1. MATPLOTLIB ::: PLOTTING LIBRARY :

PAGE 209 BLUE NOTE BOOK

In [5]:
```
# 'import matplotlib.pyplot as plt' and here we are going to work with an a -
# '%matplotlib inline'

# What is matplotlib used for Python?

# Matplotlib is a comprehensive library for creating static, animated,
# and interactive visualizations in Python. Matplotlib makes easy things easy
# and hard things possible. Create publication quality plots.
# Make interactive figures that can zoom, pan, update.

# What is matplotlib and Seaborn in Python?

# Matplotlib is a library in Python that enables users to generate visualizations like
# histograms, scatter plots, bar charts, pie charts and much more.

# Seaborn is a visualization library that is built on top of Matplotlib.
# It provides data visualizations that are typically more aesthetic and statistically sophisticated.
```

In [6]:
```
# MATPLOTLIB : In our regular working environment, Any Data we can take into the Consideration,
# even the 'image data' also -> We are going to Call that particular Data into an a -
# " GRAPHICAL REPRESENTATION "

# " GRAPHICAL REPRESENTATION " - means, nothing but if it is in a 'NUMERIC DATA', Something like,
# 'x' data   and  'y' data.
# Based on 'x' and 'y' variant, the points are going to be 'Plotted'
# Whenever i have 'x' and 'y' points, then only i can plot this
# '0' of '0' means - The 'Point' is at 'ORIGIN'.
```

NOTE : MATPLOTLIB LIBRARY :

In [7]:
```
# This Library is Specially Designed for 'DATA VISUALISATION' to Represent the Data in a -
# 'Pictorial' (or) 'Graphical format'
# Matplotlib take the supporting library of an a 'Plotly' to Represent the data effectively.

# The Advantage of Data Visualisations by Matplotlib are - (1) CLARITY, (2)ACCURACY, (3)EFFICIANCY.
# Matplotlib - is the most popular plotting library for 'python',
# it was designed to have a similar feel of Matlab's Graphical plotting library.
# In Matplotlib - We can go for the 'Customised Plotting Library'
```

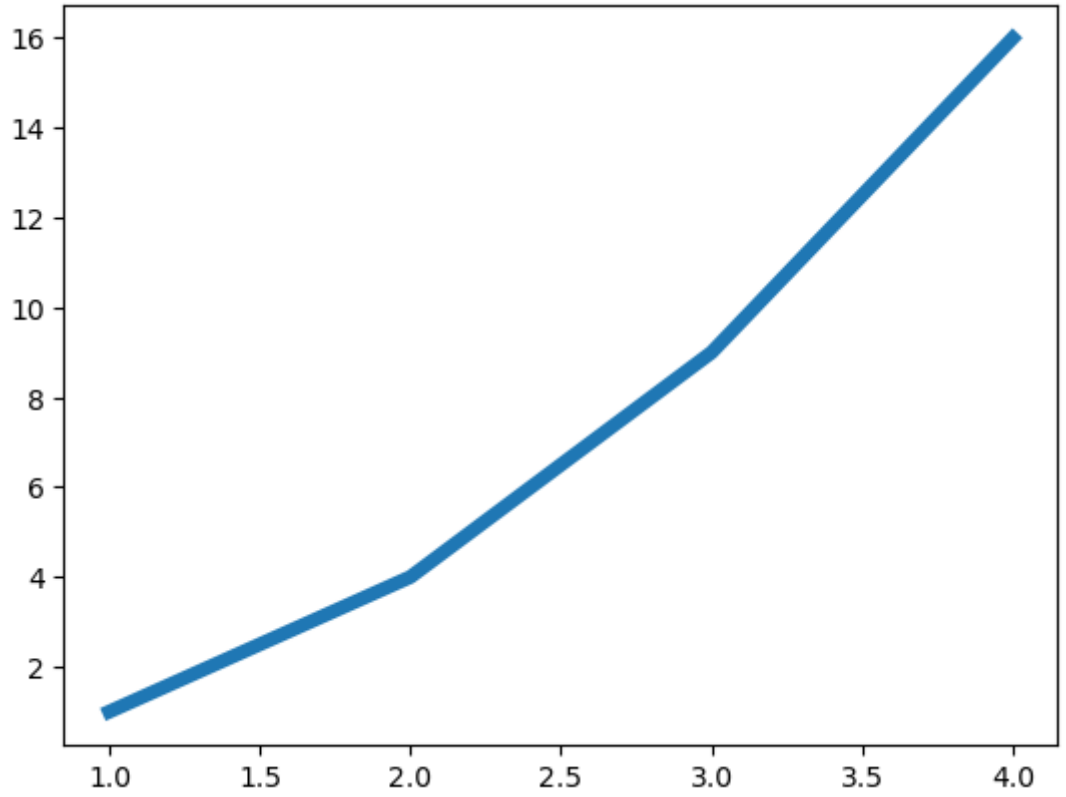
pyplot :

page 211

In [8]:
```
# pyplot = python plot

# 'Matplotlib.pyplot' - Is a Collection of Command Style Functions that make
# 'Matplotlib' to work like "Matlab".
# Each 'pyplot' function makes some changes to a figure (or) plot such as 'plot' some lines on a
# plotting area, decorate the 'plot' with labels, and many more.

# 'plt' - means, 'Matplotlib'
# '%matplotlib inline' - The exact meaning of this '%matplotlib space inline' means,
# Whenever we 'Run' the 'Execution'- at that time, we want to view the 'Output', The Output we will
# get in the 'Graphs'. That Particular 'Graph' i want to View in the Same Window.

# If i won't give this particular Syntax(matplotlib inline), it will execute the Output, But
# It will take the Second Window. That's Why We are giving "%matplotlib inline"
```

In [2]:
```
# %matplotlib inline -> To get Output results in Same Window :

import matplotlib.pyplot as plt
%matplotlib inline
```

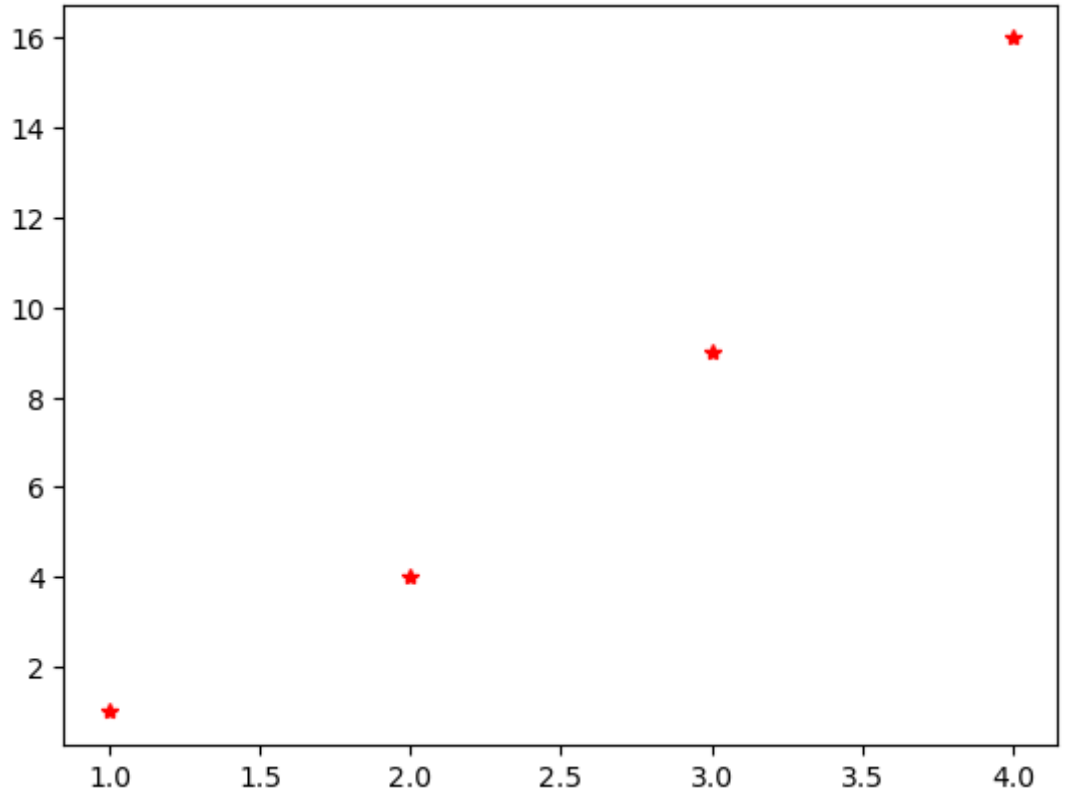
In [3]:
```
x = [1,2,3,4]
y = [1,4,9,16]

plt.plot(x,y,linewidth = 5.0)
plt.show()
```



2. DECLARE COLOUR WITH POINTER :

PAGE 213

In [4]:
```
plt.plot(x,y, 'r*')
plt.show()
```

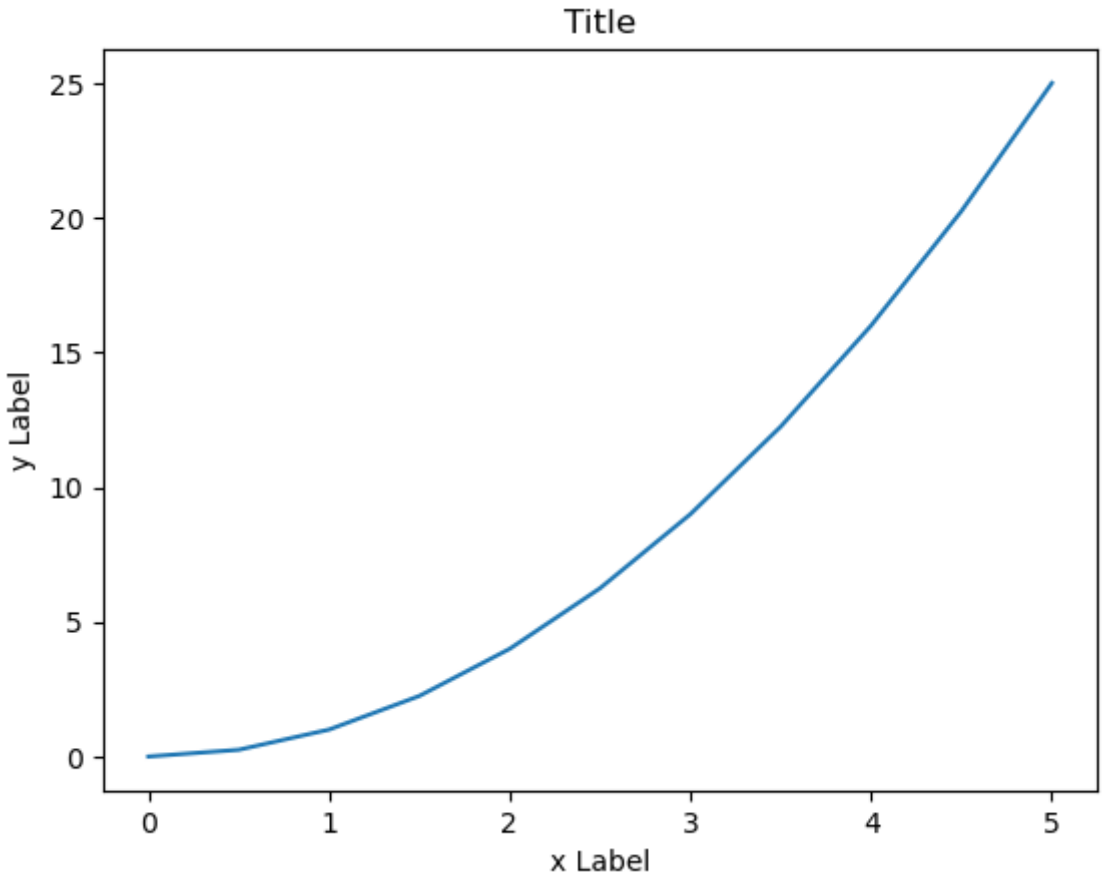


3. NAMING CONVENTION :

PAGE 215

In [56]:
```
plt.plot(x,y)   # Defaultly it will take 'The Length of the Line'
plt.xlabel("x Label")
plt.ylabel("y Label")
plt.title("Title")
```

Out[56]:
```
Text(0.5, 1.0, 'Title')
```

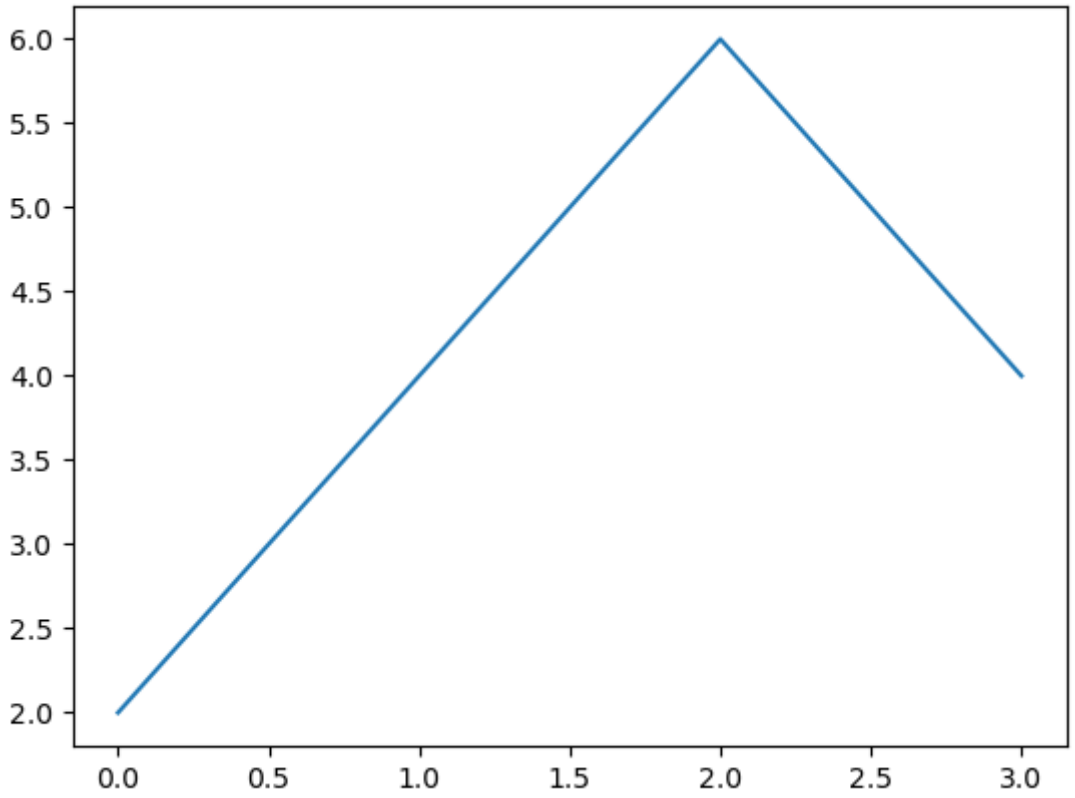


4. NOW 'POINTS', Not 'x' nor 'y' Co-ordinators :

page 215

In [18]:
```
plt.plot([2,4,6,4])  # points these are,,,
plt.xlable("Indeices")
plt.ylable("Numbers")
plt.title("Indis Vs Numbers")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16936\2342833108.py in <module>
      1 plt.plot([2,4,6,4])  # points these are,,,
----> 2 plt.xlable("Indeices")
      3 plt.ylable("Numbers")
      4 plt.title("Indis Vs Numbers")
      5 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'xlable'
```

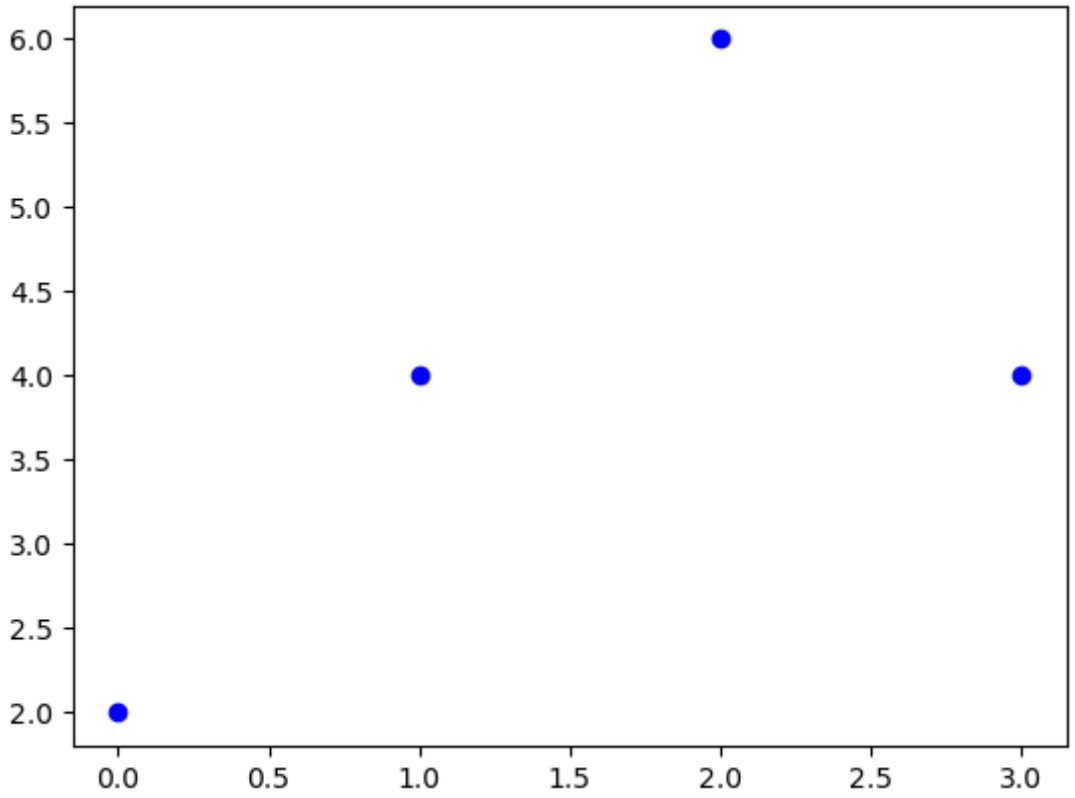


5. Point based declaration :

page 216

In [7]:
```
# Based on the 'points', 'x' and 'y' axis going to be taken dafaultly.
# 'bo' - Blue Colour Circle Shape.

plt.plot([2,4,6,4],'bo')
plt.xlable("Indeices")
plt.ylable("Numbers")
plt.title("Indis Vs Numbers")
plt.show()
```

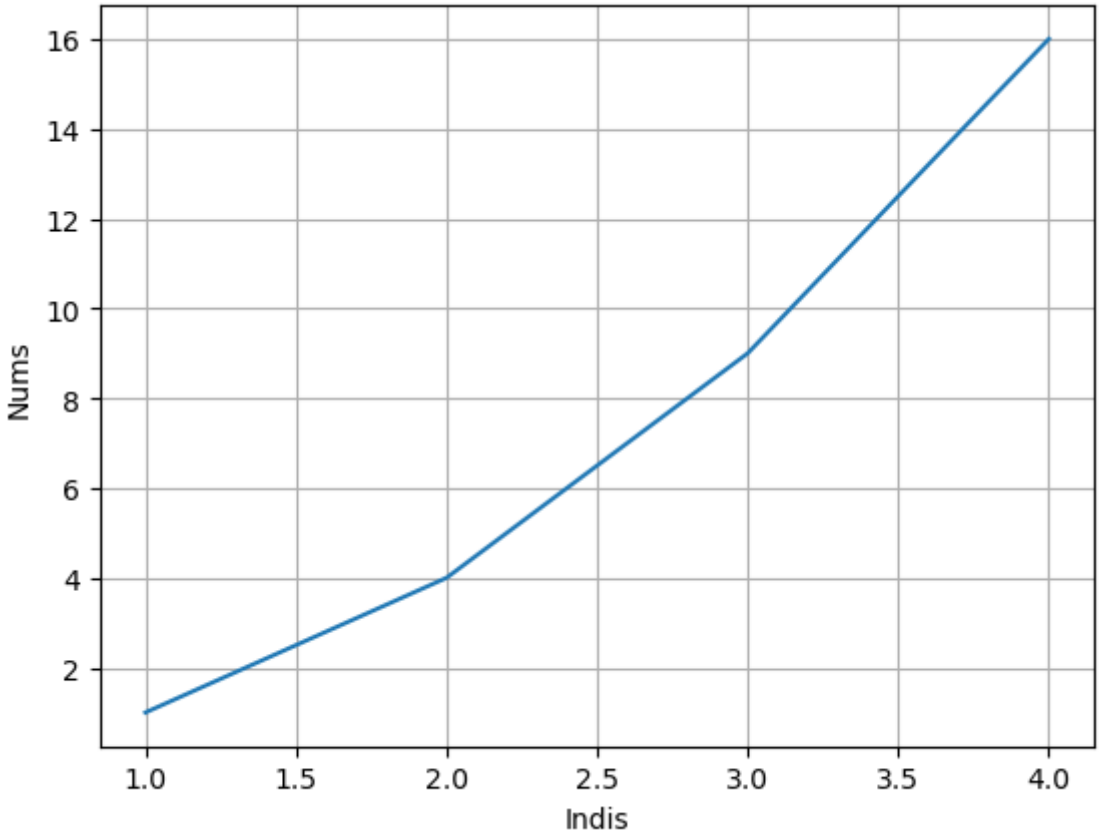
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16936\2915869008.py in <module>
      3
      4 plt.plot([2,4,6,4],'bo')
----> 5 plt.xlable("Indeices")
      6 plt.ylable("Numbers")
      7 plt.title("Indis Vs Numbers")

AttributeError: module 'matplotlib.pyplot' has no attribute 'xlable'
```



6. Grid ( ) :

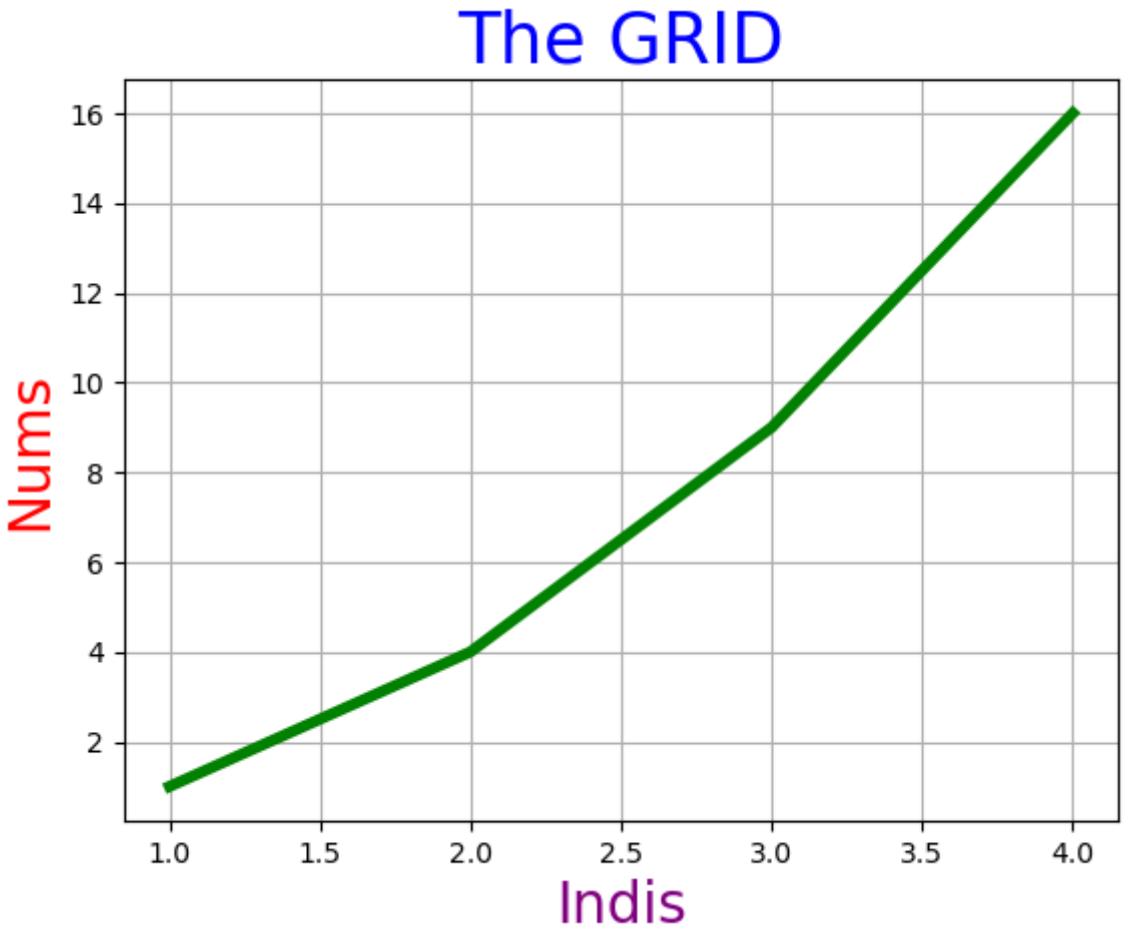
page 217

In [274]:
```
# Small Square Boxes, Something like Pixels We will see.
# It is too easy to lacate where exactly 'point' is Coming.

plt.plot([1,2,3,4],[1,4,9,16])
plt.xlabel("Indis")
plt.ylabel("Nums")
plt.grid()
plt.show()
```



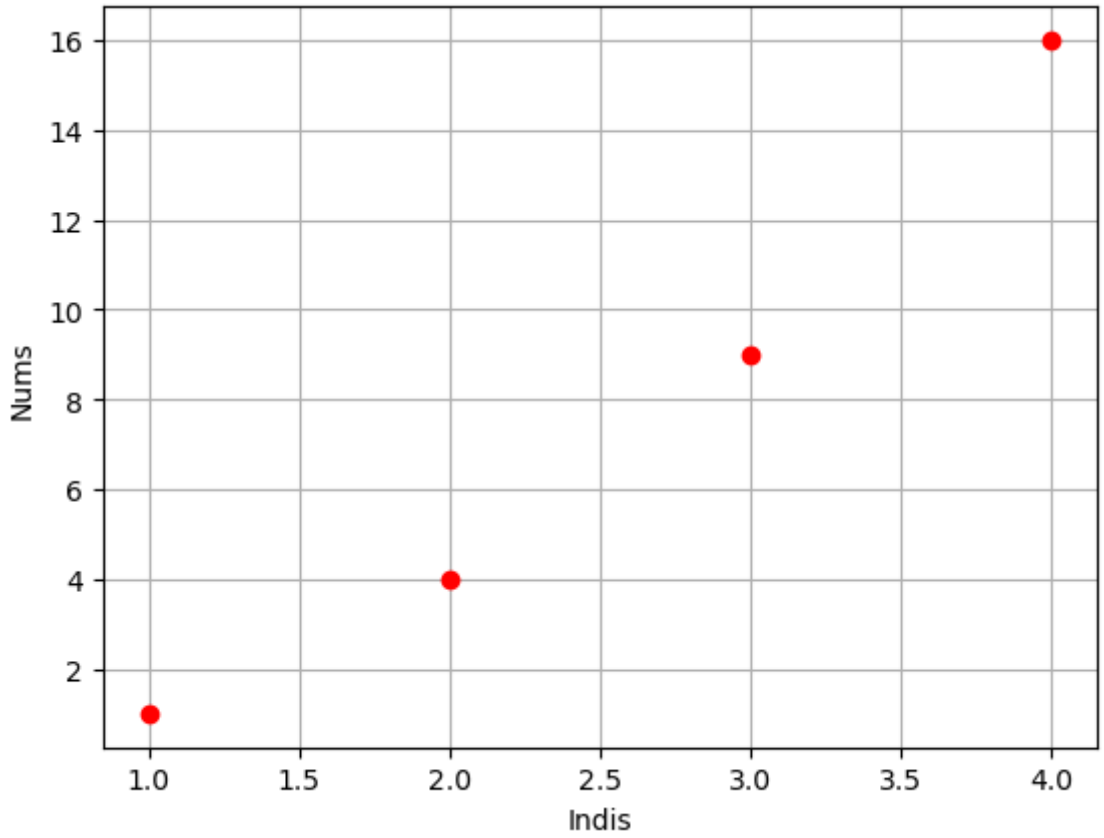
6th sum modifications :

In [289]:
```
plt.plot([1,2,3,4],[1,4,9,16], lw = 4, color = 'green')
plt.xlabel("Indis", size = 20, color = 'purple')
plt.ylabel("Nums", size = 20, color = 'red')
plt.grid()
plt.title("The GRID", color = 'b', size = 25)
plt.show()
```



7. POINTING STYLE :

In [9]:
```
# Here, plt.plot([1,2,3,4],[1,4,9,16],'ro') - Now again this is called as an a Co-ordinated base.
# We are not declaring independantly 'x' and 'y'
# Here '2 co-ordinates' -> 1st co-ordinate it will take as 'x' defaultly
# and '2nd co-ordinate' it will take as 'y' defaultly


plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.xlabel("Indis")
plt.ylabel("Nums")
plt.grid()
plt.show()
```

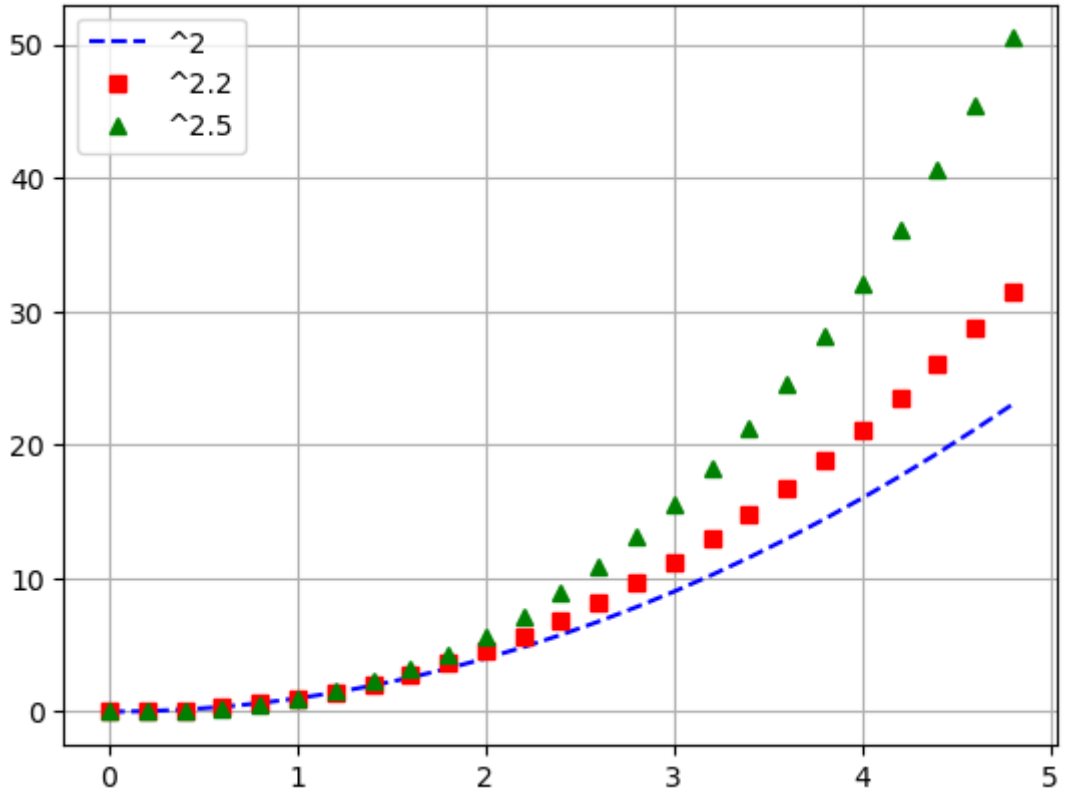


8. '3' OPERATIONS IN A SINGLE GRAPH :

PAGE 218

In [14]:
```
# Now i want to work with '3' plotting points in a Single Graph.
# Means, 3 results in single Graph:
# Here, Individual plotting also we will see
# Here, Double star(**) is a Multiplication. Means, 'Square of Square' 2square =4, 4square = 16.
```

In [15]:
```
import numpy as np
t = np.arange(0.,5.0,0.2)
plt.plot(t,t**2,'b--',label = '^2')
plt.plot(t,t**2.2,'rs',label = '^2.2')
plt.plot(t,t**2.5,'g^',label = '^2.5')
plt.grid()
plt.legend()
plt.show()
```



9. Multiple Ways of Declaration :

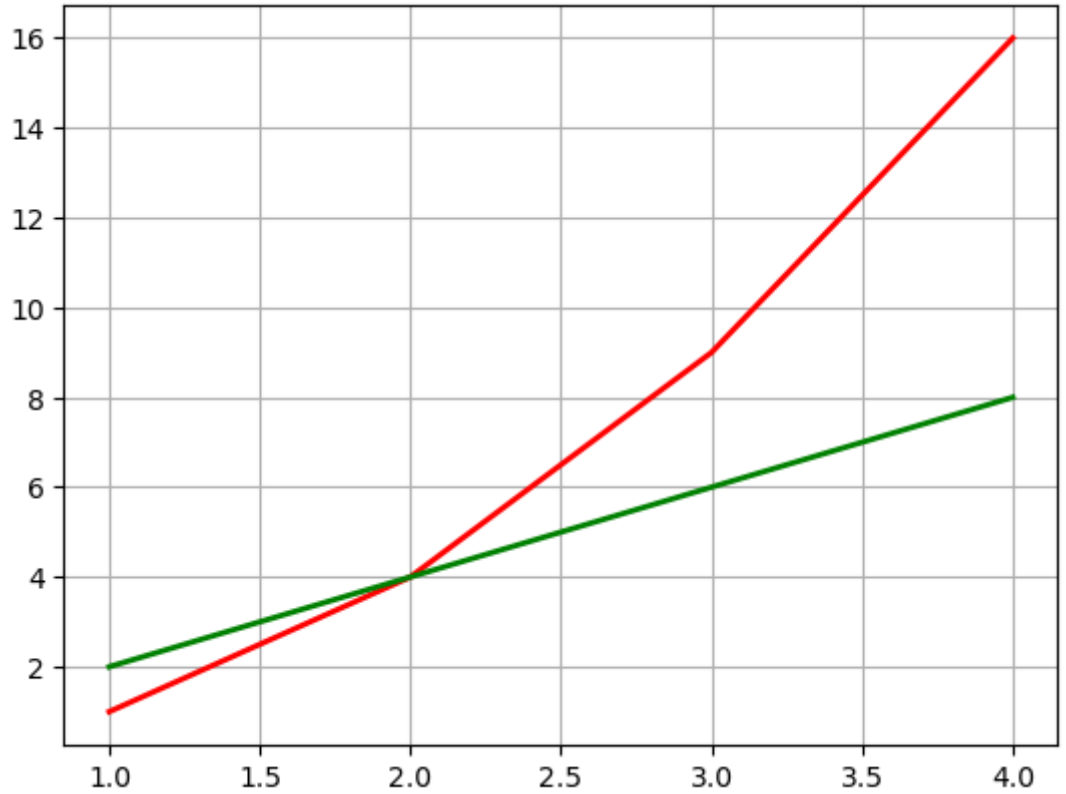
page 220

NOTE : GRAPH

In [50]:
```
# GRAPH = FIGURE
# We can Split the figure in to an a 'SUBPLOTS'
# 'SUBPLOTS' Can be done, Based on 'MATRIX FORMAT DECLARATION'
# 'SUBPLOT' eg -> 2,3 -- means 2 Rows and 3 Columns.

# The 'SUBPLOT PROPERITIES' - Such as 'x' point, 'y' point, Shape of the Point, Colour of the Point -
# Totally depends upon 'SUBPLOT NUMBER'
```

In [51]:
```
# setp = Set Properities
```

In [17]:
```
x1 = [1,2,3,4]
y1 = [1,4,9,16]

x2 = [1,2,3,4]
y2 = [2,4,6,8]

lines = plt.plot(x1,y1,x2,y2)

plt.setp(lines[0],color ='r',linewidth = 2.0)
plt.setp(lines[1],'color','g','linewidth',2.0)
plt.grid()
plt.show()
```



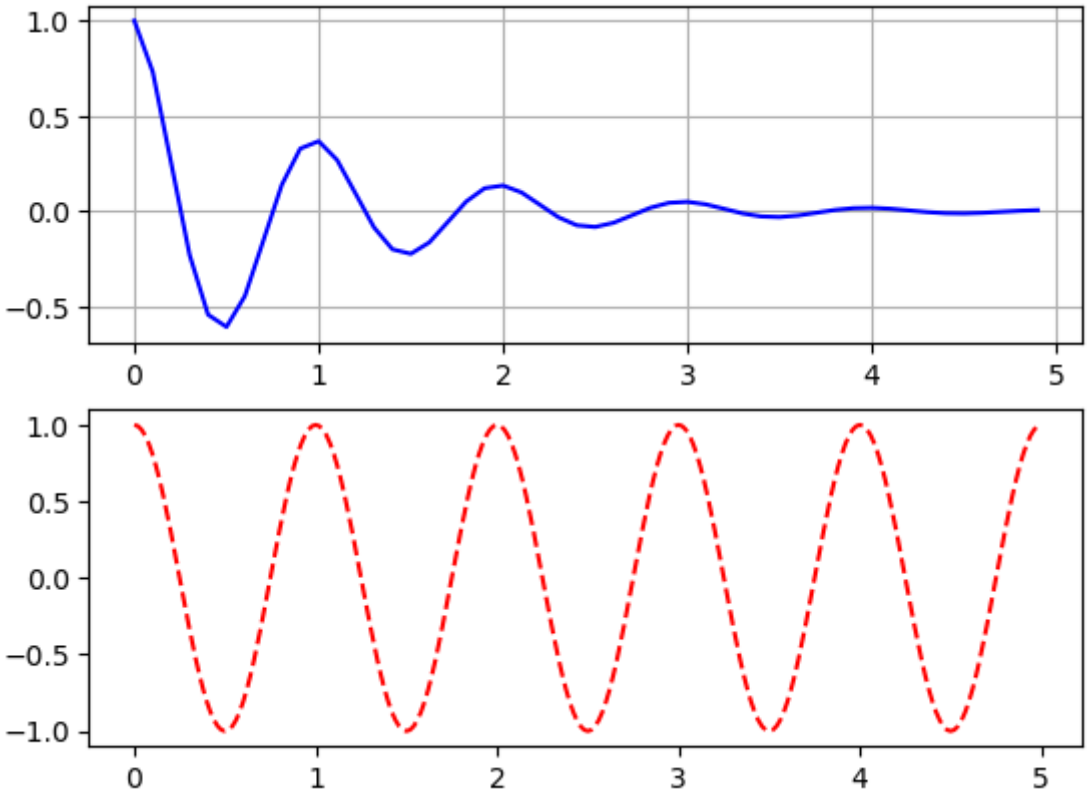
10. SUBPLOT DECLARATION CONCEPT :

Here i want to get -- " SIGN WAVE OUTPUT "

PAGE 224

In [47]:
```
# Here, def f (t): - means "in the f of (t),"

# Now 'SIN WAVE FORMULA' - return np.exp(-t) * np.cos(2 * np.pi * t)
# --> " i'm taking an a return value as an a -
# numpy (dot). exponantial of (-t) * numpy . cos of (2 * numpy . pi value * t)

# t1 = np.arange(0.0,5.0,0.1) - Here t1 = numpy.arange of (0.0 to 5.o with 0.1 step)

# plt.figure(1) -- Here, figure number we can declare, because we can take multiple figures
# based on the requirement. 2 figures, 3 figures like that independantlyl
# As per right now, figure name (1)

# plt.subplot(211) -- Here, plt . subplot of figure(211) -- Rows, Columns of 1st Subplot

# plt.plot(t1,f(t1),'b-') -- Here, plt.plot of x,y we need to declare know -
# Now instead of 'x' and 'y' i'm taking 'x' as 't1', 'y' as function of 't1' -- f(t1) and taking
# Blue Color with an a hyfen(-) --> 'b-'

# plt.plot(t2,np.cos(2 * np.pi * t2),'r--') -- Here, i'm Declaring Properities --
# plt.plot(t2('x' axis), numpy dot. cos of (2 * numpy dot. pi value * t2), Red color double hyfen --)
```

In [48]:
```
def f (t):
    return np.exp(-t) * np.cos(2 * np.pi * t)
t1 = np.arange(0.0,5.0,0.1)
t2 = np.arange(0.0,5.0,0.02)

plt.figure(1)
plt.subplot(211)
plt.grid()
plt.plot(t1,f(t1),'b-')

plt.subplot(212)
plt.plot(t2,np.cos(2 * np.pi * t2),'r--')
plt.show()
```



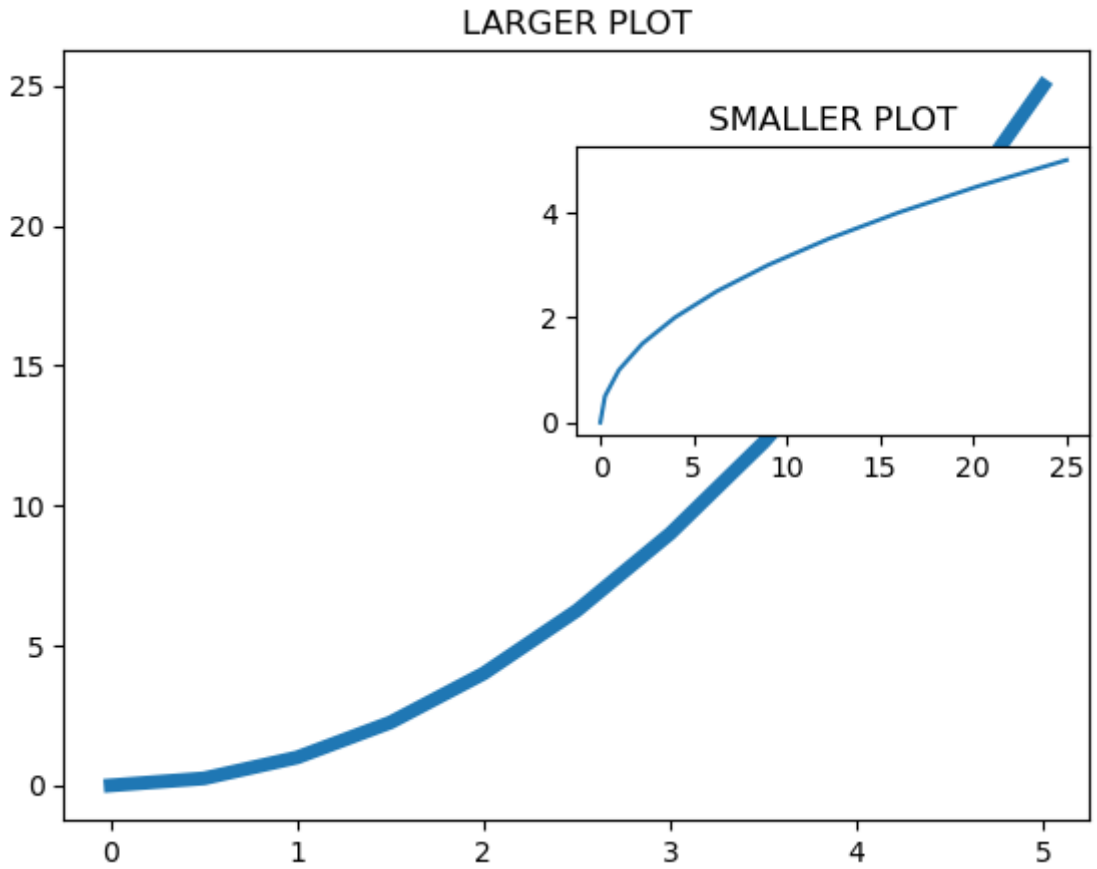
11. GRAPH OF GRAPH : INNER GRAPH

PAGE 226

In [53]:
```
# Here, We can change the Co-ordinates of 'inner point' - Called as 'Plotting of Inner Point'
# Graph of Graph --> Considered as 'Inner Graph'
```

In [201]:
```
x = np.linspace(0,5,11)
y = x ** 2

fig = plt.figure()

axes1 = fig.add_axes([0.1,0.1,0.8,0.8])
axes2 = fig.add_axes([0.5,0.5,0.4,0.3])

axes1.plot(x,y , linewidth = 5.0)  # here i was added 'linewidth'  by my own.
axes1.set_title("LARGER PLOT")

axes2.plot(y,x)   # Here, We gave 'y' to 'x' axes
axes2.set_title("SMALLER PLOT")
```

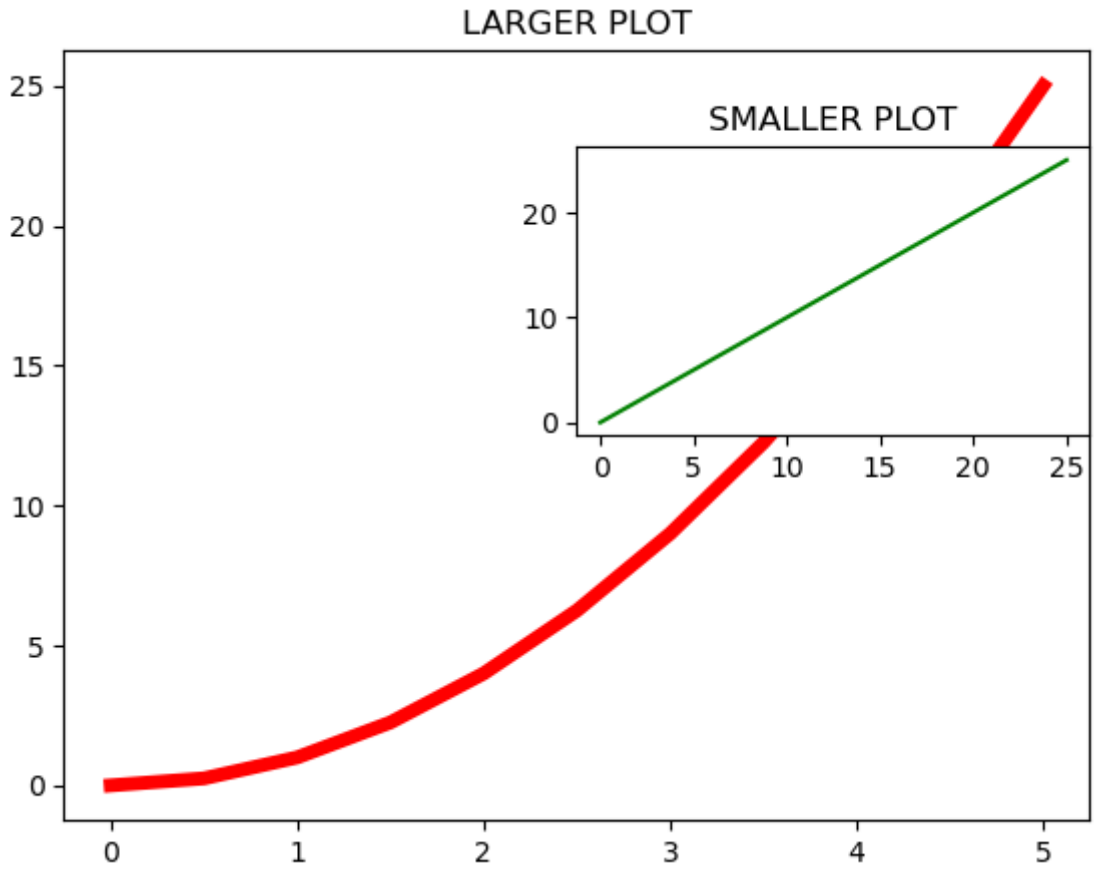
Out[201]:
```
Text(0.5, 1.0, 'SMALLER PLOT')
```



Now, Changing 'line color' and 'axes' to the 11th sum :

In [216]:
```
x = np.linspace(0,5,11)
y = x ** 2

fig = plt.figure()

axes1 = fig.add_axes([0.1,0.1,0.8,0.8])
axes2 = fig.add_axes([0.5,0.5,0.4,0.3])

axes1.plot(x,y , linewidth = 5.0, color = 'r')  # here i was added 'linewidth'  by my own.
axes1.set_title("LARGER PLOT")

axes2.plot(y,y, color = 'g')   # Here, We gave 'y' to 'x' axes
axes2.set_title("SMALLER PLOT")
```

Out[216]:
```
Text(0.5, 1.0, 'SMALLER PLOT')
```



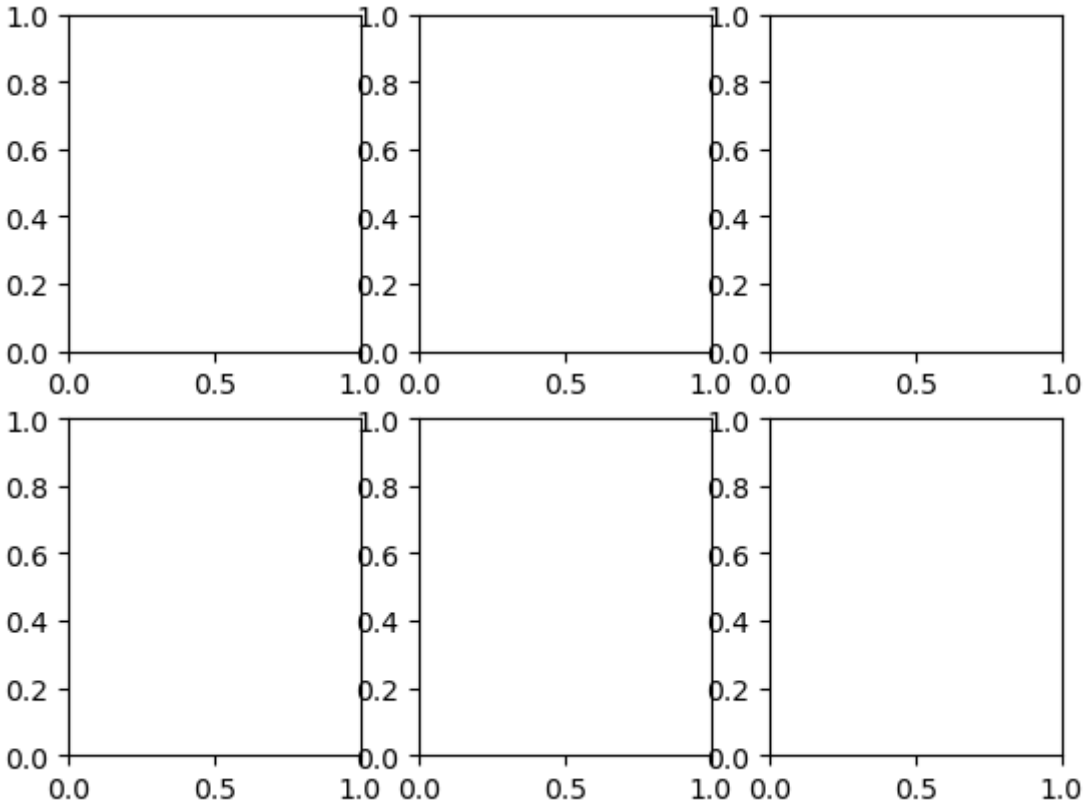
12. Calling 'SUBPLOTS' : not 'Subplot'

page 227

In [ ]:
```
# Here, 'fig,axes' are 'Two left hand variables', means we are calling 'Subplots' not an 'Subplot'
```

In [84]:
```
fig = plt.figure()
fig,axes = plt.subplots(nrows = 2,ncols = 3)
```

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



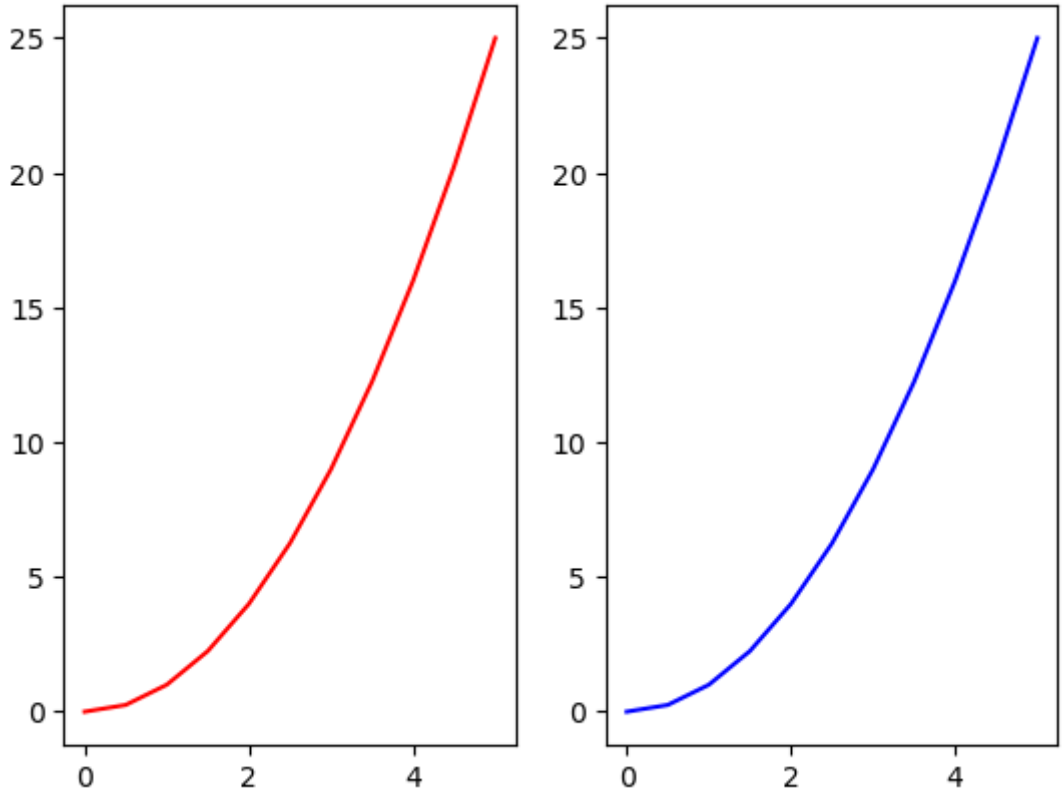
13. DECLARE THE PROPERTIES :

PAGE 228

In [85]:
```
# We can declare the Properties based on our requirement :
```

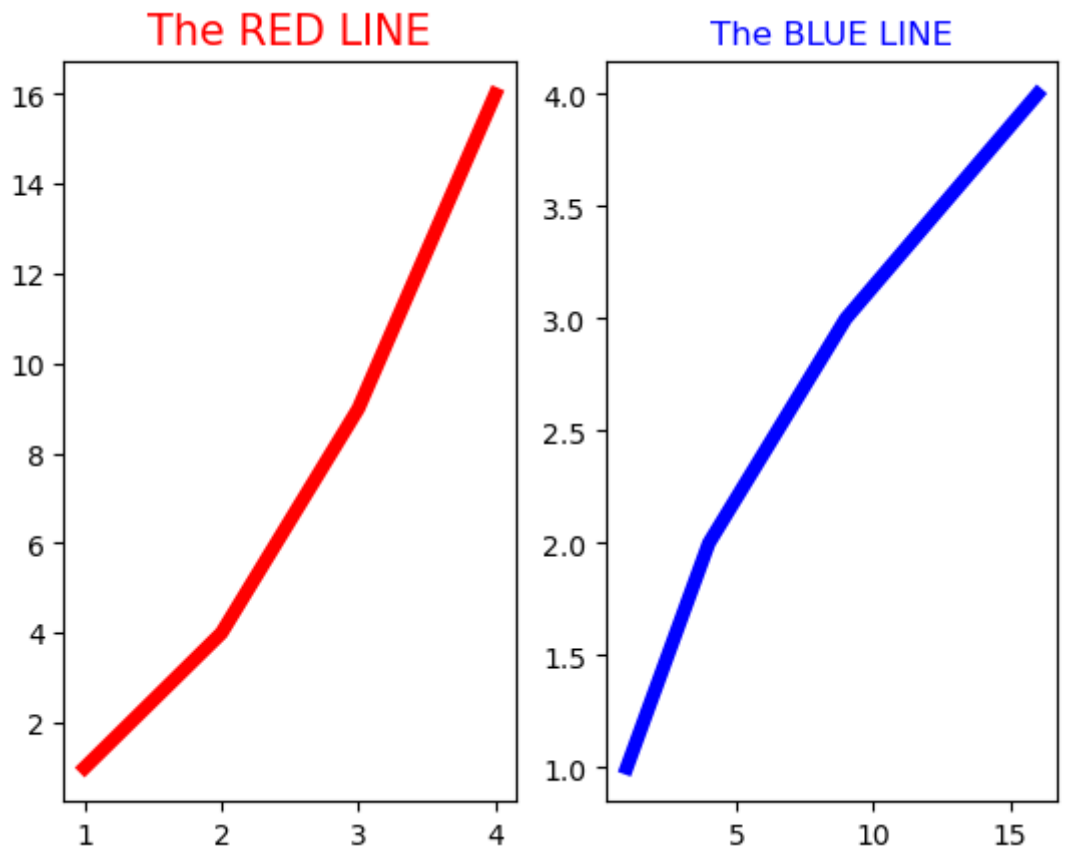
In [88]:
```
plt.subplot(1,2,1)
plt.plot(x,y,'r')
plt.subplot(1,2,2)
plt.plot(x,y,'b')
```

Out[88]:
```
[<matplotlib.lines.Line2D at 0x2d1b3ed4070>]
```



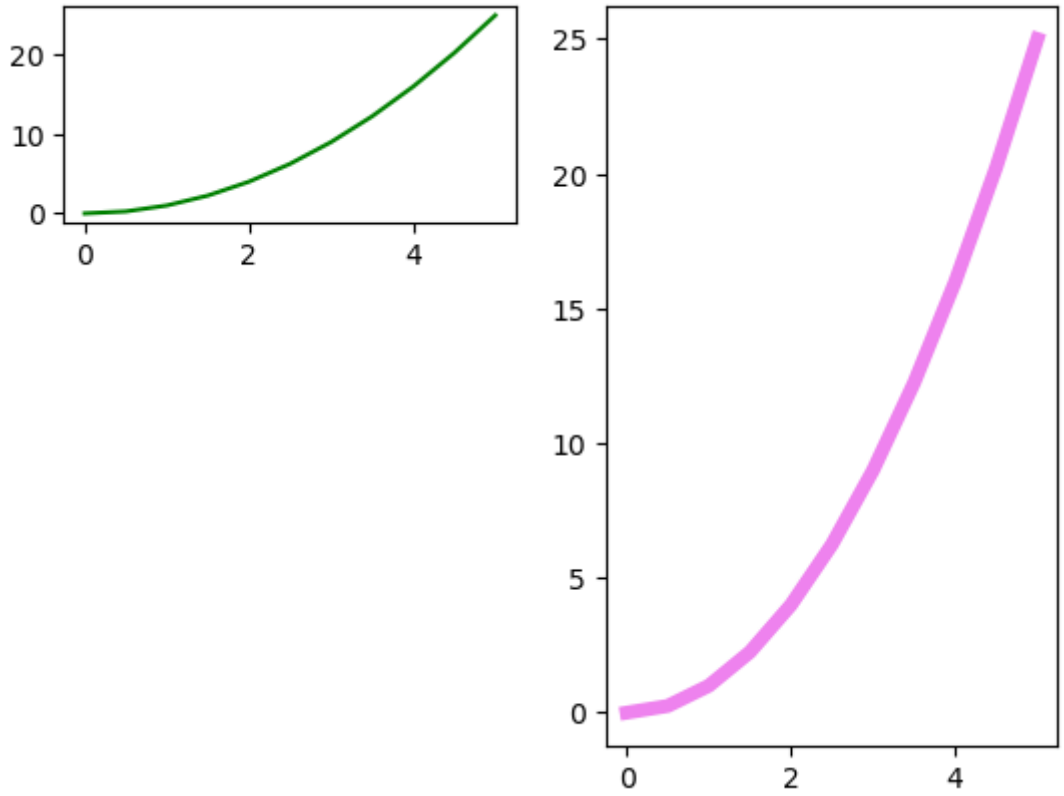
In [264]:
```
# Now, i'm adding Titles and Colors to the Titles :
# Here, System defaultly printing size = '12', Now, i'm changing it to '15' to the Title Red line.
# And also Changing the 'axes' and the 'line width(lw)'

plt.subplot(1,2,1)
plt.plot(x,y,'r', lw = 5.0)
plt.title("The RED LINE", color = 'r', size = 15)
plt.subplot(1,2,2)
plt.plot(y,x,'b', lw = 5.0)
plt.title("The BLUE LINE", color = 'b')
```

Out[264]: Text(0.5, 1.0, 'The BLUE LINE')



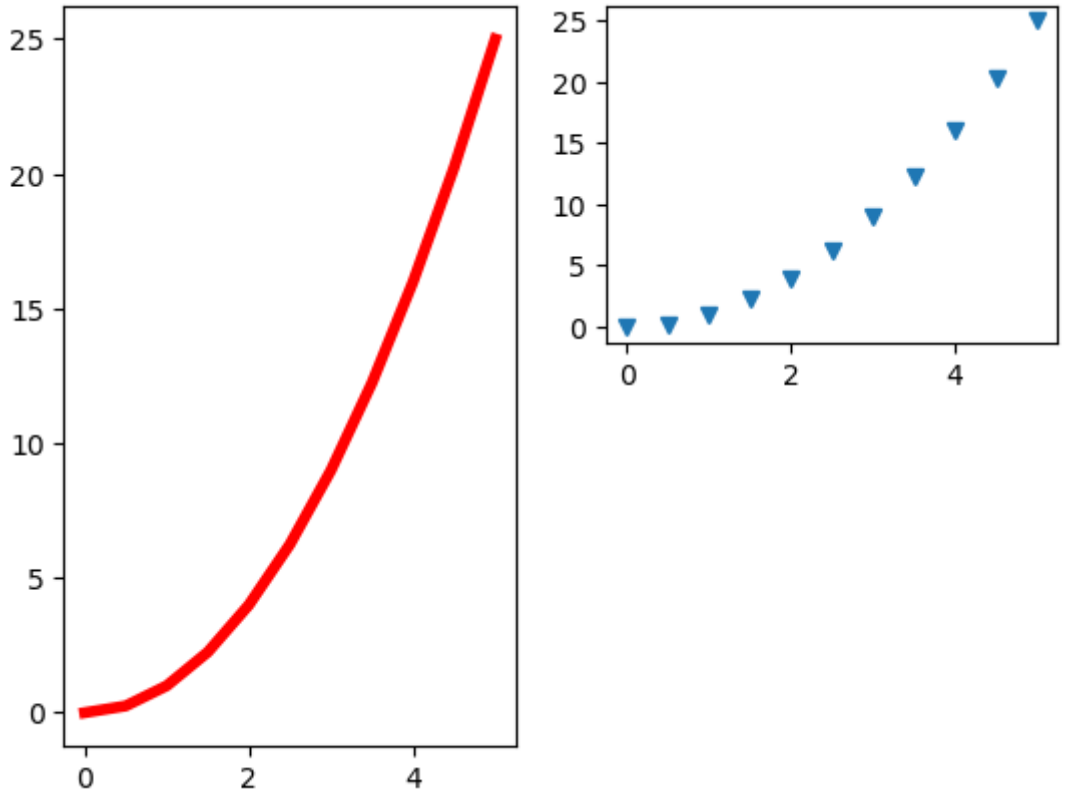
In [109]:
```
plt.subplot(3,2,1)
plt.plot(x,y,'g')
plt.subplot(1,2,2)
plt.plot(x,y,'violet', linewidth = 5.0)
```

Out[109]:
```
[<matplotlib.lines.Line2D at 0x2d1b6db4430>]
```



In [114]:
```
plt.subplot(1,2,1)
plt.plot(x,y,'r', linewidth = 4.0)
plt.subplot(2,2,2)
plt.plot(x,y,'v')
```

Out[114]:
```
[<matplotlib.lines.Line2D at 0x2d1b6e65490>]
```

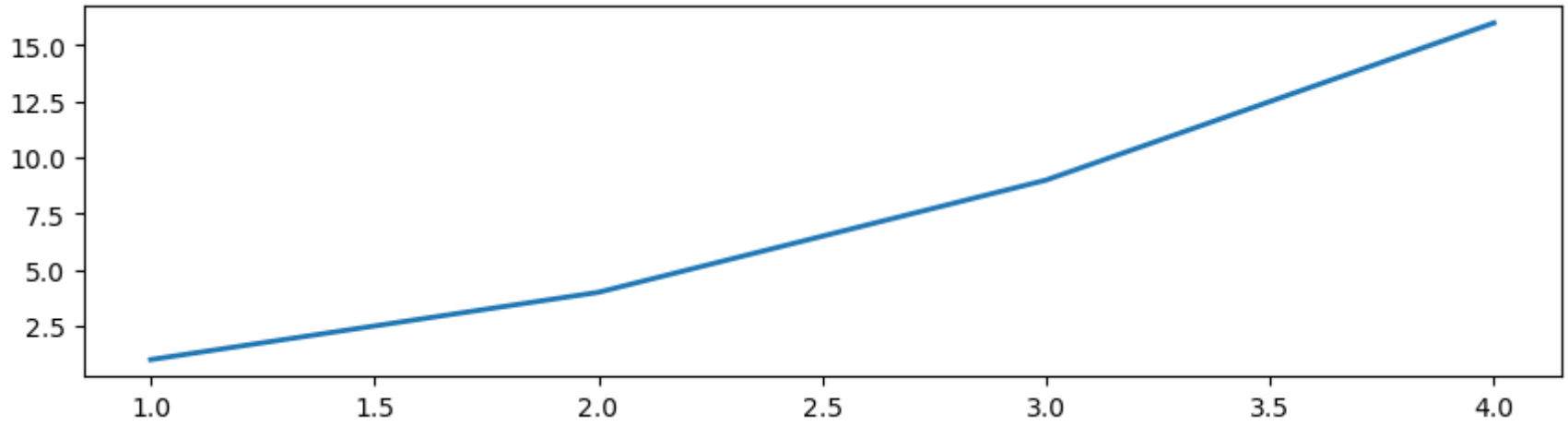


14. Now i'm Calling libraries -- matplotlib and also numpy :

In [116]:
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [221]:
```
x = [1,2,3,4]
y = [1,4,9,16]
fig = plt.figure(figsize = (8,2))  # 8 of 2 means 8 inches 'Width' and 2 inches 'Height'
ax = fig.add_axes([0,0,1,1])
ax.plot(x,y, lw = 2)
```

Out[221]:
```
[<matplotlib.lines.Line2D at 0x2d1bbeecd30>]
```

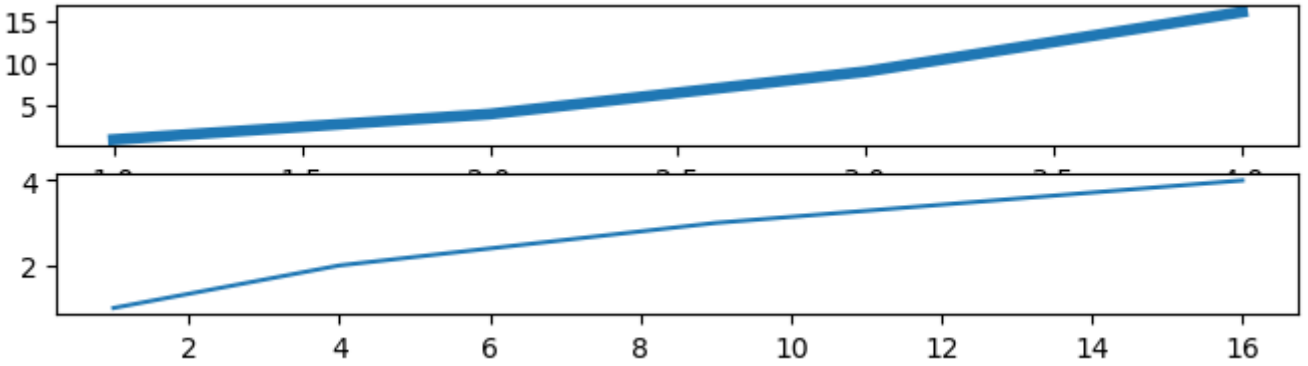


15. SUBPLOTS CONCEPT :

PAGE 230

In [225]:
```
fig,axes = plt.subplots(nrows = 2, ncols = 1, figsize = (8,2))
axes[0].plot(x,y, linewidth = 4.0)
axes[1].plot(y,x)
```

Out[225]:
```
[<matplotlib.lines.Line2D at 0x2d1bd1ba220>]
```



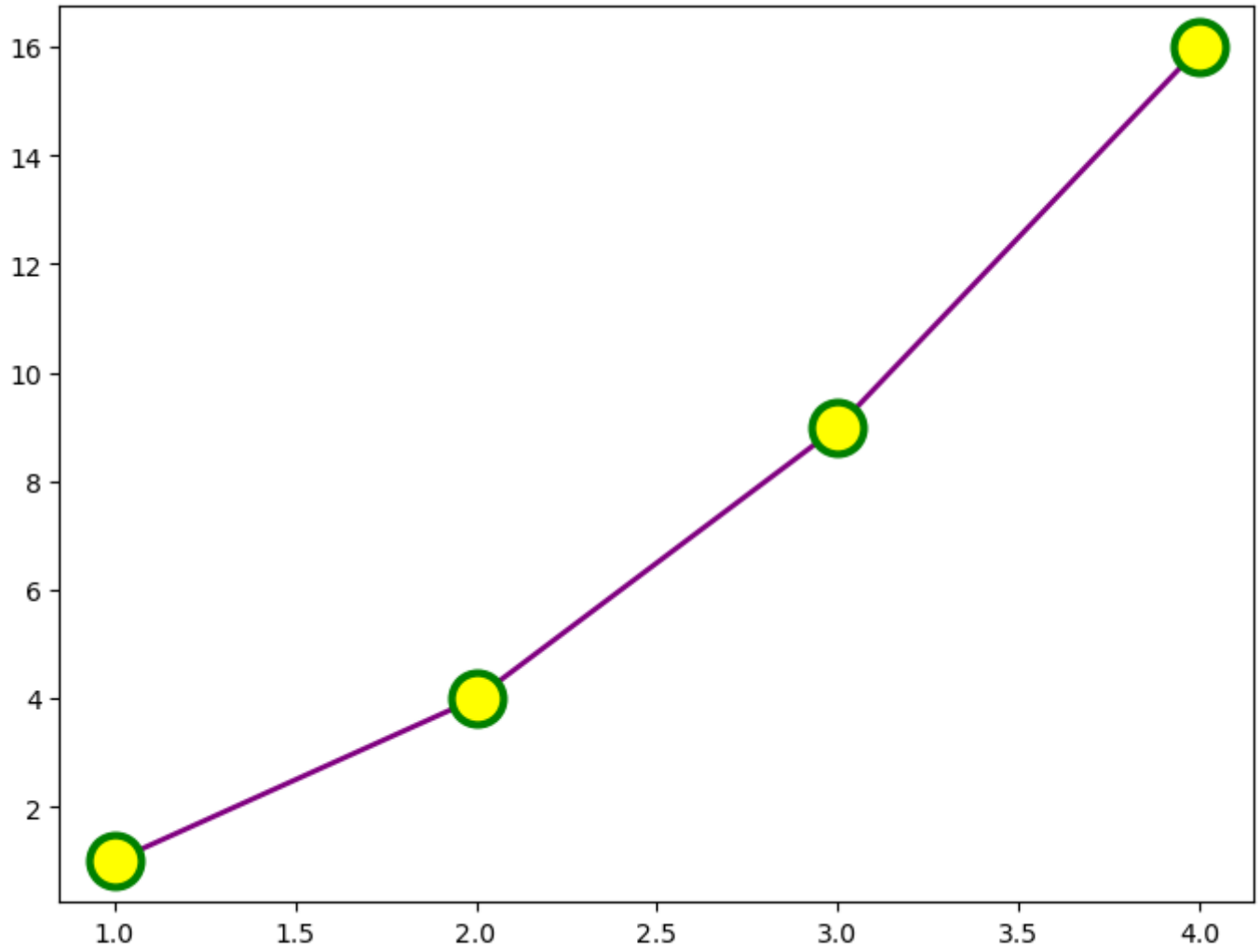
16. DECLARE COMPLETE PARAMETERS :

PAGE 230

In [ ]:
```
# 'lw' means, line width
# 'ls' means, line shape
```

In [140]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])

ax.plot(x,y,color = 'purple',lw = 2, ls = '-', marker = 'o',
        markersize = 20, markerfacecolor = 'yellow', markeredgewidth = 3, markeredgecolor = 'green')
```

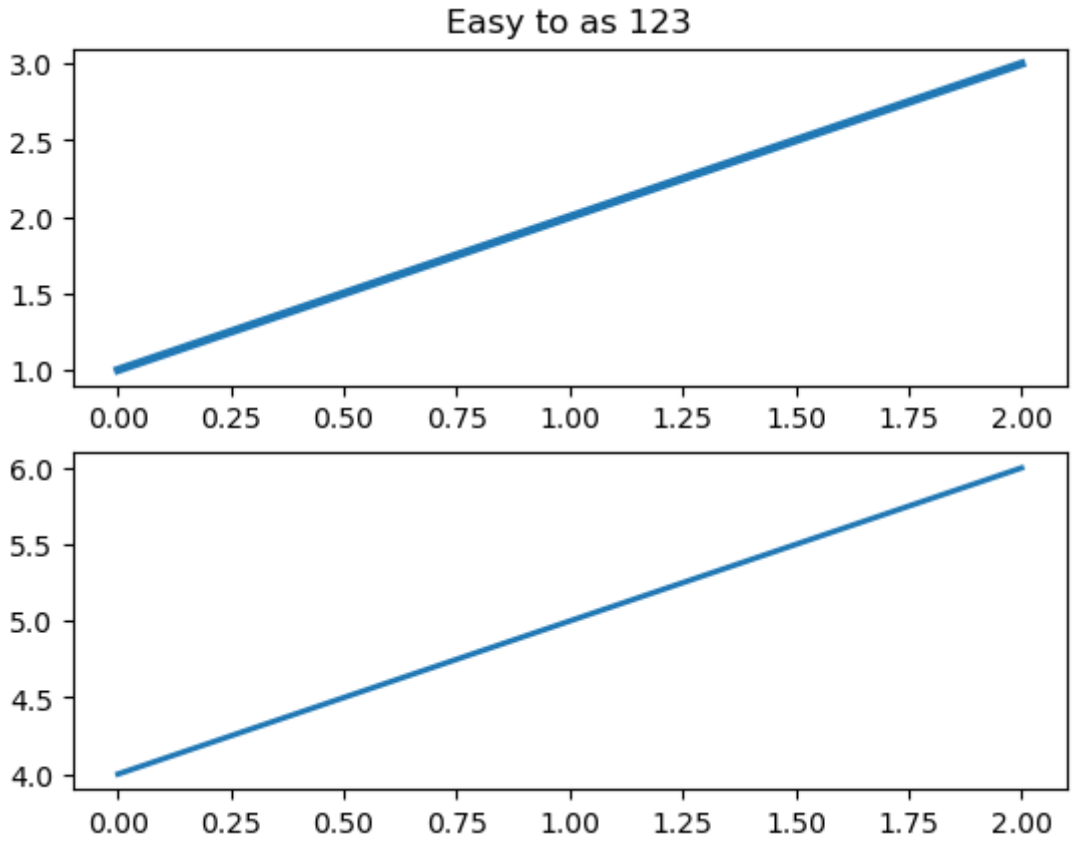
Out[140]: [<matplotlib.lines.Line2D at 0x2d1b835f0d0>]



17. Subplots : another example

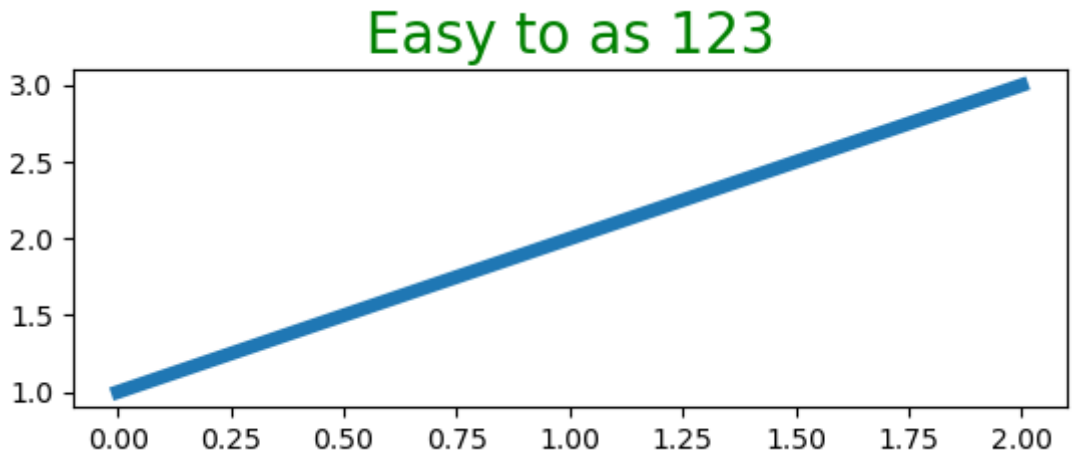
page 231

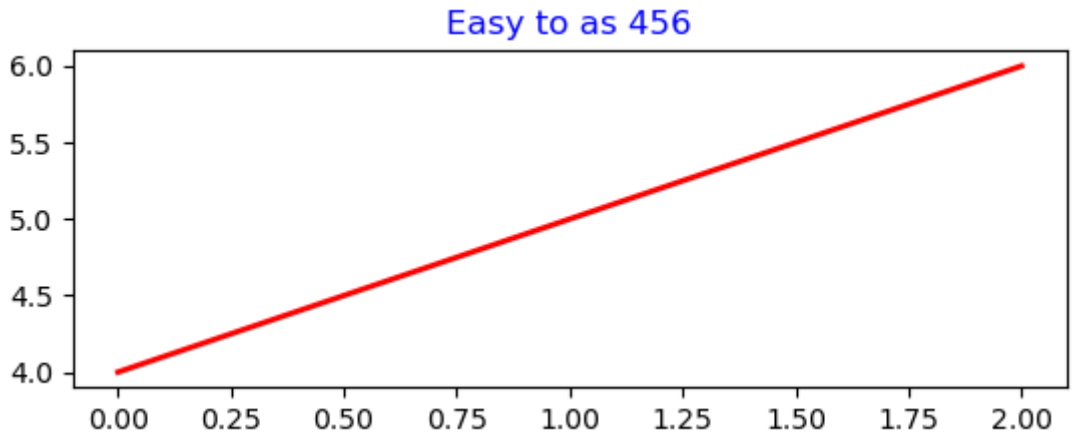
In [168]:
```
plt.figure(1)
plt.subplot(211)
plt.plot([1,2,3], lw = 3)

plt.subplot(212)
plt.plot([4,5,6], lw = 2)

plt.figure(1)
plt.subplot(211)
plt.title("Easy to as 123")
plt.show()
```



We can give 'Title' also to 2nd subplot : And also Size and colors to the Titles : 17th sum values

In [248]:
```
plt.figure(1)
plt.subplot(211)
plt.plot([1,2,3], lw = 5,)
plt.title("Easy to as 123", color = 'green', size = 20)
plt.show()

plt.subplot(212)
plt.plot([4,5,6], lw = 2.0, color = 'r')
plt.title("Easy to as 456", color = 'blue')
plt.show()
```





In [ ]: