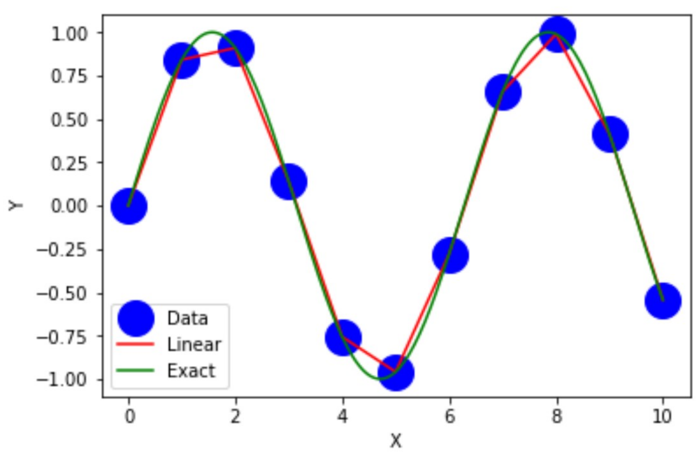
Interpolation

- Using inbuilt interpolation function
- Newton-Gregory Interpolation
- Lagrange Interpolation

In [3]:
```
import matplotlib.pyplot as pl
import numpy as np
from scipy.interpolate import interp1d

# make our tabular values
x_table = np.arange(11)
y_table = np.sin(x_table)

# linearly interpolate
x = np.linspace(0.,10.,201)

# here we create linear interpolation function
linear = interp1d(x_table,y_table,'linear')

# apply and create new array
y_linear = linear(x)

# plot results to illustrate
pl.ion()
pl.plot(x_table,y_table,'bo',markersize=20)
pl.plot(x,y_linear,'r')
pl.plot(x,np.sin(x),'g')
pl.legend(['Data','Linear','Exact'],loc='best')
pl.xlabel('X')
pl.ylabel('Y')
```

Out[3]:
```
Text(0, 0.5, 'Y')
```



In [4]:

```
# Program to interpolate using
# newton forward interpolation

# calculating u mentioned in the formula
def u_cal(u, n):

    temp = u;
    for i in range(1, n):
        temp = temp * (u - i);
    return temp;

# calculating factorial of given number n
def fact(n):
    f = 1;
    for i in range(2, n + 1):
        f *= i;
    return f;

# Driver Code

# Number of values given
n = 4;
x = [ 45, 50, 55, 60 ];

# y[][] is used for difference table
# with y[][0] used for input
y = [[0 for i in range(n)]
        for j in range(n)];
y[0][0] = 0.7071;
y[1][0] = 0.7660;
y[2][0] = 0.8192;
y[3][0] = 0.8660;

# Calculating the forward difference
# table
for i in range(1, n):
    for j in range(n - i):
        y[j][i] = y[j + 1][i - 1] - y[j][i - 1];

# Displaying the forward difference table
for i in range(n):
    print(x[i], end = "\t");
    for j in range(n - i):
        print(y[i][j], end = "\t");
    print("");

# Value to interpolate at
value = 52;

# initializing u and sum
sum = y[0][0];
u = (value - x[0]) / (x[1] - x[0]);
for i in range(1,n):
    sum = sum + (u_cal(u, i) * y[0][i]) / fact(i);

print("\nValue at", value,
    "is", round(sum, 6));

# This code is contributed by mits
```

```
45      0.7071  0.05890000000000006     -0.005700000000000038   -0.000700000000000
0339
50      0.766   0.05320000000000025     -0.006400000000000072
55      0.8192  0.04679999999999995
```

In [18]:
```
# Python3 program for implementing
# Newton divided difference formula

# Function to find the product term
def proterm(i, value, x):
    pro = 1;
    for j in range(i):
        pro = pro * (value - x[j]);
    return pro;

# Function for calculating
# divided difference table
def dividedDiffTable(x, y, n):

    for i in range(1, n):
        for j in range(n - i):
            y[j][i] = ((y[j][i - 1] - y[j + 1][i - 1]) /
                                    (x[j] - x[i + j]));
    return y;

# Function for applying Newton's
# divided difference formula
def applyFormula(value, x, y, n):

    sum = y[0][0];

    for i in range(1, n):
        sum = sum + (proterm(i, value, x) * y[0][i]);

    return sum;

# Function for displaying divided
# difference table
def printDiffTable(y, n):

    for i in range(n):
        for j in range(n - i):
            print(round(y[i][j], 4), "\t",
                                end = " ");

        print("");

# Driver Code

# number of inputs given
n = 4;
y = [[0 for i in range(10)]
        for j in range(10)];
x = [ 5, 6, 9, 11 ];

# y[][] is used for divided difference
# table where y[][0] is used for input
y[0][0] = 12;
y[1][0] = 13;
y[2][0] = 14;
y[3][0] = 16;

# calculating divided difference table
y=dividedDiffTable(x, y, n);

# displaying divided difference table
printDiffTable(y, n);

# value to be interpolated
```

In [19]:
```
def  lagrange(x,i,xm):
    """
    Evaluates  the i-th  Lagrange  polynomial
    at xbased on grid  data xm
    """
    n=len(xm)-1
    y=1
    for j in  range(n+1):
        if i!=j:
            y*=(x-xm[j])/(xm[i]-xm[j])
    return y


def  interpolation(x,xm ,ym):
    n=len(xm)-1
    lagrpoly=array([lagrange(x,i,xm) for i in range(n+1)])
    y = dot(ym ,lagrpoly)
    return y


# Example
xm = array([1,2,3,4,5,6])
ym = array([-3,0,-1,2,1,4])
xplot = linspace(0.9,6.1,100)
yplot = interpolation(xplot ,xm,ym)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-19-ce2ef674f9ef> in <module>
     20
     21 # Example
---> 22 xm = array([1,2,3,4,5,6])
     23 ym = array([-3,0,-1,2,1,4])
     24 xplot = linspace(0.9,6.1,100)

NameError: name 'array' is not defined
```

In [ ]: