TOPIC-6: Visualizing Data with Graphs- Plotting with matplotlib-2D plots- plotting the scalar and vector fields, Scatter Plots, and Graph customization.

2-D Plot

'Matplotlib' is a Python library for publication-quality 2D and 3D graphics, with support for a variety of different output formats. More information about Matplotlib is available at the project’s web site www.matplotlib.org. This web site contains detailed documentation and an extensive gallery that showcases the various types of graphs that can be generated using the Matplotlib library, together with the code for each example.

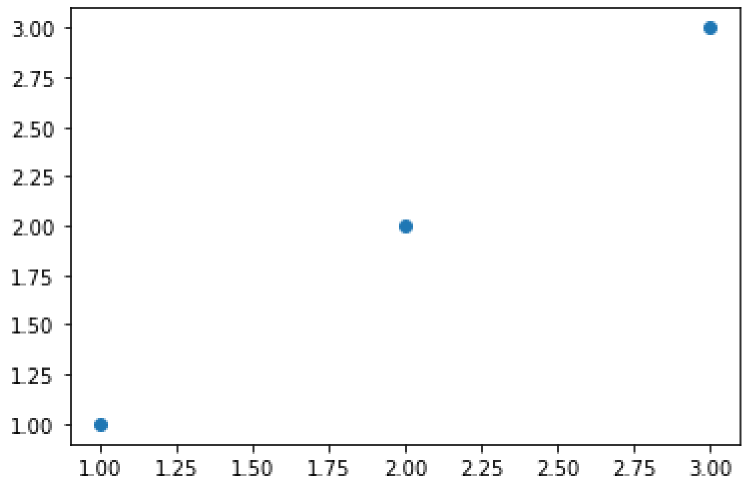
Matplotlib is imported using the following standard convention:

In [2]:
```
%matplotlib inline
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

'%matplotlib' inline configures the Matplotlib to use the “inline” backend, which results in the created figures being displayed directly in, for example, the Jupyter Notebook, rather than in a new window. The statement 'import matplotlib as mpl' imports the main Matplotlib module, and the import statement 'import matplotlib.pyplot as plt', is for convenient access to the submodule 'matplotlib.pyplot' that provides the functions that we will use to create new Figure instances.

the pyplot style:

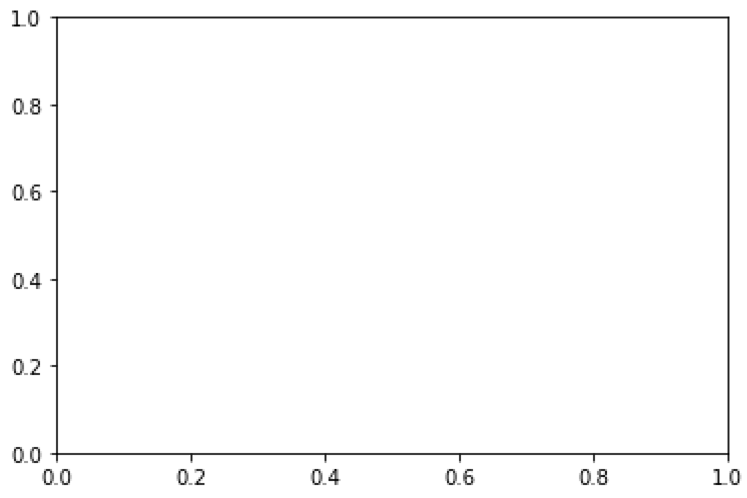
In [54]:
```
x = [1, 2, 3]
y = [1, 2, 3]
plt.plot(x, y)
plt.show()
```



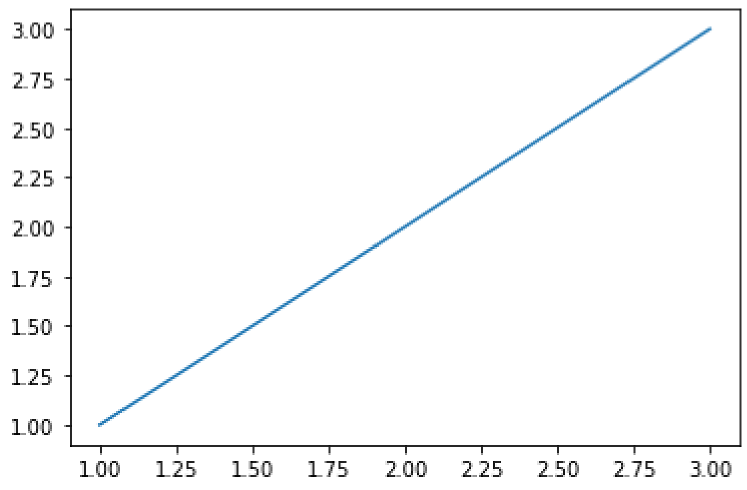
This program creates a matplotlib window that shows a line passing through the given points. Under the hood, when the plt.plot() function is called, a Figure object is created, within which the axes are created, and finally the data is plotted within the axes.

The Object Oriented(OO)-style. The following program re-creates this plot, but we'll also explicitly create the Figure object and add axes to it, instead of just calling the plot() function and relying on it to create those:

In [15]:
```
fig = plt.figure()#cretes a figure explicitly
ax = plt.axes()#includes a single axes in the figure explicitly

plt.show()
```



In [8]:
```
fig = plt.figure()
ax = plt.axes()
plt.plot(x, y)
plt.show()
```

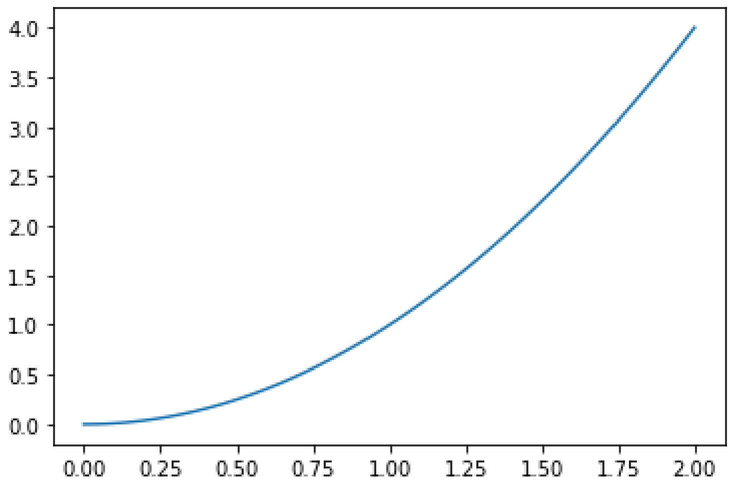


We create the Figure object using the 'figure()' function, and then we create the axes using the 'axes()' function. The 'axes()' function also adds the axes to the Figure object. The last two lines are the same as in the earlier program. This time, when we call the plot() function, it sees that a Figure object with an Axes object already exists and directly proceeds to plot the data supplied to it.

Plotting a function

In [3]:
```
#plot the function f(x)=x^2.
x = np.linspace(0, 2, 100)
```

```
plt.plot(x,x**2)
plt.show()
```

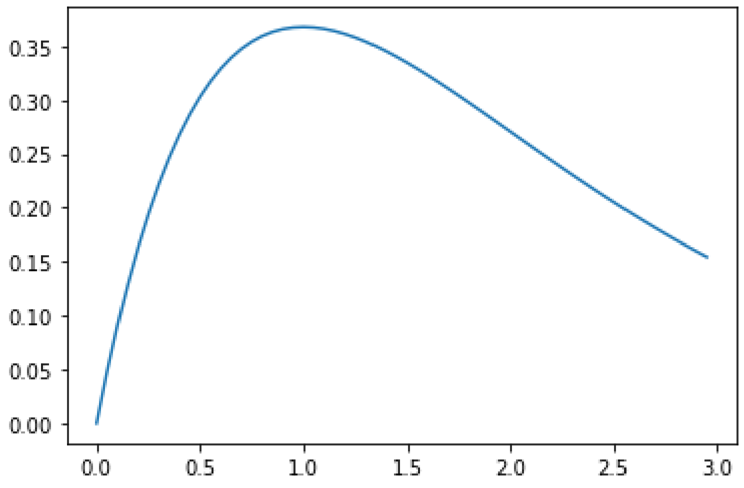


In [5]:

```
#plot f(x) = x * exp(-x).

def f(x):
    return x*np.exp(-x)
x = np.arange   ( start = 0.
                 , stop = 3.
                 , step = 0.05
                 ) # generate points between start and stop with distances of step ap
y = f(x)
plt.plot( x, y)
plt.show
```

Out[5]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

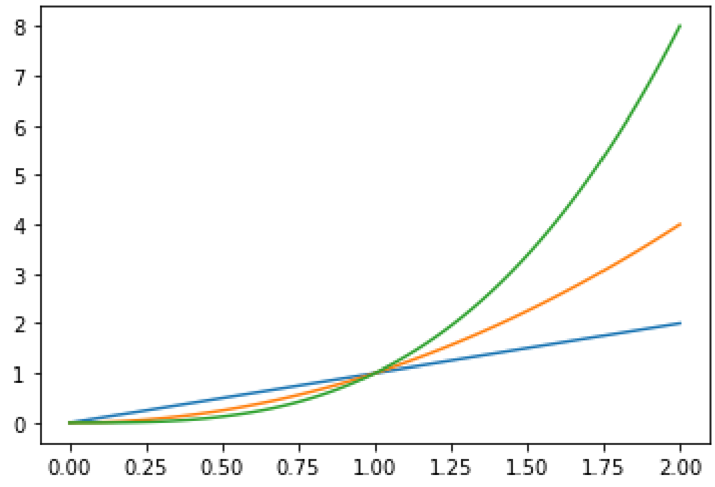


More than one plot in the same figure

the pyplot-style:

In [11]:

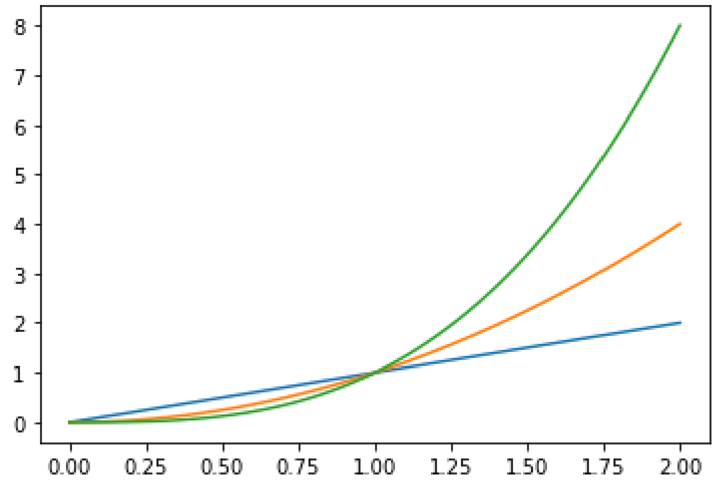
```
#plot the funnctions f(x) = x, x^2, x^3.

x = np.linspace(0, 2, 100)
plt.plot(x,x)
plt.plot(x,x**2)
plt.plot(x,x**3)
plt.show()
```



the OO-style:

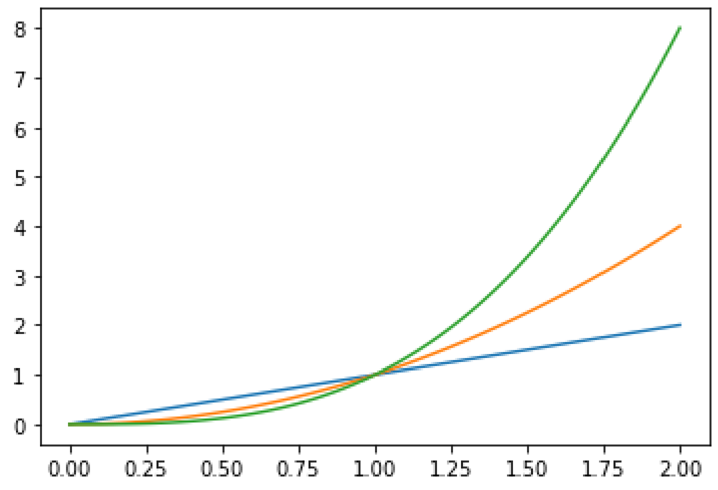
In [19]:

```python
fig = plt.figure()
ax = plt.axes()
ax.plot(x, x)  # Plot some data on the axes.
ax.plot(x, x**2)  # Plot more data on the axes...
ax.plot(x, x**3)
plt.show()
```



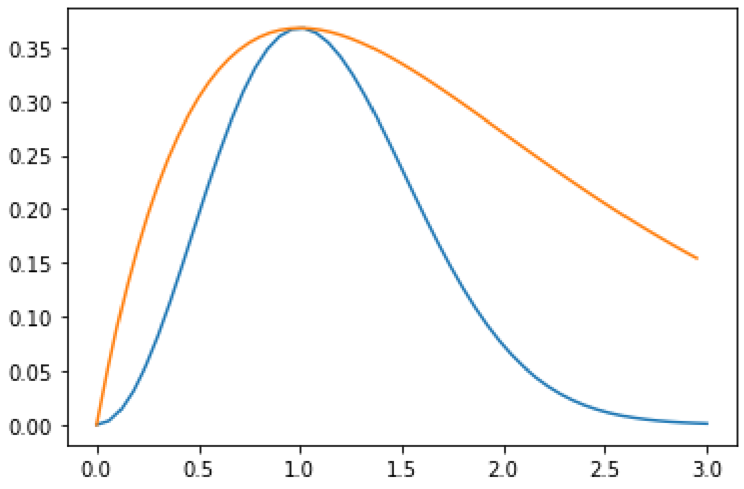
In [21]:

```python
# using the command subplots().
fig, ax = plt.subplots()
ax.plot(x, x)  # Plot some data on the axes.
ax.plot(x, x**2)  # Plot more data on the axes...
ax.plot(x, x**3)  # ... and some more.
plt.show()
```



In [6]:

```
#plot the functions f(x) = x^2 * exp(-x^2) and g(x) = x * exp(-x).

def f(x):
    return x**2*np.exp(-x**2)
x = np.linspace ( start = 0.    # lower limit
                , stop = 3      # upper limit
                , num = 51      # generate 51 points between 0 and 3
                )


def g(x):
    return x*np.exp(-x)
xx = np.arange  ( start = 0.
                , stop = 3.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap

y = f(x)
yy = g(xx)


plt.plot(x,y)
plt.plot( xx, yy)
plt.show()
```



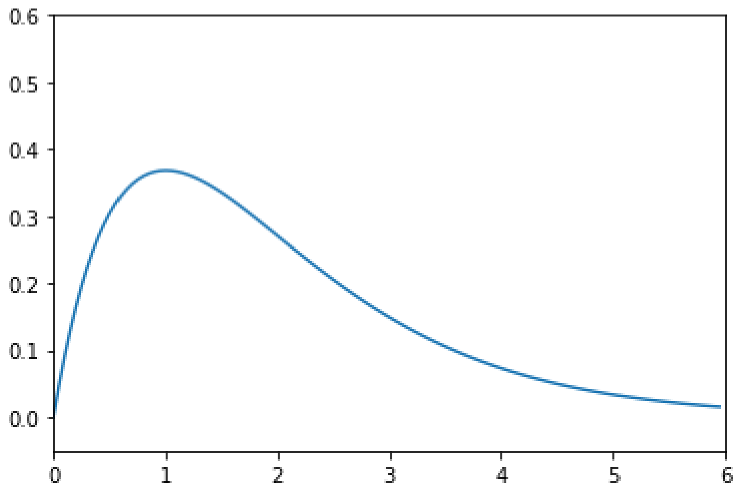
Setting the limits of the plot’s axes

we use the command plt.axis([xmin, xmax, ymin, ymax]) to set the range of the axes.

In [22]:

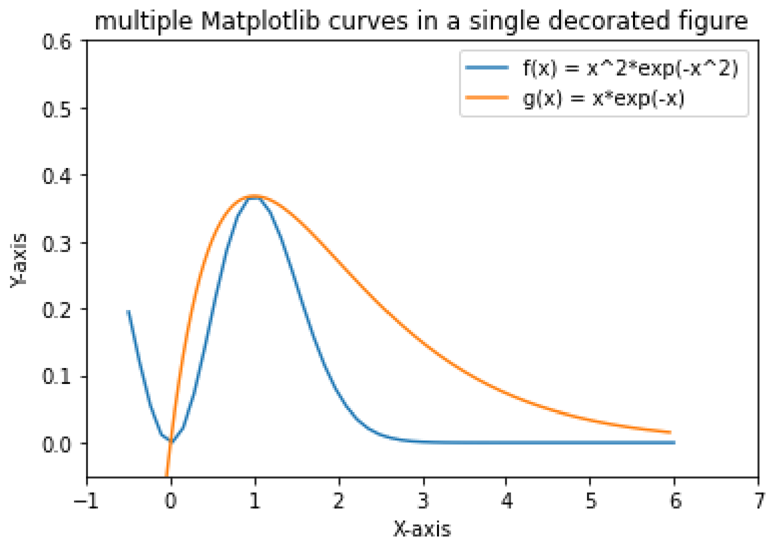
```
xx = np.arange  ( start = 0.
                , stop = 6.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap
plt.plot( xx, g(xx))
plt.axis([0, 6, -0.05, 0.6]) # [xmin, xmax, ymin, ymax]
```

Out[22]:
```
(0.0, 6.0, -0.05, 0.6)
```



Adding the axis-labels, figure-title, and legends

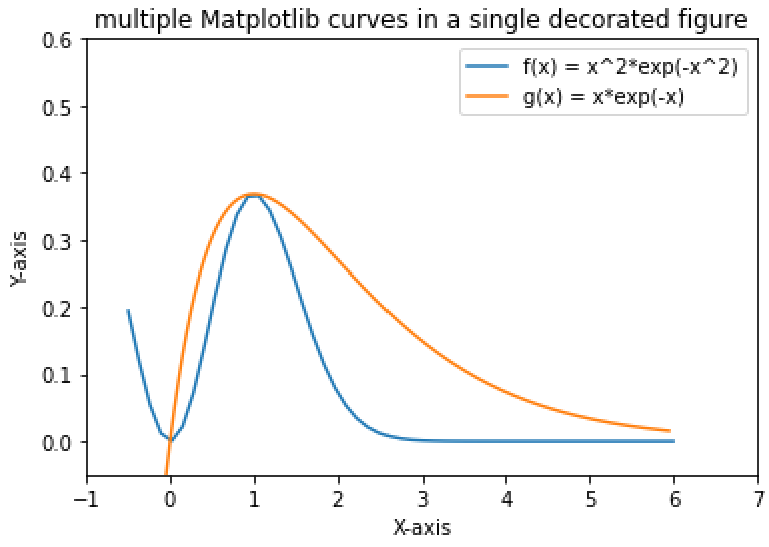
In [29]:
```
def f(x):
    return x**2*np.exp(-x**2)
x = np.linspace ( start = -0.5    # lower limit
                , stop = 6      # upper limit
                , num = 51      # generate 51 points between 0 and 3
                )


def g(x):
    return x*np.exp(-x)
xx = np.arange  ( start = -0.5
                , stop = 6.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap

y = f(x)
yy = g(xx)


plt.plot(x,y)
plt.plot( xx, yy)
plt.axis([-1, 7, -0.05, 0.6])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend( [ 'f(x) = x^2*exp(-x^2)'   # This is f(x)
            , 'g(x) = x*exp(-x)'       # This is g(x)
            ] )
plt.title('multiple Matplotlib curves in a single decorated figure')
plt.show()
```



Saving figures as external files

In [33]:

```
y = f(x)
yy = g(xx)


plt.plot(x,y)
plt.plot( xx, yy)
plt.axis([-1, 7, -0.05, 0.6])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend( [ 'f(x) = x^2*exp(-x^2)'   # This is f(x)
            , 'g(x) = x*exp(-x)'       # This is g(x)
            ] )
plt.title('multiple Matplotlib curves in a single decorated figure')
plt.savefig('multipleCurvesFullRangeDecorated.pdf') # produces a PDF file containing
plt.savefig('multipleCurvesFullRangeDecorated.png') # produces a PNG file containing
plt.show()
```



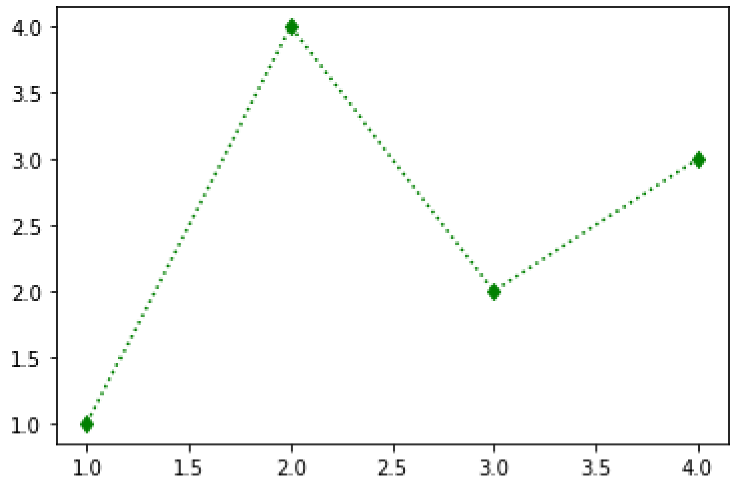
Plotting line-styles and symbols

We can set various linestyles and different symbols for the points to be plotted.

linestyle=':'. '-.', '--' and '-' are the line styles. marker='^', '1', '2', '3', '4', '8', '>', '<', 'v', 'd', 's', 'o', '+', 'x', 'X', 'd', 'D', 'h', 'H', 'p' are some of the symbols for the points to be plotted

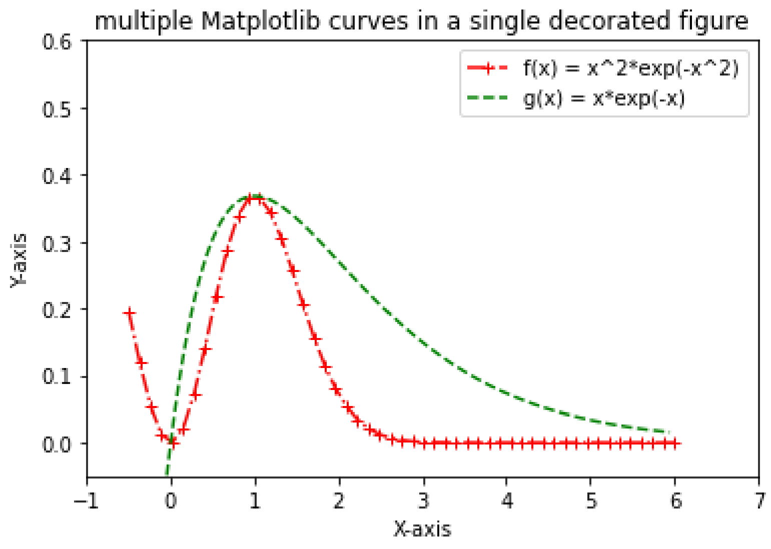
In [52]:

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
plt.plot(x, y, color = 'g', linestyle=':', marker='d')
plt.show()
```



In [48]:

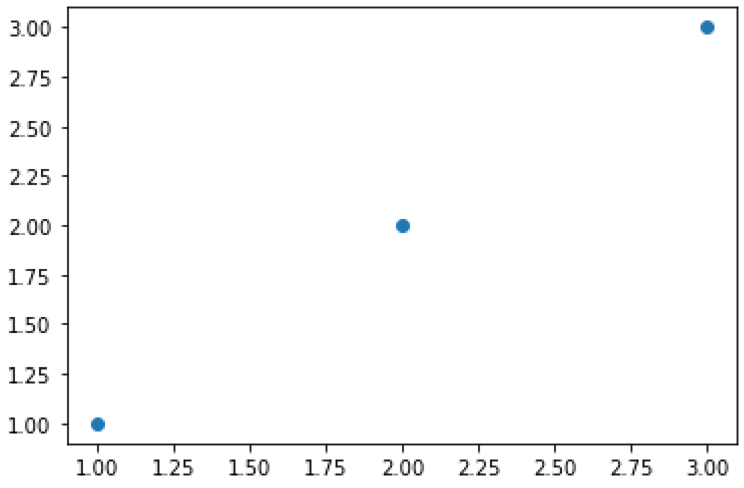
```
x = np.linspace ( start = -0.5    # lower limit
                , stop = 6      # upper limit
                , num = 51
                )

xx = np.arange  ( start = -0.5
                , stop = 6.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap

y = f(x)
yy = g(xx)


plt.plot(x,y, 'r', linestyle='-.')
plt.plot( xx, yy, 'g', linestyle='--')
plt.axis([-1, 7, -0.05, 0.6])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend( [ 'f(x) = x^2*exp(-x^2)'   # This is f(x)
            , 'g(x) = x*exp(-x)'       # This is g(x)
            ] )
plt.title('multiple Matplotlib curves in a single decorated figure')

plt.show()
```



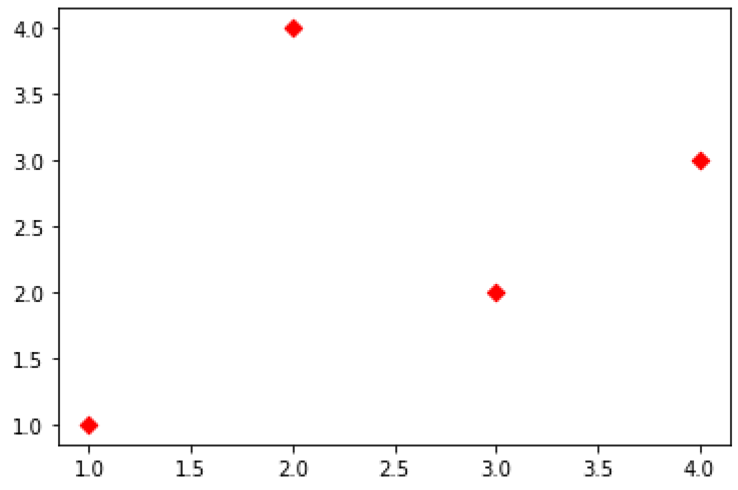
Scatter Plot

The command 'scatter()' is used to obtain a scatter plot.

In [55]:
```
x = [1, 2, 3]
y = [1, 2, 3]
plt.scatter(x, y)
plt.show()
```

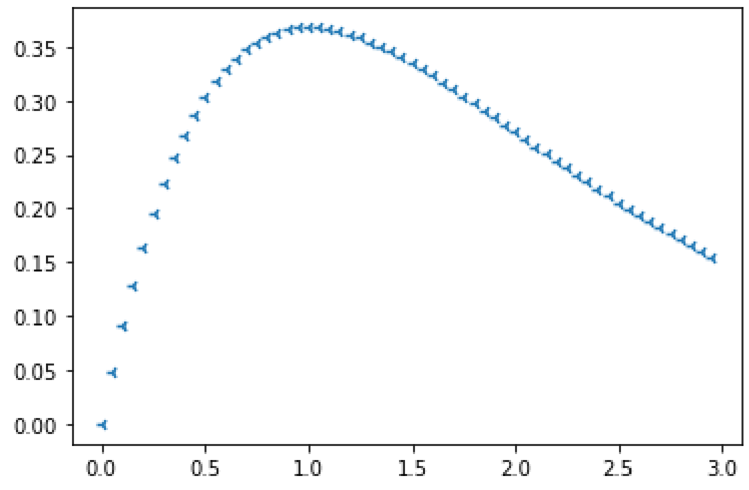


In [58]:
```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
plt.scatter(x, y, color = 'r', marker='D')
plt.show()
```



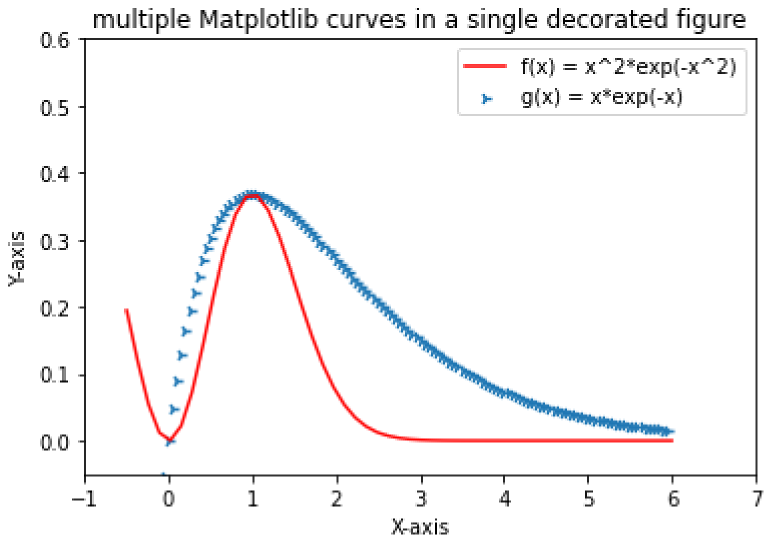
In [60]:
```
def f(x):
    return x*np.exp(-x)
x = np.arange  ( start = 0.
                , stop = 3.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap
y = f(x)
plt.scatter( x, y, marker='3')
plt.show
```

Out[60]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [64]:
```
def f(x):
    return x**2*np.exp(-x**2)
x = np.linspace ( start = -0.5    # lower limit
                , stop = 6      # upper limit
                , num = 51      # generate 51 points between 0 and 3
                )


def g(x):
    return x*np.exp(-x)
xx = np.arange  ( start = -0.5
                , stop = 6.
                , step = 0.05
                ) # generate points between start and stop with distances of step ap

y = f(x)
yy = g(xx)
```

```
plt.plot(x,y, color='r')
plt.scatter( xx, yy, marker = '4')
plt.axis([-1, 7, -0.05, 0.6])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend( [ 'f(x) = x^2*exp(-x^2)'   # This is f(x)
            , 'g(x) = x*exp(-x)'       # This is g(x)
            ] )
plt.title('multiple Matplotlib curves in a single decorated figure')
plt.show()
```

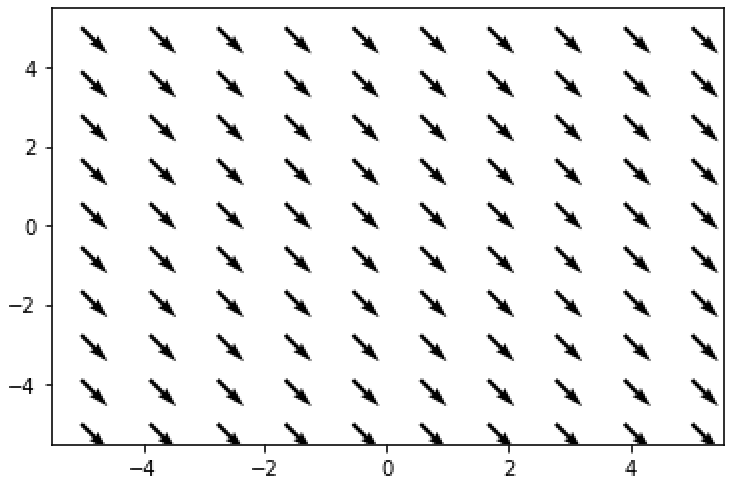


Vector Fields

Plot the vector field $\vec{F}(x, y) = i - j$.

In [2]:

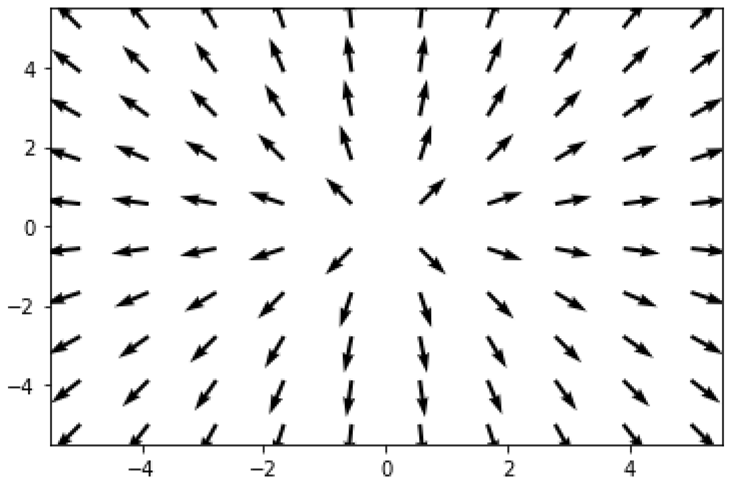
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x,y = np.meshgrid(np.linspace(-5,5,10),np.linspace(-5,5,10))

u = 1
v = -1

plt.quiver(x,y,u,v)
plt.show()
```



Plot the vector field $\vec{F}(x,y) = \frac{x}{\sqrt{x^2+y^2}} i + \frac{y}{\sqrt{x^2+y^2}} j$

In [3]:

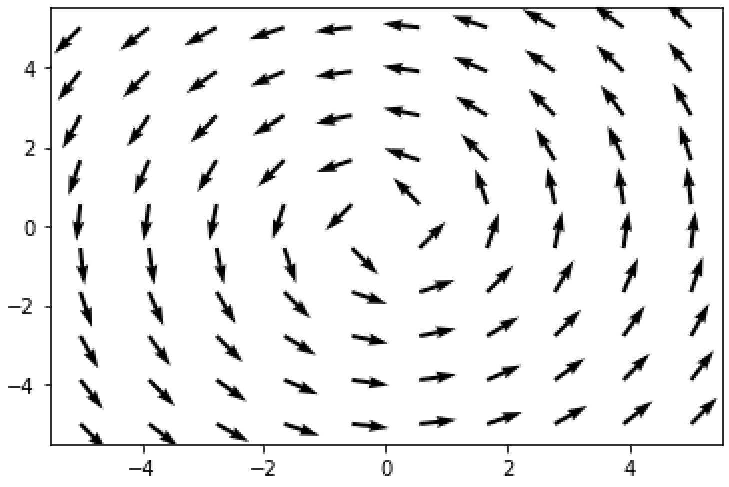
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x,y = np.meshgrid(np.linspace(-5,5,10),np.linspace(-5,5,10))

u = x/np.sqrt(x**2 + y**2)
v = y/np.sqrt(x**2 + y**2)

plt.quiver(x,y,u,v)
plt.show()
```



Plot the vector field $\vec{F}(x,y) = -\frac{y}{\sqrt{x^2+y^2}} i + \frac{x}{\sqrt{x^2+y^2}} j$

In [4]:

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

x,y = np.meshgrid(np.linspace(-5,5,10),np.linspace(-5,5,10))

u = -y/np.sqrt(x**2 + y**2)
v = x/np.sqrt(x**2 + y**2)

plt.quiver(x,y,u,v)
plt.show()
```



Visualizing data from CSV file in Python

CSV stands for 'Comma-Separated Values'. It means the data(values) in a CSV file are separated by a delimiter i.e., comma. Data in a CSV file is stored in tabular format with an extension of .csv. Generally, CSV files are used with Google spreadsheets or Microsoft Excel sheets. A CSV file contains a number of records with the data spread across rows and columns. We are going to visualize data from a CSV file in Python.

To extract the data in CSV file, CSV module must be imported in our program as follows:

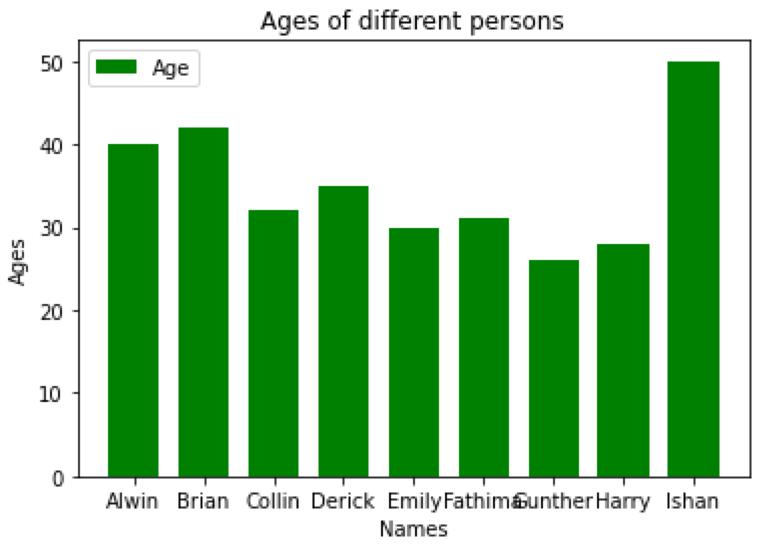
In [4]:

```
import csv
rows=[]
with open('biostats.csv') as File:
    Line_reader = csv.reader(File)
    header=next(Line_reader)
    for row in Line_reader:
        rows.append(row)

print(header)
print(rows)
```

```
['ï»¿Name', 'Sex', 'Age']
[['Alwin', 'M', '40'], ['Brian', 'M', '42'], ['Collin', 'M', '32'], ['Derick', 'M',
'35'], ['Emily', 'F', '30'], ['Fathima', 'F', '31'], ['Gunther', 'M', '26'], ['Harr
y', 'M', '28'], ['Ishan', 'M', '50']]
```

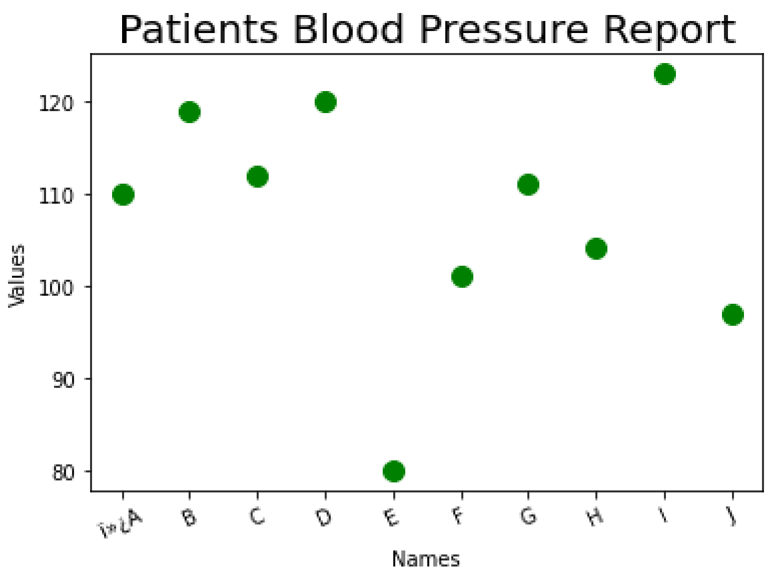
In [11]:

```
import matplotlib.pyplot as plt
import csv

x = []
y = []

with open('biostats.csv','r') as csvfile:
    plots = csv.reader(csvfile, delimiter = ',')

    for row in plots:
        x.append(row[0])
        y.append(int(row[2]))

plt.bar(x, y, color = 'g', width = 0.72, label = "Age")
plt.xlabel('Names')
plt.ylabel('Ages')
plt.title('Ages of different persons')
plt.legend()
plt.show()
```



In [3]:

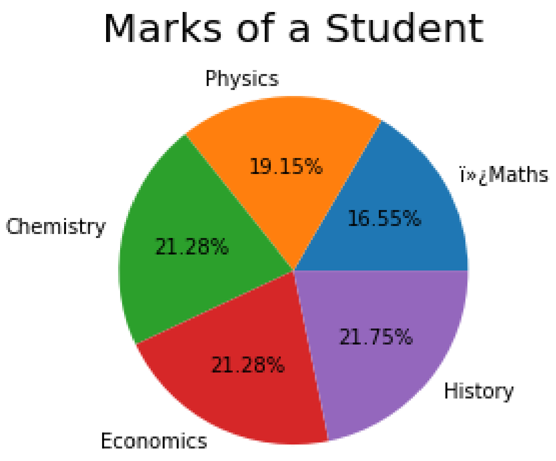
```
import matplotlib.pyplot as plt
import csv

Names = []
Values = []

with open('bldprs_measure.csv','r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    for row in lines:
        Names.append(row[0])
        Values.append(int(row[1]))

plt.scatter(Names, Values, color = 'g',s = 100)
plt.xticks(rotation = 25)
plt.xlabel('Names')
plt.ylabel('Values')
plt.title('Patients Blood Pressure Report', fontsize = 20)

plt.show()
```



In [4]:

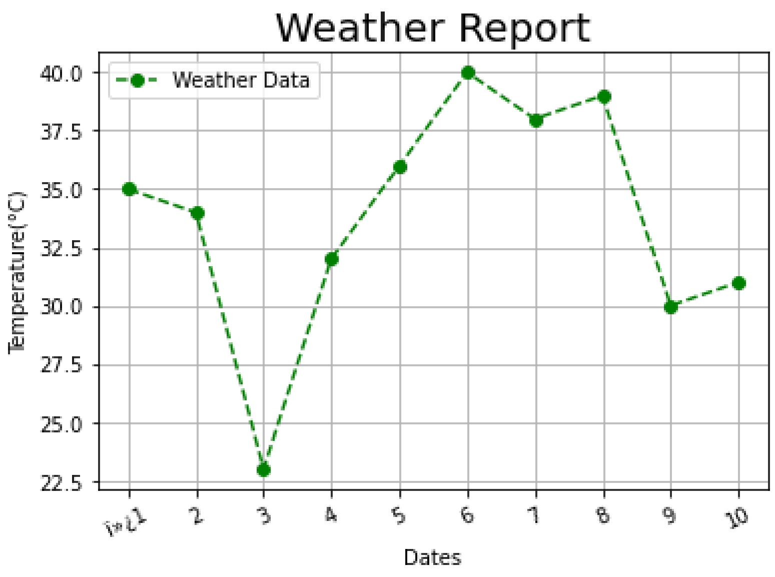
```
import matplotlib.pyplot as plt
import csv

Subjects = []
Scores = []
```

```
with open('SubjectMarks.csv', 'r') as csvfile:
    lines = csv.reader(csvfile, delimiter = ',')
    for row in lines:
        Subjects.append(row[0])
        Scores.append(int(row[1]))

plt.pie(Scores,labels = Subjects,autopct = '%.2f%%')
plt.title('Marks of a Student', fontsize = 20)
plt.show()
```



In [5]:

```
import matplotlib.pyplot as plt
import csv

x = []
y = []

with open('Weatherdata.csv','r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    for row in lines:
        x.append(row[0])
        y.append(int(row[1]))

plt.plot(x, y, color = 'g', linestyle = 'dashed',
         marker = 'o',label = "Weather Data")

plt.xticks(rotation = 25)
plt.xlabel('Dates')
plt.ylabel('Temperature(°C)')
plt.title('Weather Report', fontsize = 20)
plt.grid()
plt.legend()
plt.show()
```


Weather Report
Weather Data
Temperature(°C)
Dates
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
1
2
3
4
5
6
7
8
9
10

```
In [ ]:
```