Prepared by -: Vithusan

MatplotLib Tutorial:

What it is: A powerful Python library for data visualization.

What it does: Creates various plots and charts:

- Line charts (trends over time/space)
- Bar charts (comparisons & frequencies)
- Scatter plots (relationships & clusters)
- Histograms (data distributions)
- Box plots (group comparisons & outliers)

Why use it:

- **Bring data to life:** Make trends, patterns, and insights visually apparent.
- **Communicate effectively:** Share findings through clear and informative plots.
- **Analyze data efficiently:** Visual exploration can reveal hidden relationships and anomalies.

In [41]:
```
# Import Matplotlib library
import matplotlib.pyplot as plt

# Display plots directly in the Jupyter Notebook
%matplotlib inline
```

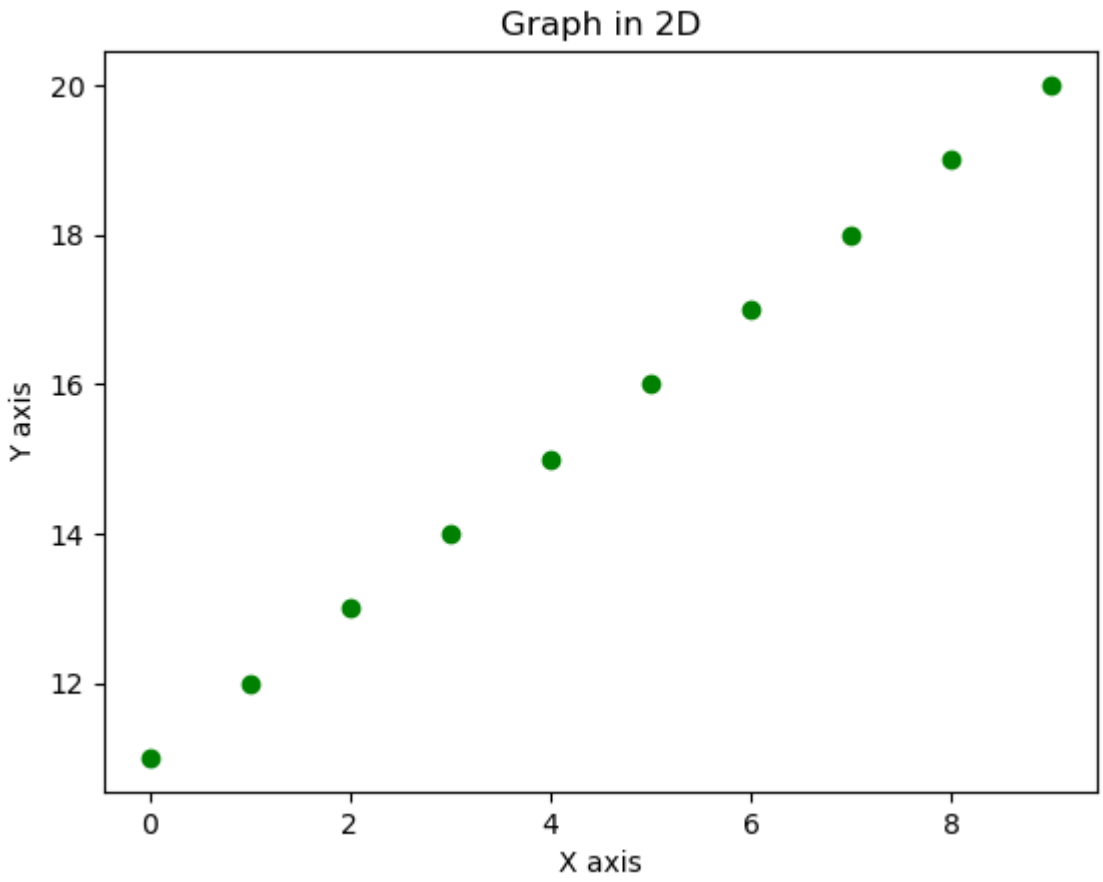
- `%matplotlib inline` is a Jupyter Notebook magic command that ensures plots are displayed inline, directly below the code cells that generate them.

In [42]:
```
import numpy as np
```

In [43]:
```
#simple Example

x = np.arange(0, 10)
y = np.arange(11, 21)
```

In [44]:
```
#Plotting using matplotlib

plt.scatter(x, y, c = "g")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Graph in 2D")
```

Out[44]: `Text(0.5, 1.0, 'Graph in 2D')`



scatter

- A scatter plot is a type of data visualization that displays individual data points on a two-dimensional graph.
- Each point represents the values of two variables, one plotted along the x-axis and the other along the y-axis.

Purpose:

- **Visualizing Relationships:** Scatter plots are particularly useful for revealing relationships or correlations between two variables.
- **Identifying Patterns:** Patterns such as trends, clusters, or outliers become apparent when examining the arrangement of points.

In [45]:
```
# Save the current figure as an image file named "firstOutput.png"
plt.savefig("firstOutput.png")
```

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

Matplotlib's `plt.plot()` Function:

- **Purpose:** Used for creating line plots, depicting the relationship between two variables.
- **Parameters:**
  - `x` and `y` : Arrays or sequences representing data points along the X and Y axes.
  - `label` : Provides a label for the line, useful for legend.
  - `color` : Specifies line color.
  - `linestyle` : Defines line style ('-', '--', ':', etc.).
  - `linewidth` : Sets line width.
  - `marker` : Indicates marker style for data points.
  - `markersize` : Specifies marker size.
- **Key Concepts:**
  - **Line Styles:**
    - `'-'` (solid line)
    - `'--'` (dashed line)
    - `':'` (dotted line)
    - `'-.'` (dash-dot line)
  - **Colors:**
    - `'b'` (blue)
    - `'g'` (green)
    - `'r'` (red)
    - `'c'` (cyan)
    - `'m'` (magenta)
    - `'y'` (yellow)
    - `'k'` (black)
    - `'w'` (white)
  - **Markers:**
    - `'o'` (circle)
    - `'s'` (square)
    - `'^'` (triangle up)
    - `'v'` (triangle down)
    - `'+'` (plus)
    - `'*'` (star)
- **Labels and Titles:**
  - `xlabel` and `ylabel` : Add labels to the X and Y axes.
  - `title` : Sets the title of the plot.
- **Legend:**
  - `legend` : Displays a legend if labels are provided, helping to identify multiple lines.
- **Show:**
  - `show()` : Displays the plot.
- **Customization:**

- The function is highly customizable, allowing users to adjust various aspects of the plot for better visualization.

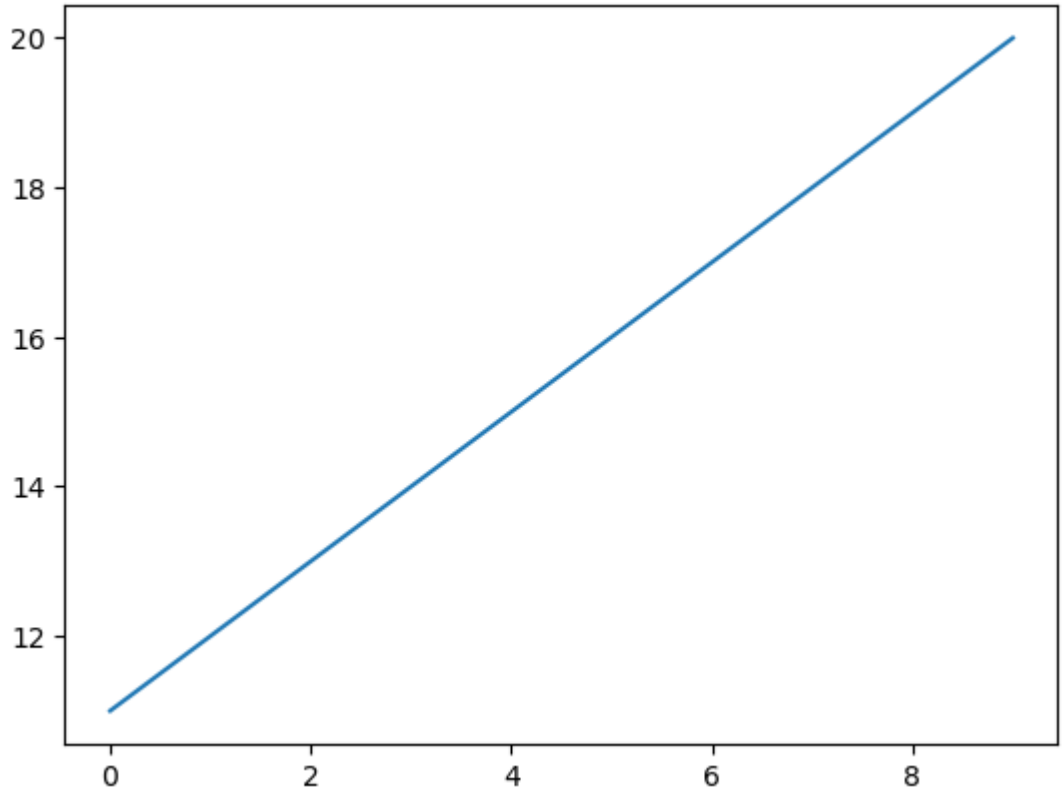
- **Versatility:**
  - Suitable for various types of line plots, making it a fundamental tool for visualizing trends, patterns, and relationships in data.

Example Usage:

```
plt.plot(x, y, label='Line Plot', color='blue', linestyle='-', linewidth=
2, marker='o', markersize=8)
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Line Plot Example")
plt.legend()
plt.show()
```

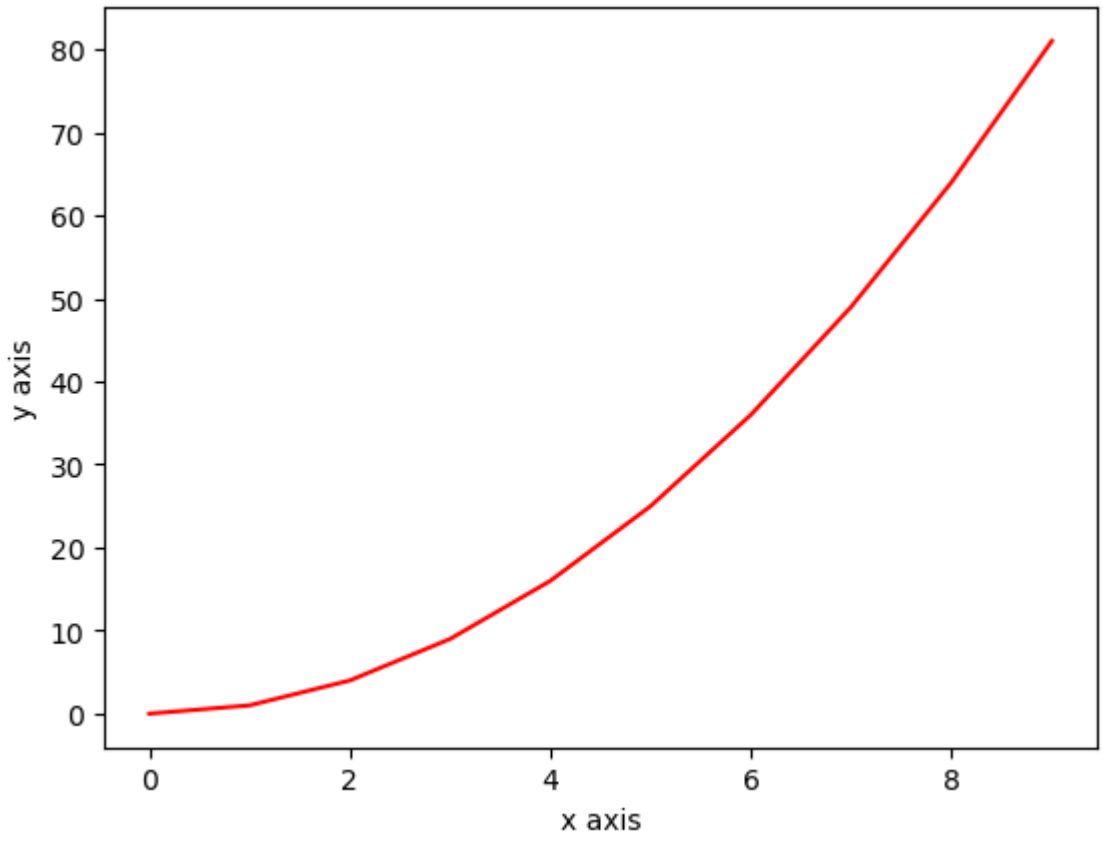
In [46]:
```
plt.plot(x , y)
```

Out[46]:
```
[<matplotlib.lines.Line2D at 0x1f7801ca790>]
```



In [59]:
```
y = x * x
plt.plot(x, y, "r-")
#plt.plot(x, y, "r*-")
#plt.plot(x, y, "ro")

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

Out[59]:
```
Text(0, 0.5, 'y axis')
```

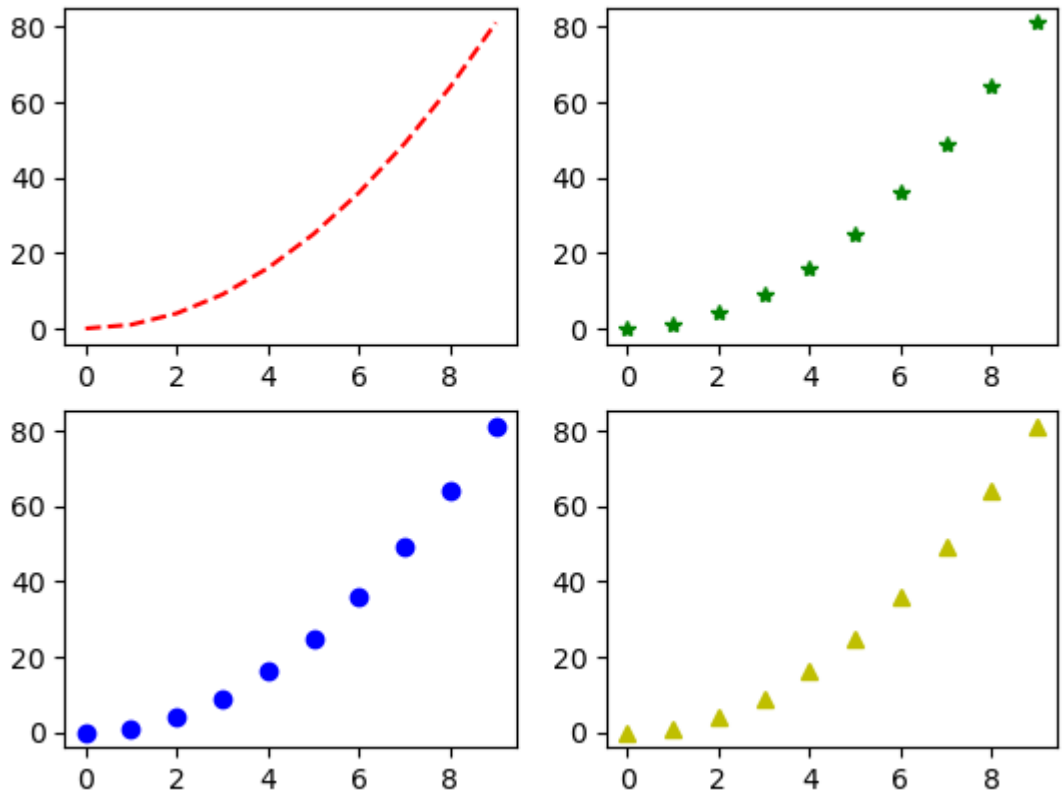


`plt.subplot()`

- **Purpose:**
  - Enables the creation of subplots within a single Matplotlib figure.
- **Functionality:**
  - Organizes multiple plots in a grid format, facilitating side-by-side visualizations.
- **Parameters:**
  - `nrows` : Specifies the number of rows in the subplot grid.
  - `ncols` : Specifies the number of columns in the subplot grid.
  - `index` : Represents the index of the current subplot being created.
- **Indexing System:**
  - Subplots are arranged in a grid, and indexing starts from 1.
  - Indexing proceeds from left to right, and when a row is filled, it continues to the next row.
- **Usage:**
  - Ideal for displaying related visualizations in a compact and organized manner.

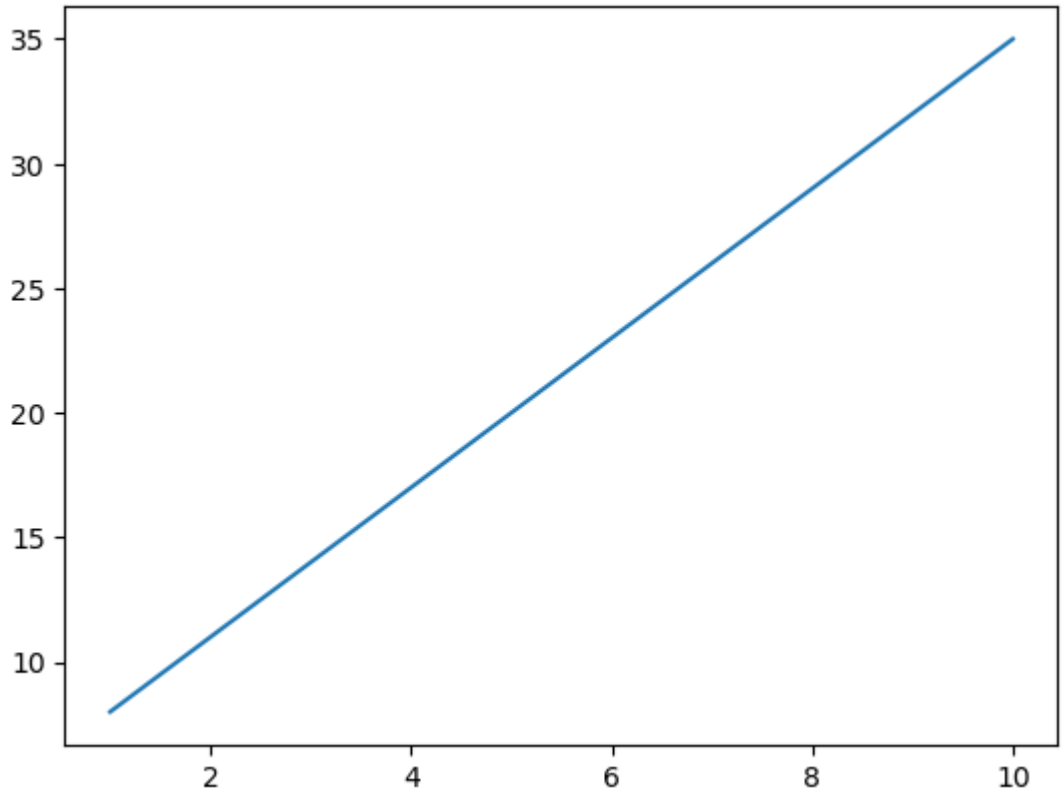
In [66]:
```
#Creating subplots

plt.subplot(2, 2, 1)
plt.plot(x, y, "r--")

plt.subplot(2, 2, 2)
plt.plot(x, y, "g*")

plt.subplot(2, 2, 3)
plt.plot(x, y, "bo")

plt.subplot(2, 2, 4)
plt.plot(x, y, "y^")
```

Out[66]:
```
[<matplotlib.lines.Line2D at 0x1f783f97f10>]
```



In [68]:
```
x = np.arange(1, 11)
y = 3 * x + 5

plt.plot(x, y)
```

Out[68]:
```
[<matplotlib.lines.Line2D at 0x1f78407bd90>]
```

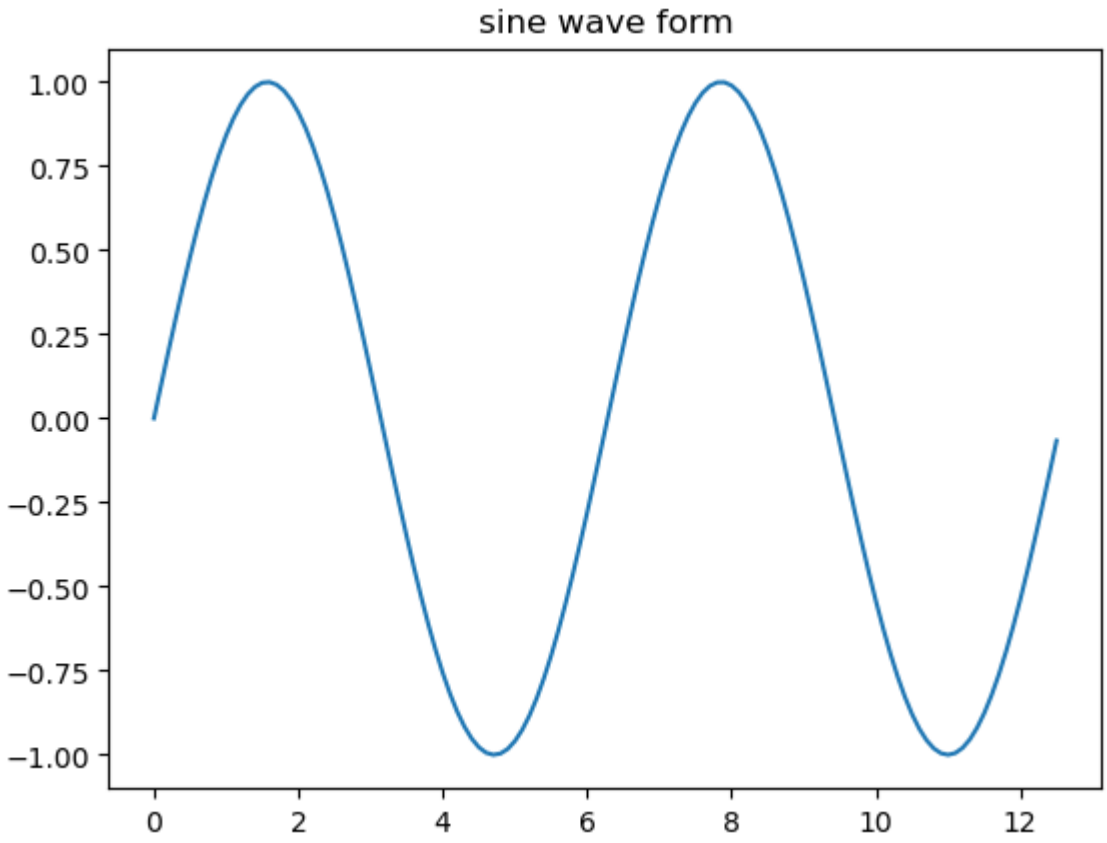


In [69]:
```
np.pi
```

Out[69]:
```
3.141592653589793
```

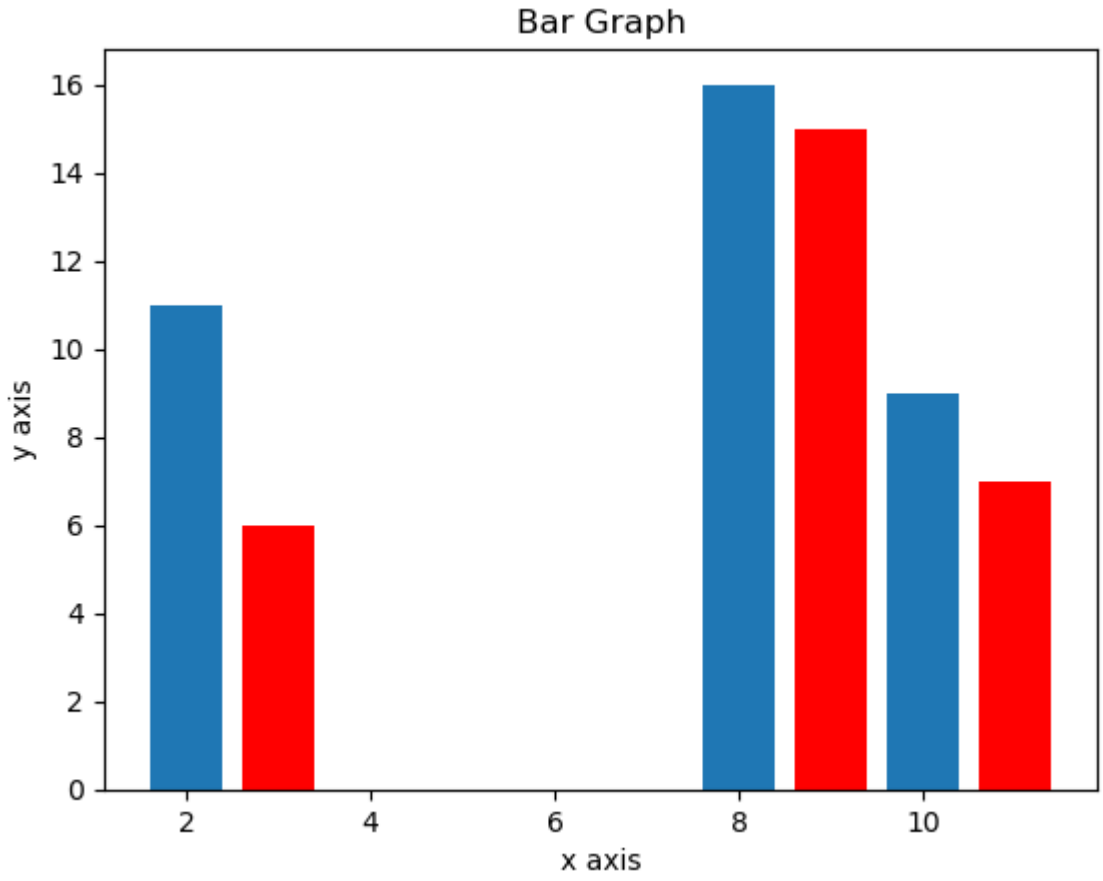
In [71]:
```
#Compute the x and y coordinates for points an a sine curve

x = np.arange(0, 4 * np.pi, 0.1)
y = np.sin(x)
plt.title("sine wave form")

plt.plot(x ,y)
```

Out[71]:
```
[<matplotlib.lines.Line2D at 0x1f783df6710>]
```



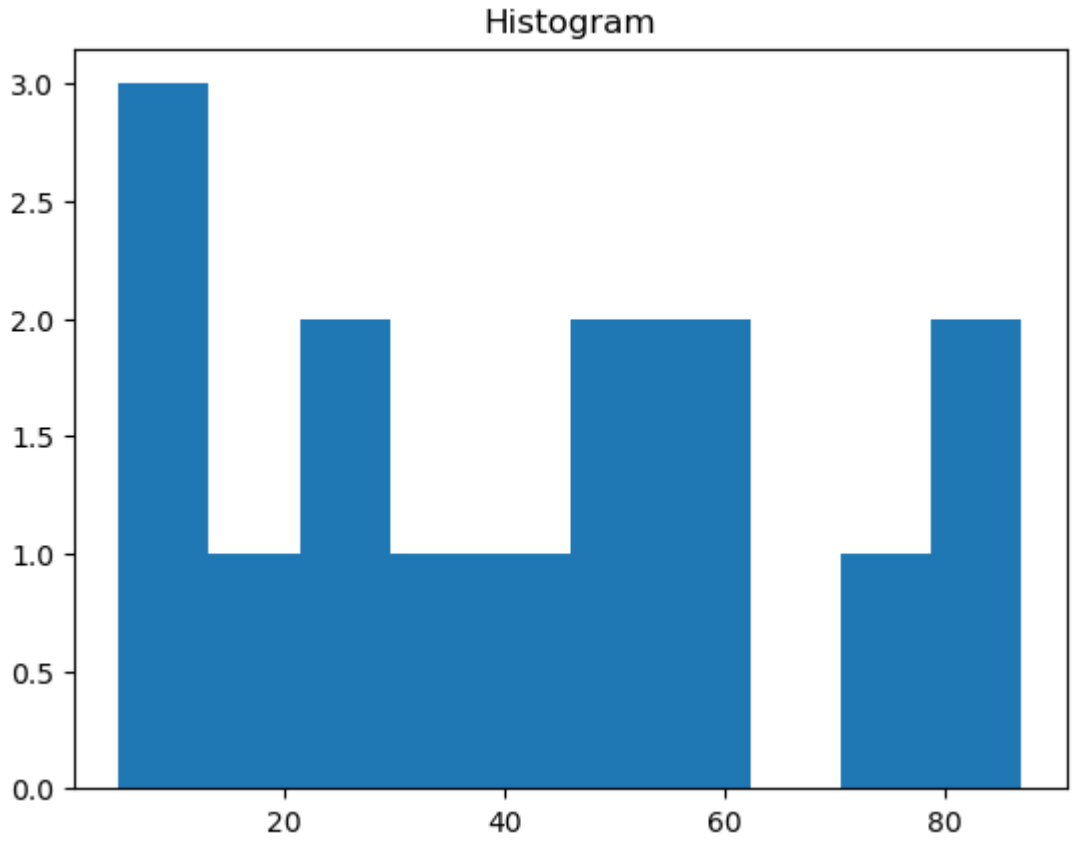
Bar plot

In [81]:
```
x = [2, 8, 10]
y = [11, 16, 9]

x2 = [3, 9, 11]
y2 = [6, 15, 7]

plt.bar(x, y)
plt.bar(x2, y2, color = "r")
plt.title("Bar Graph")
plt.xlabel("x axis")
plt.ylabel("y axis")

plt.show()
```



Histograms

In [83]:
```
a = np.array([22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27])
plt.hist(a)
plt.title("Histogram")
plt.show()
```

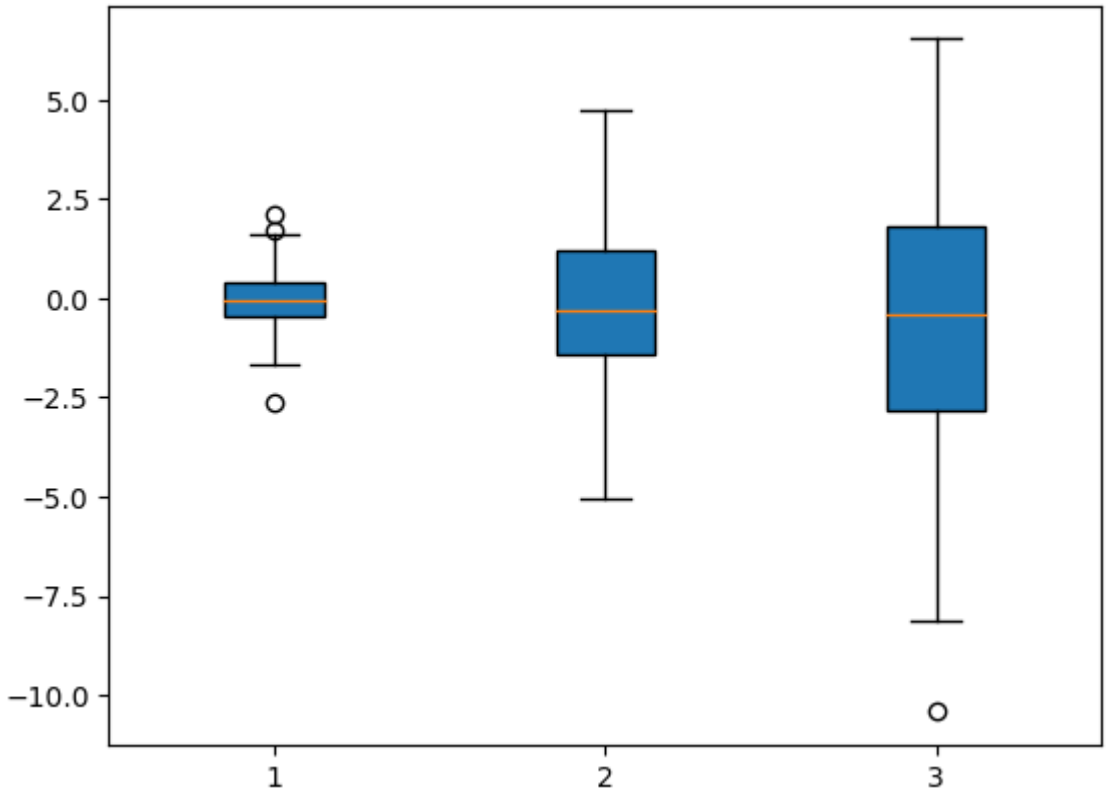


plt.hist()

- **Purpose:**
  - Generates histograms, a visual representation of the distribution of a dataset.
- **Functionality:**
  - Divides the data into intervals (bins) and displays the frequency or count of values within each bin.
- **Parameters:**
  - `a` : The input data for which the histogram is to be created.
  - `bins` : Specifies the number of bins or the specific bin edges.
- **Output:**
  - A bar plot where each bar represents the count or frequency of data points within a specified range (bin).
- **Key Characteristics:**
  - **Bin Count:** Determines the number of intervals the data is divided into.
  - **Bin Range:** Defines the range of values covered by each bin.
- **Usage:**
  - Gain insights into the distribution and central tendency of the data.
  - Identify patterns, outliers, or concentrations within the dataset.

Box Plot

In [89]:
```
data = [np.random.normal(0, std, 100) for std in range(1, 4)]

#rectangular box plot
plt.boxplot(data, vert = True, patch_artist = True);
```

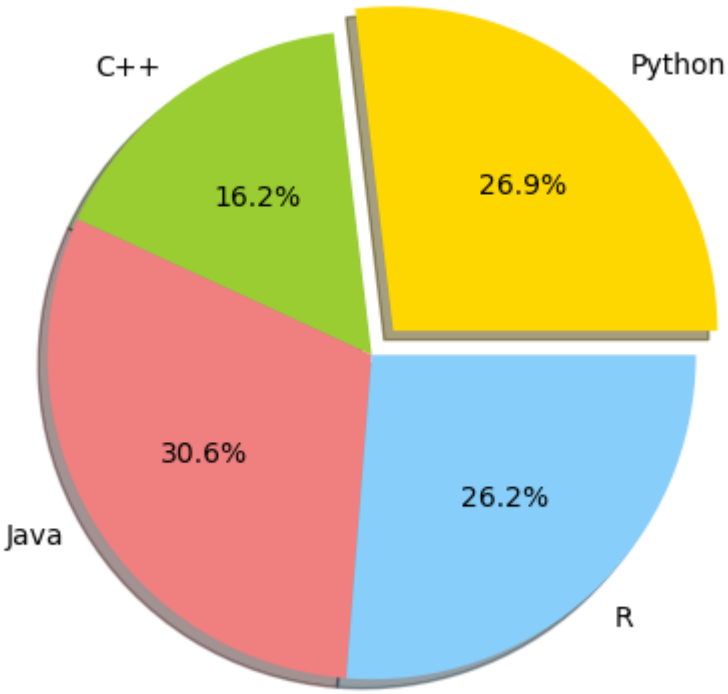


In [91]:
```
data
```

Out[91]:
```
[array([ 0.06032657,  0.68436389, -0.42514008, -0.74704656,  0.23746571,
        -0.96785008,  0.2095686 ,  1.09422767,  0.21209055,  1.12391569,
         1.51783275,  0.36802388,  1.00574672,  1.39681891, -0.02970333,
         0.62419029,  0.02739368, -0.31167069,  0.46571622, -0.9716651 ,
        -0.41544186,  0.46085238,  0.32702312, -0.1473449 , -2.64487029,
        -1.09732735, -0.07575041,  0.44001815, -0.36685821,  0.83537067,
         1.31364817, -0.46173614,  0.20281016, -1.61731416, -0.64806197,
         0.60574429, -0.55814564,  0.36773806, -0.33019822, -0.3998947 ,
        -0.1927217 ,  0.39890611,  1.53470209, -1.23556129, -0.41302393,
        -0.47805476,  1.18137959, -0.44547071, -0.22342064,  0.15564783,
        -0.18580244,  0.56174638,  0.67349172, -0.58761144, -0.55623725,
         1.73216388, -0.00496076, -0.38083597, -0.04203319,  0.7666134 ,
         0.02021349,  0.79412372,  0.25350907,  0.43059129, -0.95850922,
         0.41239229, -0.15039506, -0.43699621, -0.34558405, -0.75583223,
        -1.35528573,  1.09166227,  0.26828567, -0.18104523, -0.81718416,
        -0.32237535, -0.5040276 ,  0.06912407, -0.29258522, -1.67679775,
        -0.33725733,  1.601023  , -0.0142079 ,  0.40524339,  0.26850859,
        -0.17637274, -0.67797913, -0.44661869, -0.04798882,  2.10754514,
        -1.12680137, -0.55279162,  0.59337299,  0.40876542, -0.11069786,
```

Pie Chart

In [93]:
```
#Data to plot
labels = "Python", "C++", "Java", "R"
sizes = [215, 130, 245, 210]
colors = ["gold", "yellowgreen", "lightcoral", "lightskyblue"]
explode = (0.1, 0, 0, 0) #explode 1st slice

#plot
plt.pie(sizes, explode = explode, labels = labels, colors = colors, autopct = "%1.

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



2024.01.15

[Find me on Linkedin (https://bit.ly/3U2fXs6)](https://bit.ly/3U2fXs6)

[Matplot Lib Github Repository (https://bit.ly/3vxFoYh)](https://bit.ly/3vxFoYh)