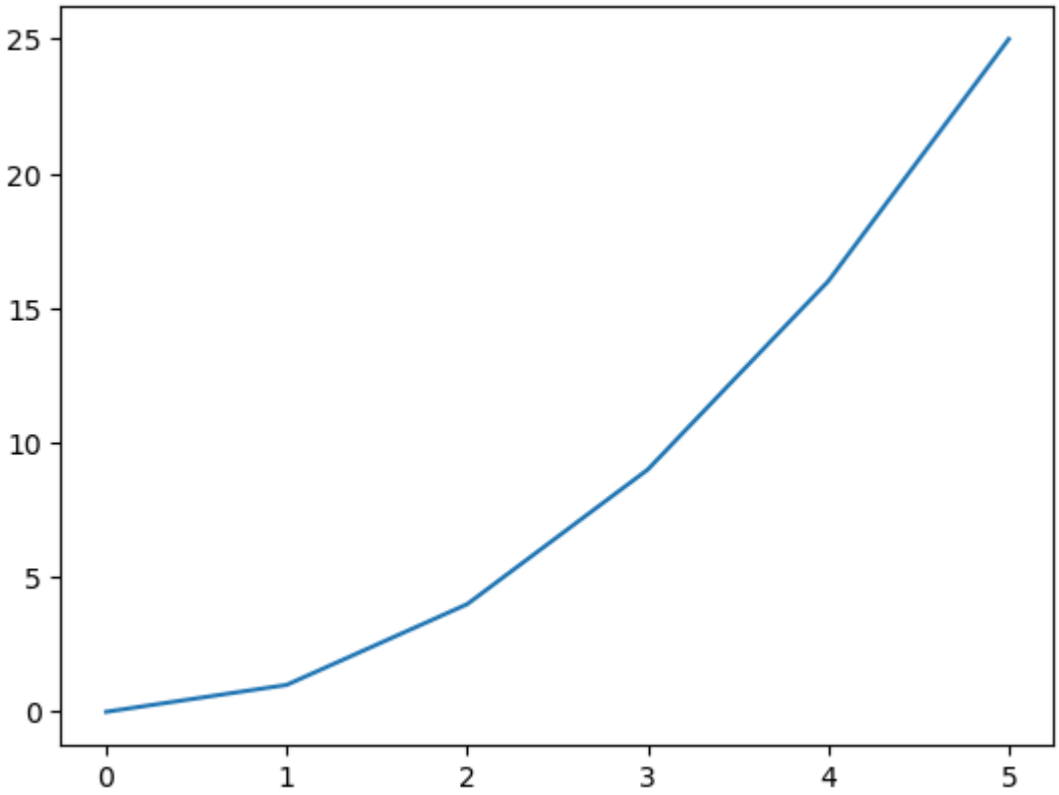
Visualizing the different types of plots in matplotlib

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

1 - Line Plot (plot)

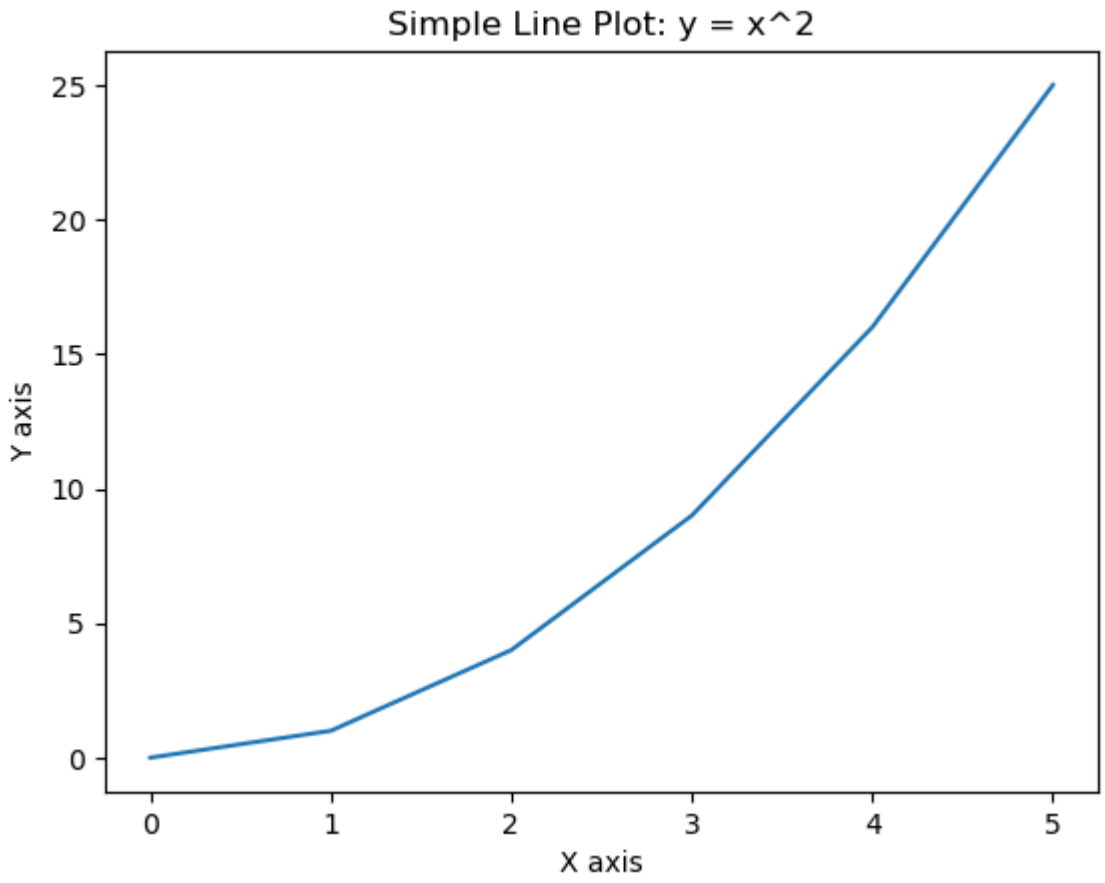
In [4]:
```
# used for displaying data points connected by lines

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]

plt.plot(x,y)
```

Out[4]: [<matplotlib.lines.Line2D at 0x2ae42088470>]



In [5]:
```
import matplotlib.pyplot as plt

# Create data
x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]  # y = x^2

# Create a plot
plt.plot(x, y)

# Add labels and a title
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Simple Line Plot: y = x^2")
```

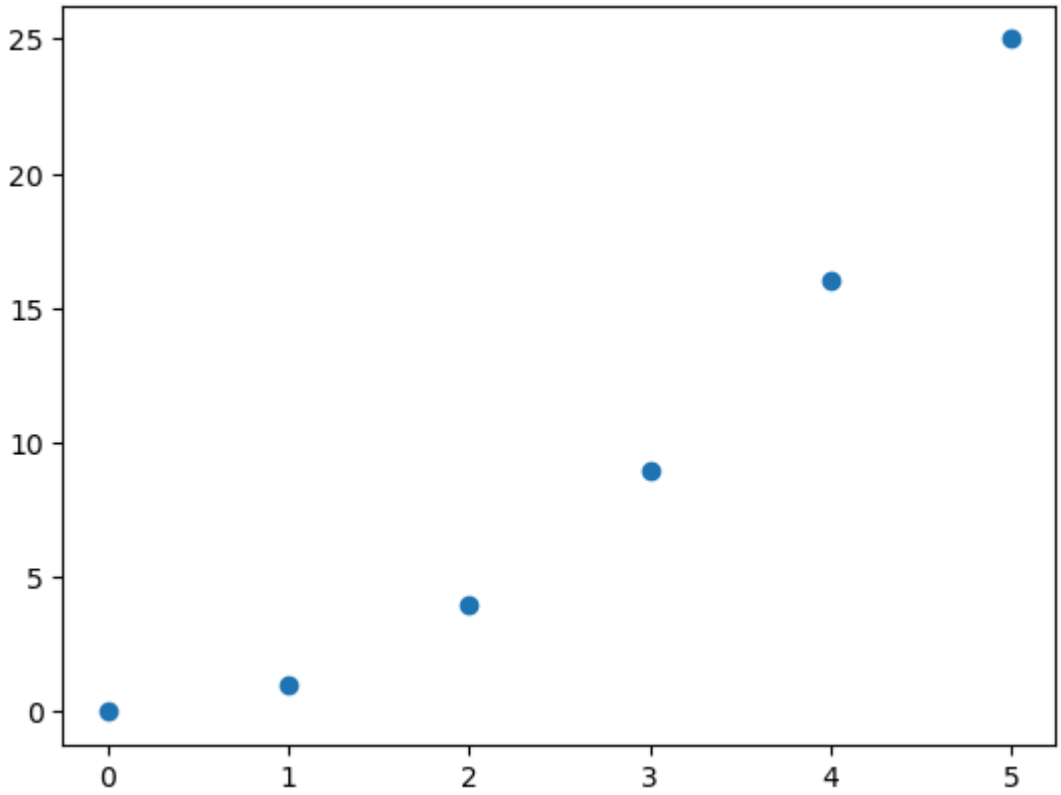
```
# Display the plot
plt.show()
```



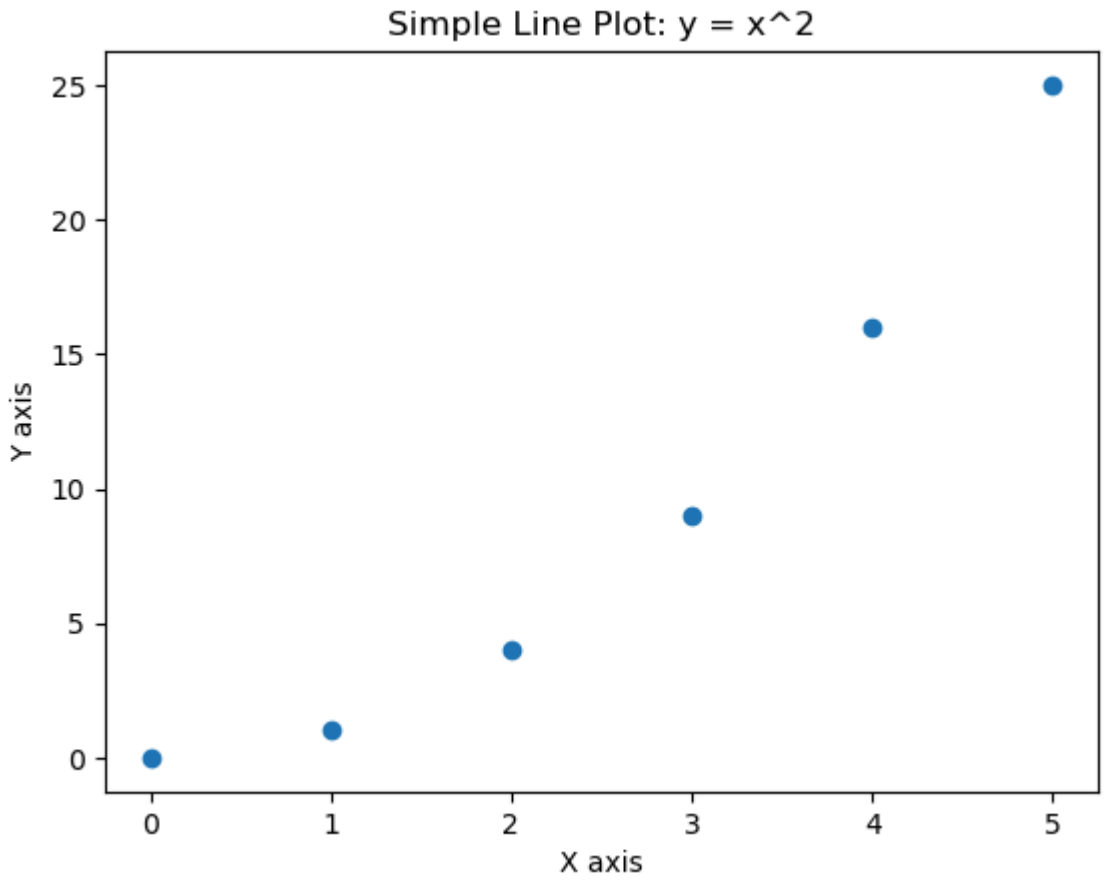
2 - Scatter Plot (scatter)

In [7]:
```
# Displays individual data points as dots in a 2D space, useful for relationship

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.scatter(x, y)
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x2ae4229f320>
```



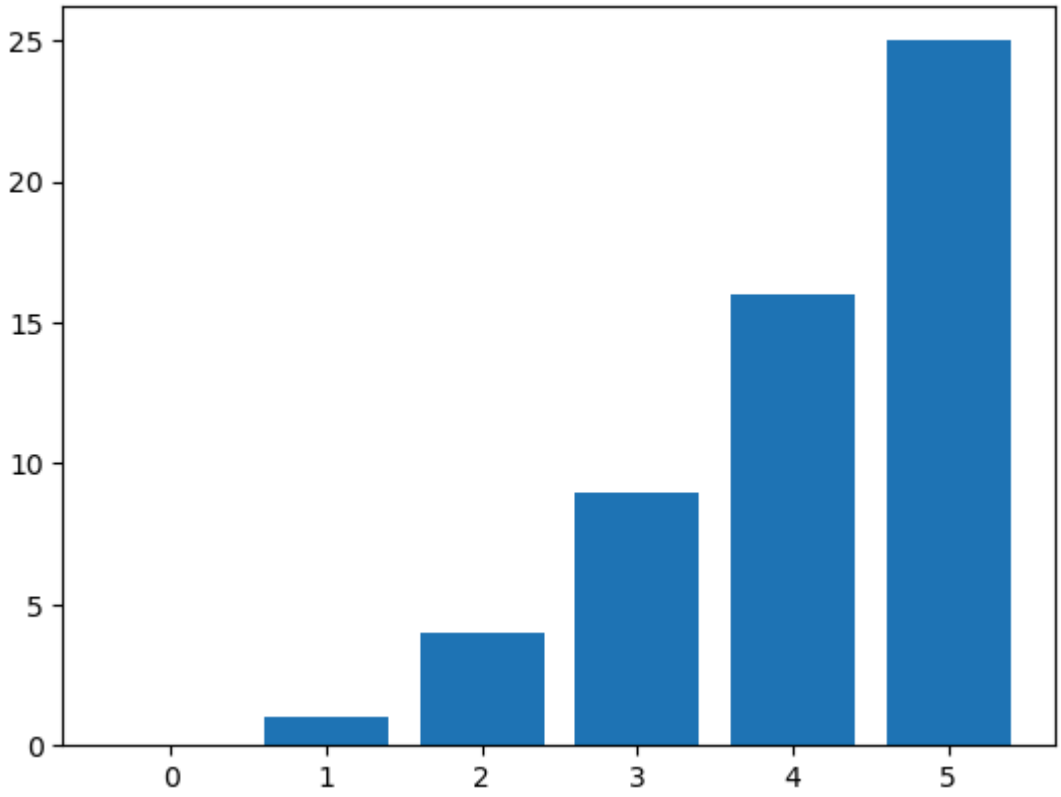
In [8]:
```python
import matplotlib.pyplot as plt

# Create data
x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]  # y = x^2

# Create a plot
plt.scatter(x, y)

# Add labels and a title
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Simple Line Plot: y = x^2")

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



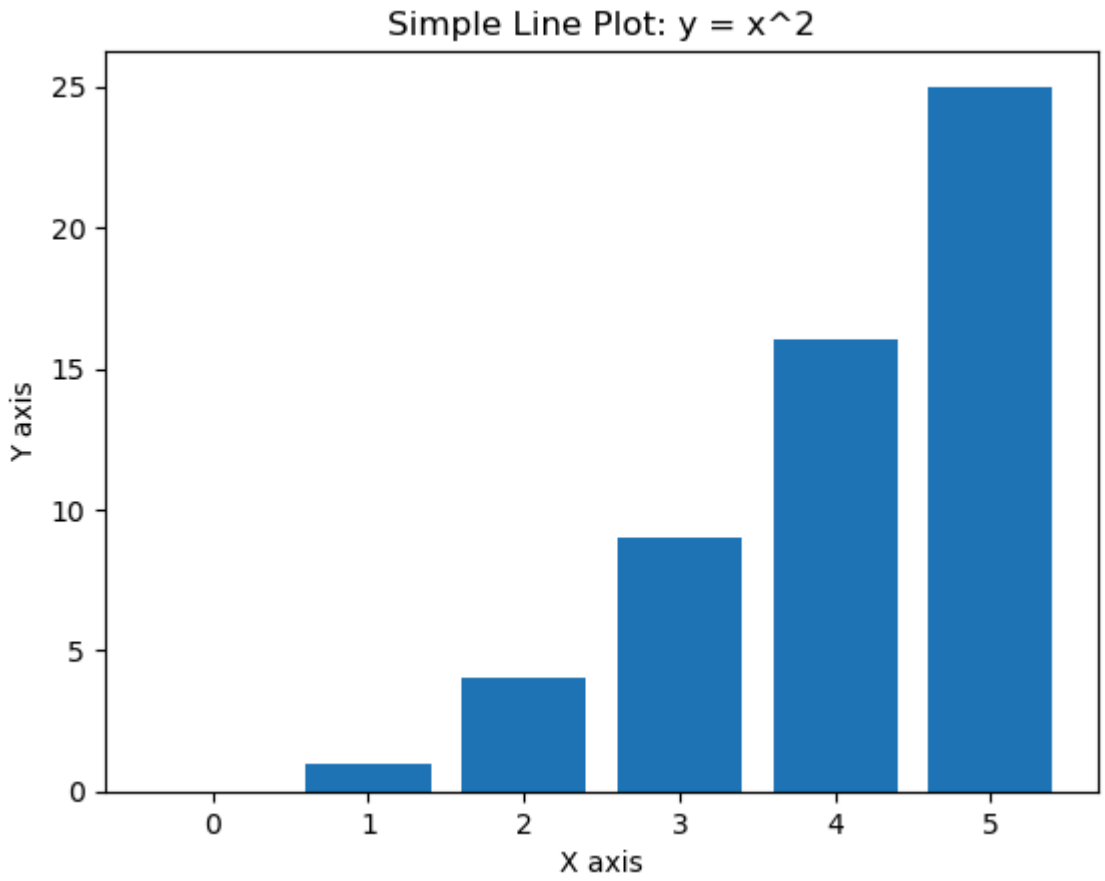
3 - Bar Plot (bar)

In [10]:
```
# Displays rectangular bars representing data values. Useful for categorical dat

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.bar(x,y)
```

Out[10]:
```
<BarContainer object of 6 artists>
```



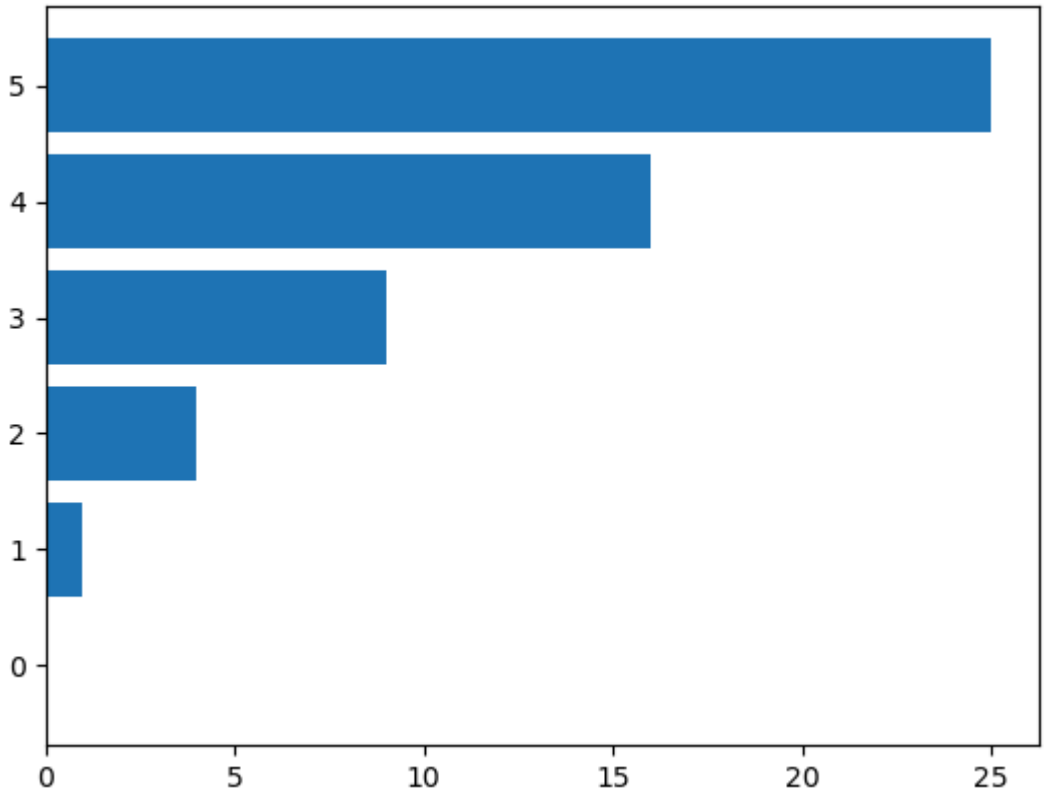
In [11]:
```python
import matplotlib.pyplot as plt

# Create data
x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]  # y = x^2

# Create a plot
plt.bar(x, y)

# Add labels and a title
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Simple Line Plot: y = x^2")

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



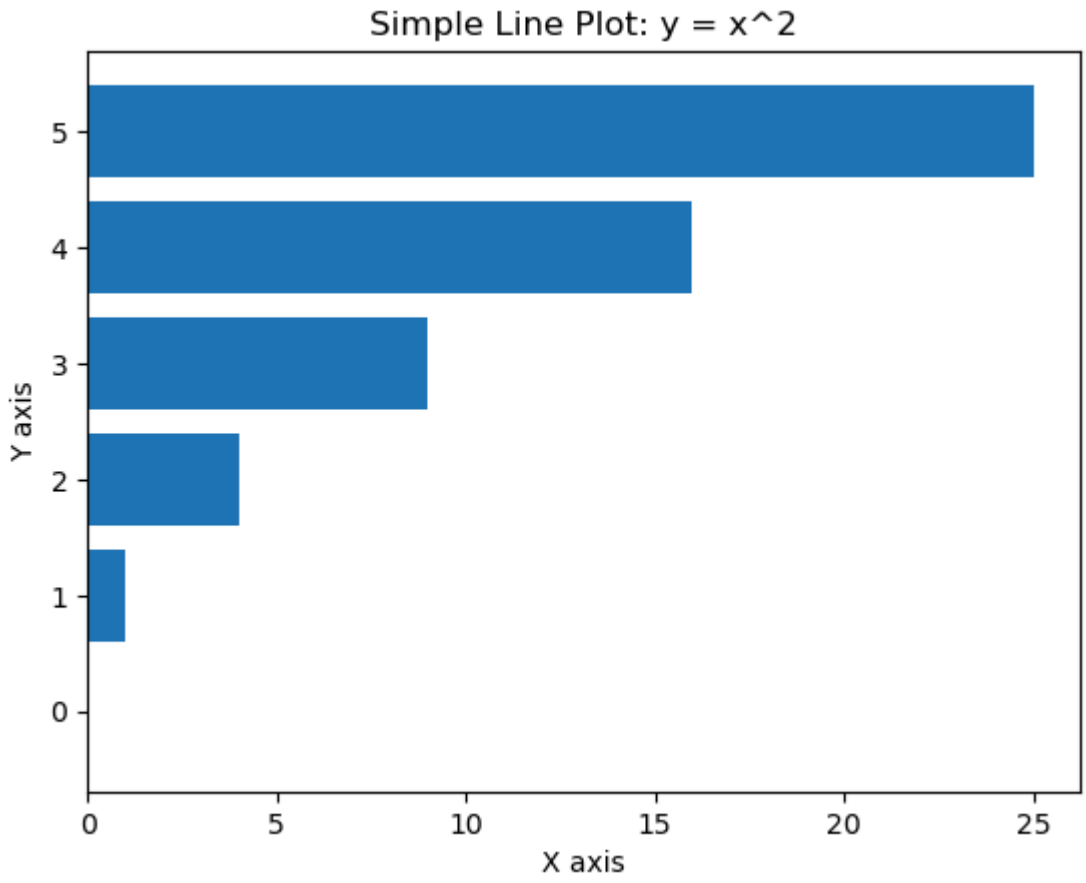
4 - Horizontal Bar plot (barh)

In [13]:
```
# Similar to the bar plot but bars are horizontal.

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.barh(x,y)
```

Out[13]:
```
<BarContainer object of 6 artists>
```



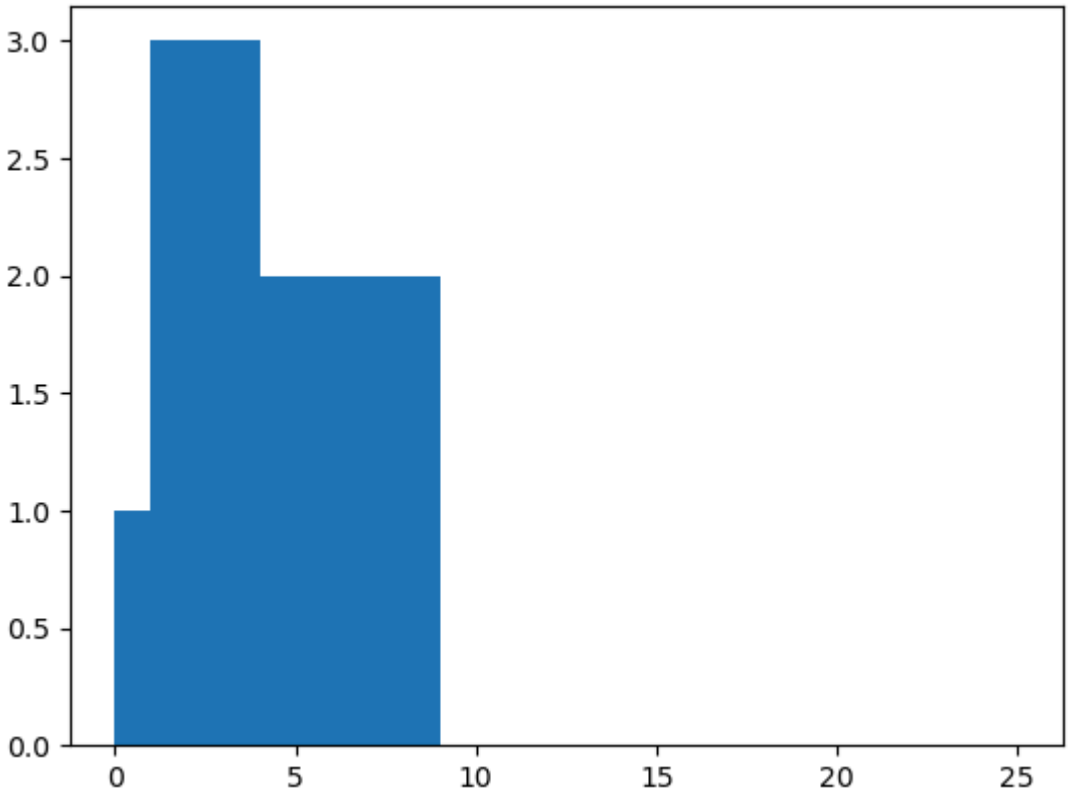
In [14]:
```python
import matplotlib.pyplot as plt

# Create data
x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]  # y = x^2

# Create a plot
plt.barh(x, y)

# Add labels and a title
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Simple Line Plot: y = x^2")

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



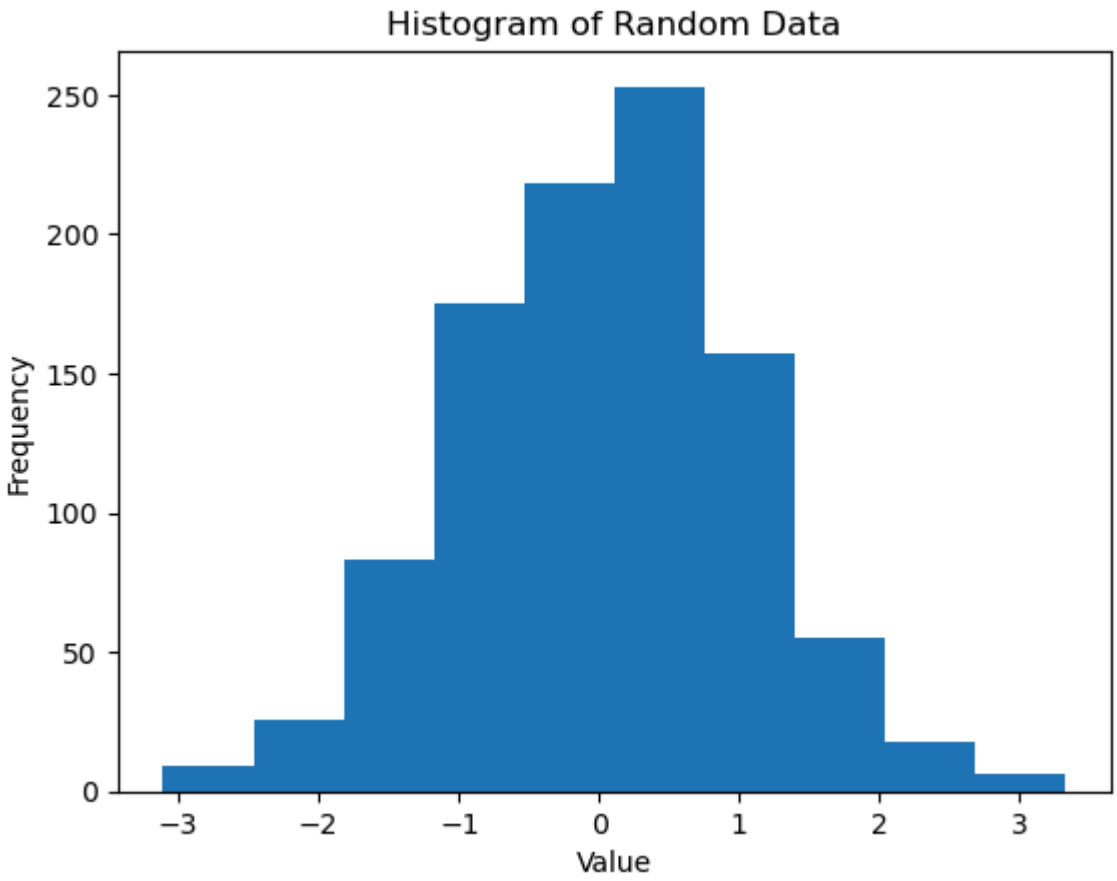
5 - Histogram (hist)

In [16]:
```
# Shows the distribution of a dataset by dividing data into bins.

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.hist(x,y)
```

Out[16]:
```
(array([1., 3., 2., 0., 0.]),
 array([ 0.,  1.,  4.,  9., 16., 25.]),
 <BarContainer object of 5 artists>)
```



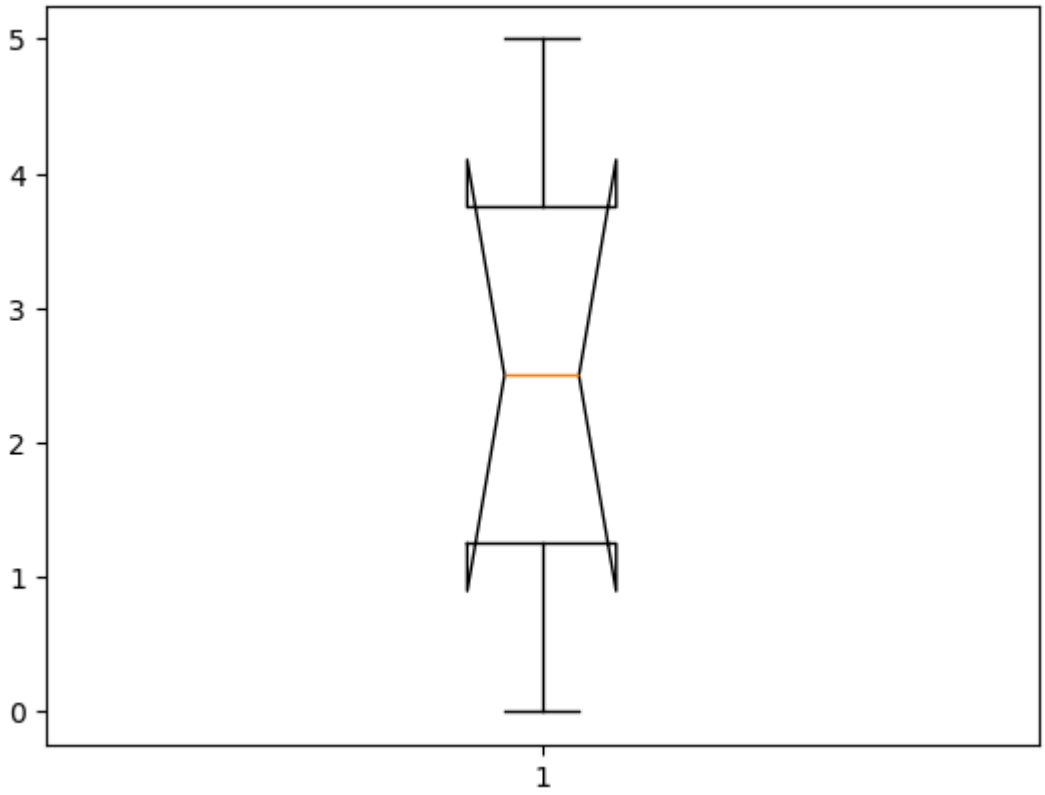
In [17]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Generate random data (1000 points from a standard normal distribution)
data = np.random.randn(1000)

# Create a histogram
plt.hist(data, bins=10)

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Random Data')

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



6 - Box Plot (boxplot)

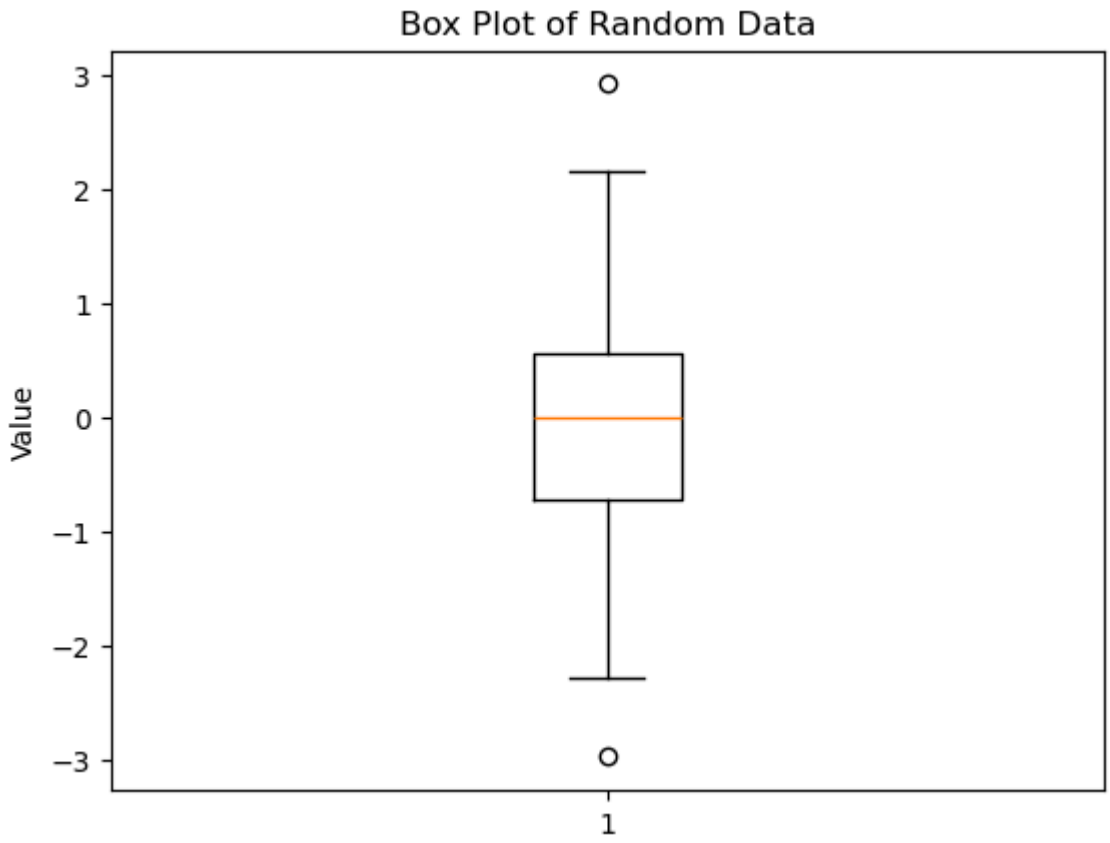
In [19]:
```
# Displays the distribution of a dataset based on a five-number summary (minimum

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.boxplot(x,y)
```

Out[19]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2ae4345fd70>,
  <matplotlib.lines.Line2D at 0x2ae43498050>],
 'caps': [<matplotlib.lines.Line2D at 0x2ae434982f0>,
  <matplotlib.lines.Line2D at 0x2ae434984a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2ae4345fad0>],
 'medians': [<matplotlib.lines.Line2D at 0x2ae434987a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2ae43498a10>],
 'means': []}
```



In [20]:

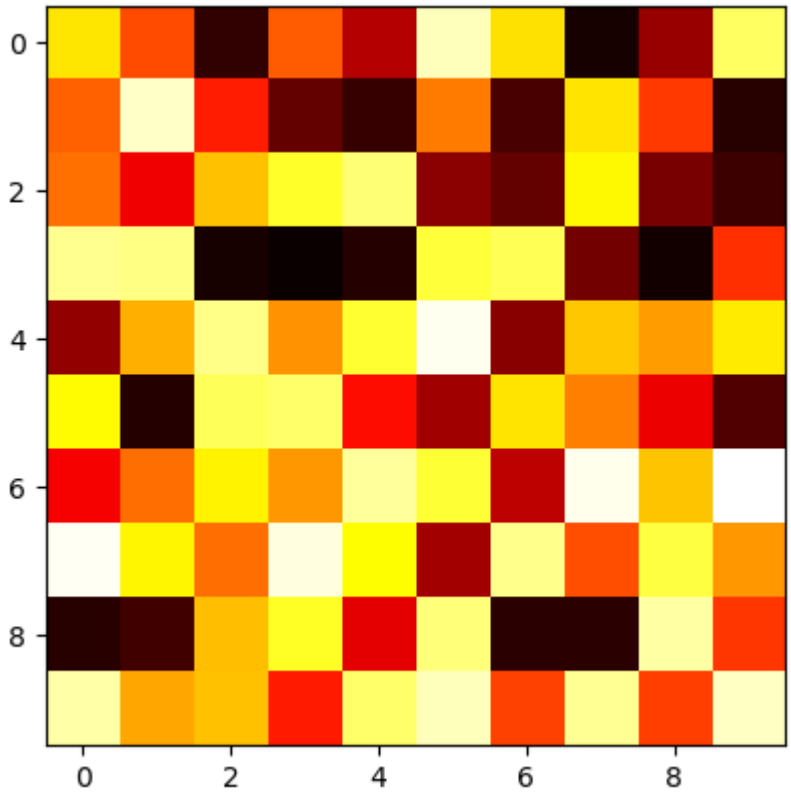
```
import matplotlib.pyplot as plt
import numpy as np

# Generate some random data (100 random values from a normal distribution)
data = np.random.randn(100)

# Create the box plot
plt.boxplot(data)

# Add title and labels
plt.title("Box Plot of Random Data")
plt.ylabel("Value")

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



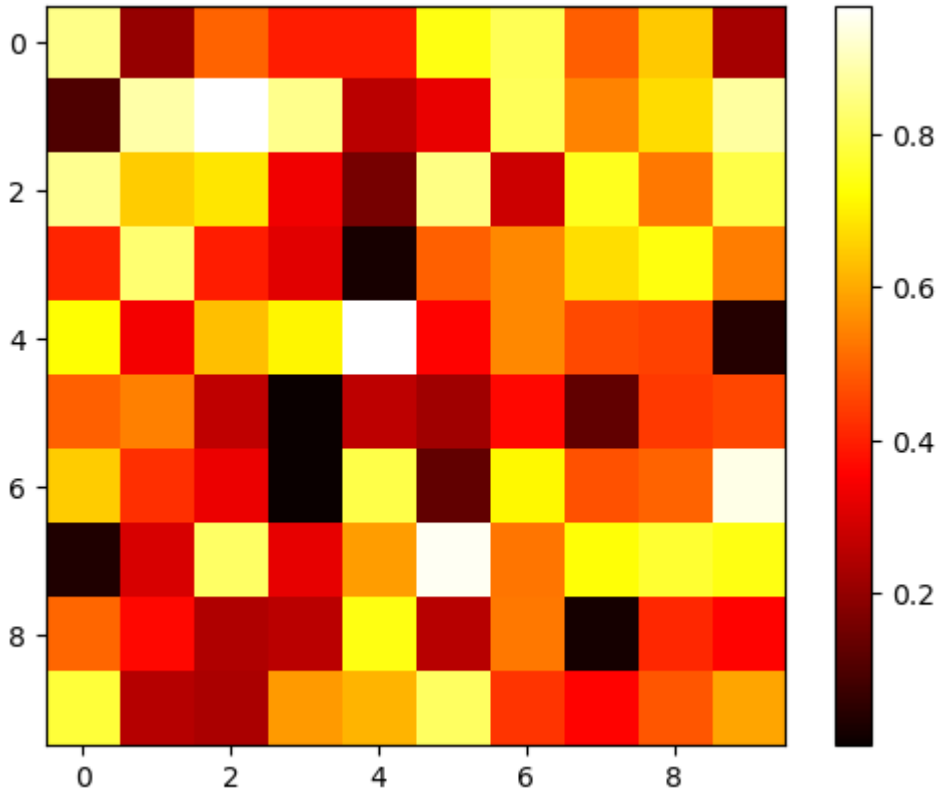
7- Heatmap (imshow,pcolormesh)

In [22]:
```
#Represents matrix-like data with colors, often used for visualizing data in two

import numpy as np
data=np.random.rand(10,10)
plt.imshow(data, cmap='hot')
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x2ae42146ed0>
```



In [23]:
```
#Displays the contours of a three-dimensional surface in two dimensions. Good fo

import matplotlib.pyplot as plt
import numpy as np

# Create a 2D array of random values (10x10 matrix)
data = np.random.rand(10, 10)

# Display the data as an image using the 'hot' colormap
plt.imshow(data, cmap='hot')

# Add a color bar
plt.colorbar()

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

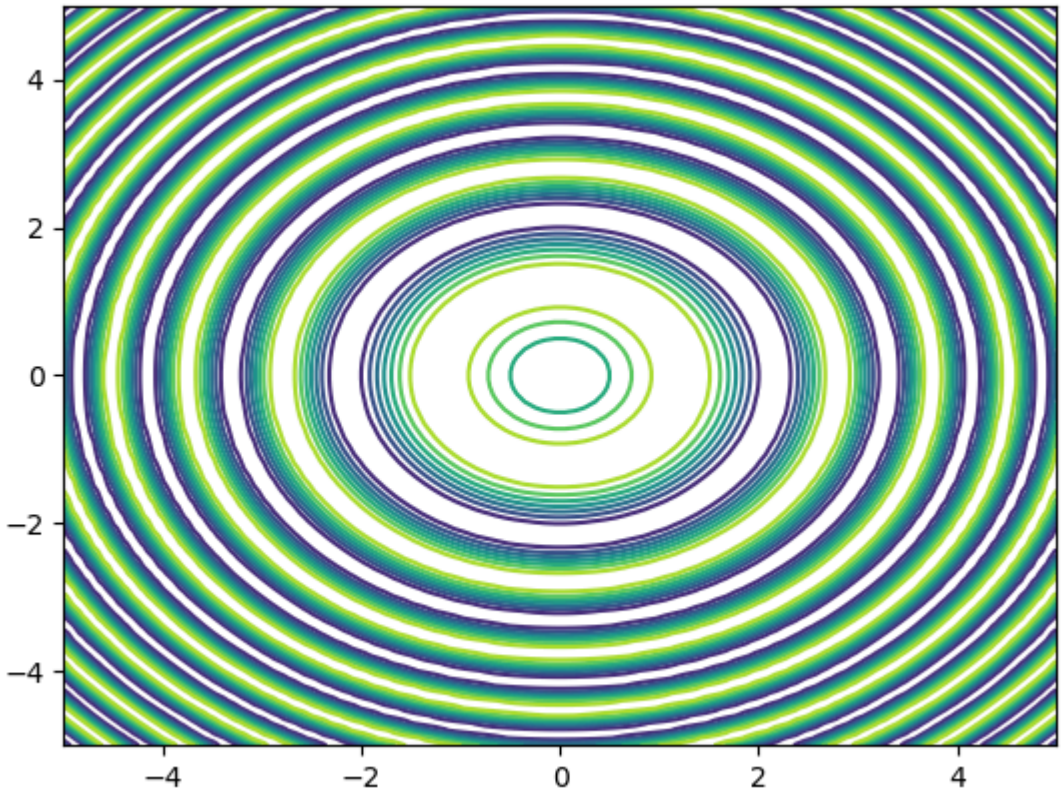


9 - Contour Plot (contour)

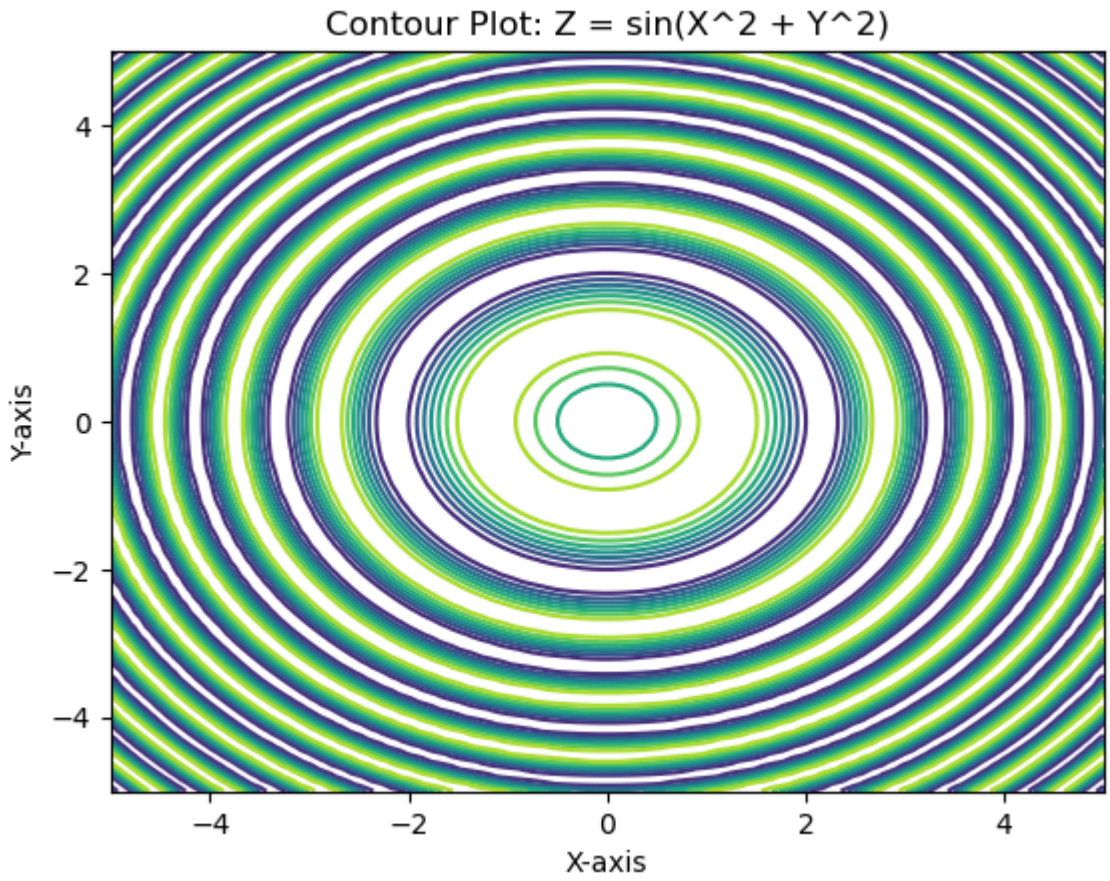
In [25]:
```
# Displays the contours of a three-dimensional surface in two dimensions. Good f

# Create a grid of points (x and y)
x = np.linspace(-5, 5, 100)  # 100 points from -5 to 5
y = np.linspace(-5, 5, 100)  # 100 points from -5 to 5

# Create a meshgrid for plotting
X, Y = np.meshgrid(x, y)

# Calculate Z values for each (x, y) point
Z = np.sin(X**2 + Y**2)
```

In [26]:
```
plt.contour(X, Y, Z)
```

Out[26]:
```
<matplotlib.contour.QuadContourSet at 0x2ae436cbb30>
```



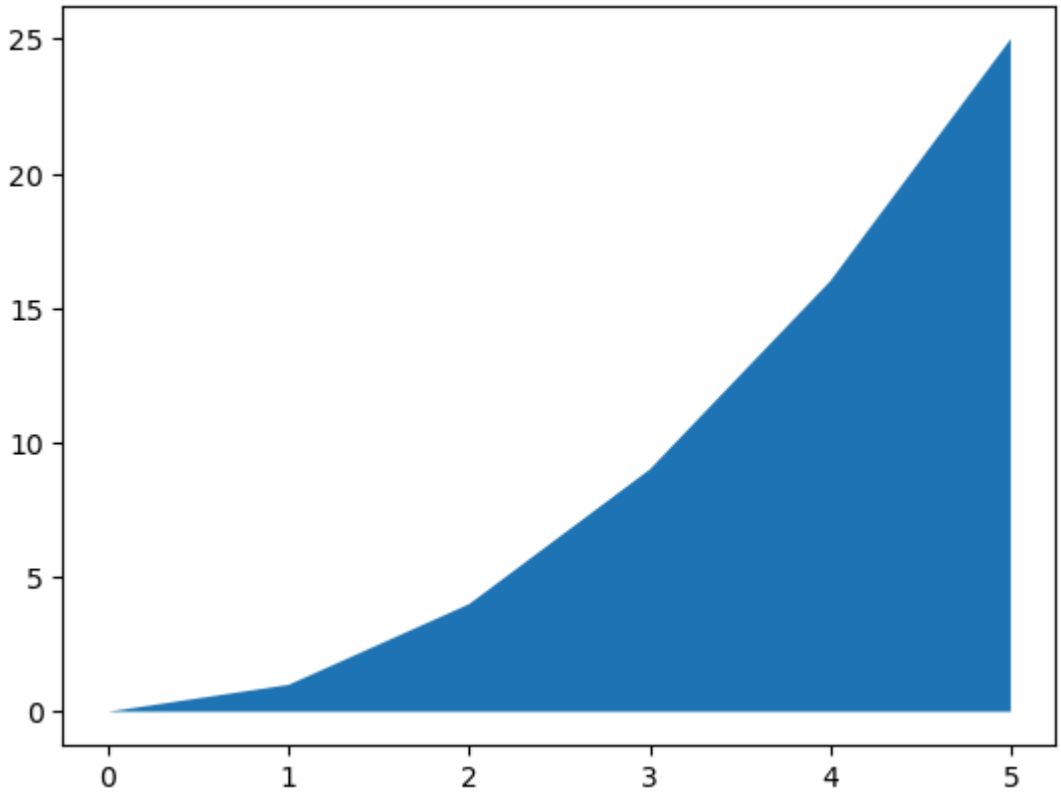
In [27]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Create a grid of points (x and y)
x = np.linspace(-5, 5, 100)  # 100 points from -5 to 5
y = np.linspace(-5, 5, 100)  # 100 points from -5 to 5

# Create a meshgrid for plotting
X, Y = np.meshgrid(x, y)

# Calculate Z values for each (x, y) point
Z = np.sin(X**2 + Y**2)

# Create the contour plot
plt.contour(X, Y, Z)

# Add title and labels
plt.title('Contour Plot: Z = sin(X^2 + Y^2)')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

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



10 - Area Plot (fill_between)

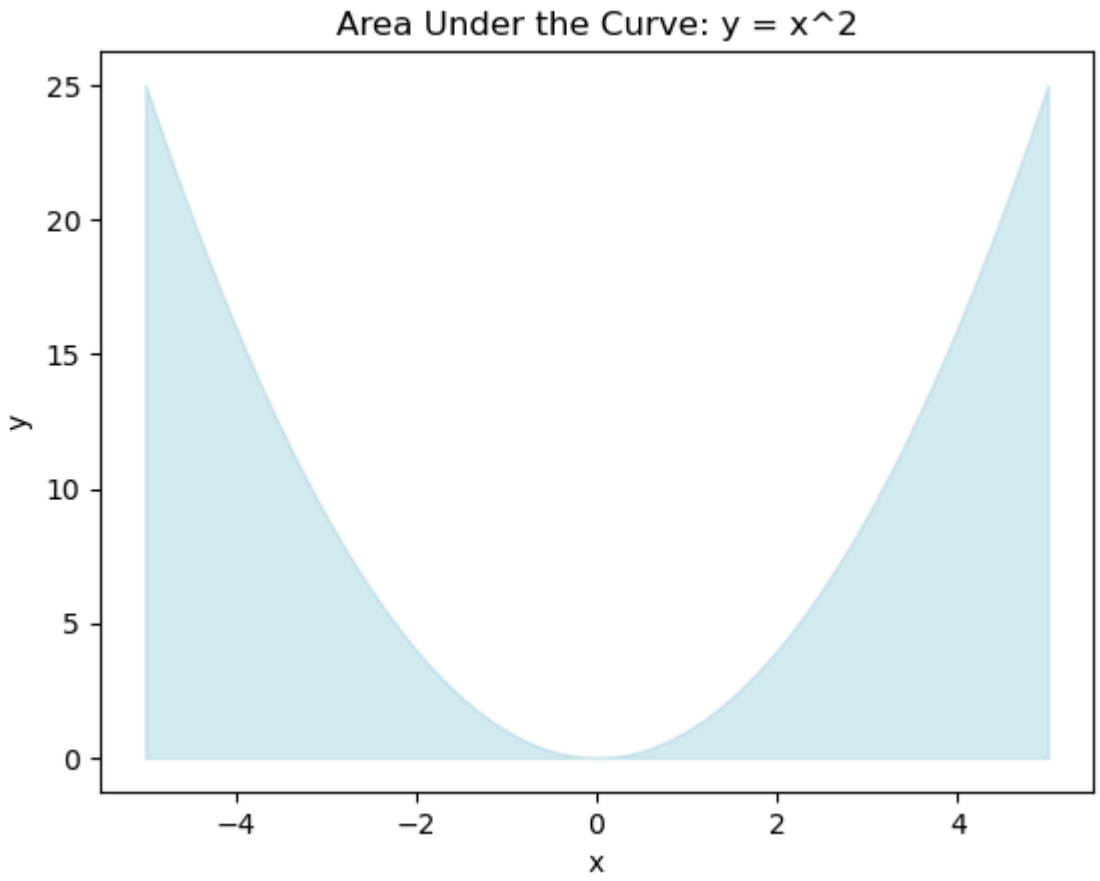
In [29]:
```
# Similar to line plots but with shaded areas between lines. Useful for showing

x = [0, 1, 2, 3, 4, 5]
y = [i**2 for i in x]
plt.fill_between(x,y)
```

Out[29]:
```
<matplotlib.collections.PolyCollection at 0x2ae423411f0>
```

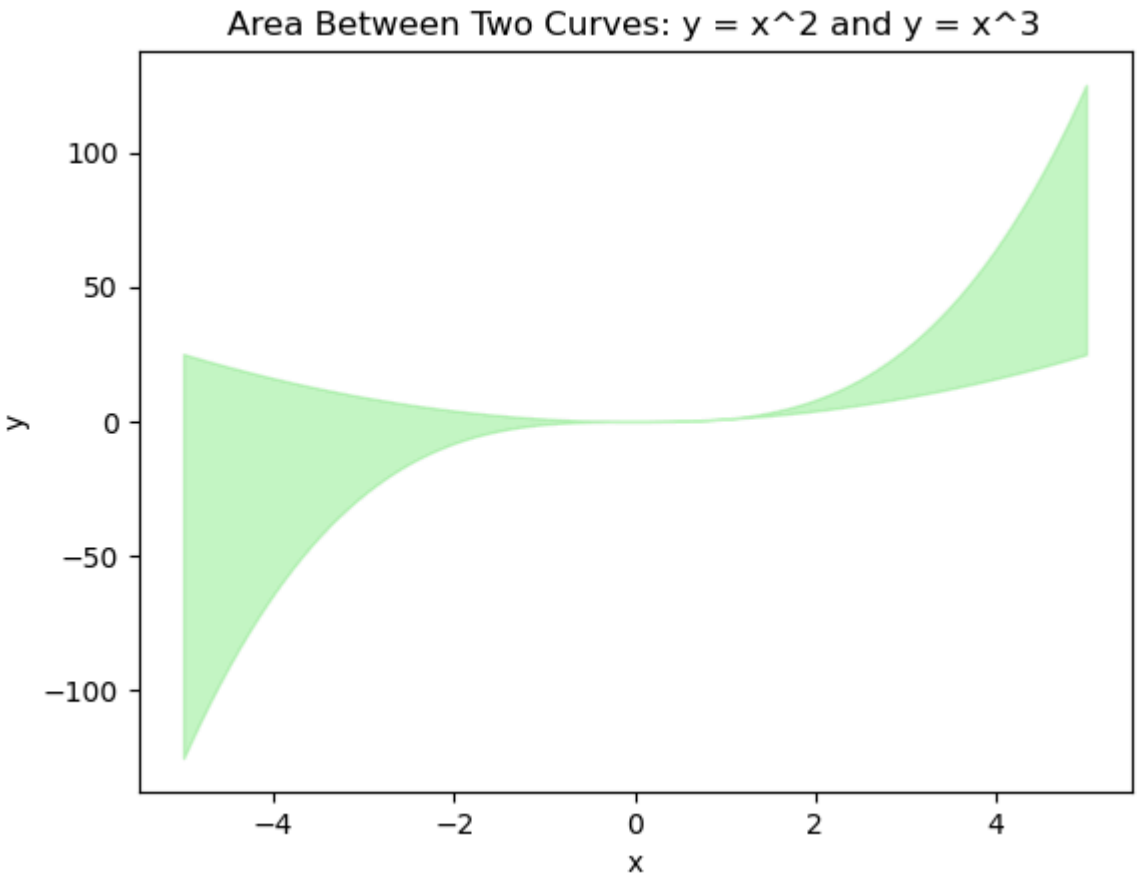


In [30]:

```python
import matplotlib.pyplot as plt
import numpy as np

# Create an array of x values from -5 to 5
x = np.linspace(-5, 5, 100)

# Calculate the corresponding y values (for example, y = x^2)
y = x**2

# Fill the area between the curve and the x-axis
plt.fill_between(x, y, color='lightblue', alpha=0.5)

# Add title and labels
plt.title('Area Under the Curve: y = x^2')
plt.xlabel('x')
plt.ylabel('y')

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



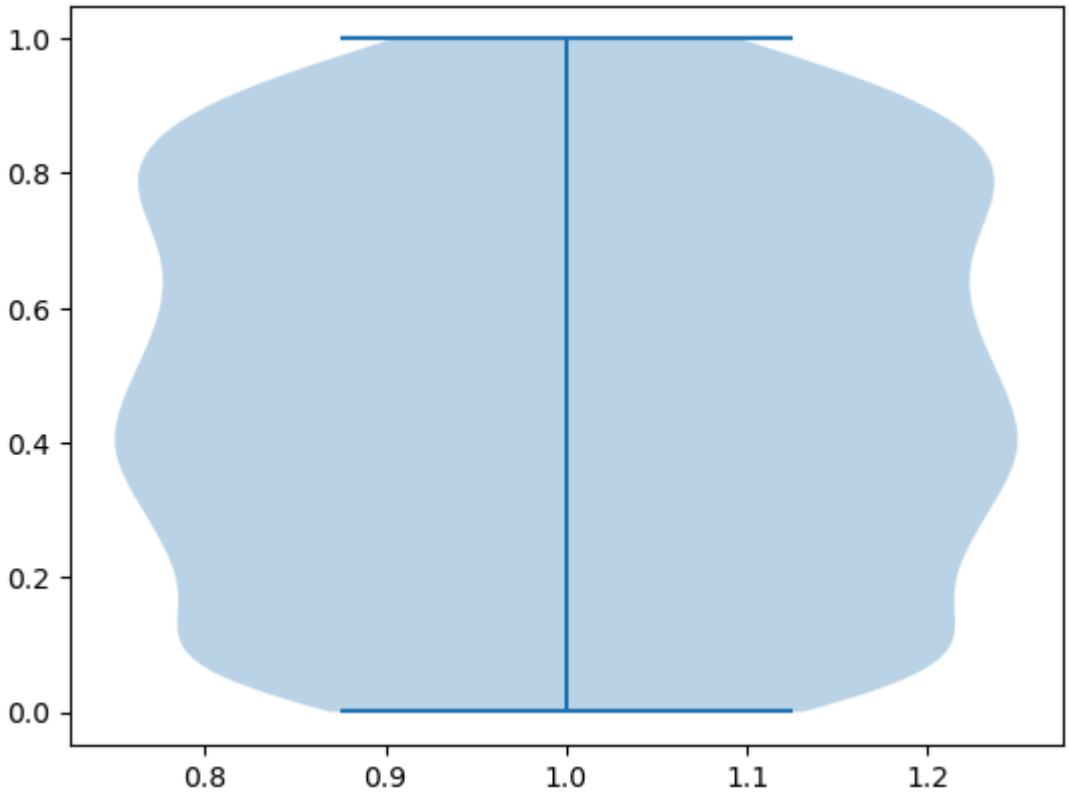
In [31]:
```
# filling between two curves

y1 = x**2
y2 = x**3

# Fill the area between the two curves
plt.fill_between(x, y1, y2, color='lightgreen', alpha=0.5)

# Add title and labels
plt.title('Area Between Two Curves: y = x^2 and y = x^3')
plt.xlabel('x')
plt.ylabel('y')

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



11 - Violin Plot (violinplot)

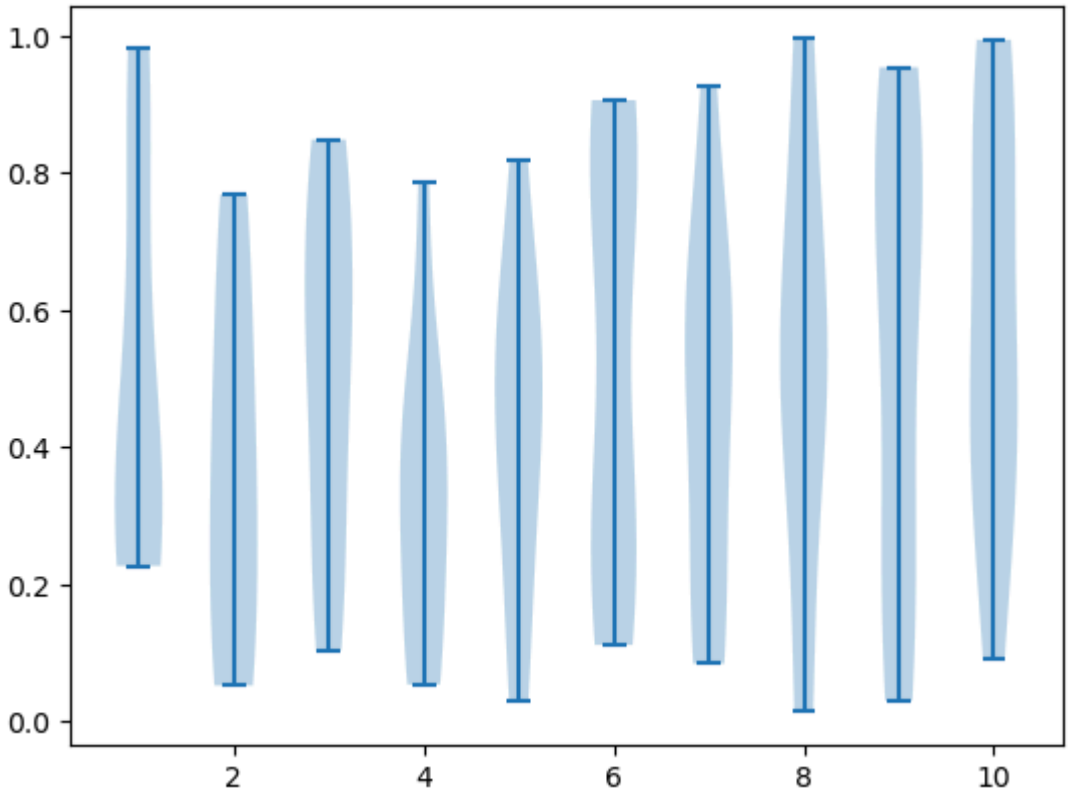
In [33]:
```
# Combines aspects of box plot and kernel density plot. Useful for comparing dis

data = np.random.rand(1000)
plt.violinplot(data)
```

Out[33]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x2ae438d3440>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x2ae438d0c80>,
 'cmins': <matplotlib.collections.LineCollection at 0x2ae437c5be0>,
 'cbars': <matplotlib.collections.LineCollection at 0x2ae438c09e0>}
```

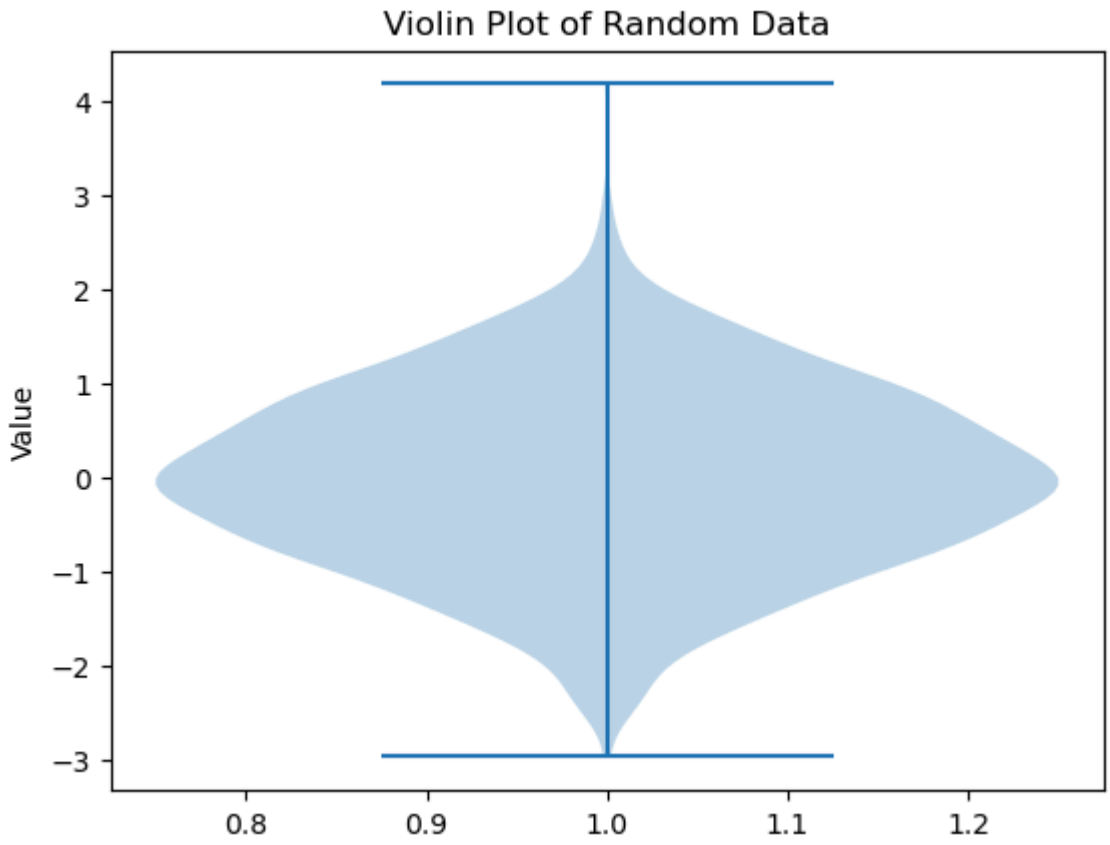


In [34]:
```
data = np.random.rand(10, 10)
plt.violinplot(data)
```

Out[34]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x2ae4380fb60>,
  <matplotlib.collections.PolyCollection at 0x2ae43832300>,
  <matplotlib.collections.PolyCollection at 0x2ae43749e80>,
  <matplotlib.collections.PolyCollection at 0x2ae438d2720>,
  <matplotlib.collections.PolyCollection at 0x2ae43692bd0>,
  <matplotlib.collections.PolyCollection at 0x2ae4380f650>,
  <matplotlib.collections.PolyCollection at 0x2ae437c7710>,
  <matplotlib.collections.PolyCollection at 0x2ae438fd460>,
  <matplotlib.collections.PolyCollection at 0x2ae4380d310>,
  <matplotlib.collections.PolyCollection at 0x2ae44a401a0>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x2ae435b6270>,
 'cmins': <matplotlib.collections.LineCollection at 0x2ae44a40c80>,
 'cbars': <matplotlib.collections.LineCollection at 0x2ae44a41a60>}
```

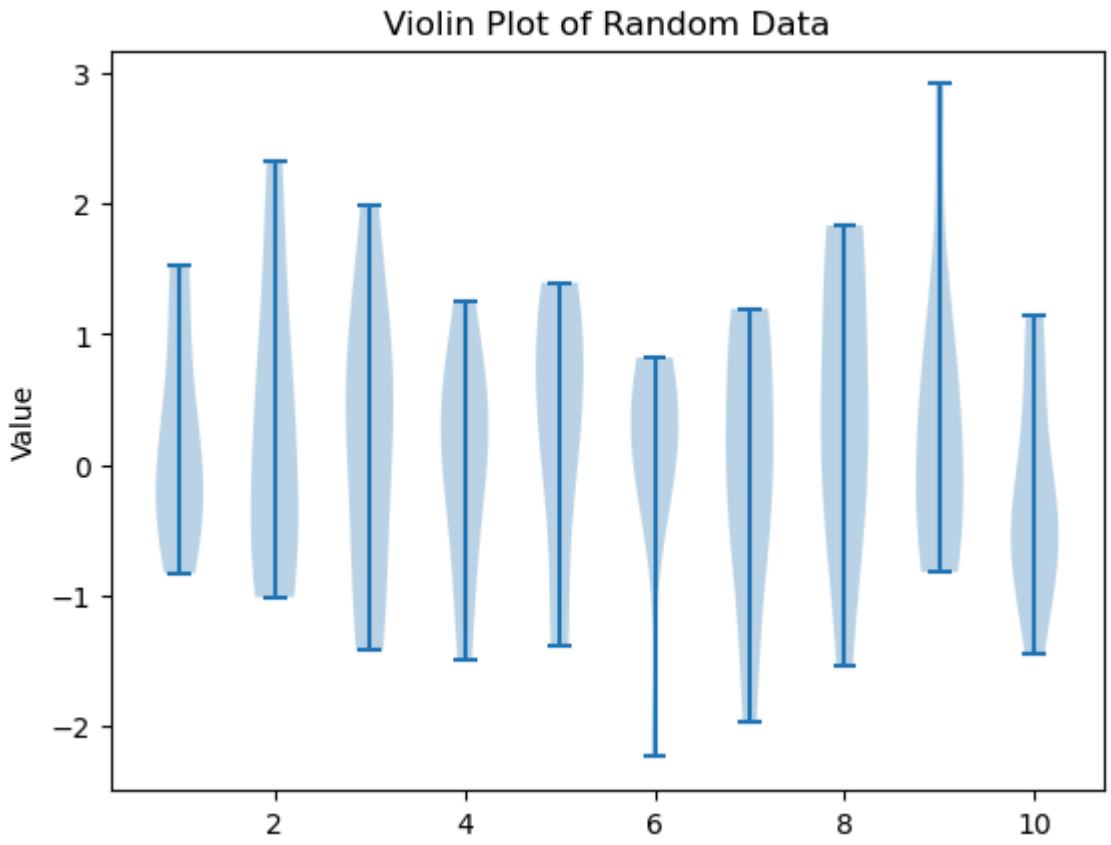


In [35]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Generate random data (1000 points from a normal distribution)
data = np.random.randn(1000)

# Create the violin plot
plt.violinplot(data)

# Add title and labels
plt.title('Violin Plot of Random Data')
plt.ylabel('Value')

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



In [36]:

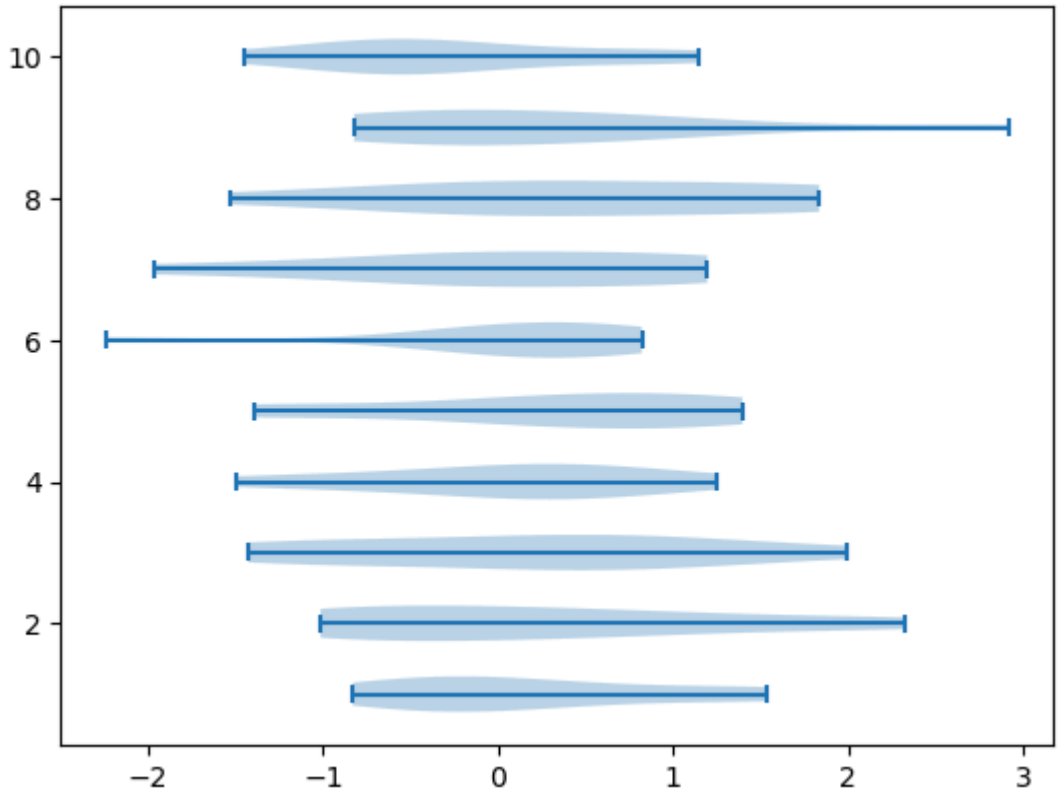
```
import matplotlib.pyplot as plt
import numpy as np

# Generate random data (1000 points from a normal distribution)
data = np.random.randn(10, 10)

# Create the violin plot
plt.violinplot(data)

# Add title and labels
plt.title('Violin Plot of Random Data')
plt.ylabel('Value')

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



In [37]:
```
## Creating multiple violin plots

# Generate two sets of random data
data1 = np.random.randn(1000)
data2 = np.random.randn(1000) + 2  # Shifted distribution

# Create a violin plot for both datasets
plt.violinplot([data1, data2])

# Add title and labels
plt.title('Violin Plot of Two Datasets')
plt.ylabel('Value')

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

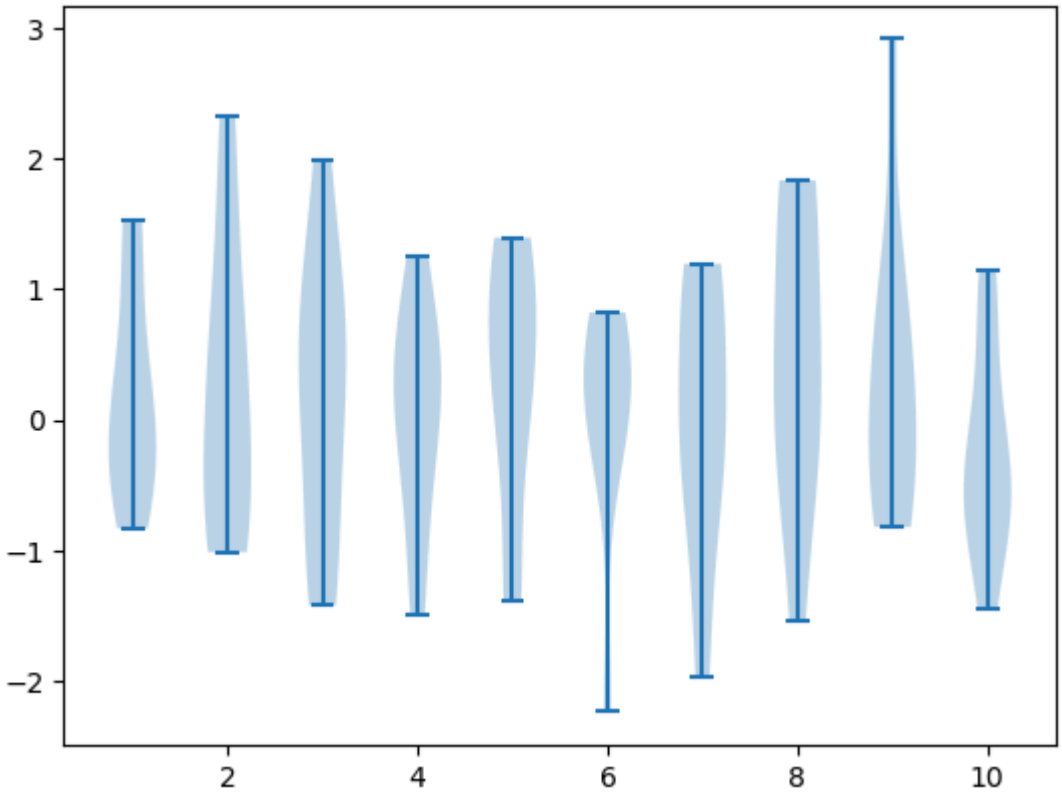
In [38]:
```
plt.violinplot(data, vert=False)
```

Out[38]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x2ae43941b80>,
  <matplotlib.collections.PolyCollection at 0x2ae44a8d5e0>,
  <matplotlib.collections.PolyCollection at 0x2ae4379e960>,
  <matplotlib.collections.PolyCollection at 0x2ae4397ec90>,
  <matplotlib.collections.PolyCollection at 0x2ae44adf050>,
  <matplotlib.collections.PolyCollection at 0x2ae43967380>,
  <matplotlib.collections.PolyCollection at 0x2ae438d3e30>,
  <matplotlib.collections.PolyCollection at 0x2ae43447b30>,
  <matplotlib.collections.PolyCollection at 0x2ae437c77a0>,
  <matplotlib.collections.PolyCollection at 0x2ae43964f50>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x2ae43965ca0>,
 'cmins': <matplotlib.collections.LineCollection at 0x2ae44afcdd0>,
 'cbars': <matplotlib.collections.LineCollection at 0x2ae44a8d100>}
```



In [39]:
```
plt.violinplot(data, vert=True)
```

Out[39]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x2ae43967d10>,
  <matplotlib.collections.PolyCollection at 0x2ae43967230>,
  <matplotlib.collections.PolyCollection at 0x2ae44afd1c0>,
  <matplotlib.collections.PolyCollection at 0x2ae44afde50>,
  <matplotlib.collections.PolyCollection at 0x2ae44abdd90>,
  <matplotlib.collections.PolyCollection at 0x2ae43965490>,
  <matplotlib.collections.PolyCollection at 0x2ae4397ea50>,
  <matplotlib.collections.PolyCollection at 0x2ae44a71370>,
  <matplotlib.collections.PolyCollection at 0x2ae43967fe0>,
  <matplotlib.collections.PolyCollection at 0x2ae4374bb00>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x2ae44abf290>,
 'cmins': <matplotlib.collections.LineCollection at 0x2ae44a70a70>,
 'cbars': <matplotlib.collections.LineCollection at 0x2ae44abfcb0>}
```

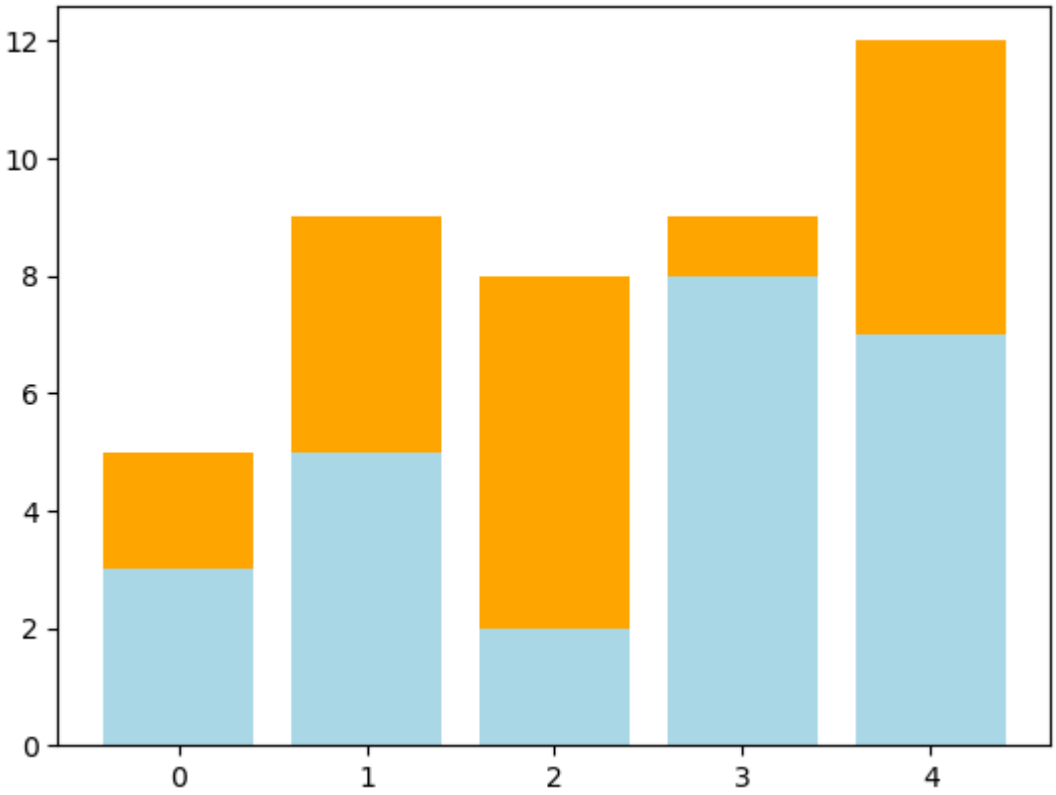


12 - Stacked Bar Plot (bar with stacked = True)

In [41]:
```
# Displays bars stacked on top of each other, useful for showing parts of a whol


# Example x-values (categories)
x = np.arange(5)  # x = [0, 1, 2, 3, 4]

# Example y-values for two datasets (stacked on top of each other)
y1 = np.array([3, 5, 2, 8, 7])  # First dataset
y2 = np.array([2, 4, 6, 1, 5])  # Second dataset
```

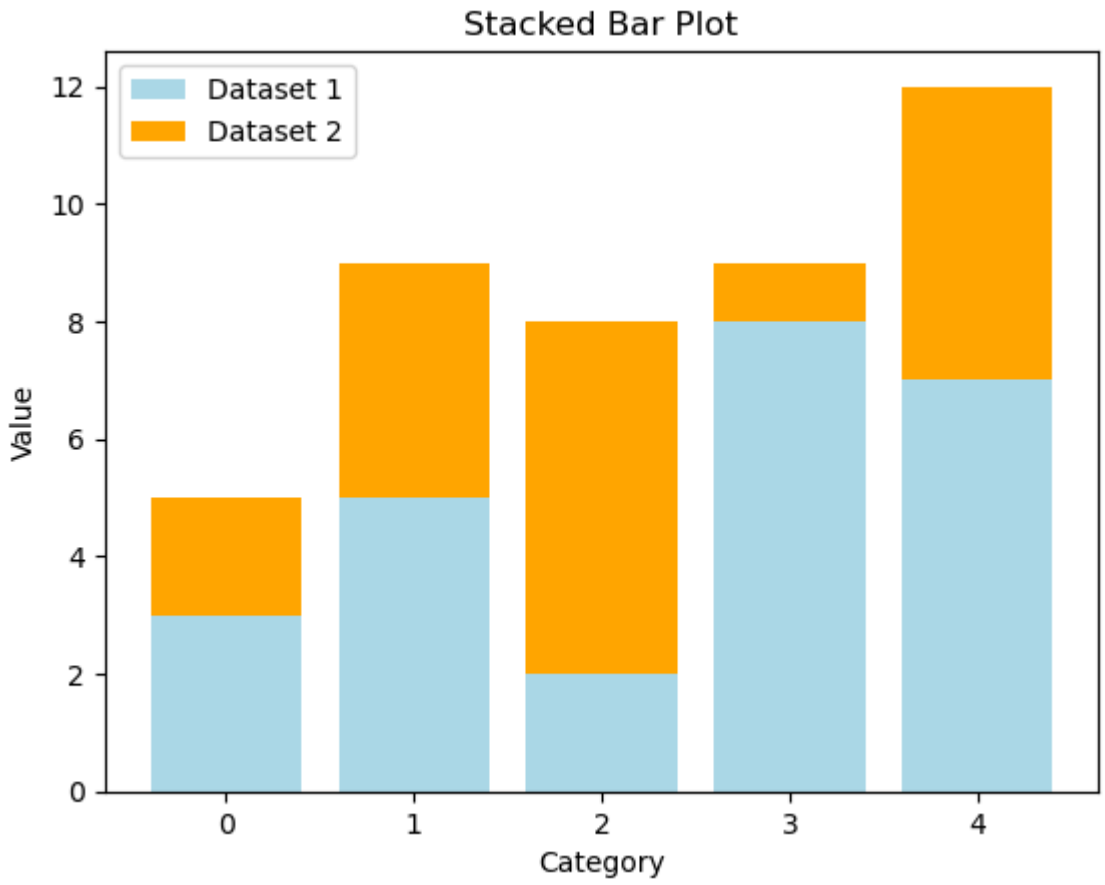
In [42]:
```
plt.bar(x, y1, label='Dataset 1', color='lightblue')  # Bar for y1
plt.bar(x, y2, bottom=y1, label='Dataset 2', color='orange')  # Bar for y2, stac
```

Out[42]:
```
<BarContainer object of 5 artists>
```

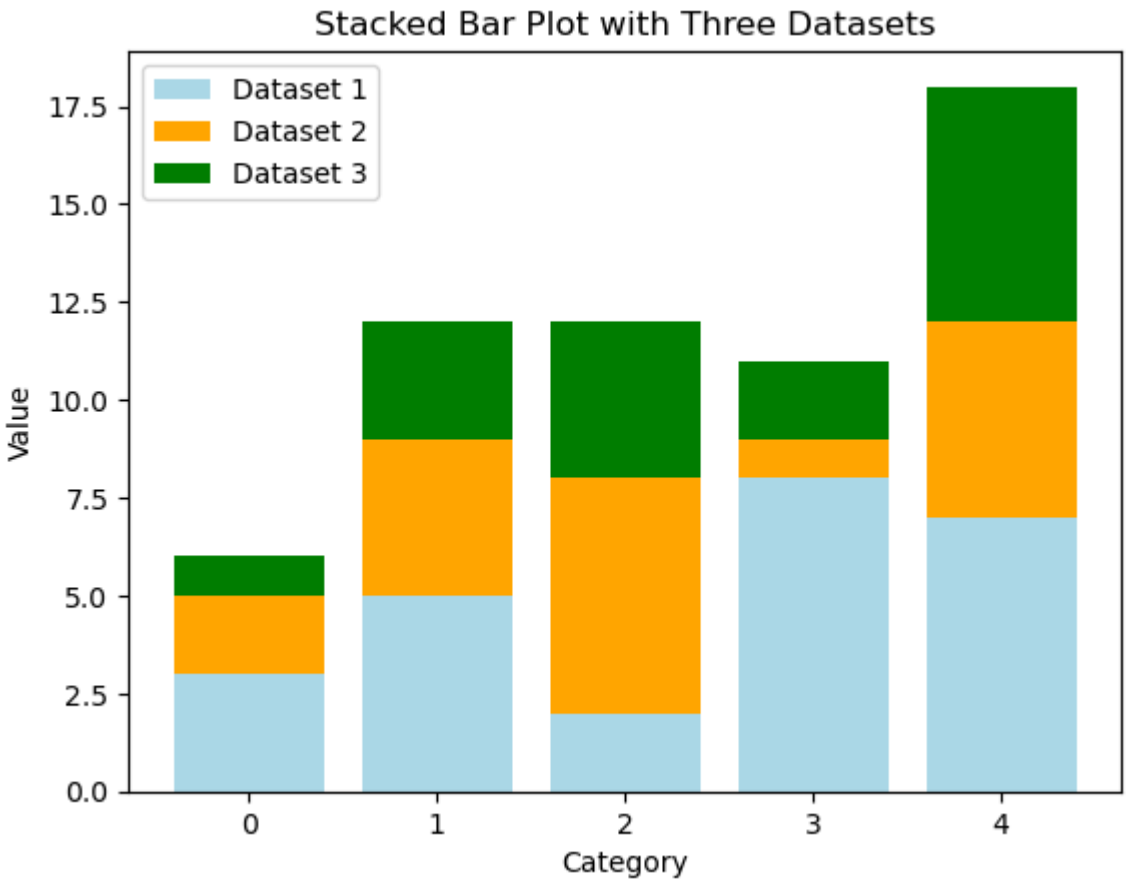


In [43]:

```
import matplotlib.pyplot as plt
import numpy as np

# Example x-values (categories)
x = np.arange(5)  # x = [0, 1, 2, 3, 4]

# Example y-values for two datasets (stacked on top of each other)
y1 = np.array([3, 5, 2, 8, 7])  # First dataset
y2 = np.array([2, 4, 6, 1, 5])  # Second dataset

# Create the stacked bar plot
plt.bar(x, y1, label='Dataset 1', color='lightblue')  # Bar for y1
plt.bar(x, y2, bottom=y1, label='Dataset 2', color='orange')  # Bar for y2, stac

# Add labels and title
plt.xlabel('Category')
plt.ylabel('Value')
plt.title('Stacked Bar Plot')
plt.legend()  # Show legend for the datasets

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



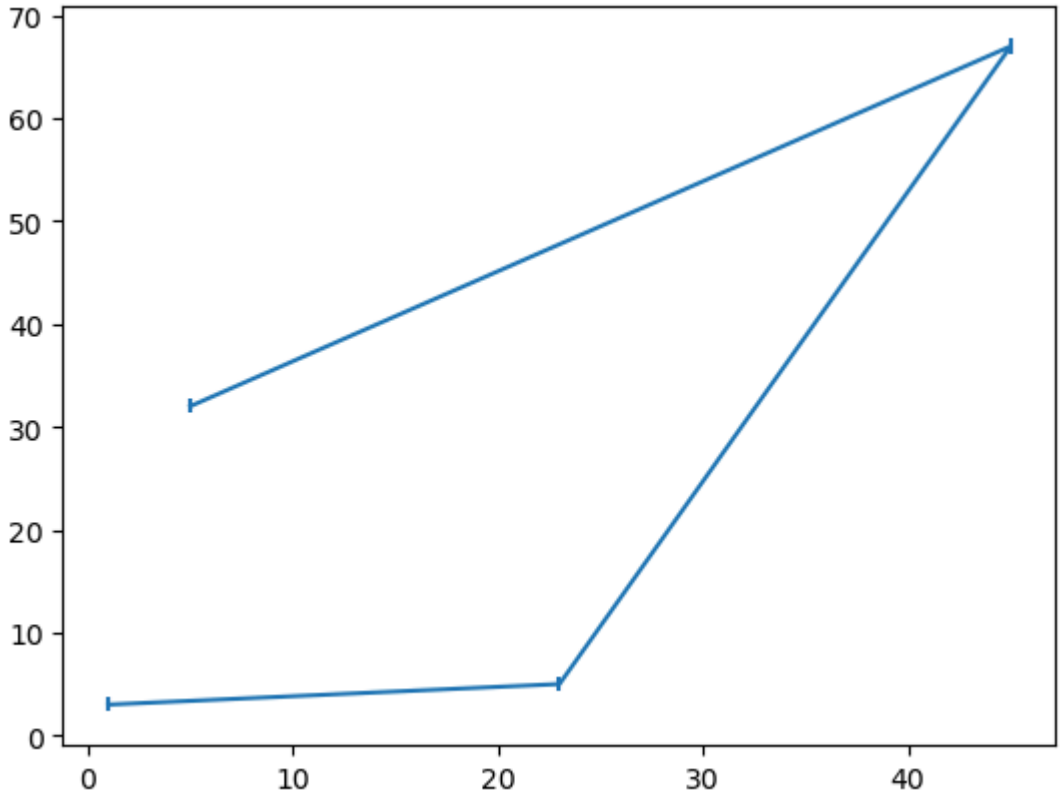
In [44]:
```
# Example y-values for three datasets
y1 = np.array([3, 5, 2, 8, 7])
y2 = np.array([2, 4, 6, 1, 5])
y3 = np.array([1, 3, 4, 2, 6])

# Create the stacked bar plot for three datasets
plt.bar(x, y1, label='Dataset 1', color='lightblue')
plt.bar(x, y2, bottom=y1, label='Dataset 2', color='orange')
plt.bar(x, y3, bottom=y1 + y2, label='Dataset 3', color='green')  # Stacked on t

# Add labels and title
plt.xlabel('Category')
plt.ylabel('Value')
plt.title('Stacked Bar Plot with Three Datasets')
plt.legend()

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



13 - Error Bar Plot

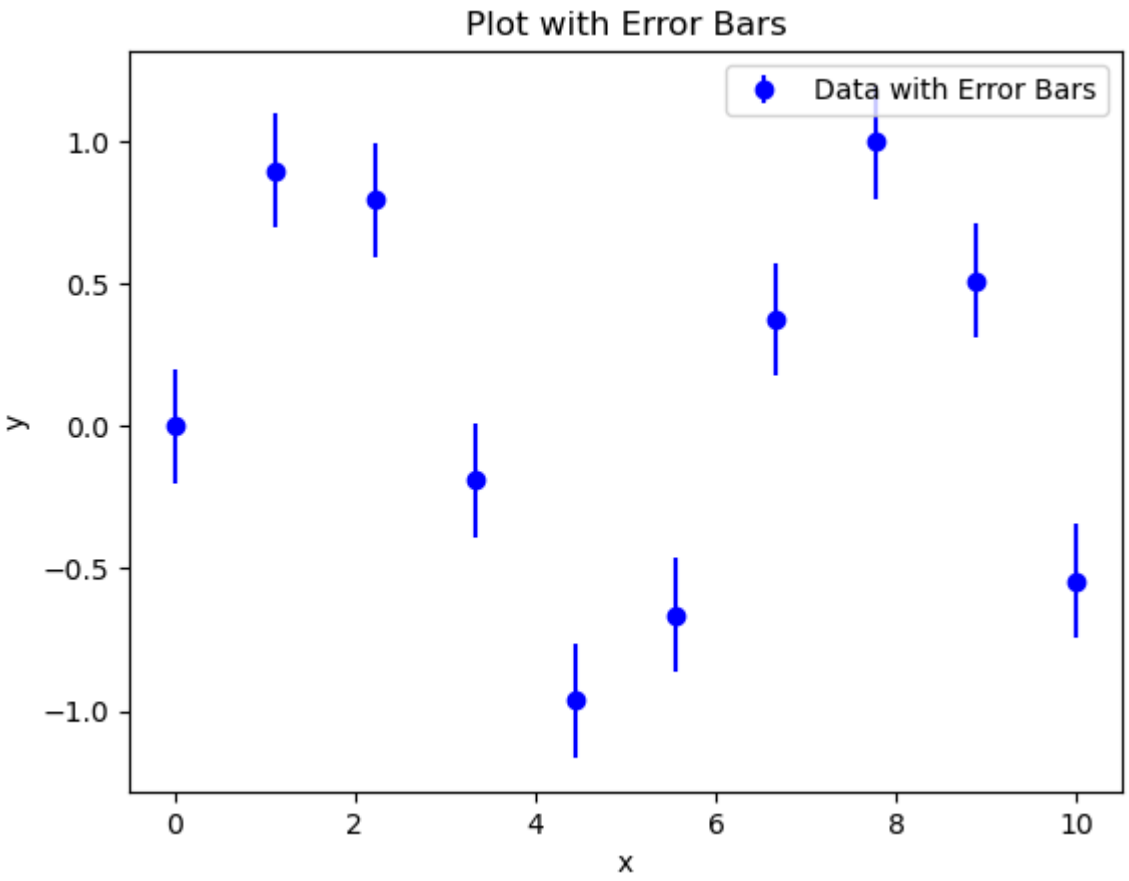
In [46]:
```
# Used to display errors or uncertainty in data points.

x = [1,23,45,5]
y = [3,5,67,32]

error=0.7
plt.errorbar(x, y, yerr=error)
```

Out[46]:
```
<ErrorbarContainer object of 3 artists>
```



In [47]:

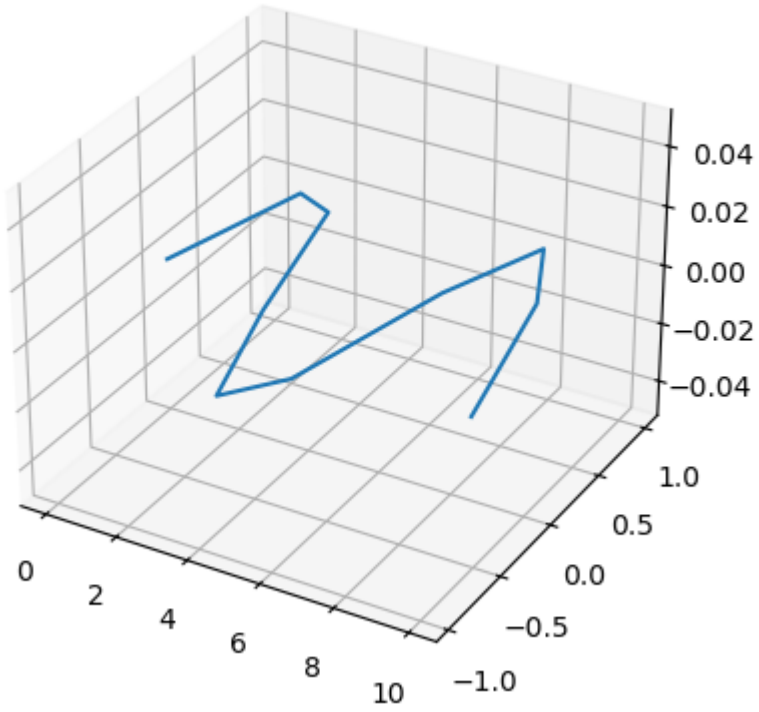
```
import matplotlib.pyplot as plt
import numpy as np

# Example x-values (independent variable)
x = np.linspace(0, 10, 10)

# Example y-values (dependent variable)
y = np.sin(x)

# Example y-error values (uncertainty in y-values)
error = 0.2  # Constant error for all points

# Plot with error bars
plt.errorbar(x, y, yerr=error, fmt='o', label='Data with Error Bars', color='blu

# Add title and labels
plt.title('Plot with Error Bars')
plt.xlabel('x')
plt.ylabel('y')

# Show legend
plt.legend()

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



14 - 3D Plot (Axes3D)

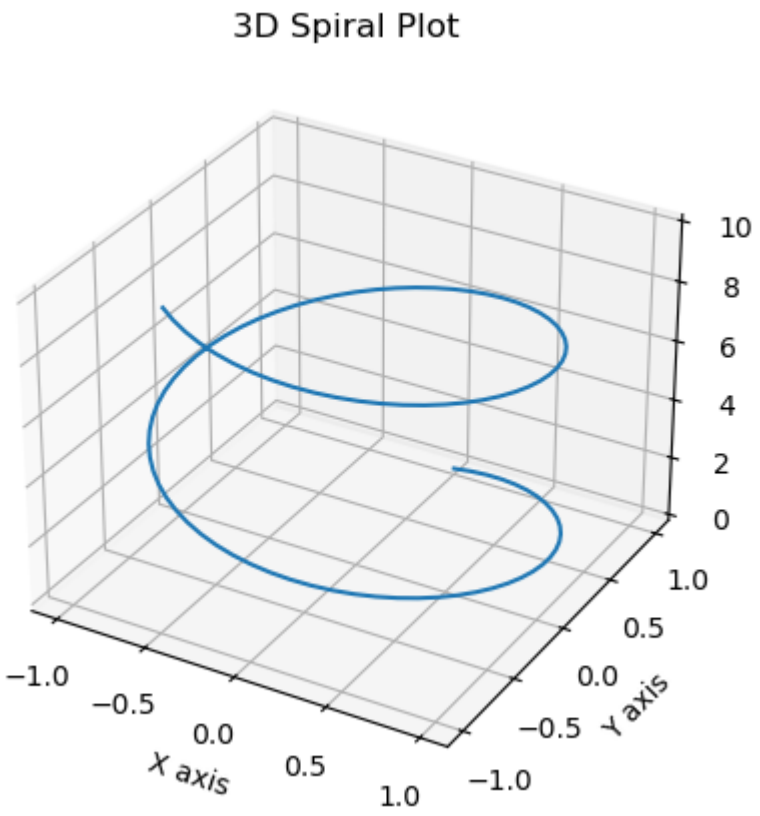
In [49]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y)
```

Out[49]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x2ae434e2d50>]
```



In [50]:

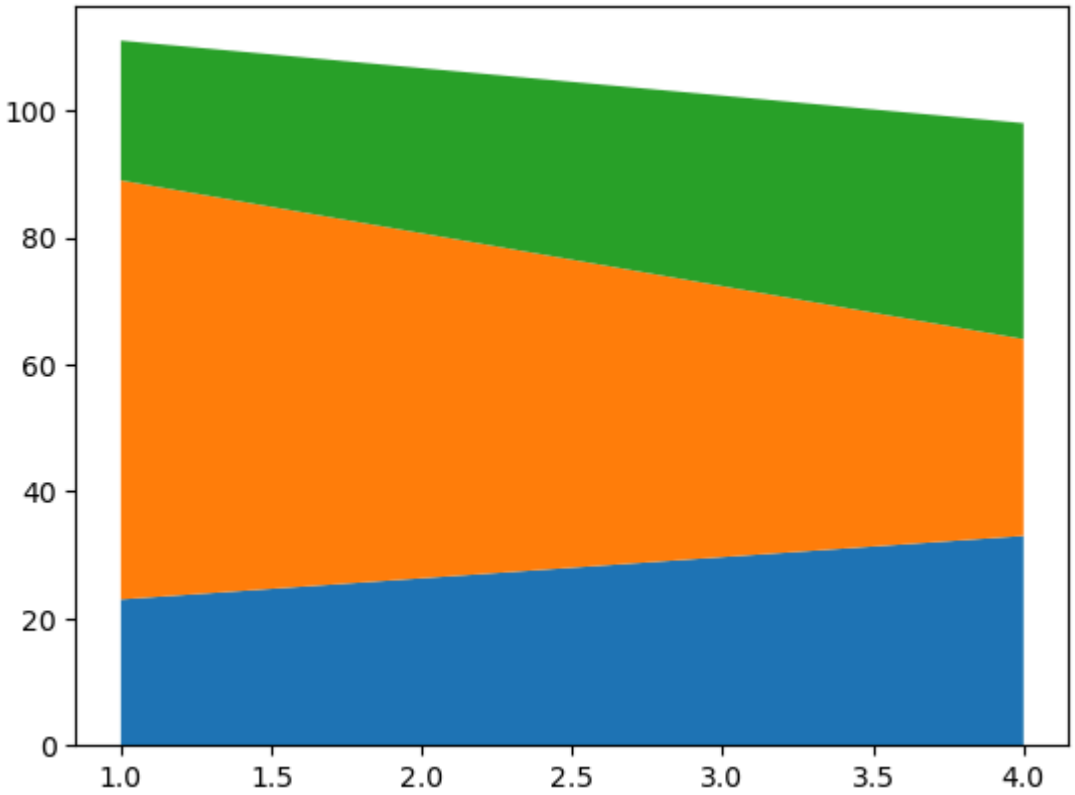
```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Generate data (example: a spiral curve in 3D)
t = np.linspace(0, 10, 100)
x = np.sin(t)
y = np.cos(t)
z = t

# Create a figure and 3D axes
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot the data
ax.plot(x, y, z)

# Add labels and title
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
ax.set_title('3D Spiral Plot')

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



15 - Stacked Area plot

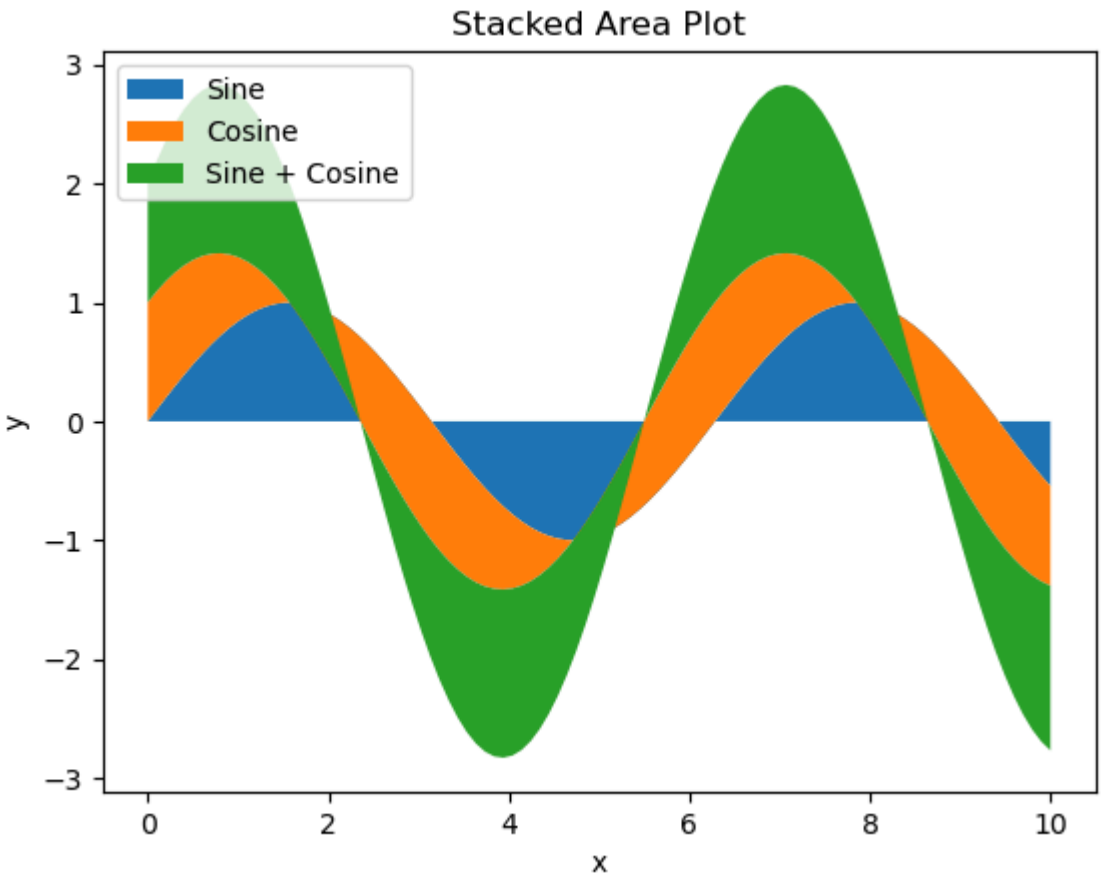
In [52]:
```
#  variation of the area plot that shows multiple datasets stacked on top of eac

x = [1,4]
y1 = [23,33]
y2 = [66,31]
y3 = [22,34]
plt.stackplot(x, y1, y2, y3)
```

Out[52]:
```
[<matplotlib.collections.PolyCollection at 0x2ae43601d60>,
 <matplotlib.collections.PolyCollection at 0x2ae423bf410>,
 <matplotlib.collections.PolyCollection at 0x2ae423a0590>]
```



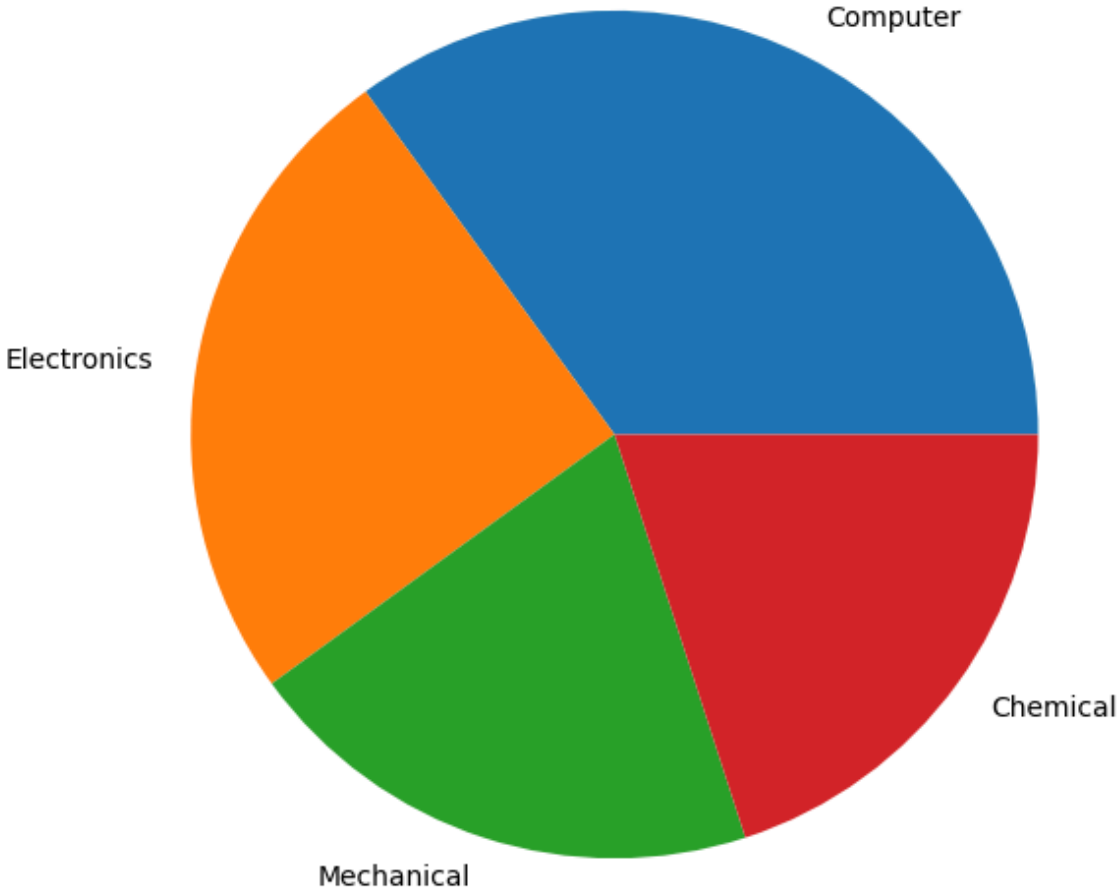
In [53]:

```
import matplotlib.pyplot as plt
import numpy as np

# Example x-values (categories or time points)
x = np.linspace(0, 10, 100)

# Example y-values for three datasets (to be stacked)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.sin(x) + np.cos(x)

# Create the stacked area plot
plt.stackplot(x, y1, y2, y3, labels=['Sine', 'Cosine', 'Sine + Cosine'])

# Add title and labels
plt.title('Stacked Area Plot')
plt.xlabel('x')
plt.ylabel('y')

# Display the legend
plt.legend(loc='upper left')

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



16 - Pie Chart (pie)

In [55]:
```
plt.figure(figsize=(7,7))

x10 = [35, 25, 20, 20]

labels = ['Computer', 'Electronics', 'Mechanical', 'Chemical']

plt.pie(x10, labels=labels);

plt.show()
```



In [ ]: