Matplotlib

What is Matplotlib?

- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Installation

In [3]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\teks108\anaconda3\lib\site-pack
ages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\teks108\anaconda3\lib\sit
e-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\teks108\anaconda3\lib\site-pa
ckages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\teks108\anaconda3\lib\si
te-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\teks108\anaconda3\lib\si
te-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\teks108\anaconda3\lib\site-pac
kages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\teks108\anaconda3\lib\site
-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\teks108\anaconda3\lib\site-p
ackages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\teks108\anaconda3\li
b\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\teks108\anaconda3\lib
\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\teks108\anaconda3\lib\site-packag
es (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

Importing Matplotlib

- After successfully installing Matplotlib, You can use this command to import Matplotlib on your system.

In [4]:
```
import matplotlib
```

Checking Matplotlib Version

The version string is stored under **version** attribute.

In [16]:
```
import matplotlib

print(matplotlib.__version__)
```

```
3.7.2
```

Matplotlib Pyplot

Pyplot

- Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

In [17]:
```
import matplotlib.pyplot as plt
```

Types of Matplotlib

Matplotlib comes with a wide variety of plots. Plots help to understand trends, and patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

- Matplotlib Line Plot
- Matplotlib Bar Plot
- Matplotlib Histograms Plot
- Matplotlib Scatter Plot
- Matplotlib Pie Charts
- Matplotlib Area Plot

Matplotlib Line Plot

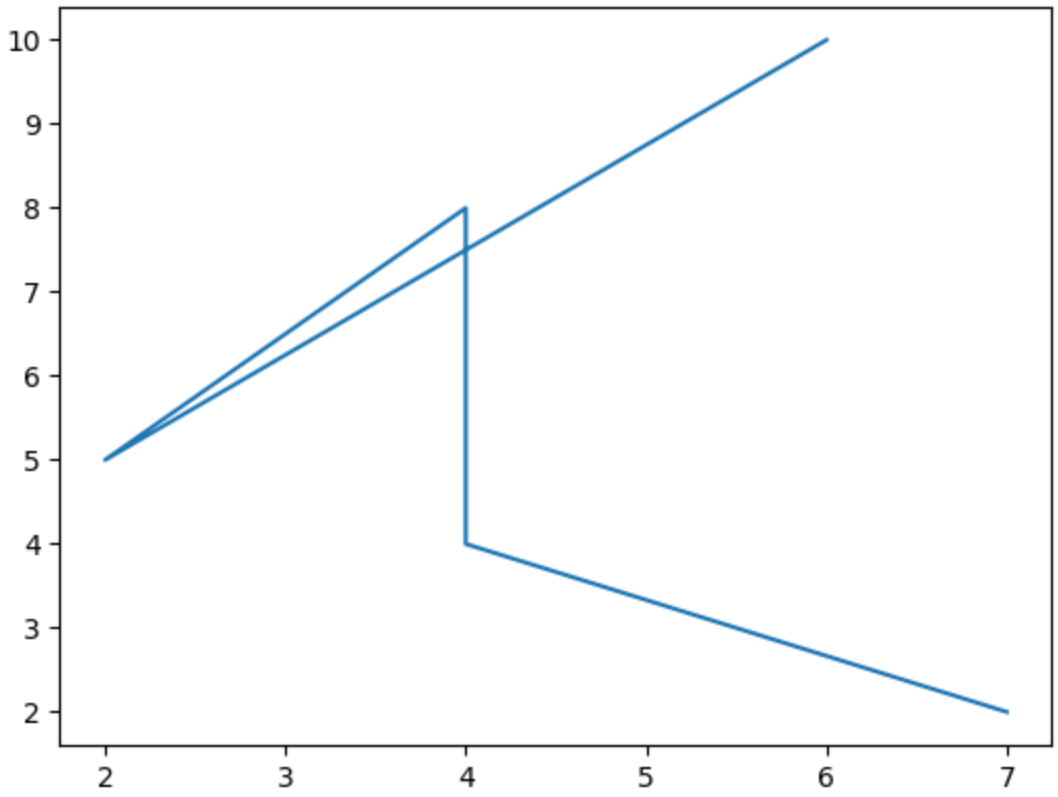
In [6]:
```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [6, 2, 4, 4, 7]

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

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

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



Matplotlib Bar Plot

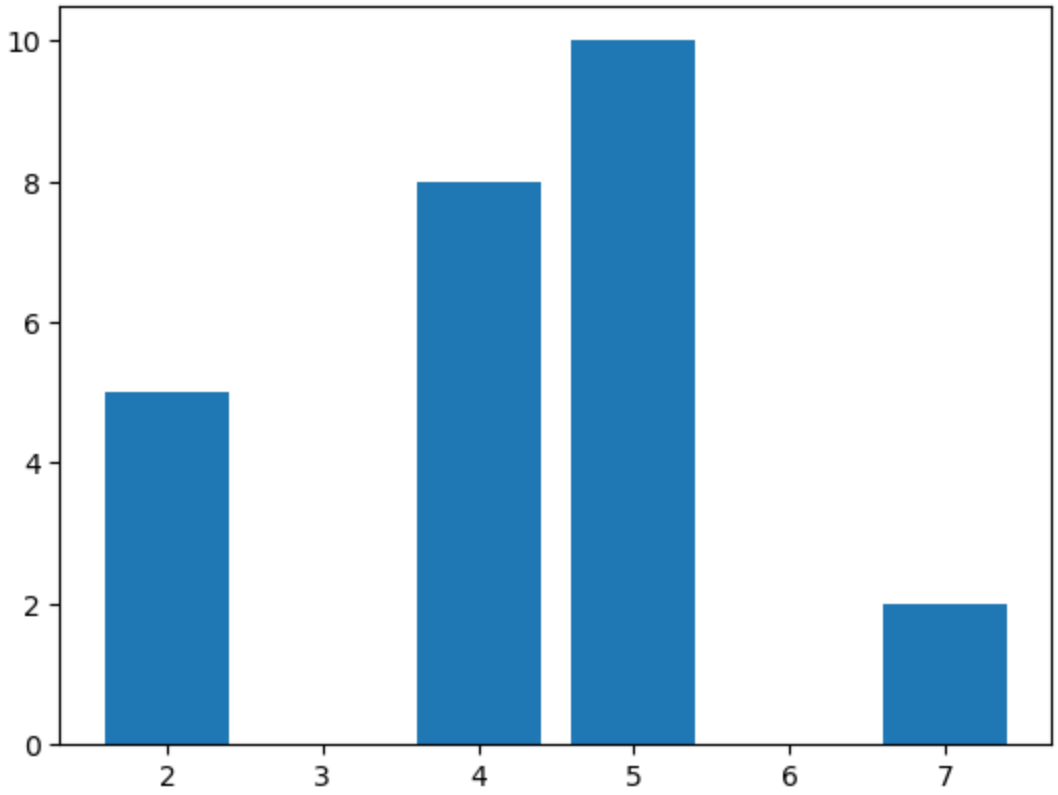
In [8]:
```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 4, 4, 7]

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

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

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



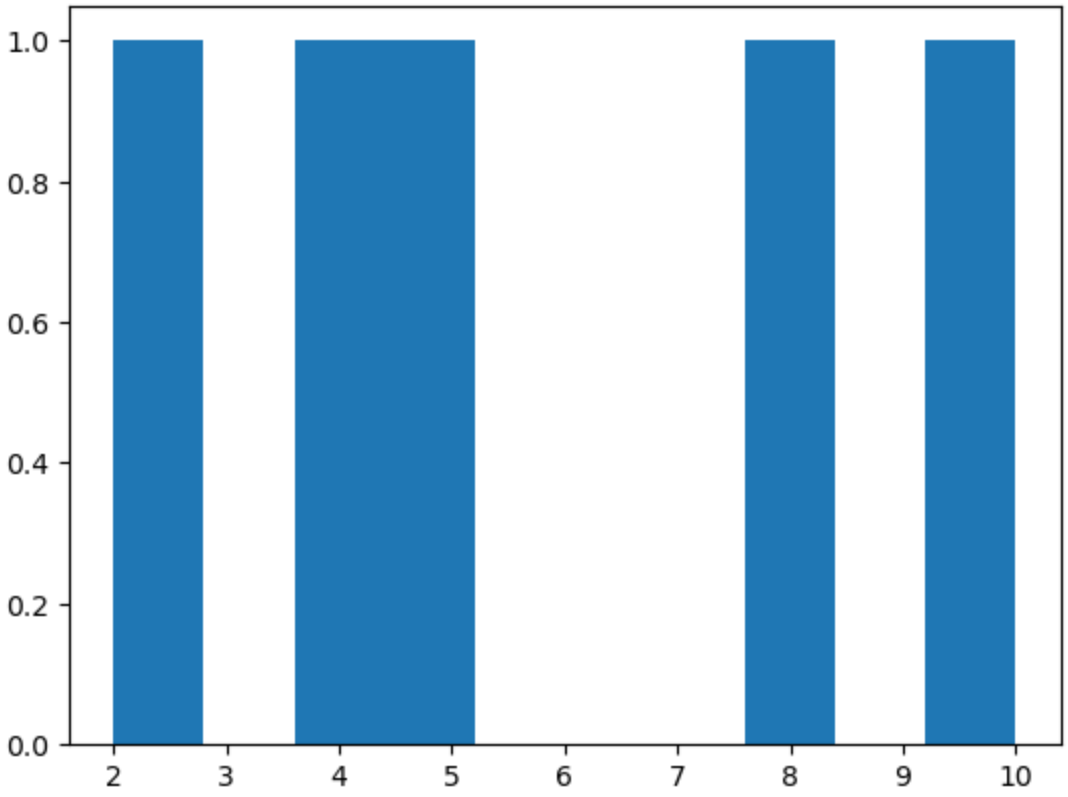
Matplotlib Histograms Plot

In [9]:
```
# importing matplotlib module
from matplotlib import pyplot as plt

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

# Function to plot histogram
plt.hist(y)

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



Matplotlib Scatter Plot

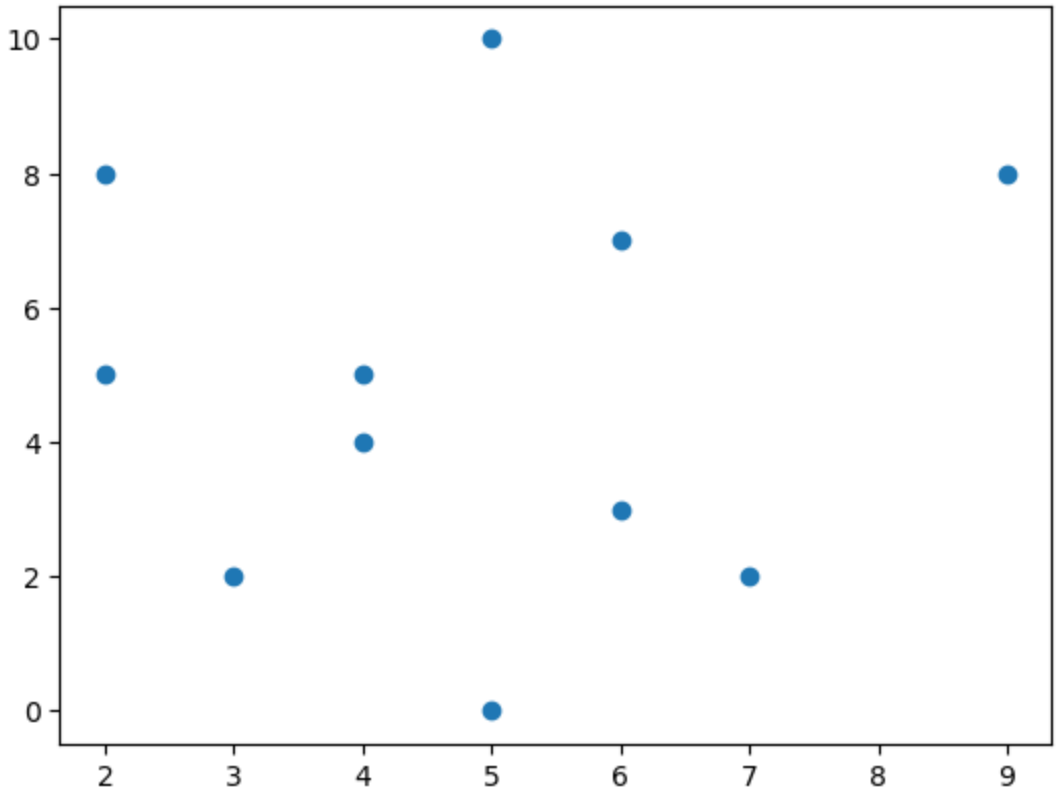
In [11]:
```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7, 4, 6, 3, 6, 2, 5]

# Y-axis values
y = [10, 5, 8, 4, 2, 5, 3, 2, 7, 8, 0]

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

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

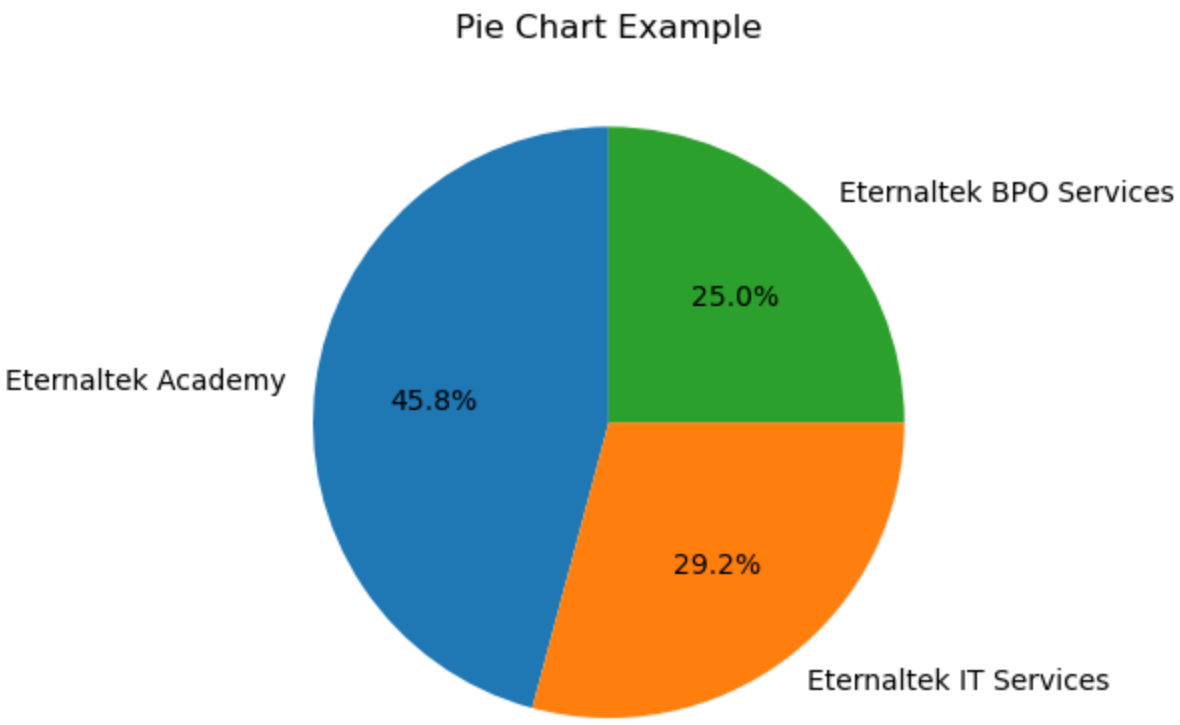


Matplotlib Pie Charts

In [14]:
```
import matplotlib.pyplot as plt

# Data for the pie chart
labels = ['Eternaltek Academy', 'Eternaltek IT Services', 'Eternaltek BPO Services']
sizes = [55, 35, 30]

# Plotting the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart Example')
plt.show()
```



Matplotlib Area Plot

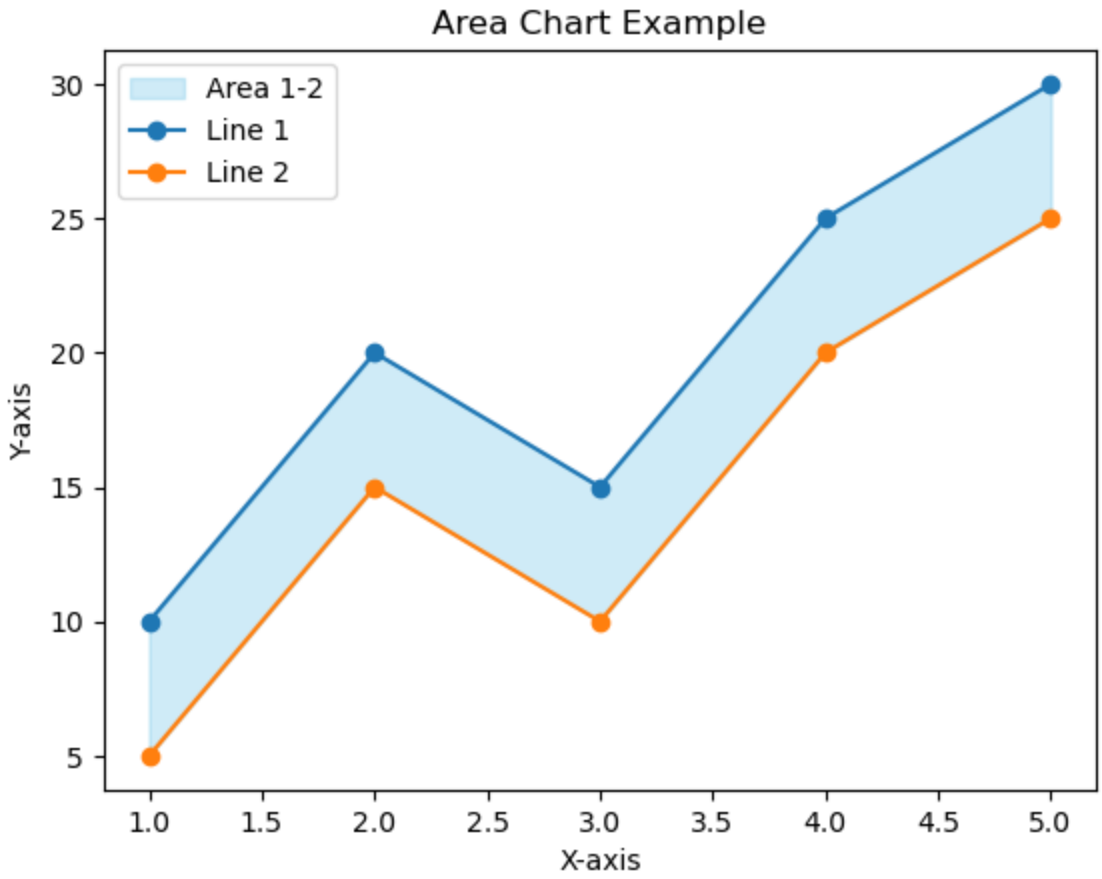
In [15]:
```
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y1, y2 = [10, 20, 15, 25, 30], [5, 15, 10, 20, 25]

# Area Chart
plt.fill_between(x, y1, y2, color='skyblue', alpha=0.4, label='Area 1-2')
plt.plot(x, y1, label='Line 1', marker='o')
plt.plot(x, y2, label='Line 2', marker='o')

# Labels and Title
plt.xlabel('X-axis'), plt.ylabel('Y-axis'), plt.title('Area Chart Example')

# Legend and Display
plt.legend(), plt.show()
```



Out[15]: (<matplotlib.legend.Legend at 0x158313e6250>, None)

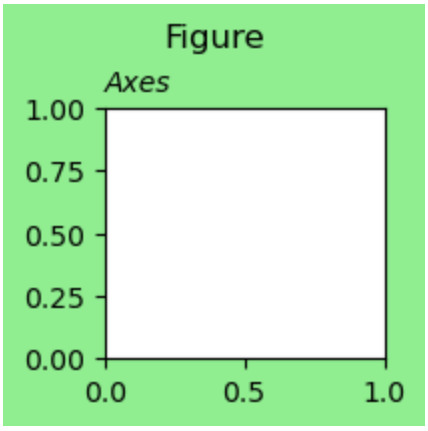
Introduction to Figures

In [21]:
```
#Constrained layout automatically adjusts subplots



fig = plt.figure(figsize=(2, 2), facecolor='lightgreen', layout='constrained')
fig.suptitle('Figure')
ax = fig.add_subplot()
ax.set_title('Axes', loc='left', fontstyle='oblique', fontsize='medium')
```

Out[21]: Text(0.0, 1.0, 'Axes')

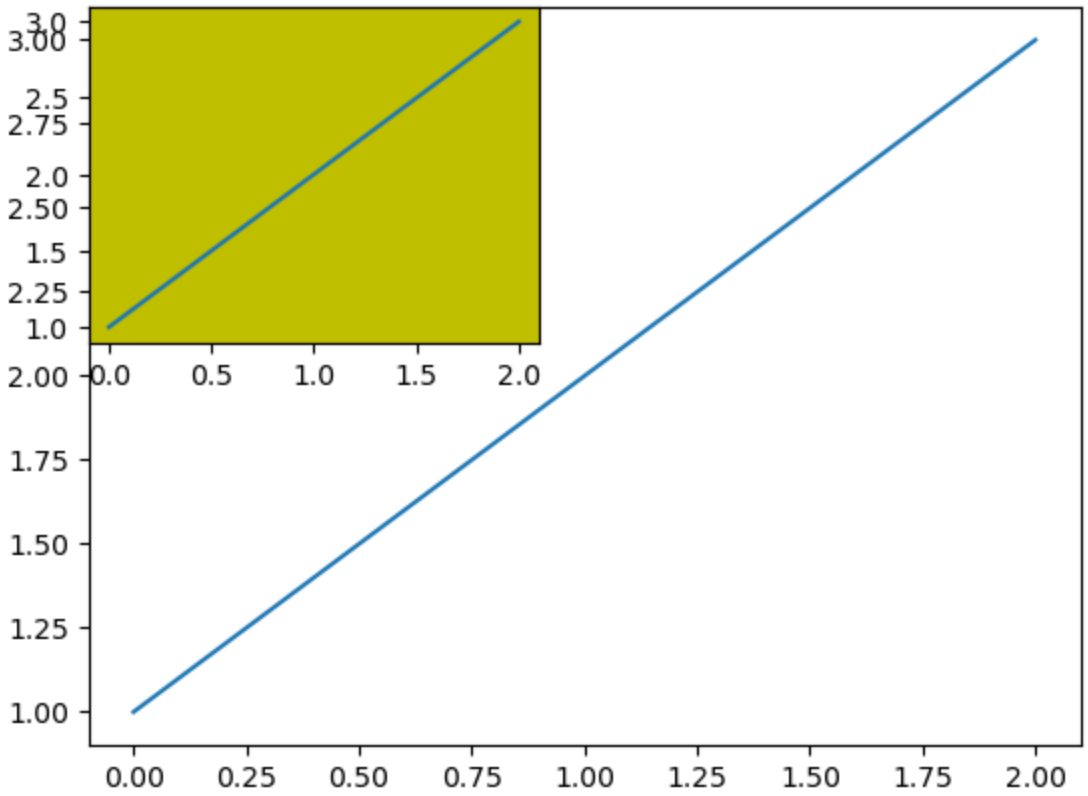


subplot() function

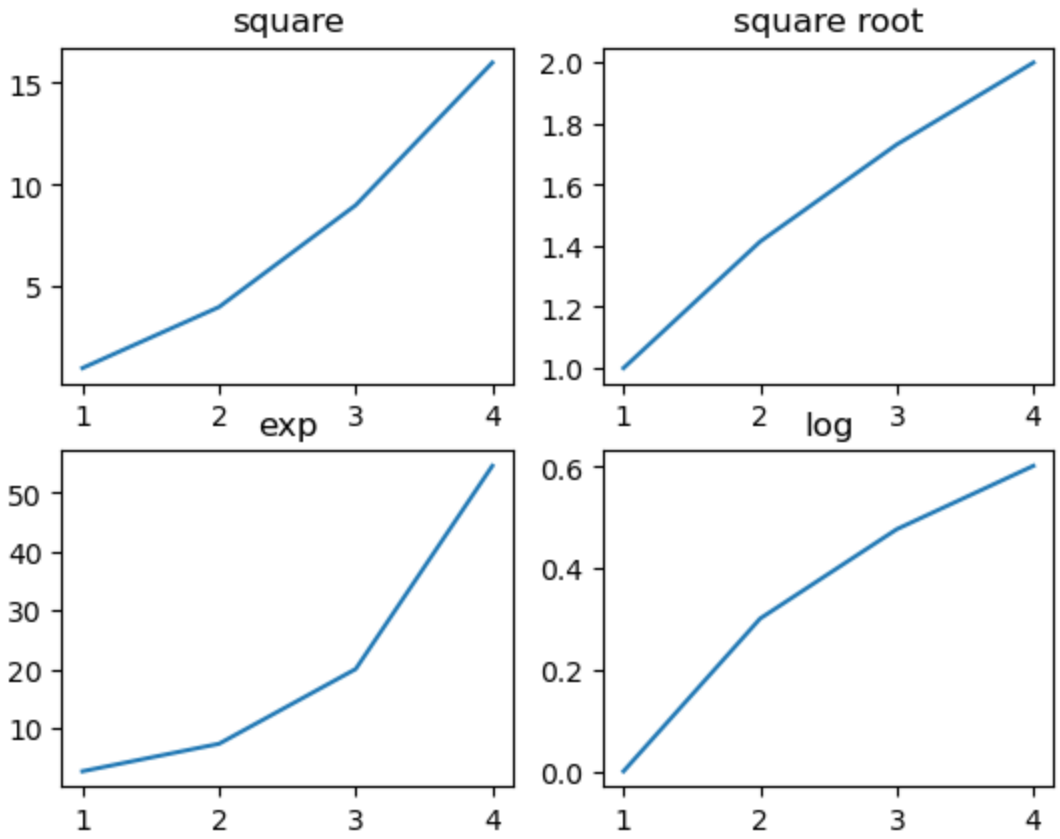
In [23]:
```
# plt.subplot(subplot(nrows, ncols, index)
```

In [24]:
```
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot([1,2,3])
ax2 = fig.add_subplot(221, facecolor='y')
ax2.plot([1,2,3])
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x158312a6a50>]
```

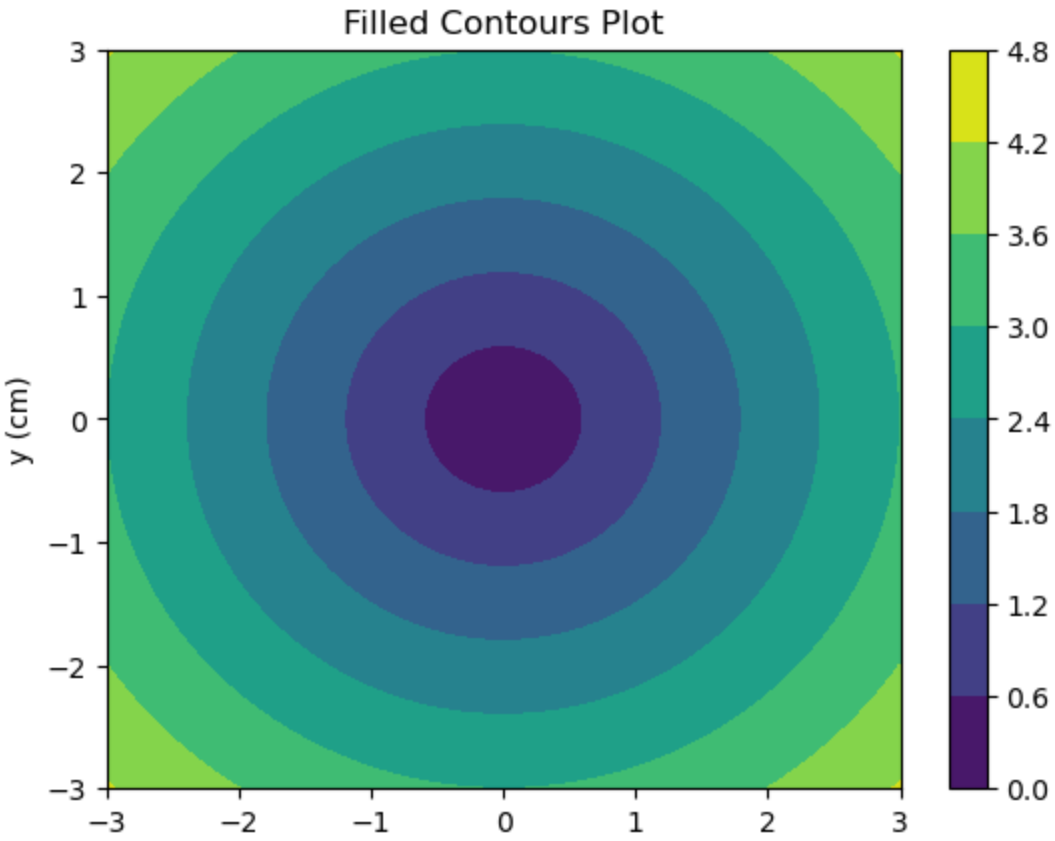


In [25]:
```
fig,a =  plt.subplots(2,2)
import numpy as np
x = np.arange(1,5)
a[0][0].plot(x,x*x)
a[0][0].set_title('square')
a[0][1].plot(x,np.sqrt(x))
a[0][1].set_title('square root')
a[1][0].plot(x,np.exp(x))
a[1][0].set_title('exp')
a[1][1].plot(x,np.log10(x))
a[1][1].set_title('log')
plt.show()
```



Matplotlib Contour Plot

In [26]:
```
xlist = np.linspace(-3.0, 3.0, 100)
ylist = np.linspace(-3.0, 3.0, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = np.sqrt(X**2 + Y**2)
fig,ax=plt.subplots(1,1)
cp = ax.contourf(X, Y, Z)
fig.colorbar(cp) # Add a colorbar to a plot
ax.set_title('Filled Contours Plot')
#ax.set_xlabel('x (cm)')
ax.set_ylabel('y (cm)')
plt.show()
```



Matplotlib Violin Plot

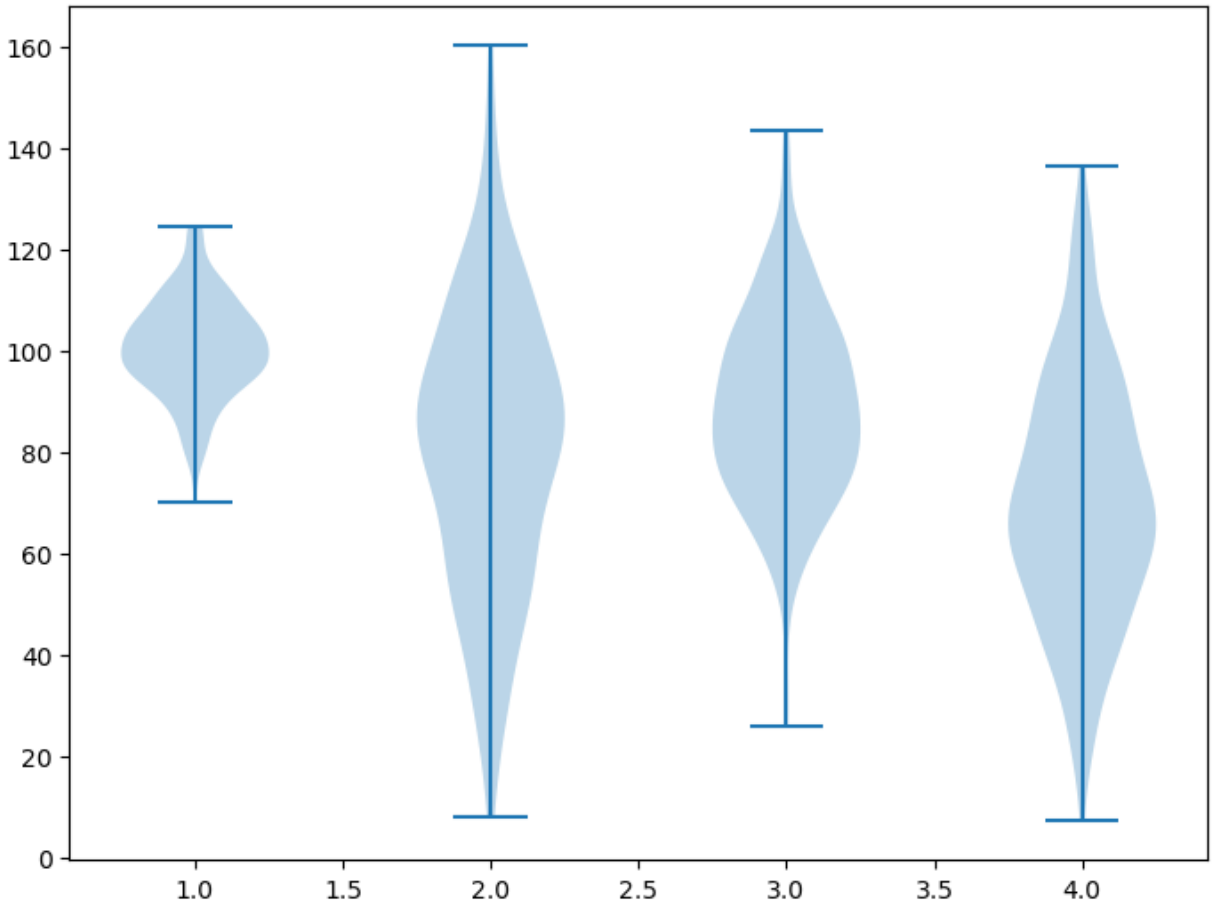
In [27]:
```
np.random.seed(10)
collectn_1 = np.random.normal(100, 10, 200)
collectn_2 = np.random.normal(80, 30, 200)
collectn_3 = np.random.normal(90, 20, 200)
collectn_4 = np.random.normal(70, 25, 200)

## combine these different collections into a list
data_to_plot = [collectn_1, collectn_2, collectn_3, collectn_4]

# Create a figure instance
fig = plt.figure()

# Create an axes instance
ax = fig.add_axes([0,0,1,1])

# Create the boxplot
bp = ax.violinplot(data_to_plot)
plt.show()
```



In [ ]: