APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Data Visualization using Matplotlib

Recap

- Pandas
  - Series
  - DataFrame
- Importing the Data/ Export the data
- Cleaning of Data
  - Handling Missing Data
    - isna, dropna, fillna
  - Duplicates
    - isduplicated, drop_duplicates
  - Outliers
- Statisticals/mathematical analysis
- Data Visualization

Day Objectives

- Data viz. using Python
- Tools for Data Viz.
- History of Matplotlib
- Different plots
  - Line plot
  - Scatter Plot
  - Histogram
  - Bar Graph
  - Box plot
  - Pie Chart

Data viz.

Finding the insights from the data

Tools for Data viz.

- MS PowerBI
- Tableau
- MS Excel/ GSheets
- Datalab -> GCP
- SAS

Programming

- Python -> matplotlib, seaborn, plotly (OS, Entriprize), Bokeh, geoplotlib)
- R-Programming -> ggplot
- JSV,

Data Visualization using Matplotlib

- John D. Hunter -> Neurobioligist -> Matlab -> Matplotlib
- It is classified into 3 layers
  - Backend Layer
  - Artist layer
  - Scripting layer
    - pyplot

In [1]:

```
1 import matplotlib
```

In [2]:

```
1 matplotlib.__version__
```

Out[2]:

```
'3.2.2'
```

In [3]:

```
1 from matplotlib import pyplot as plt
2 import matplotlib.pyplot as plt
```

In [5]:

```
1 print(plt.__doc__)
```

```
`matplotlib.pyplot` is a state-based interface to matplotlib. It provides
a MATLAB-like way of plotting.

pyplot is mainly intended for interactive plots and simple cases of
programmatic plot generation::

    import numpy as np
    import matplotlib.pyplot as plt

    x = np.arange(0, 5, 0.1)
    y = np.sin(x)
    plt.plot(x, y)

The object-oriented API is recommended for more complex plots.
```

Line Plot

it is used to identify the changes in the data W.r.to time

In [6]:

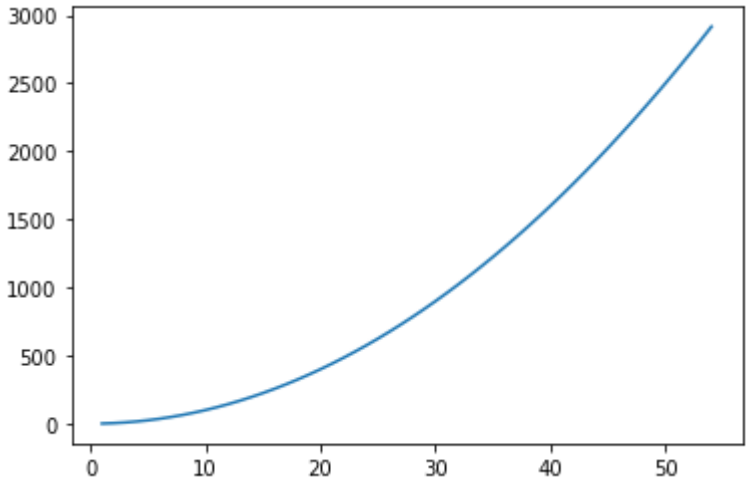
```
1 import numpy as np
```

In [7]:

```
1 x = np.arange(1, 55)
2 y = x ** 2
```

In [8]:

```
1 plt.plot(x, y)
2 plt.show()
```

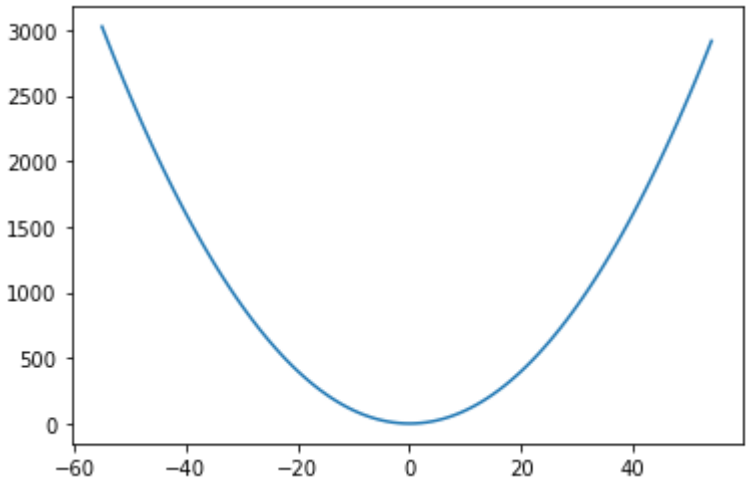


In [10]:

```
1 x = np.arange(-55, 55)
2 y = x ** 2
3
4 plt.plot(x, y)
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x23547b8f610>]
```

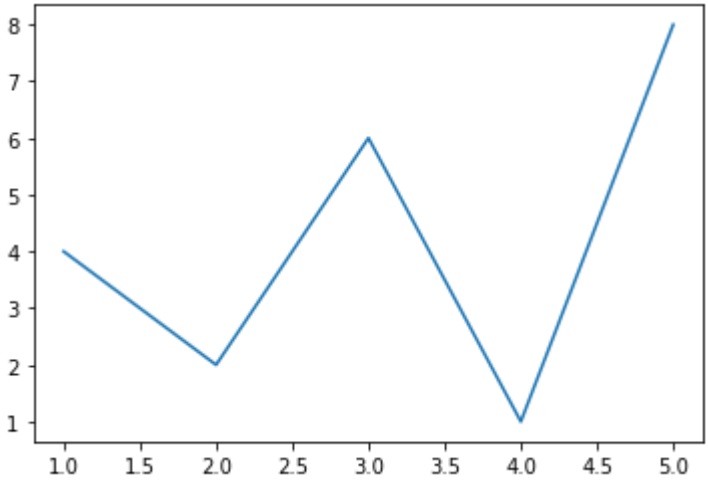


In [11]:

```
x = [1,2,3,4,5]
y = [4,2,6,1,8]

plt.plot(x, y)
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x23547cfc490>]
```



Applications

- Stock Market -> price w.r.to time
- Temp
- ECG -> Pulse
- Sales

In [12]:

```
import pandas as pd

url = "https://raw.githubusercontent.com/AP-State-Skill-Development-Corporation/Dataset
```

In [13]:

```
df = pd.read_csv(url)


df.head()
```

Out[13]:

	date	open	high	low	close	volume	Name
0	2013-02-08	27.35	27.71	27.31	27.55	33318306	MSFT
1	2013-02-11	27.65	27.92	27.50	27.86	32247549	MSFT
2	2013-02-12	27.88	28.00	27.75	27.88	35990829	MSFT
3	2013-02-13	27.93	28.11	27.88	28.03	41715530	MSFT
4	2013-02-14	27.92	28.06	27.87	28.04	32663174	MSFT

In [14]:

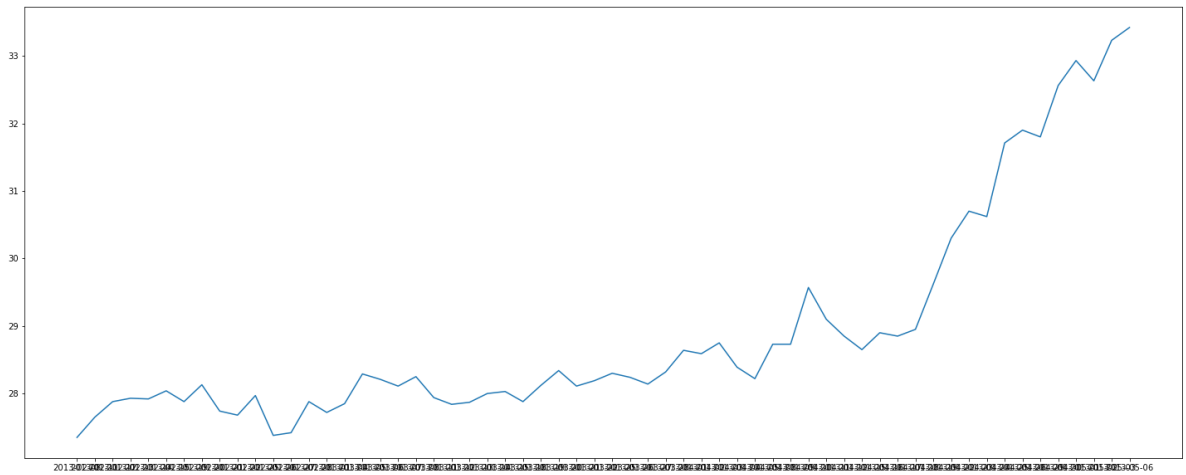
```
df.shape
```

Out[14]:

```
(1259, 7)
```

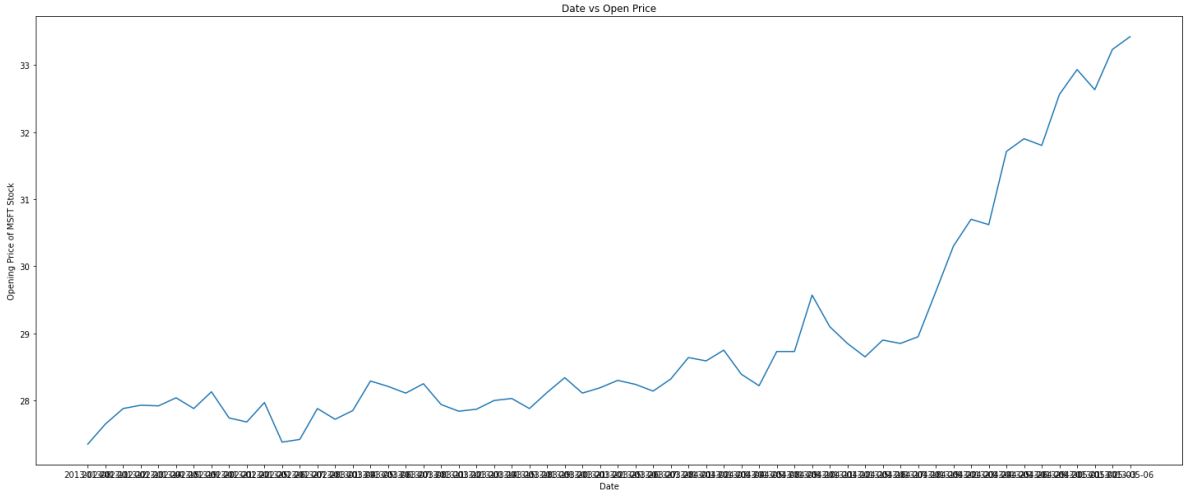
In [16]:

```
plt.figure(figsize = (25, 10)) #w, h

df1 = df.head(60)


plt.plot(df1['date'], df1['open'])
plt.show()
```



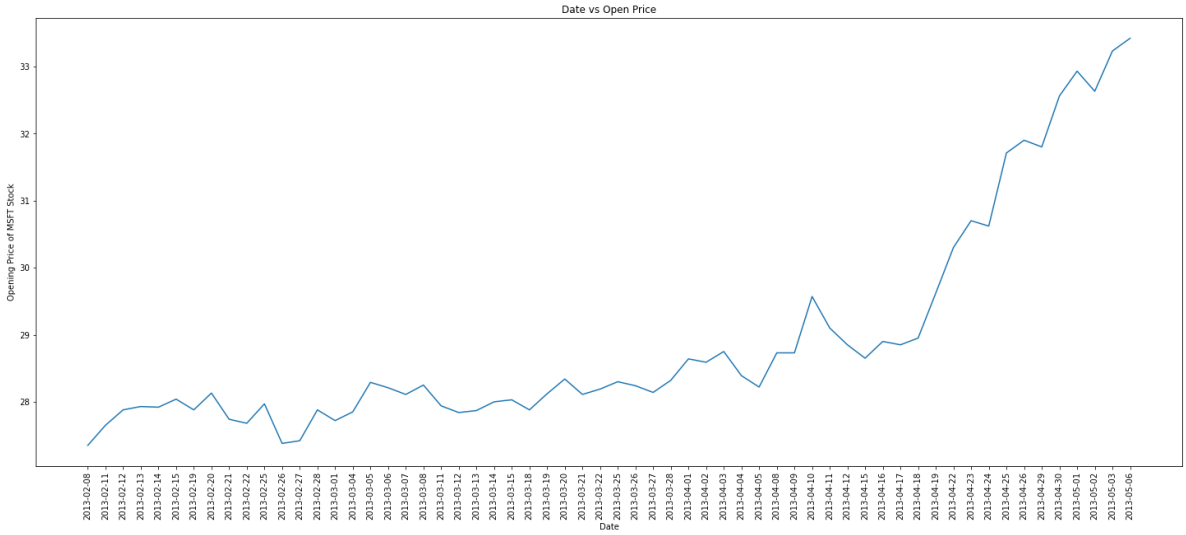
In [17]:

```
plt.figure(figsize = (25, 10)) #w, h

df1 = df.head(60)


plt.plot(df1['date'], df1['open'])
plt.xlabel('Date')
plt.ylabel("Opening Price of MSFT Stock")
plt.title("Date vs Open Price")

plt.show()
```



In [20]:

```
plt.figure(figsize = (25, 10)) #w, h

df1 = df.head(60)


plt.plot(df1['date'], df1['open'])
plt.xlabel('Date')
plt.ylabel("Opening Price of MSFT Stock")
plt.title("Date vs Open Price")

plt.xticks(df1['date'], rotation=90)

plt.show()
```

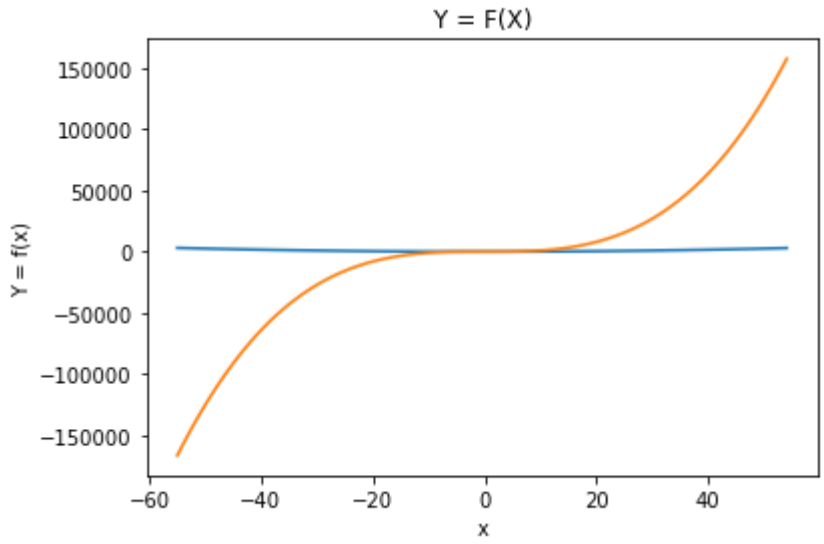


In [24]:

```
x = np.arange(-55, 55)


plt.plot(x, x ** 2)
plt.plot(x, x ** 3)
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
```

Out[24]:

```
Text(0.5, 1.0, 'Y = F(X)')
```



In [25]:

```
help(plt.plot)
```

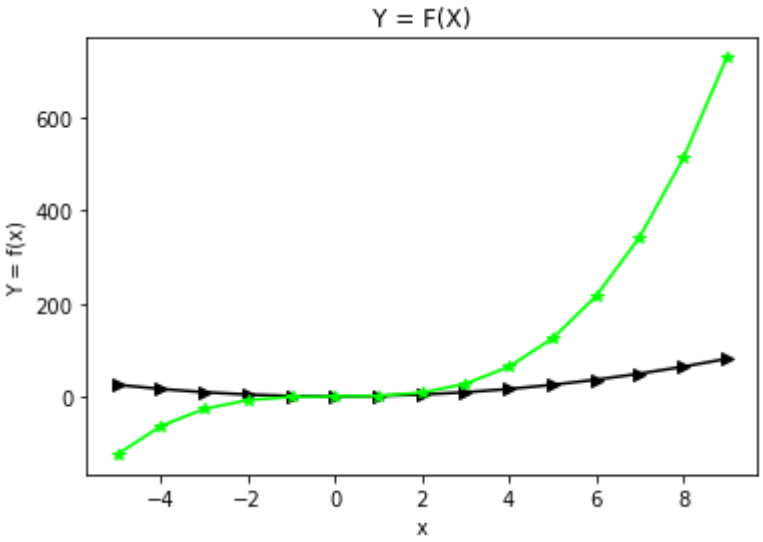
```
Help on function plot in module matplotlib.pyplot:

plot(*args, scalex=True, scaley=True, data=None, **kwargs)
    Plot y versus x as lines and/or markers.

    Call signatures::

        plot([x], y, [fmt], *, data=None, **kwargs)
        plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

    The coordinates of the points or line nodes are given by *x*, *y*.

    The optional parameter *fmt* is a convenient way for defining basic
    formatting like color, marker and linestyle. It's a shortcut string
    notation described in the *Notes* section below.

    >>> plot(x, y)        # plot x and y using default line style and colo
r
    >>> plot(x, y, 'bo')  # plot x and y using blue circle markers
```

In [27]:

```
x = np.arange(-5, 10)


plt.plot(x, x ** 2, c = 'black', marker = '>')
plt.plot(x, x ** 3, c = '#00ff00', marker = '*')
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
```

Out[27]:

```
Text(0.5, 1.0, 'Y = F(X)')
```

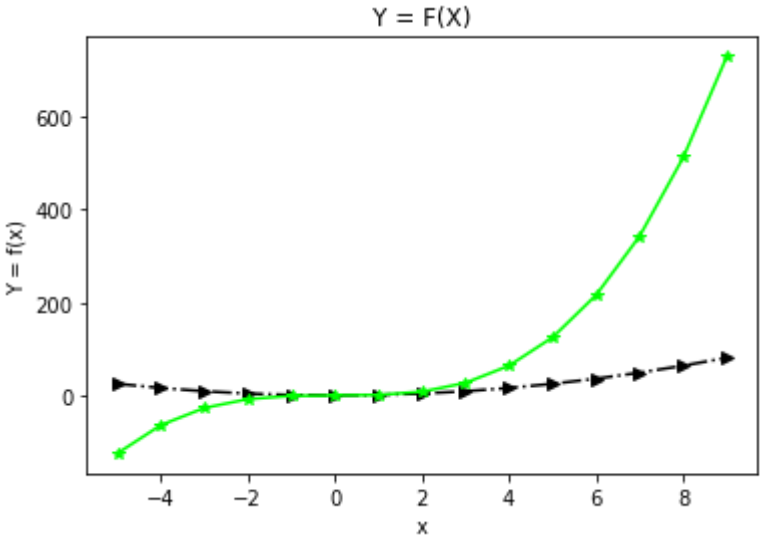


In [28]:

```
x = np.arange(-5, 10)


plt.plot(x, x ** 2, c = 'black', marker = '>', linestyle = '-.')
plt.plot(x, x ** 3, c = '#00ff00', marker = '*')
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
```

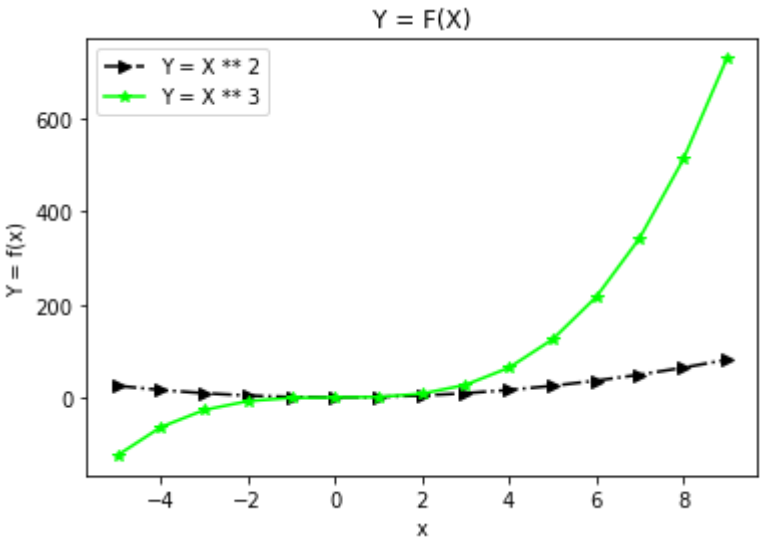
Out[28]:

```
Text(0.5, 1.0, 'Y = F(X)')
```



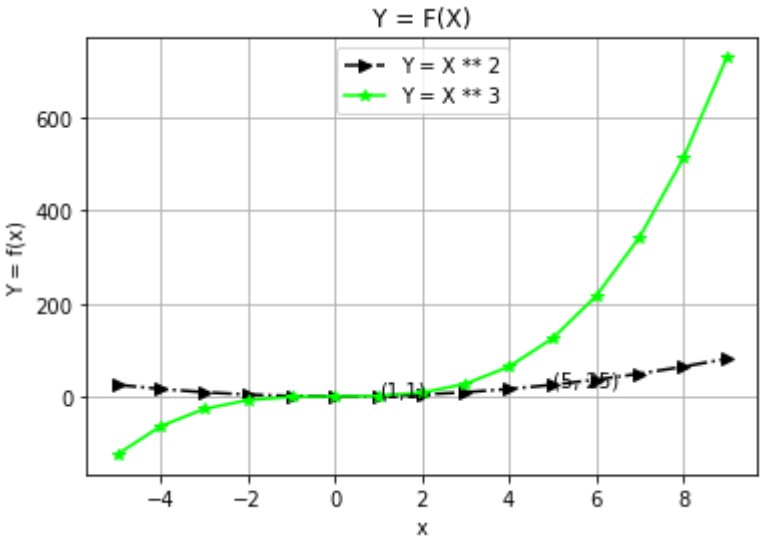
In [29]:

```
x = np.arange(-5, 10)


plt.plot(x, x ** 2, c = 'black', marker = '>', linestyle = '-.')
plt.plot(x, x ** 3, c = '#00ff00', marker = '*')
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
plt.legend(['Y = X ** 2', 'Y = X ** 3'])

plt.show()
```



In [38]:

```
x = np.arange(-5, 10)


plt.plot(x, x ** 2, c = 'black', marker = '>', linestyle = '-.')
plt.plot(x, x ** 3, c = '#00ff00', marker = '*')
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
plt.legend(['Y = X ** 2', 'Y = X ** 3'], loc = 'upper center')
plt.text(1,1, "(1,1)")
plt.text(5, 25, "(5, 25)")
plt.grid()
#plt.axis("off")

plt.show()
```



Scatter Plot

identifying the relationship between two features

- +ve
- -ve
- neutral

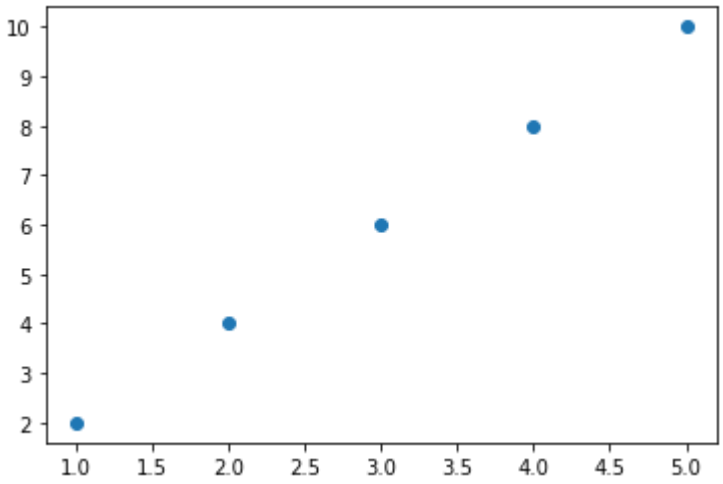
Vectorization

In [78]:

```
x = [1,2,3,4,5]
y = np.array([2,4,6,8,10])
y2 = -y
y3 = [5,5,5,5,5]
```

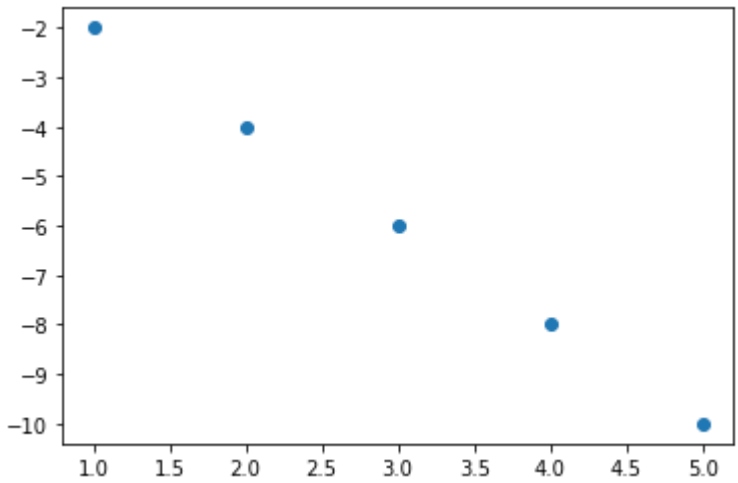
In [43]:

```
1 plt.scatter(x, y)
2 plt.show()
```



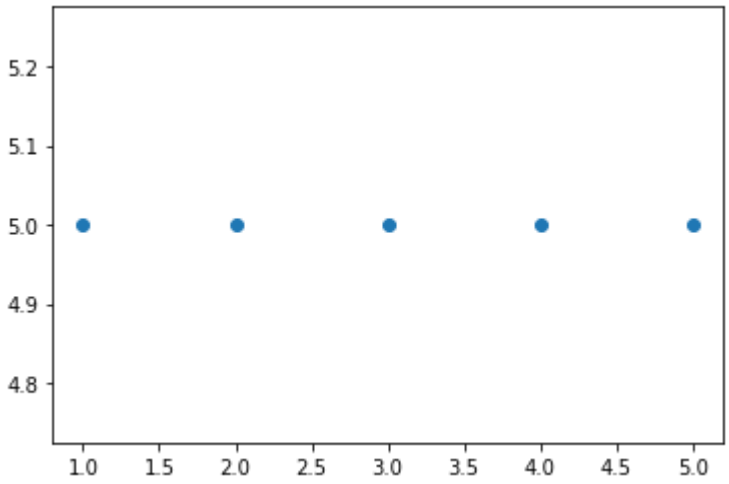
In [45]:

```
1 plt.scatter(x, y2)
2 plt.show()
```



In [46]:

```
1 plt.scatter(x, y3)
2 plt.show()
```



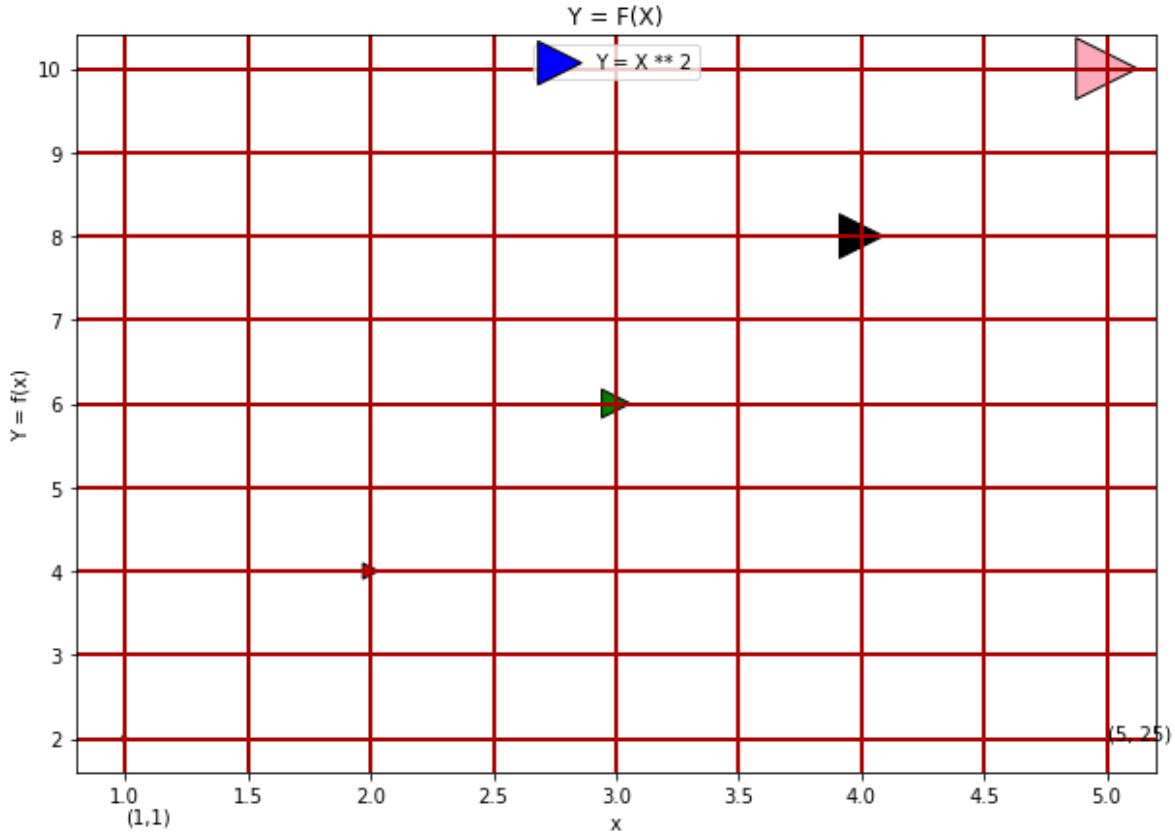
In [48]:

```
1 help(plt.scatter)
```

```
Help on function scatter in module matplotlib.pyplot:

scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=Non
e, vmax=None, alpha=None, linewidths=None, verts=<deprecated parameter>, e
dgecolors=None, *, plotnonfinite=False, data=None, **kwargs)
    A scatter plot of *y* vs. *x* with varying marker size and/or color.

    Parameters
    ----------
    x, y : scalar or array-like, shape (n, )
        The data positions.

    s : scalar or array-like, shape (n, ), optional
        The marker size in points**2.
        Default is ``rcParams['lines.markersize'] ** 2``.

    c : array-like or list of colors or color, optional
        The marker colors. Possible values:
```

In [79]:

```
plt.figure(figsize=(10, 7))
plt.scatter(x, y, marker = '>', c = ['b', 'r', 'g', 'black', '#ffaabb'], alpha = 1.0, s
plt.xlabel("x")
plt.ylabel("Y = f(x)")
plt.title("Y = F(X)")
plt.legend(['Y = X ** 2'], loc = 'upper center')
plt.text(1,1, "(1,1)")
plt.text(5, 2, "(5, 25)")
plt.grid(color='#aa0000', linestyle='-', linewidth=2)
#plt.axis("off")

plt.show()
```



In [63]:

```
help(plt.grid)
```

```
        contains: callable
        dash_capstyle: {'butt', 'round', 'projecting'}
        dash_joinstyle: {'miter', 'round', 'bevel'}
        dashes: sequence of floats (on/off ink in points) or (None, None)
        data: (2, N) array or two 1D arrays
        drawstyle or ds: {'default', 'steps', 'steps-pre', 'steps-mid', 's
teps-post'}, default: 'default'
        figure: `.Figure`
        fillstyle: {'full', 'left', 'right', 'bottom', 'top', 'none'}
        gid: str
        in_layout: bool
        label: object
        linestyle or ls: {'-', '--', '-.', ':', '', (offset, on-off-seq),
...}
        linewidth or lw: float
        marker: marker style
        markeredgecolor or mec: color
        markeredgewidth or mew: float
        markerfacecolor or mfc: color
```

Bar Graph

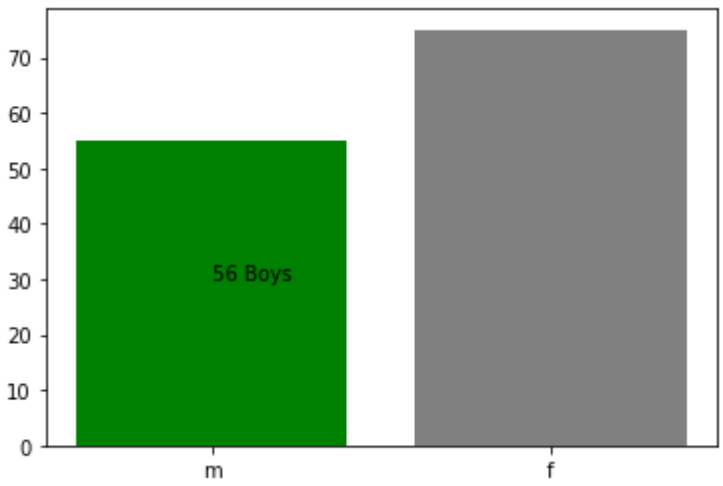
count of the categorical data

In [73]:

```
x = ['m', 'f']
y = [55, 75]

plt.bar(x, y, color = ['green', 'grey'])
plt.text(0, 30, "56 Boys")
```

Out[73]:

```
Text(0, 30, '56 Boys')
```



In [74]:

```
1 help(plt.text)
```

```
Help on function text in module matplotlib.pyplot:

text(x, y, s, fontdict=None, withdash=<deprecated parameter>, **kwargs)
    Add text to the axes.

    Add the text *s* to the axes at location *x*, *y* in data coordinates.

    Parameters
    ----------
    x, y : scalars
        The position to place the text. By default, this is in data
        coordinates. The coordinate system can be changed using the
        *transform* parameter.

    s : str
        The text.

    fontdict : dictionary, optional, default: None
        A dictionary to override the default text properties. If fontdict
```

In [ ]:

```
1
```