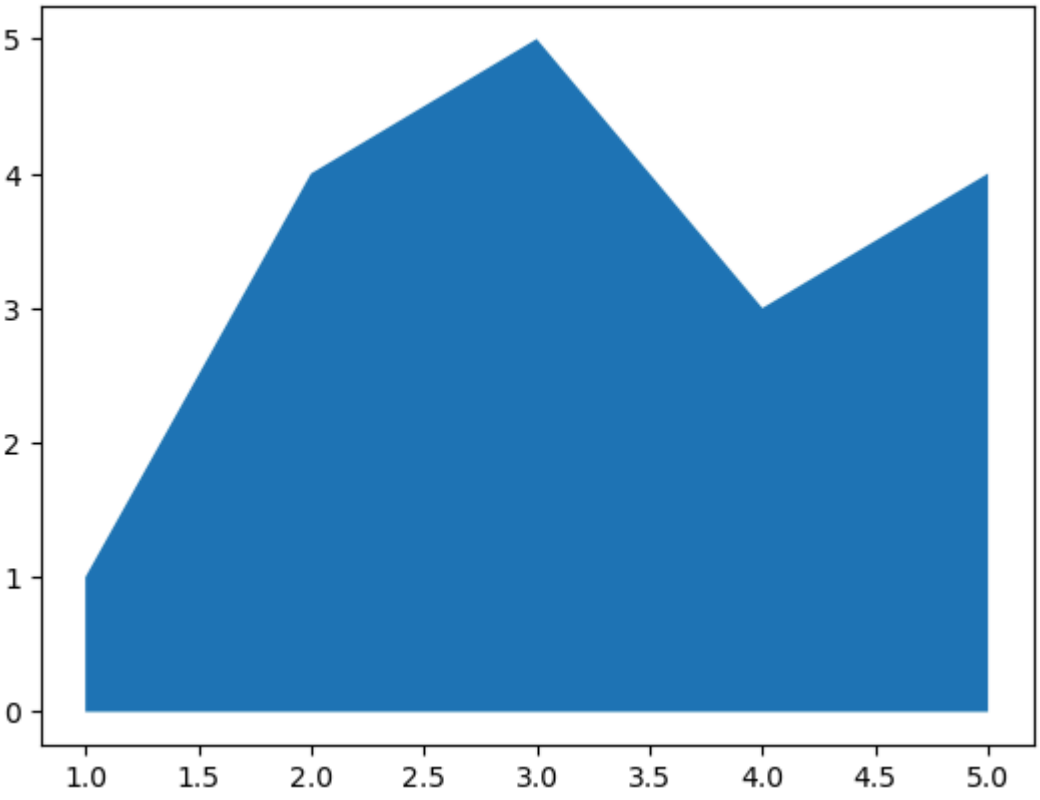
Box plot

In [39]:
```
data3=np.random.randn(100)  # it generates 100 random values from normal distrib
plt.boxplot(data3)
plt.show()
```



Area Chart

In [40]:
```
x12=range(1,6)  # these are the sequence of x-values (1,6)
y12=[1,4,5,3,4]
plt.fill_between(x12,y12)  # here the fills the area bet the curve
plt.show()
```

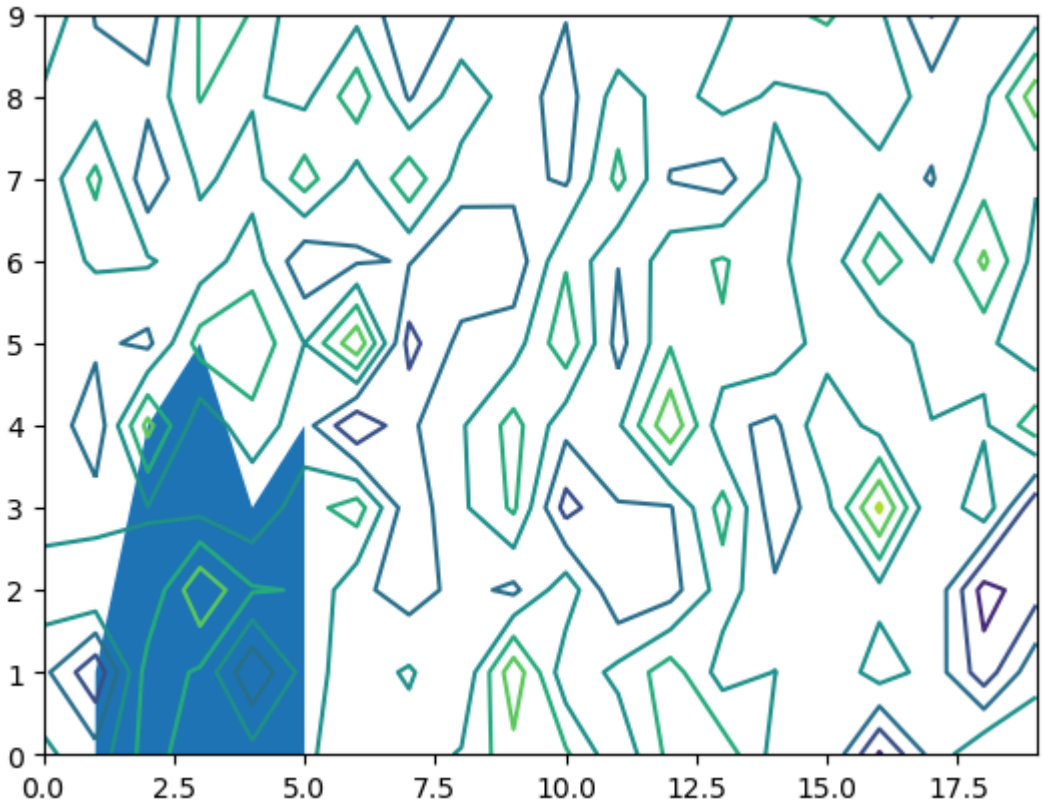


In [41]:
```
#here stackplot also use in this function
plt.stackplot(x12,y12)
```

Out[41]:
```
[<matplotlib.collections.PolyCollection at 0x21e7980f0b0>]
```

Contour plots

In [42]:
```
# here we create the martix
matrix1=np.random.randn(10,20)
cp=plt.contour(matrix1)  #This simulates 3D data for plotting in 2D.
plt.show()
```



Styles of Matplotlib plots

In [43]:
```
print(plt.style.available) # it veiw list of all styles in matplotlib
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid',
'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'graysc
ale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-
v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-d
eep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seabo
rn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-tick
s', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [44]:
```
plt.style.use('seaborn-bright')
```

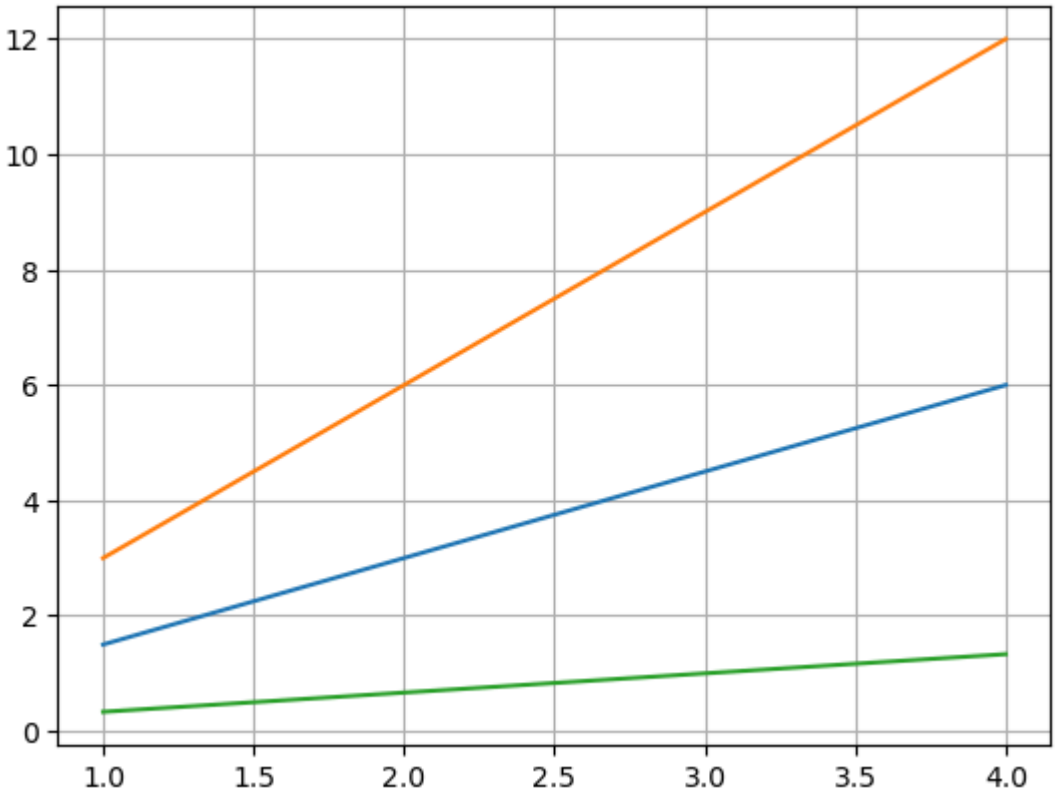
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\matplotlib\style\core.py:137, in use(style)
    136 try:
--> 137     style = _rc_params_in_file(style)
    138 except OSError as err:

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:870, in _rc_params_in_f
ile(fname, transform, fail_on_error)
    869 rc_temp = {}
--> 870 with _open_file_or_url(fname) as fd:
    871     try:

File ~\anaconda3\Lib\contextlib.py:137, in _GeneratorContextManager.__enter__(sel
f)
    136 try:
--> 137     return next(self.gen)
    138 except StopIteration:

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:847, in _open_file_or_u
rl(fname)
    846 fname = os.path.expanduser(fname)
--> 847 with open(fname, encoding='utf-8') as f:
    848     yield f

FileNotFoundError: [Errno 2] No such file or directory: 'seaborn-bright'

The above exception was the direct cause of the following exception:

OSError                                   Traceback (most recent call last)
Cell In[44], line 1
----> 1 plt.style.use('seaborn-bright')

File ~\anaconda3\Lib\site-packages\matplotlib\style\core.py:139, in use(style)
    137         style = _rc_params_in_file(style)
    138     except OSError as err:
--> 139         raise OSError(
    140             f"{style!r} is not a valid package style, path of style "
    141             f"file, URL of style file, or library style name (library "
    142             f"styles are listed in `style.available`)") from err
    143 filtered = {}
    144 for k in style:  # don't trigger RcParams.__getitem__('backend')

OSError: 'seaborn-bright' is not a valid package style, path of style file, URL o
f style file, or library style name (library styles are listed in `style.availabl
e`)
```

In [49]:
```
import matplotlib.pyplot as plt
plt.style.available
```

Out[49]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

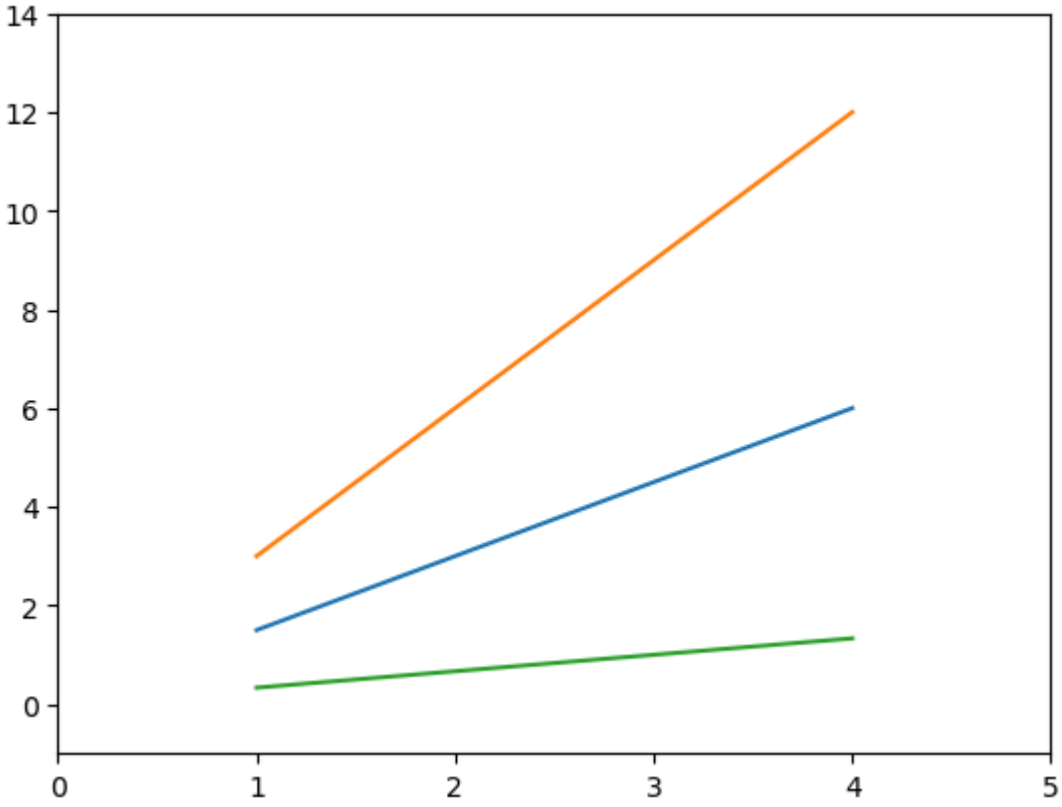
Adding a Grid

In [45]:
```
x15=np.arange(1,5)

plt.plot(x15, x15*1.5,x15, x15*3.0, x15, x15/3.0)

plt.grid(True)

plt.show()
```

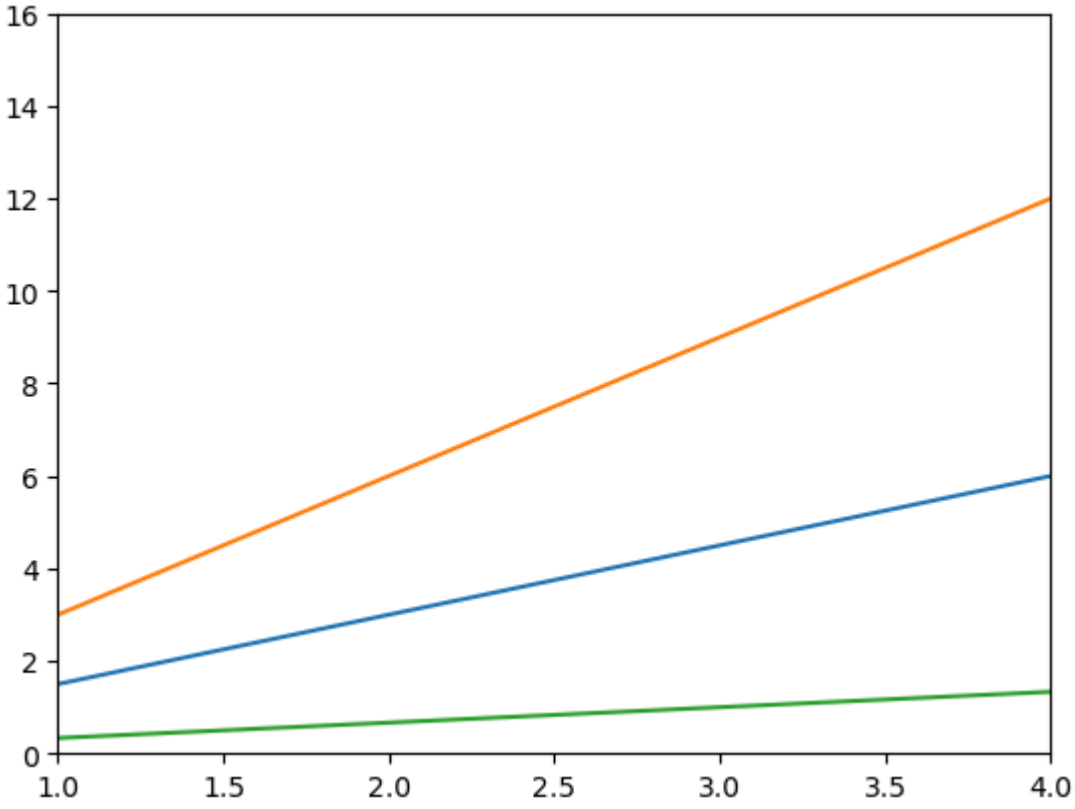


Handling axes

In [46]:
```
x15=np.arange(1,5)

plt.plot(x15, x15*1.5,x15, x15*3.0, x15, x15/3.0)

plt.axis()

plt.axis([0,5,-1,14])
plt.show()
```

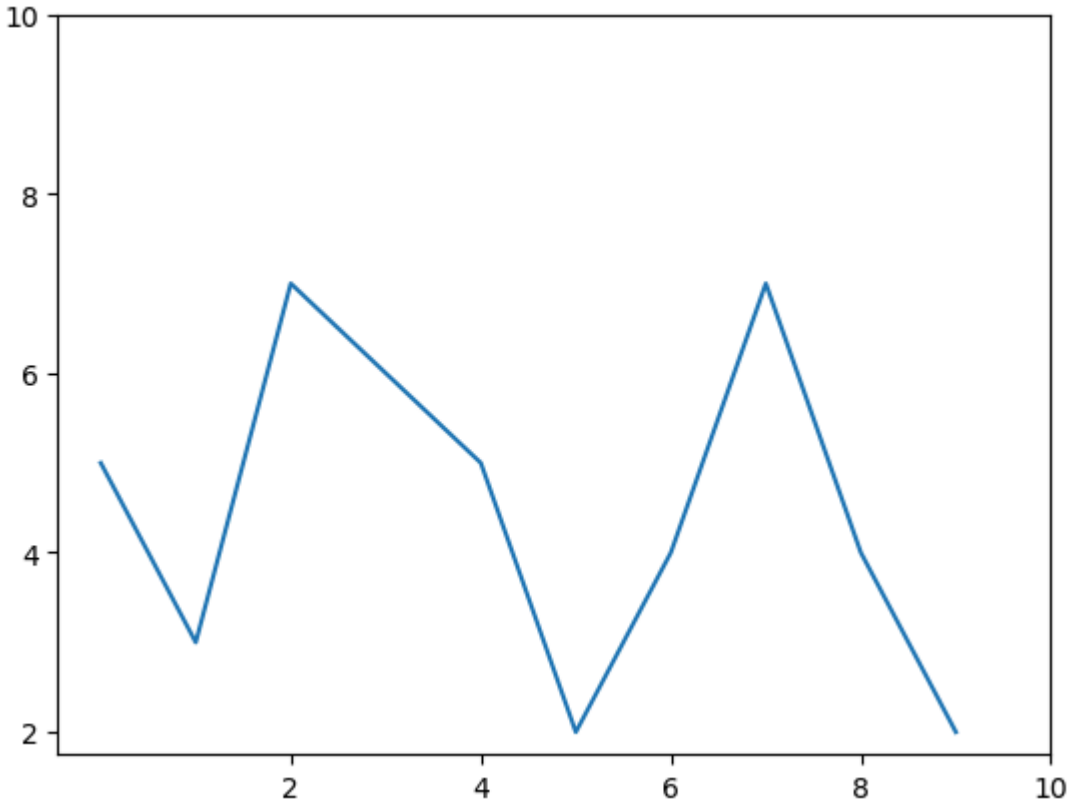


In [47]:
```
x15=np.arange(1,5)

plt.plot(x15, x15*1.5,x15, x15*3.0, x15, x15/3.0)

plt.xlim([1.0,4.0])

plt.ylim([0.0,16.0])
plt.show()
```



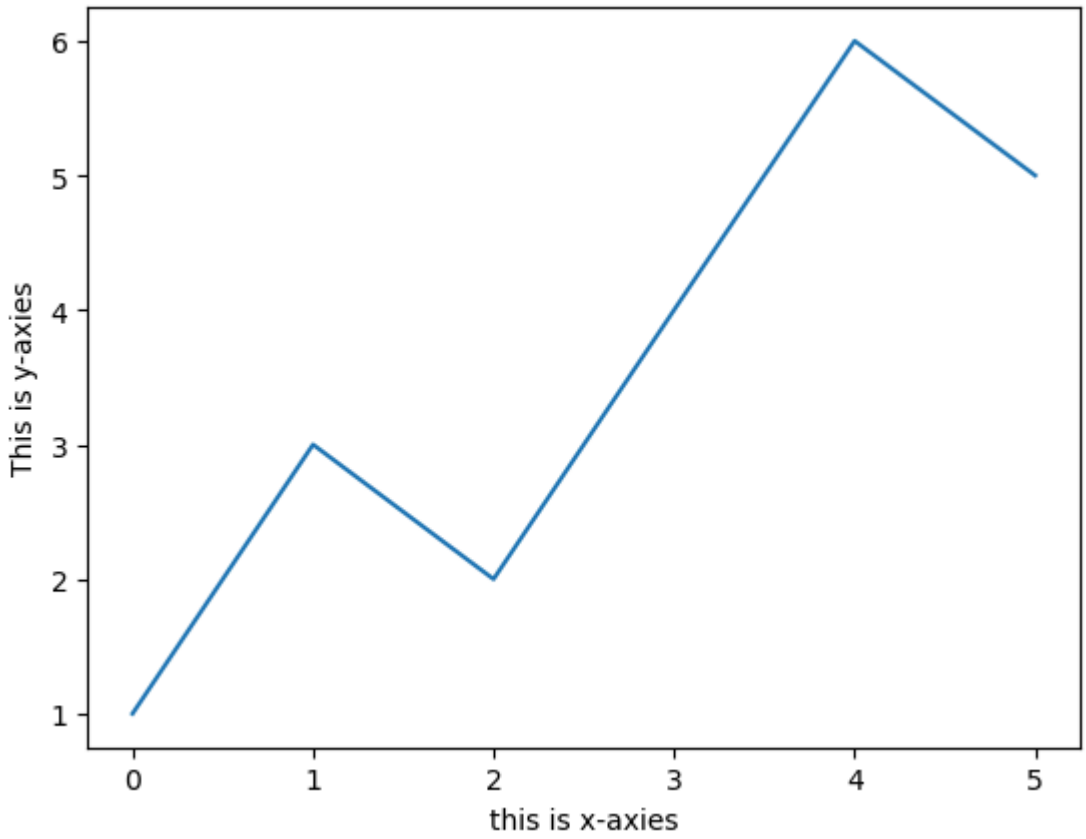
Handling X and Y ticks

In [54]:
```
import matplotlib.pyplot as plt
u=[5,3,7,6,5,2,4,7,4,2]
plt.plot(u)
plt.xticks([2,4,6,8,10])
plt.yticks([2,4,6,8,10])
plt.show()
```



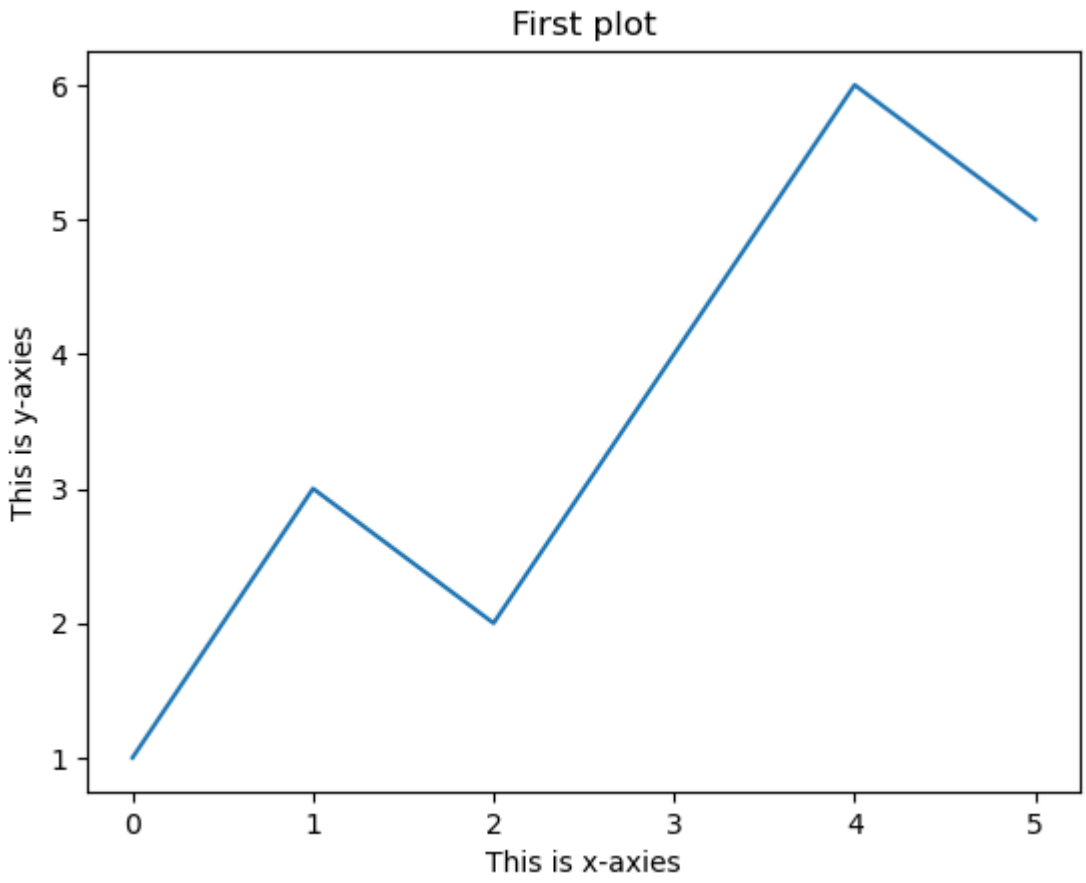
Adding labels

In [56]:
```
plt.plot([1,3,2,4,6,5])
plt.xlabel('this is x-axies')
plt.ylabel('This is y-axies')
plt.show()
```



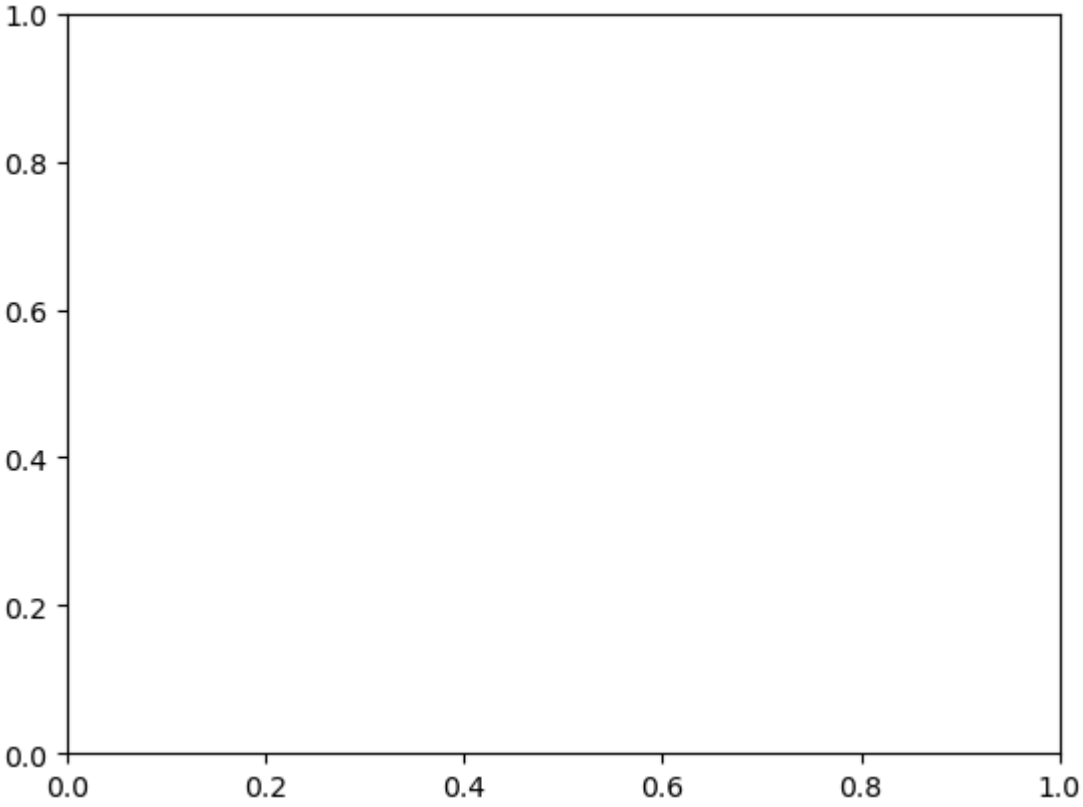
Adding a title

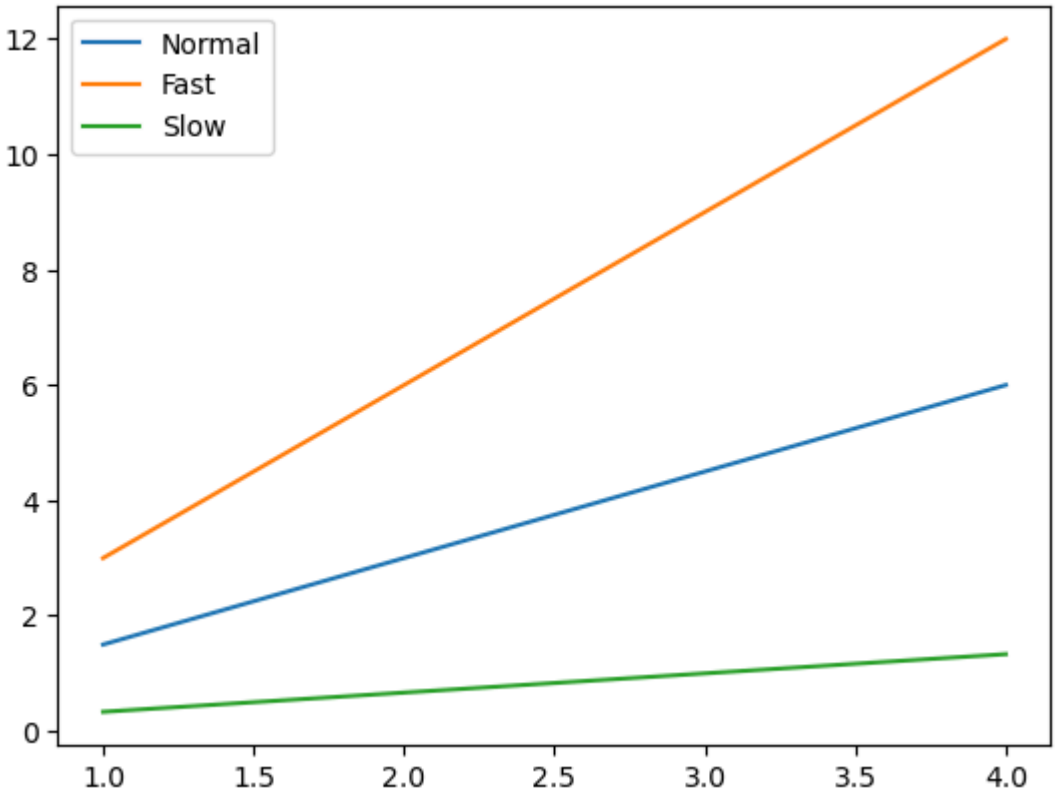
In [58]:
```
plt.plot([1,3,2,4,6,5])
plt.title('First plot')
plt.xlabel('This is x-axies')
plt.ylabel('This is y-axies')
plt.show()
```



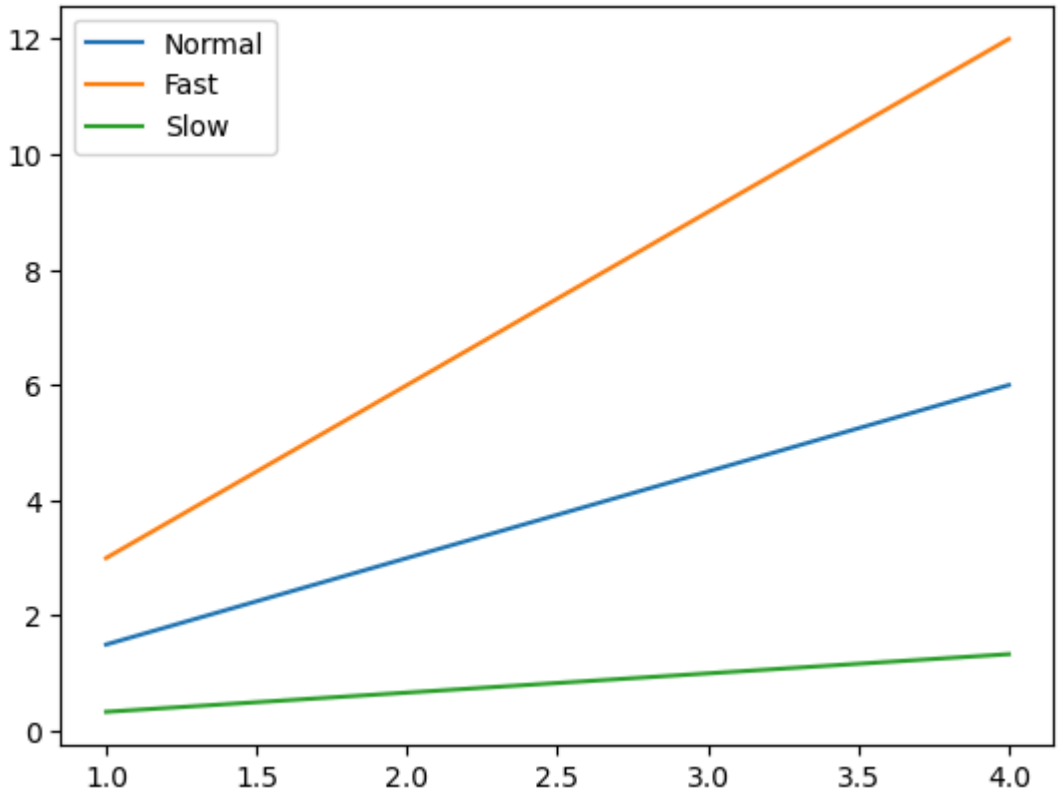
Adding a legend

In [60]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5)
ax.plot(x16,x16*3.0)
ax.plot(x16,x16/3.0)

ax.legend(['Normal','Fast','Slow'])
plt.show()
```



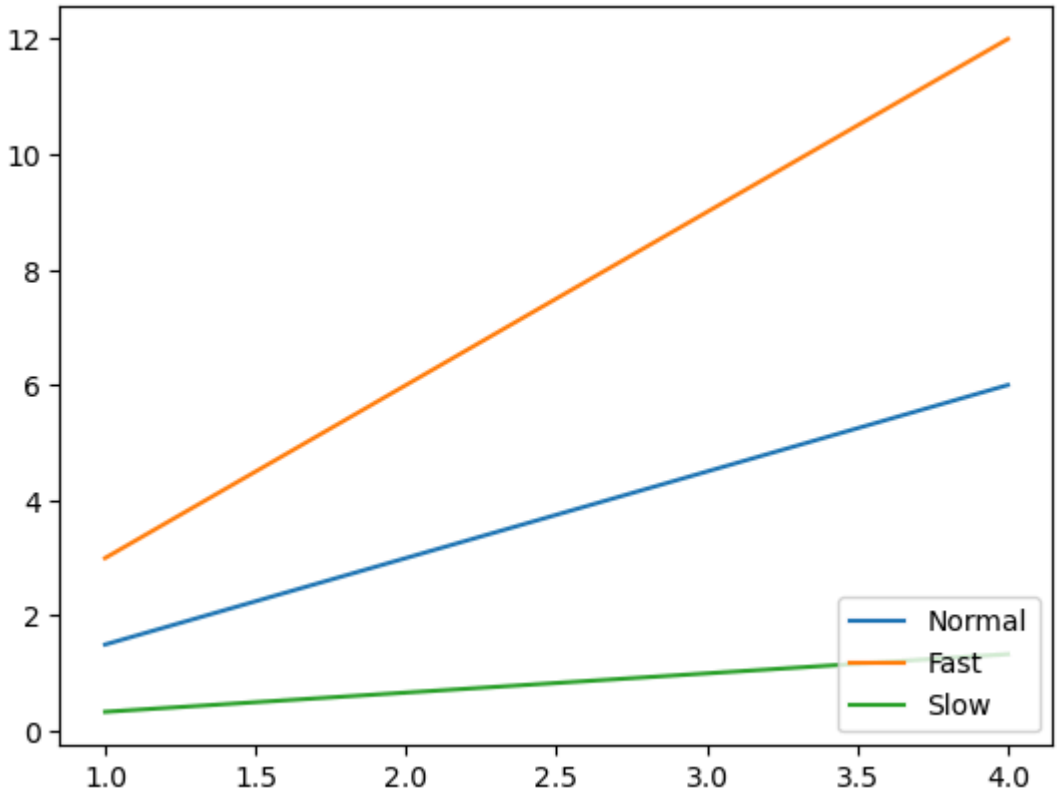


In [63]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5,label='Normal')
ax.plot(x16,x16*3.0,label='Fast')
ax.plot(x16,x16/3.0,label='Slow')

ax.legend(loc=0)
plt.show()
```

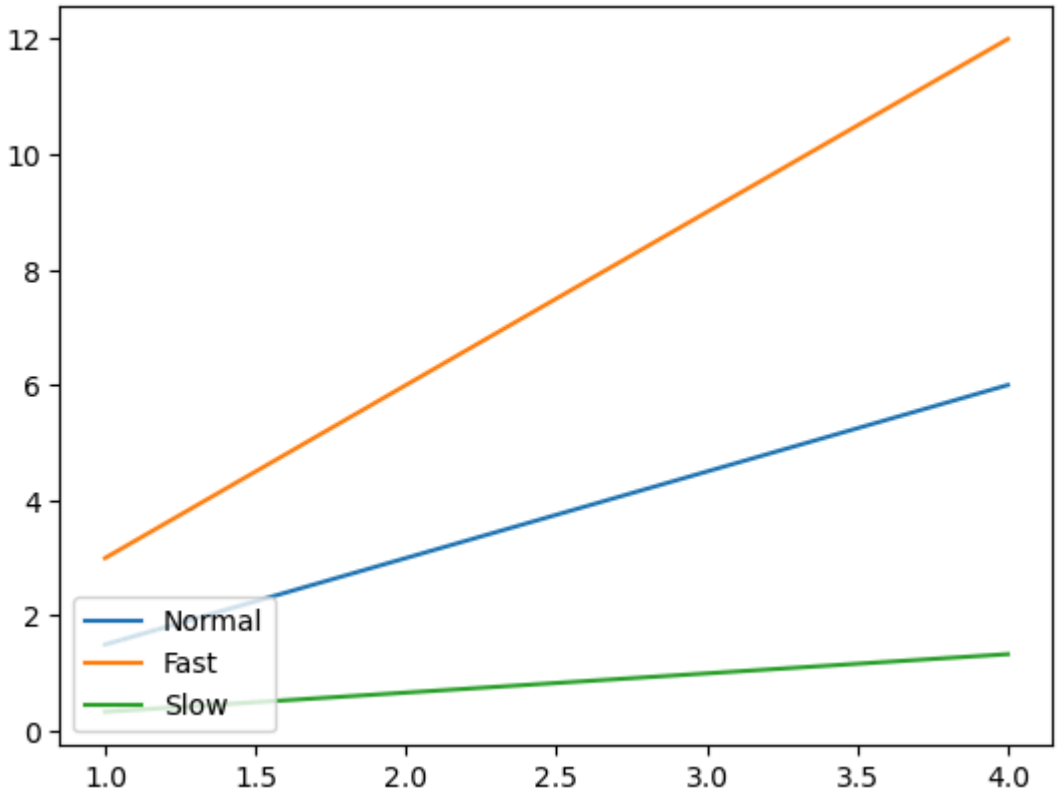


ax.legend(loc=0) # let Matplotlib decide the optimal location ax.legend(loc=1) # upper right corner ax.legend(loc=2) # upper left corner ax.legend(loc=3) # lower left corner ax.legend(loc=4) # lower right corner ax.legend(loc=5) # right ax.legend(loc=6) # center left ax.legend(loc=7) # center right ax.legend(loc=8) # lower center ax.legend(loc=9) # upper center ax.legend(loc=10) # center

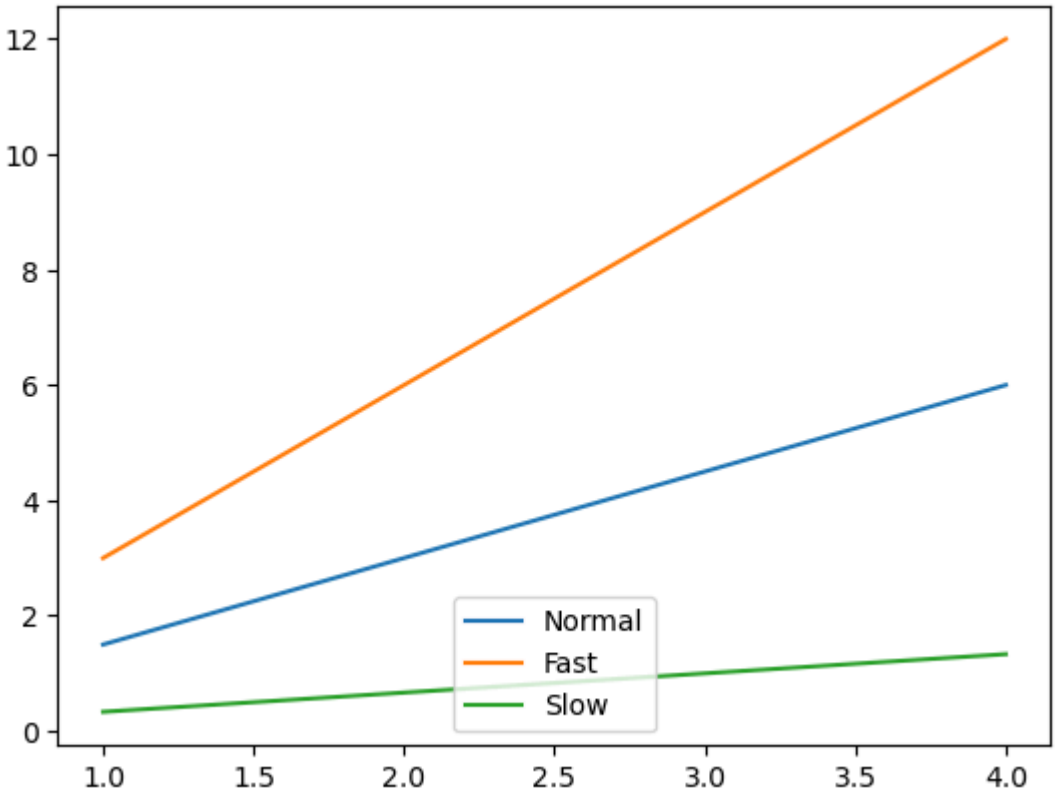
In [64]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5,label='Normal')
ax.plot(x16,x16*3.0,label='Fast')
ax.plot(x16,x16/3.0,label='Slow')

ax.legend(loc=4)
plt.show()
```



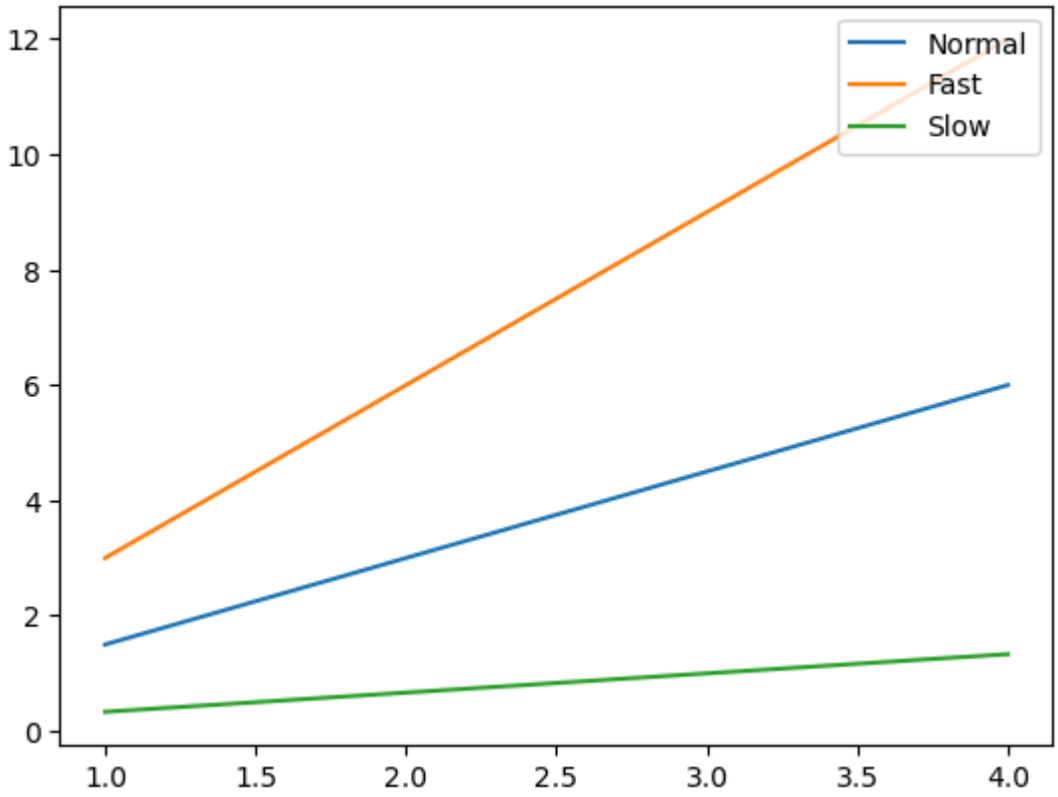
In [65]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5,label='Normal')
ax.plot(x16,x16*3.0,label='Fast')
ax.plot(x16,x16/3.0,label='Slow')

ax.legend(loc=3)
plt.show()
```



In [66]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5,label='Normal')
ax.plot(x16,x16*3.0,label='Fast')
ax.plot(x16,x16/3.0,label='Slow')

ax.legend(loc=8)
plt.show()
```

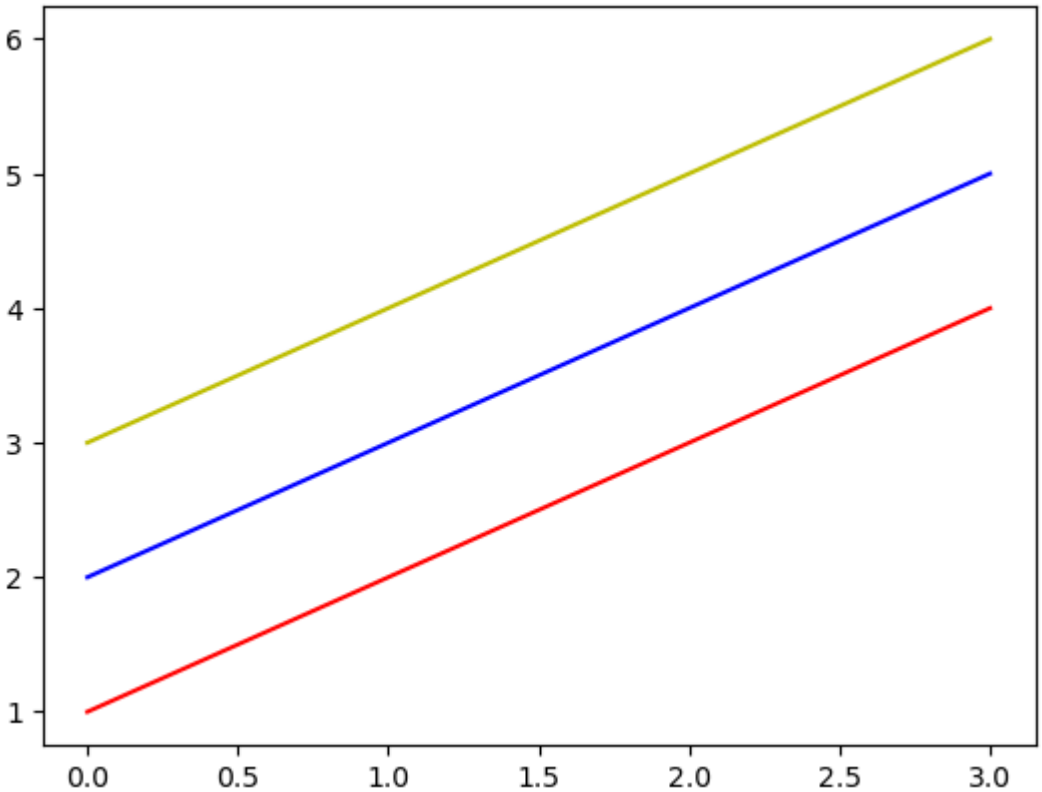


In [67]:
```
x16=np.arange(1,5)

fig,ax=plt.subplots()
ax.plot(x16,x16*1.5,label='Normal')
ax.plot(x16,x16*3.0,label='Fast')
ax.plot(x16,x16/3.0,label='Slow')

ax.legend(loc=1)
plt.show()
```



Control colours

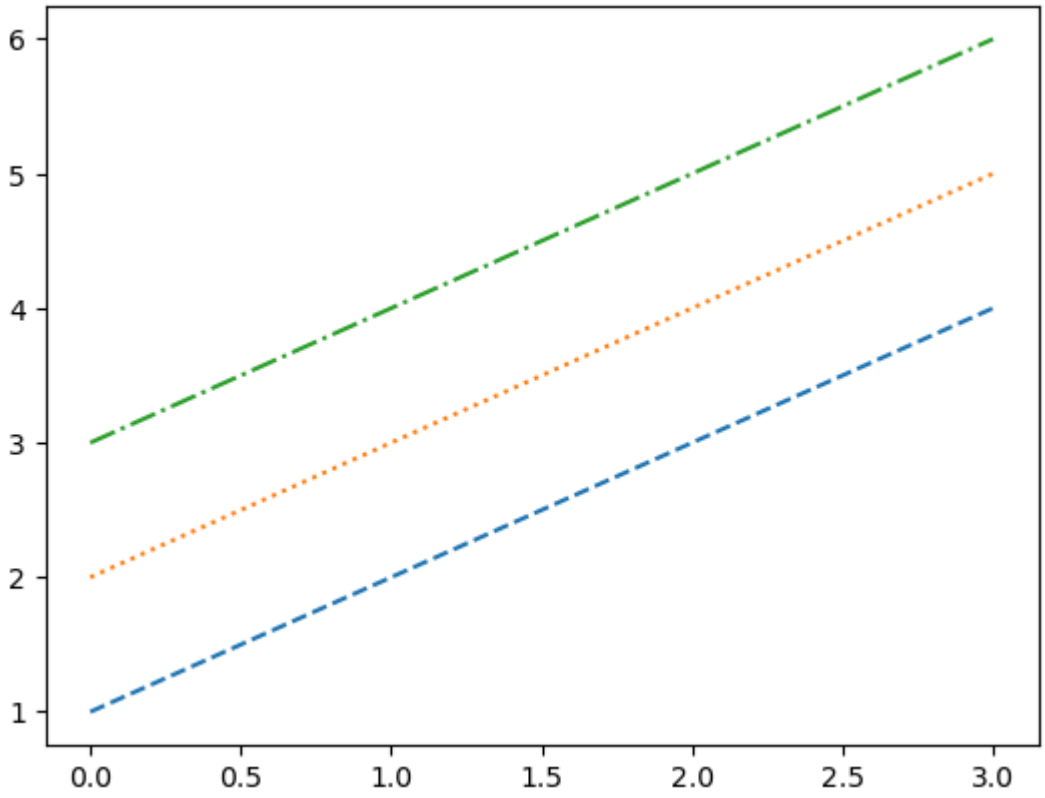
In [70]:
```
x17=np.arange(1,5)
plt.plot(x17,'r')
plt.plot(x17+1,'b')
plt.plot(x17+2,'y')
plt.show()
```



The colour names and colour abbreviations are given in the following table:- Colour abbreviation Colour name b blue c cyan g green k black m magenta r red w white y yellow

Control line styles

In [71]:
```
x17=np.arange(1,5)
plt.plot(x17,'--',x17+1,':',x17+2,'-.')
plt.show()
```



Style abbreviation Style solid line -- dashed line -. dash-dot line : dotted line

In [ ]:
```

```