chapter 02 - visualizing the data graphs

In [1]:
```
simplest = [1,2,3] # working with lists
```

In [2]:
```
simplest[0], simplest[1], simplest[2]
```

Out[2]:
```
(1, 2, 3)
```

In [3]:
```
stringlist = ['a string','b string','c string']

stringlist[0], stringlist[1] , stringlist[2]
```

Out[3]:
```
('a string', 'b string', 'c string')
```

In [4]:
```
emptylist = [] #empty list
emptylist
```

Out[4]:
```
[]
```

In [5]:
```
emptylist.append('rahul')
emptylist
```

Out[5]:
```
['rahul']
```

In [6]:
```
emptylist.append('machine learning engineer ') # appending the items in list
emptylist
```

Out[6]:
```
['rahul', 'machine learning engineer ']
```

In [7]:
```
simpletuple = (1,2,3)
```

In [8]:
```
simpletuple[0], simpletuple[1],simpletuple[2]
```

Out[8]:
```
(1, 2, 3)
```

In [9]:
```
l = [1,2,3]
for item in l:
  print(item)
```

```
1
2
3
```

In [10]:
```
l = [1,2,3]
for index, item in enumerate(l): #enumarating the items
  print(index, item)
```

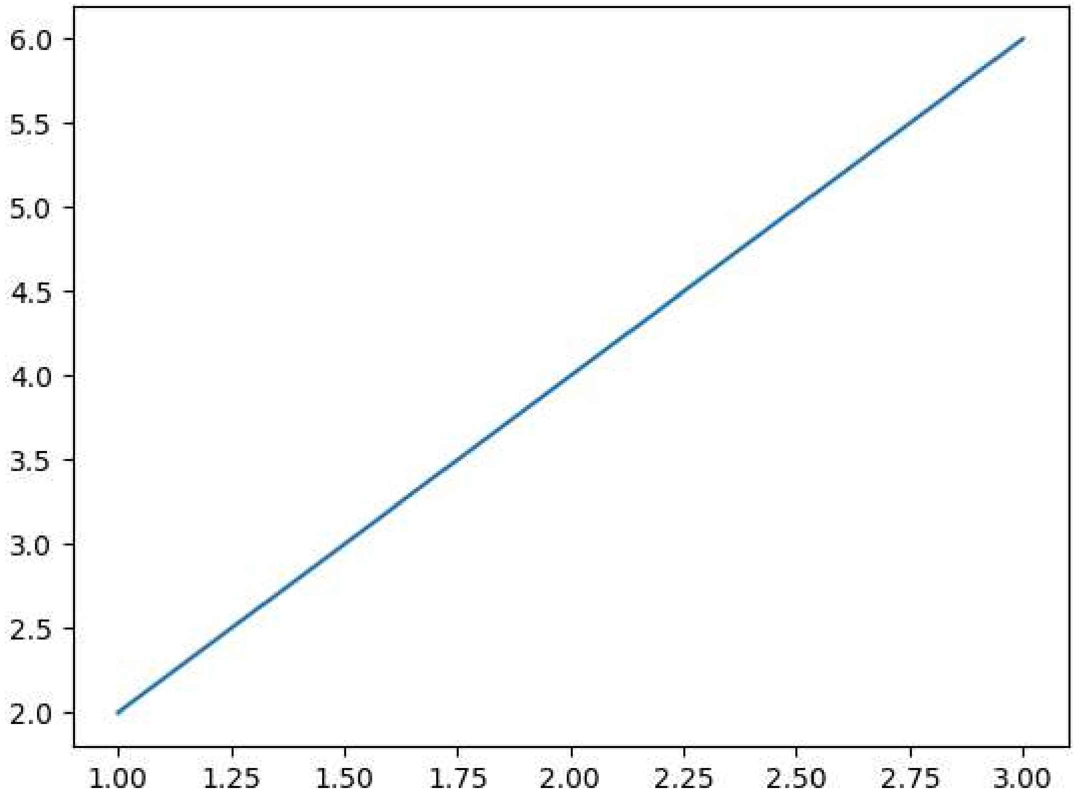
```
0 1
1 2
2 3
```

creating the graphs with the matplotlib

In [11]:
```
x_numbers = [1,2,3]
y_numbers = [2,4,6]
```

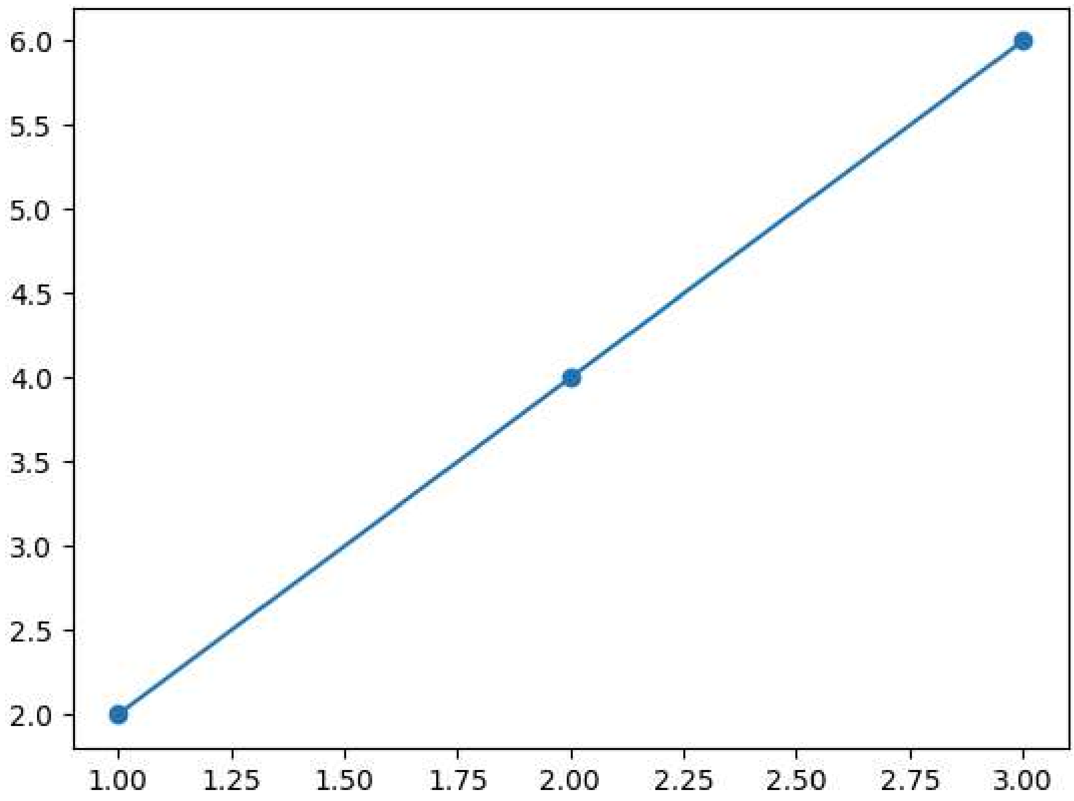
In [12]:
```
from pylab import plot, show
```

In [13]:
```
plot(x_numbers, y_numbers)
show()
```



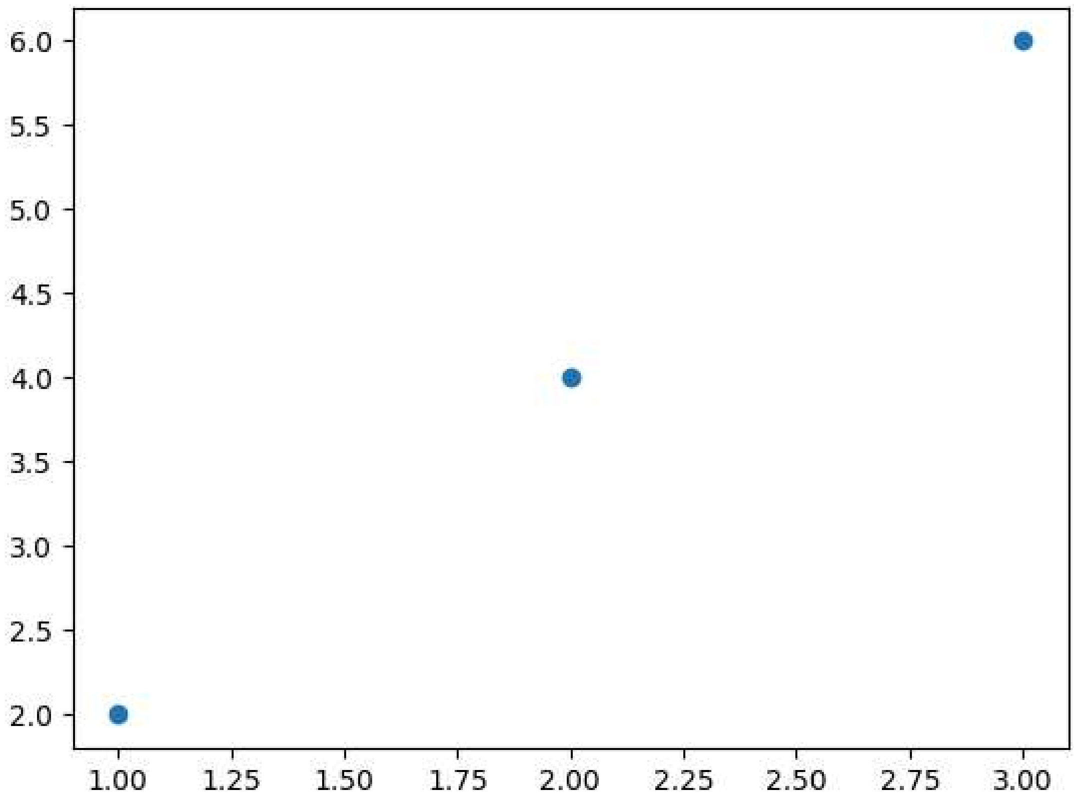
```
In [14]: plot(x_numbers, y_numbers , marker = 'o')
```

```
Out[14]: [<matplotlib.lines.Line2D at 0x7e9dcc13cee0>]
```



```
In [15]: plot(x_numbers, y_numbers ,'o')
```

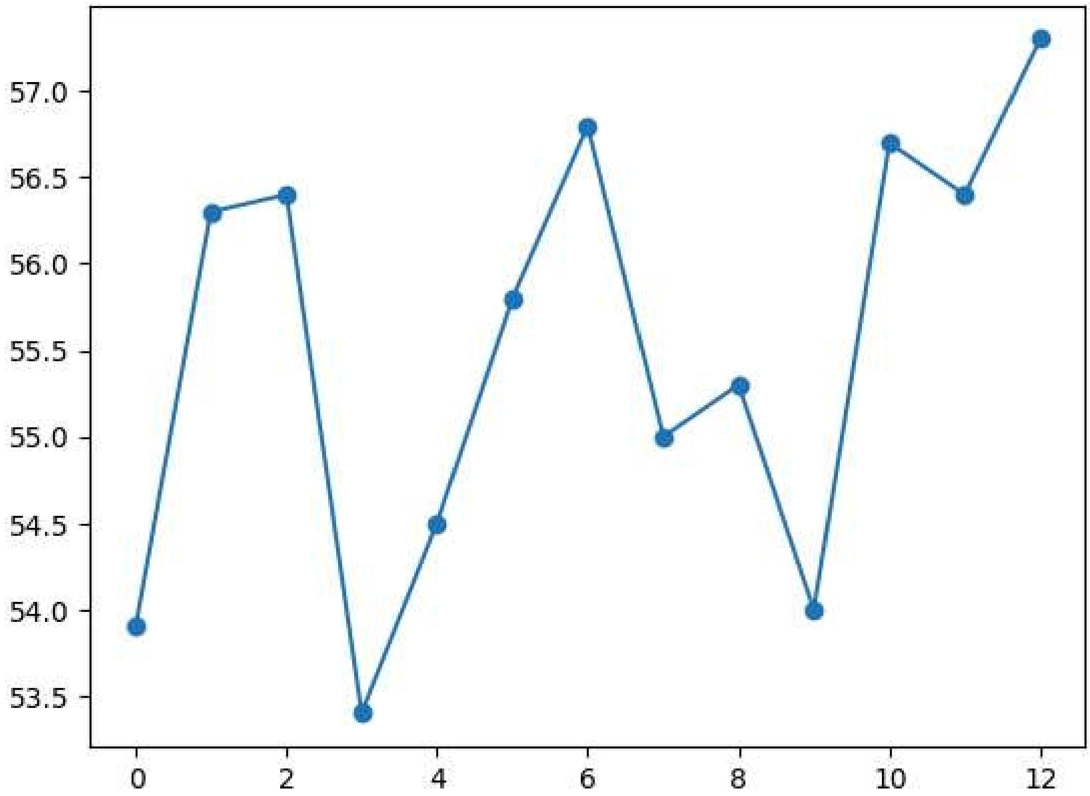
```
Out[15]: [<matplotlib.lines.Line2D at 0x7e9dcc1c20e0>]
```



graphing the avg temprature of the in newyork city

In [16]:
```
nyc_temp = [53.9, 56.3, 56.4,53.4,54.5,55.8,56.8,55.0,55.3,54.0,56.7,56.4,57.
3]
plot(nyc_temp, marker = 'o')
```

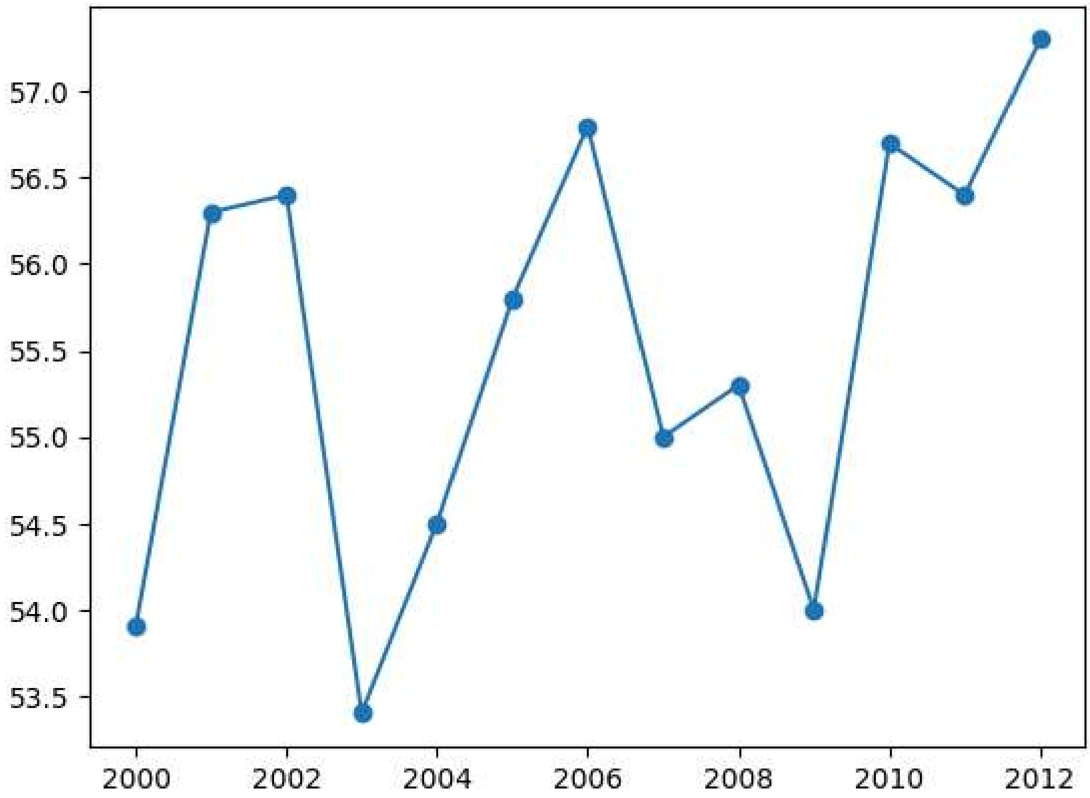
Out[16]:
```
[<matplotlib.lines.Line2D at 0x7e9dcc053430>]
```



adding the year range

In [17]:

```
nyc_temp = [53.9, 56.3, 56.4,53.4,54.5,55.8,56.8,55.0,55.3,54.0,56.7,56.4,57.3]
years = range(2000, 2013)
plot(years,nyc_temp, marker = 'o')
show()
```



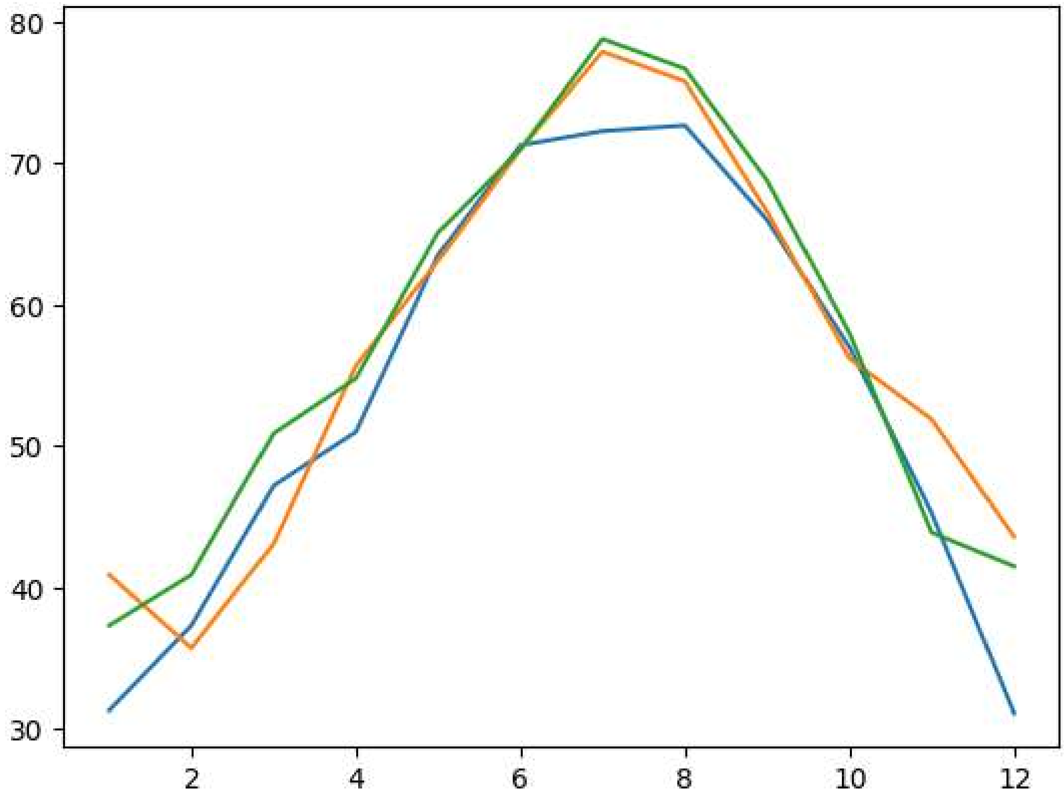
ploting the graph of warangal of specific years

In [18]:

```
warangal_temp_2000 = [31.3,37.3,47.2,51.0,63.5,71.3,72.3,72.7,66.0,57.0,45.3,31.1]
warangal_temp_2006 = [40.9,35.7,43.1,55.7,63.1,71.0,77.9,75.8,66.6,56.2,51.9,43.6]
warangal_temp_2012 = [37.3,40.9,50.9,54.8,65.1,71.0,78.8,76.7,68.8,58.0,43.9,41.5]
months = range(1,13)
plot(months,warangal_temp_2000,months,warangal_temp_2006, months,warangal_temp_2012)
```

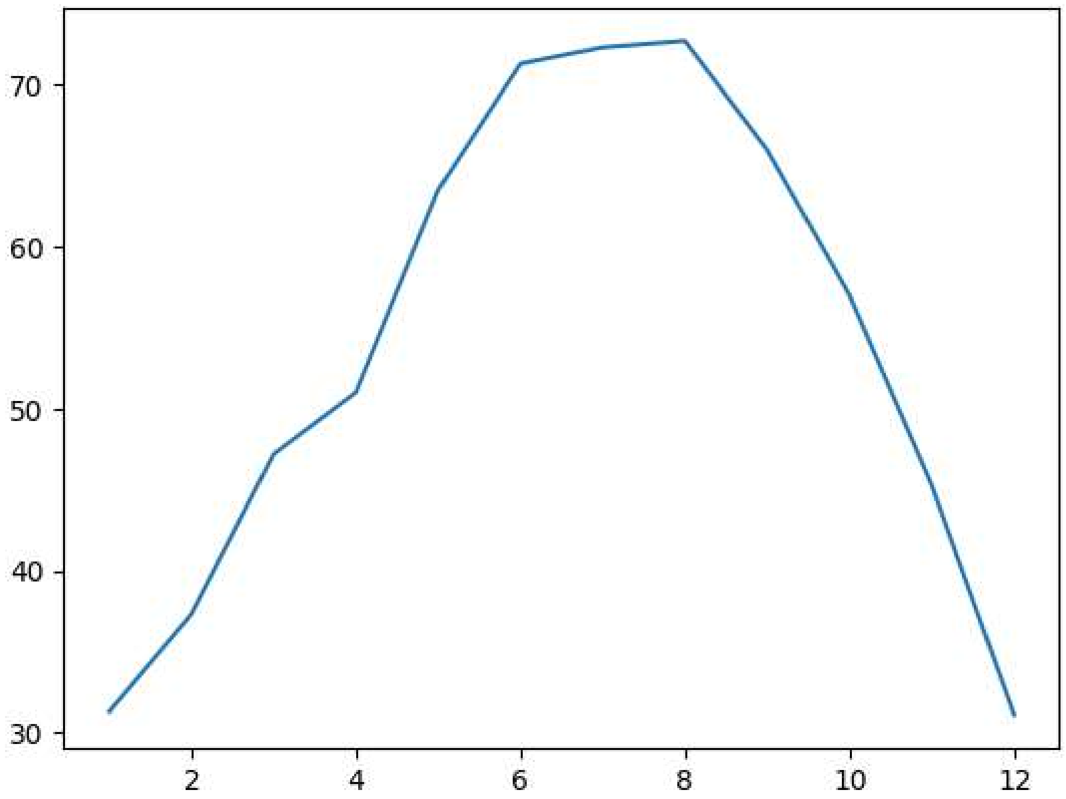
Out[18]:

```
[<matplotlib.lines.Line2D at 0x7e9dbd759d50>,
 <matplotlib.lines.Line2D at 0x7e9dbd759d80>,
 <matplotlib.lines.Line2D at 0x7e9dbd759e70>]
```



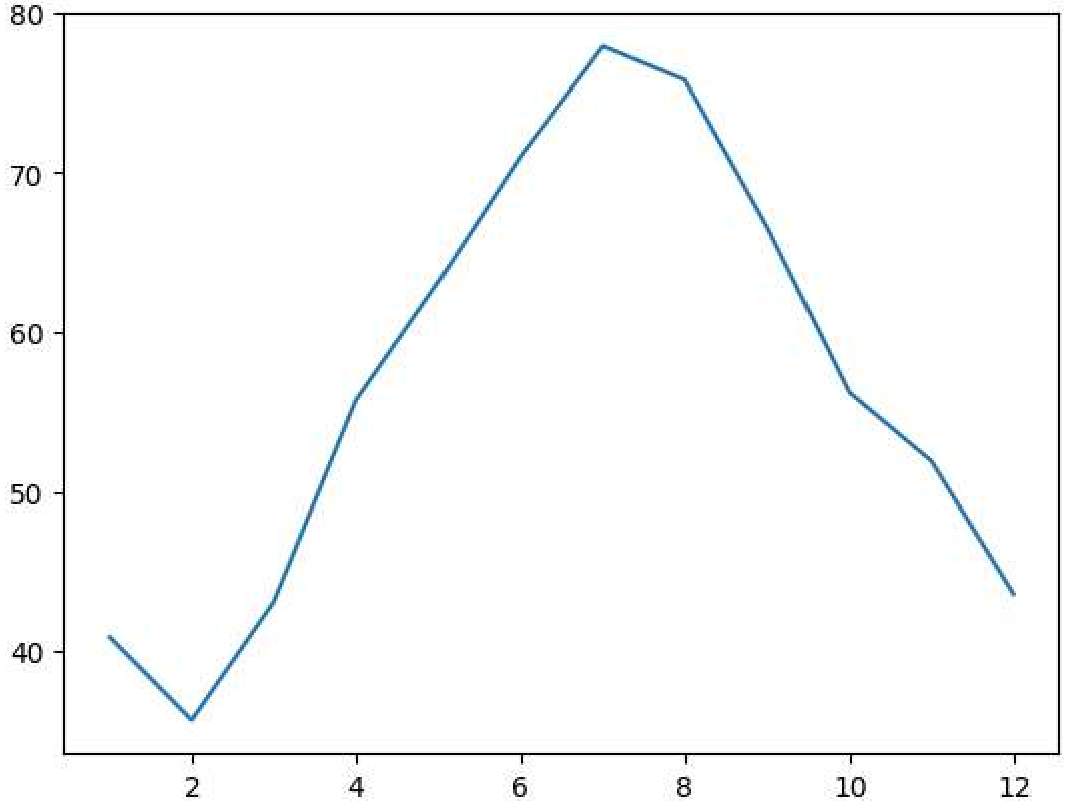
```
In [19]: plot(months,warangal_temp_2000)
```

```
Out[19]: [<matplotlib.lines.Line2D at 0x7e9dcc0ebb80>]
```



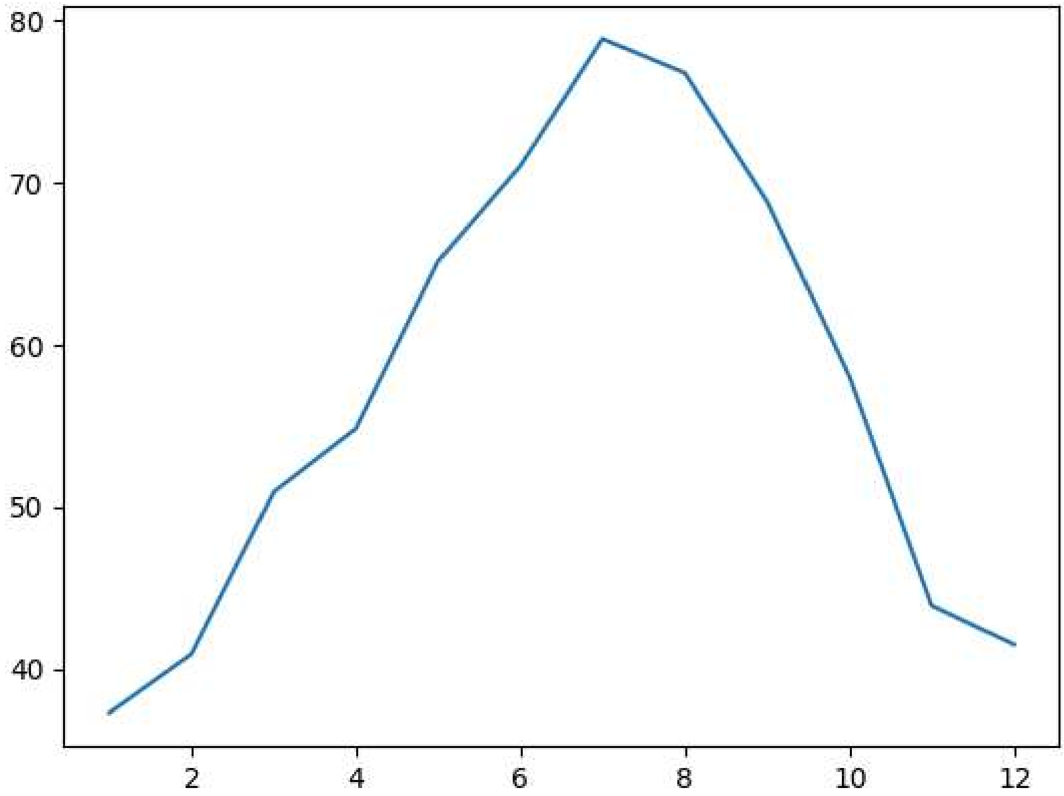
In [20]:
```
plot(months,warangal_temp_2006)
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x7e9dbd7e4190>]
```



In [21]:
```
plot(months,warangal_temp_2012)
```

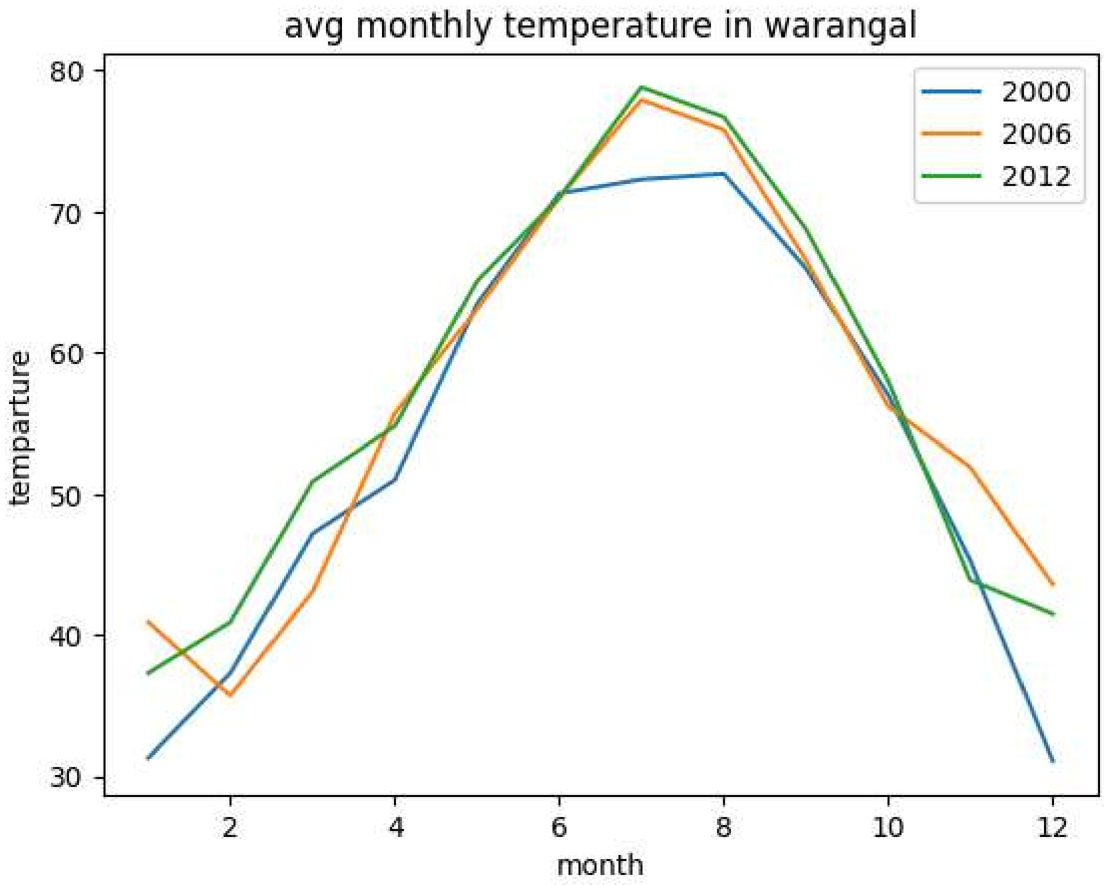
Out[21]:
```
[<matplotlib.lines.Line2D at 0x7e9dcc10e6e0>]
```



adding the titles lables

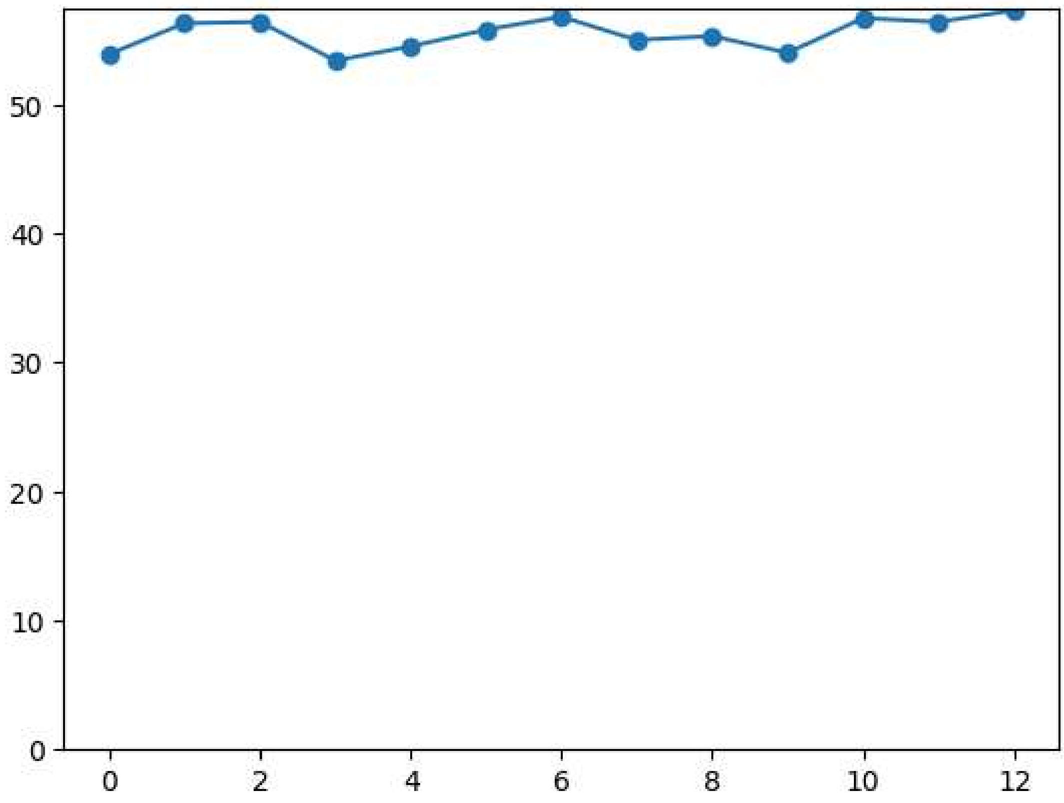
In [22]:
```
from pylab import plot, show, title, xlabel, ylabel, legend
plot(months, warangal_temp_2000, months, warangal_temp_2006, months, warangal_temp_2012)
title('avg monthly temperature in warangal')
xlabel('month')
ylabel('temparture')
legend([2000,2006,2012])
```

Out[22]:
```
<matplotlib.legend.Legend at 0x7e9dcc26c490>
```



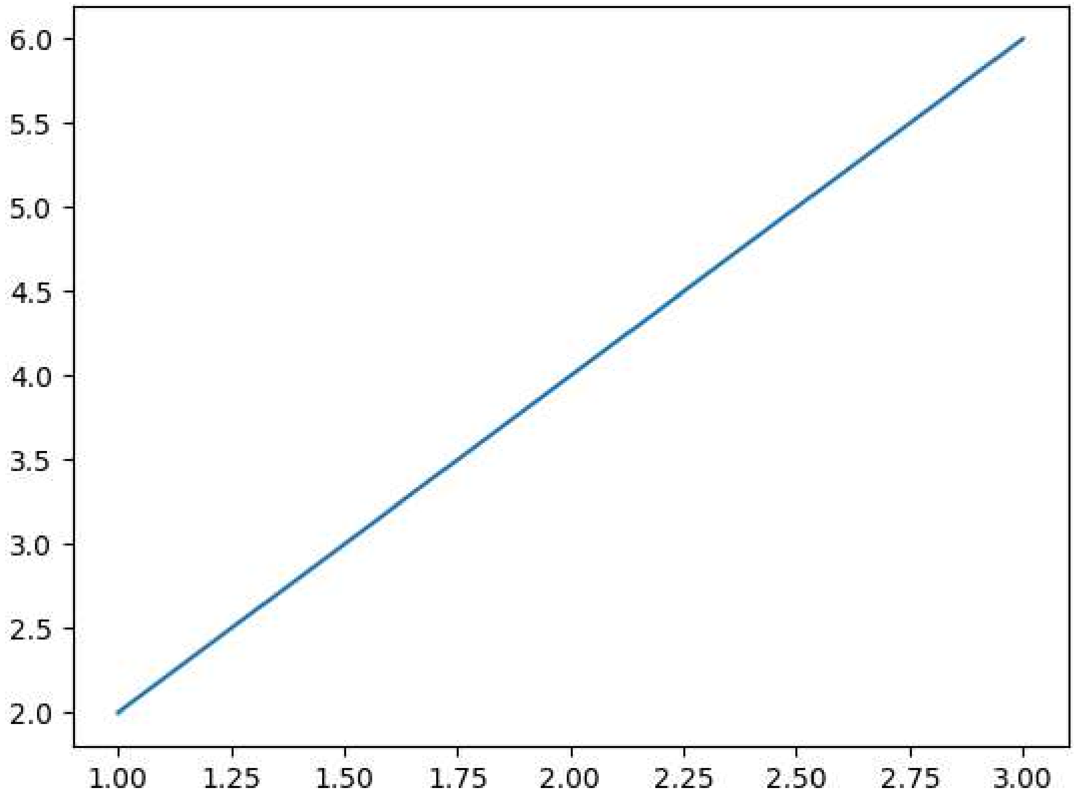
```
In [23]: nyc_temp = [53.9, 56.3, 56.4,53.4,54.5,55.8,56.8,55.0,55.3,54.0,56.7,56.4,57.3]
         plot(nyc_temp, marker = 'o')
         from pylab import axis
         axis()
         axis(ymin=0)
```

```
Out[23]: (-0.6000000000000001, 12.6, 0.0, 57.495)
```



plotting using pyplot

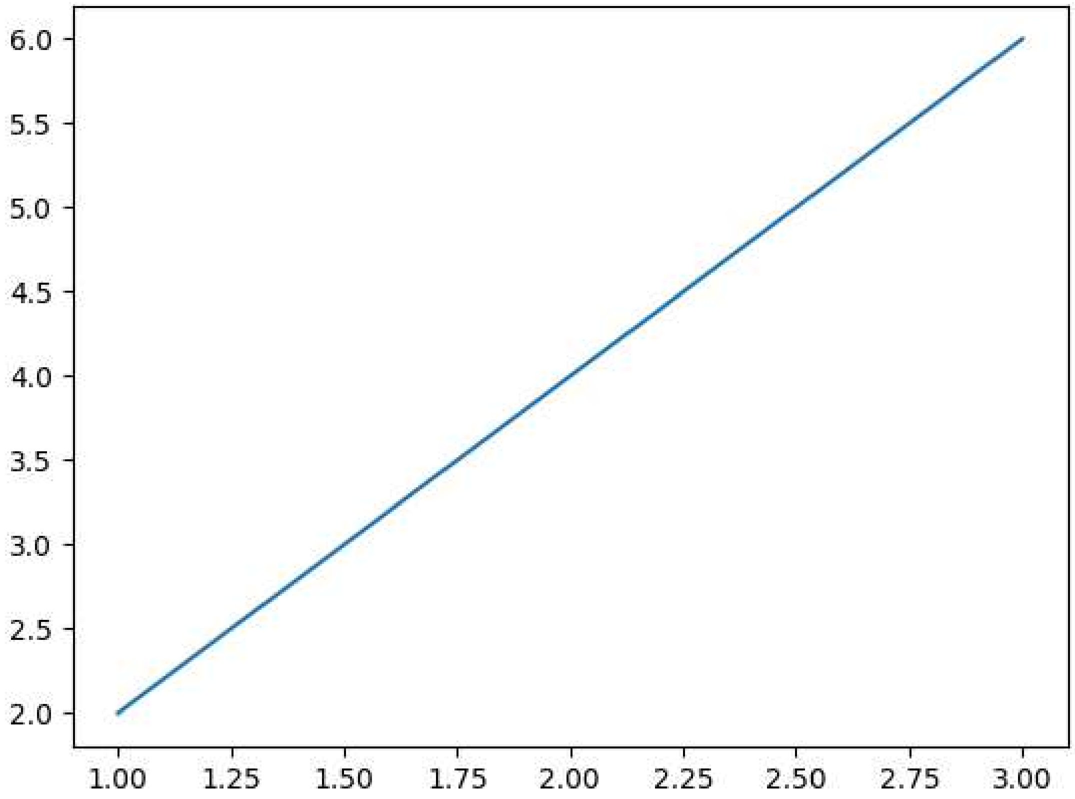
In [24]:
```
import matplotlib.pyplot
def create_graph():
  x_numbers = [1,2,3]
  y_numbers = [2,4,6]

  matplotlib.pyplot.plot(x_numbers, y_numbers)
  matplotlib.pyplot.show()

if __name__ == '__main__':
  create_graph()
```



In [25]:
```
import matplotlib.pyplot as plt
```

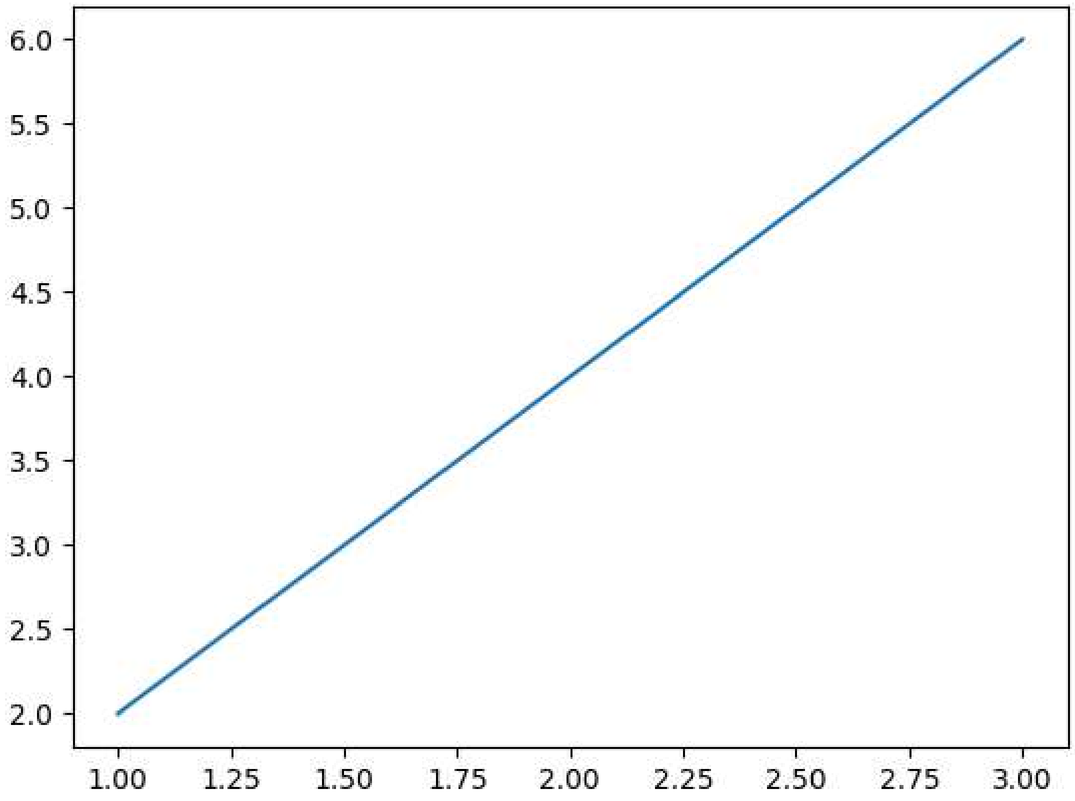
In [26]:

```
def create_graph():
  x_numbers = [1,2,3]
  y_numbers = [2,4,6]
  plt.plot(x_numbers , y_numbers)
  plt.show()

if __name__ == '__main__':
  create_graph()
```



In [27]:

```
from pylab import plot, savefig
x = [1,2,3]
y = [2,4,6]
plot(x,y)
savefig('mygraph.png')
```

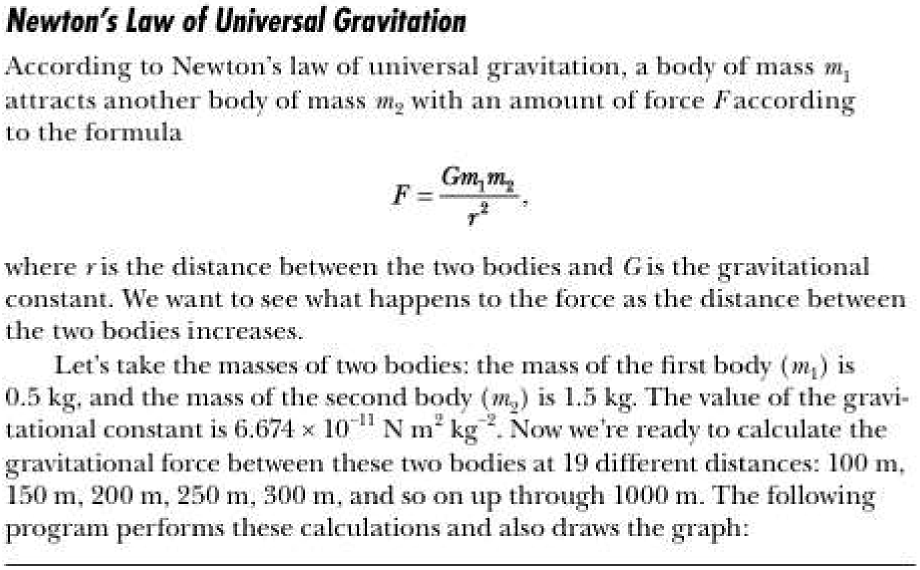


In [28]:

```
savefig('c:\mygraph.png')
```

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

plotting with teh formulas



Newton's Law of Universal Gravitation

According to Newton's law of universal gravitation, a body of mass m_1 attracts another body of mass m_2 with an amount of force F according to the formula

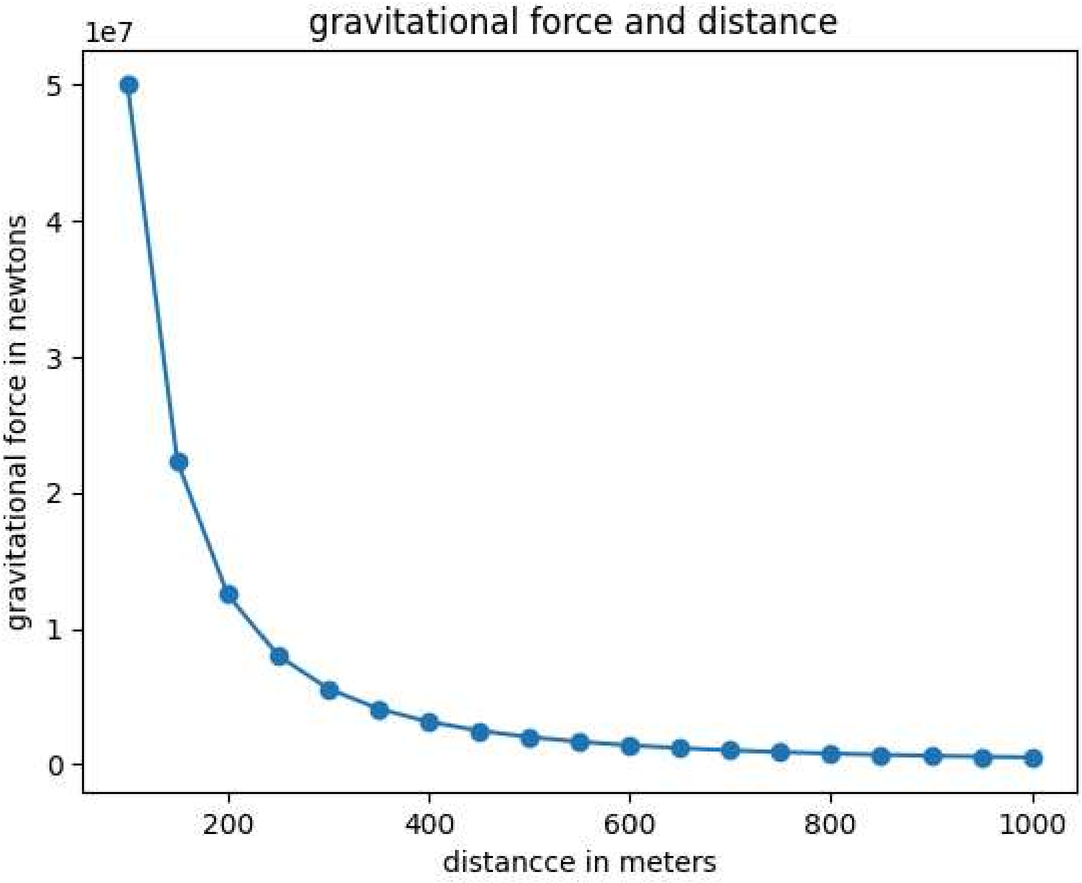
$$F = \frac{Gm_1m_2}{r^2},$$

where r is the distance between the two bodies and G is the gravitational constant. We want to see what happens to the force as the distance between the two bodies increases.

Let's take the masses of two bodies: the mass of the first body (m_1) is 0.5 kg, and the mass of the second body (m_2) is 1.5 kg. The value of the gravitational constant is 6.674×10^{-11} N m^2 kg^{-2}. Now we're ready to calculate the gravitational force between these two bodies at 19 different distances: 100 m, 150 m, 200 m, 250 m, 300 m, and so on up through 1000 m. The following program performs these calculations and also draws the graph:

In [29]:

```
# the relationship between gravitational force and distance between two bodies

import matplotlib.pyplot as plt

def draw_graph(x,y):
  plt.plot(x,y, marker = 'o')
  plt.xlabel('distancce in meters')
  plt.ylabel('gravitational force in newtons')
  plt.title('gravitational force and distance ')
  plt.show()

def generate_F_r():
  #generate values for r
  r = range(100,1001, 50 )
  #empty list to store the calculated values of F
  F = []

  #constant ,g
  G = 6.674*(10**11)
  #two masses
  m1 = 0.5
  m2 = 1.5

  #calculating force and add it to the list ,F
  for dist in r :
    force =G*(m1*m2)/(dist**2)
    F.append(force)

  #call the draw_graph func
  draw_graph(r, F)


if __name__== '__main__':

  generate_F_r()
```



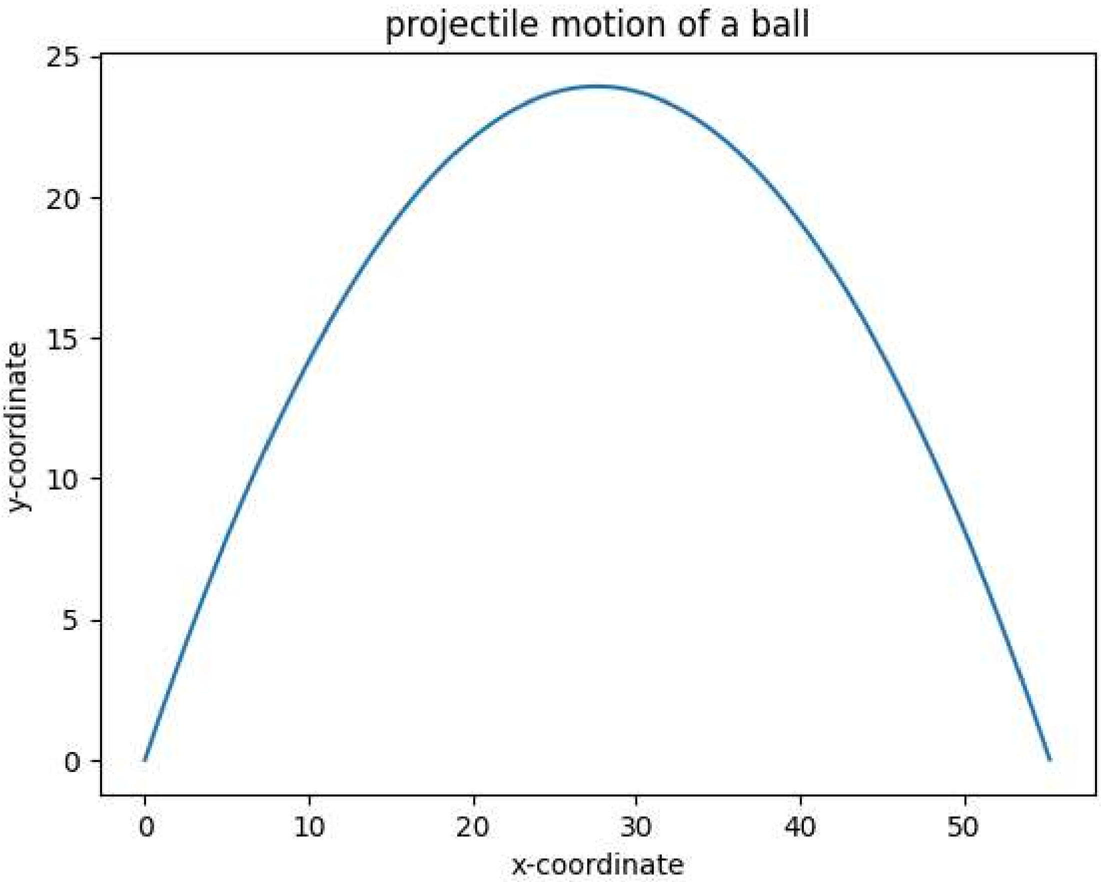
In [30]:

```
#generate equally spaced floating point numbers between two given values

def frange(start,final,increment):

  numbers = []
  while start < final:
    numbers.append(start)
    start = start + increament

  return numbers
```

In [31]:

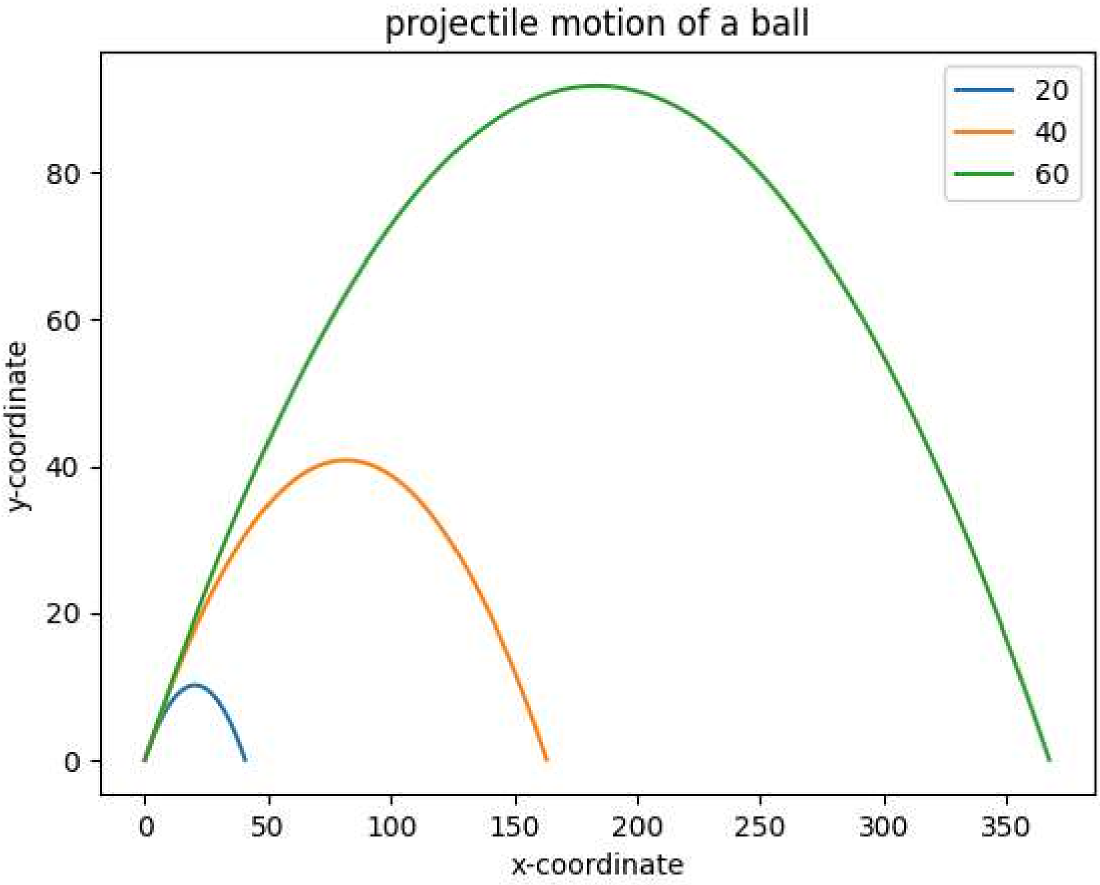
```
# drawing the trajectory of a body in projectile motion

from matplotlib import pyplot as plt
import math

def draw_graph(x,y):
  plt.plot(x,y)
  plt.xlabel('x-coordinate')
  plt.ylabel('y-coordinate')
  plt.title('projectile motion of a ball')

def frange(start,final,interval):
  numbers = []
  while start < final:
    numbers.append(start)
    start = start + interval

  return numbers


def draw_trajectory(u, theta):
  theta = math.radians(theta)
  g = 9.8

  #time of flight
  t_flight = 2*u*math.sin(theta)/g
  #find the time intervals
  intervals = frange(0, t_flight, 0.001)

  #list of x and y coordinates
  x = []
  y = []
  for t in intervals:
    x.append(u*math.cos(theta)*t)
    y.append(u*math.sin(theta)*t - 0.5*g*t*t)

  draw_graph(x,y)


if __name__ == '__main__':
  try :
    u = float(input('enter the initial velocity (m/s):'))
    theta = float(input('enter the angle of projection (degrees):'))

  except ValueError:
    print('you entered incorrect an invalid input')

  else:
    draw_trajectory(u, theta)
    plt.show()



#
```

```
enter the initial velocity (m/s):25
enter the angle of projection (degrees):60
```



In [32]:

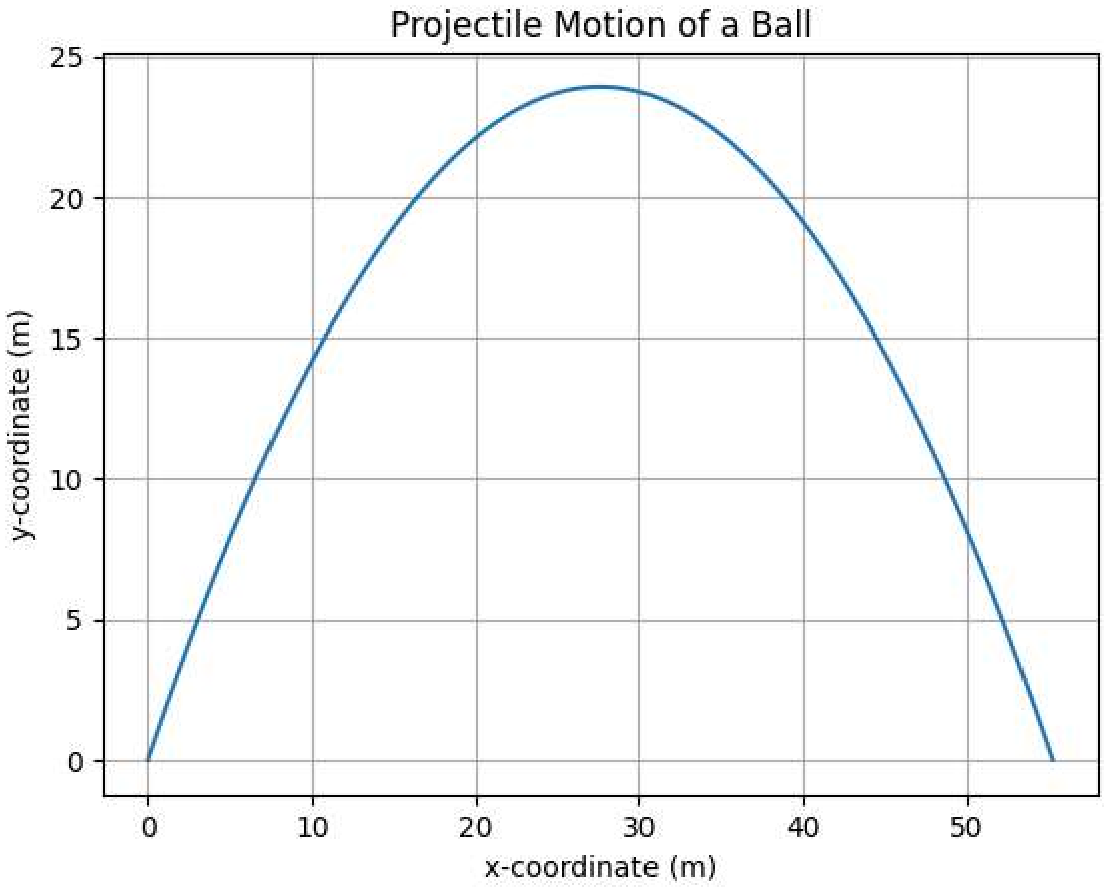
```
if __name__ == '__main__':   # comparing the trajectory differnt intial velocites

  #list of the three different intial velocities
  u_list = [20, 40 , 60 ]
  theta = 45
  for u in u_list:
    draw_trajectory(u, theta)

  #add a legend and show the graph
  plt.legend(['20', '40','60'])
  plt.show()
```



In [33]:

```
from matplotlib import pyplot as plt
import math

def draw_graph(x, y):
    plt.plot(x, y)
    plt.xlabel('x-coordinate (m)')
    plt.ylabel('y-coordinate (m)')
    plt.title('Projectile Motion of a Ball')
    plt.grid()

def frange(start, final, interval):
    numbers = []
    while start < final:
        numbers.append(start)
        start += interval
    return numbers

def draw_trajectory(u, theta):
    g = 9.8

    # Convert angle from degrees to radians
    theta_rad = math.radians(theta)

    # Time of flight
    t_flight = 2 * u * math.sin(theta_rad) / g
    # Find the time intervals
    intervals = frange(0, t_flight, 0.001)

    # List of x and y coordinates
    x = []
    y = []
    for t in intervals:
        x.append(u * math.cos(theta_rad) * t)
        y.append(u * math.sin(theta_rad) * t - 0.5 * g * t * t)

    draw_graph(x, y)

if __name__ == '__main__':
    try:
        u = float(input('Enter the initial velocity (m/s): '))
        theta = float(input('Enter the angle of projection (degrees): '))

        # Check for valid input
        if u < 0 or theta < 0:
            raise ValueError("Initial velocity and angle must be non-negativ
e.")

    except ValueError as e:
        print(f'Invalid input: {e}')

    else:
        draw_trajectory(u, theta)
        plt.show()
```

```
Enter the initial velocity (m/s): 25
Enter the angle of projection (degrees): 60
```



programming challenges

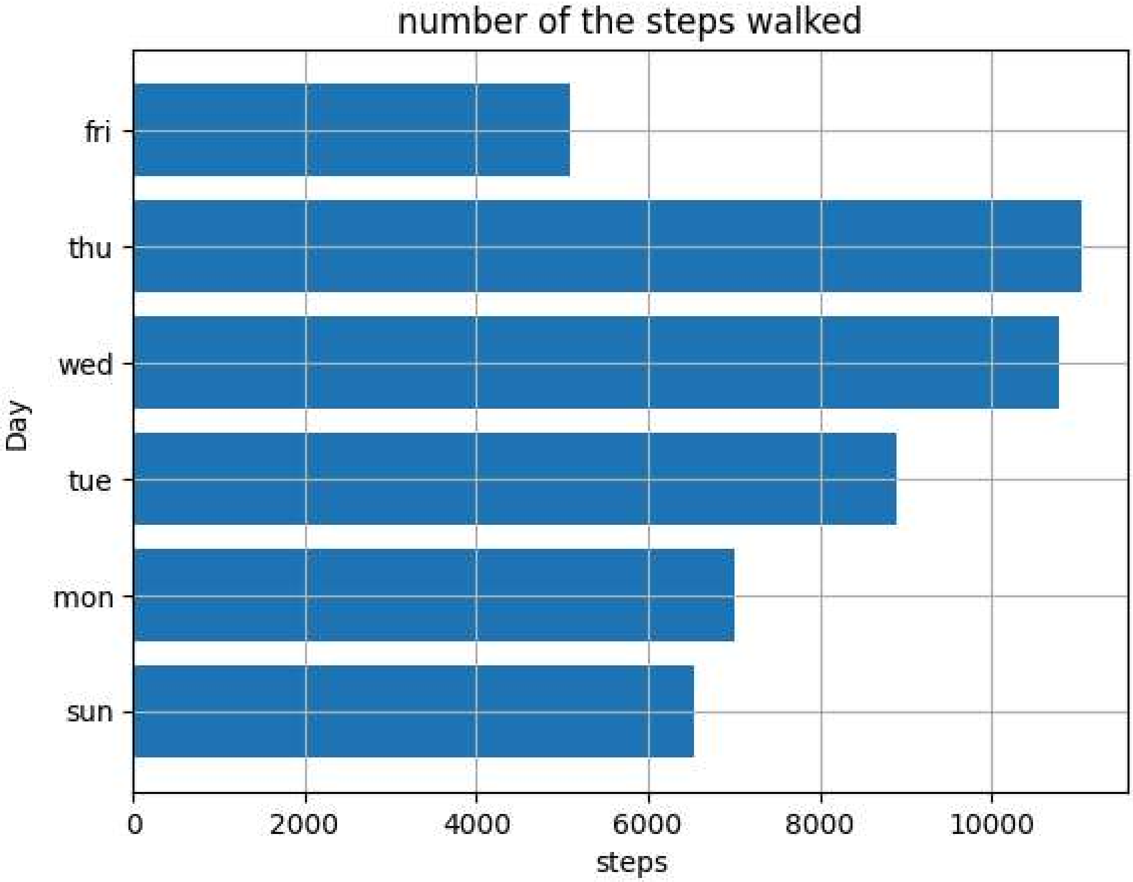
exploring the quadratic func visually

In [34]:
```
x_values = [-1, 1 ,2,3, 4,5] #quadratic function calculator
#assume the values of x
for x in x_values:
  #calculating the value of the quadratic function
  y = x**2 + 2*x + 1
  print('x={0} y = {1}'.format(x,y))
```

```
x=-1 y = 0
x=1 y = 4
x=2 y = 9
x=3 y = 16
x=4 y = 25
x=5 y = 36
```

example of drawing a horzontal bar chart

In [35]:
```
import matplotlib.pyplot as plt
def create_bar_chart(data,labels):
  #number of bars
  num_bars = len(data)
  #this list is the point on the y_ axis where each
  #bar is centered here it will be [1,2,3,4...]
  positions = range(1,num_bars+1)
  plt.barh(positions, data, align = 'center')
  #set the label of each bar
  plt.yticks(positions,labels[:num_bars])
  plt.xlabel('steps')
  plt.ylabel('Day')
  plt.title('number of the steps walked')
  #turns on the grid which may assist in visual estimation
  plt.grid()
  plt.show()


if __name__ =='__main__':
  #number of the steps i walked during the past week
  steps = [6534, 7000, 8900, 10786, 11045,5095]
  #corresponding days
  labels = ['sun',' mon', 'tue', 'wed','thu','fri', 'sat']
  create_bar_chart(steps,labels)
```



In [36]:

```python
def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        fib_series = [0, 1]
        for i in range(2, n):
            fib_series.append(fib_series[-1] + fib_series[-2])
        return fib_series

# Example usage
n = 10  # Number of terms
result = fibonacci(n)
print(result)  # Output the Fibonacci series
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

chapter ends.

In [36]: