```
import matplotlib as plot
import numpy as np
import sympy as sym       #Lib for Symbolic Math
from matplotlib import pyplot

def objective(x):
  return (x+3)**2

def derivative(x):
  return 2*(x + 3)

def gradient_descent(alpha, start, max_iter):
  x_list = list()
  x= start;
  x_list.append(x)
  for i in range(max_iter):
    gradient = derivative(x);
    x = x - (alpha*gradient);
    x_list.append(x);
  return x_list

x = sym.symbols('x')
expr = (x+3)**2.0;
grad = sym.Derivative(expr,x)
print("{}".format(grad.doit()) )
grad.doit().subs(x,2)
```

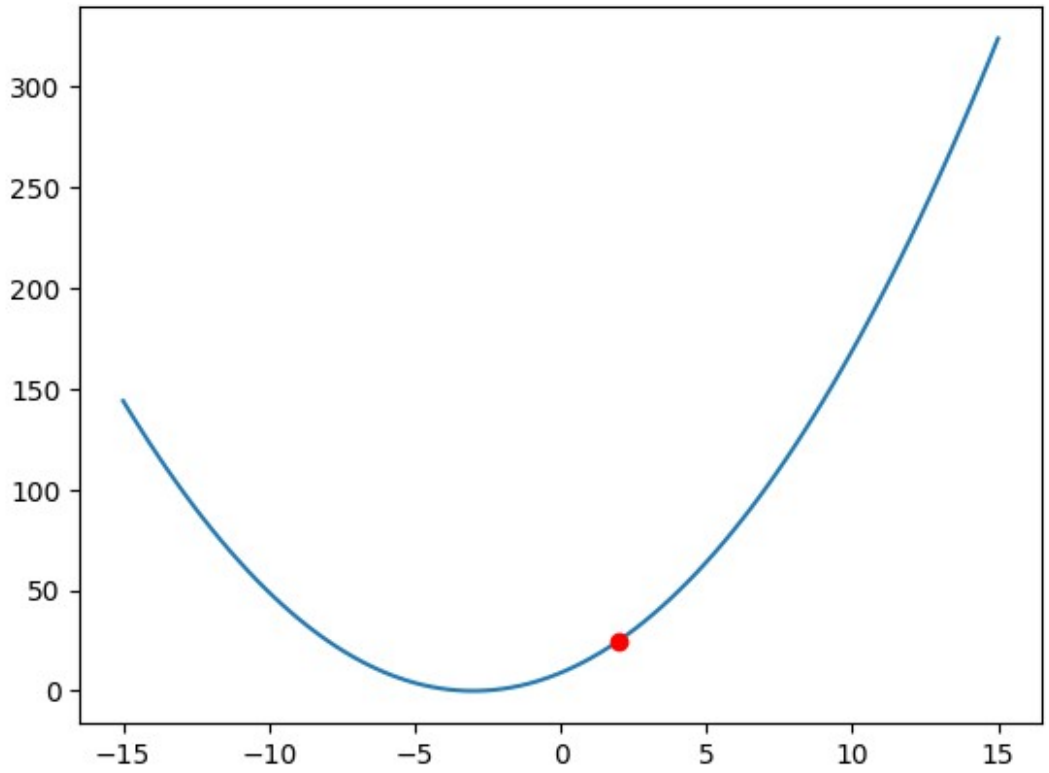
```
2.0*(x + 3)**1.0
```

```
10.0000000000000
```

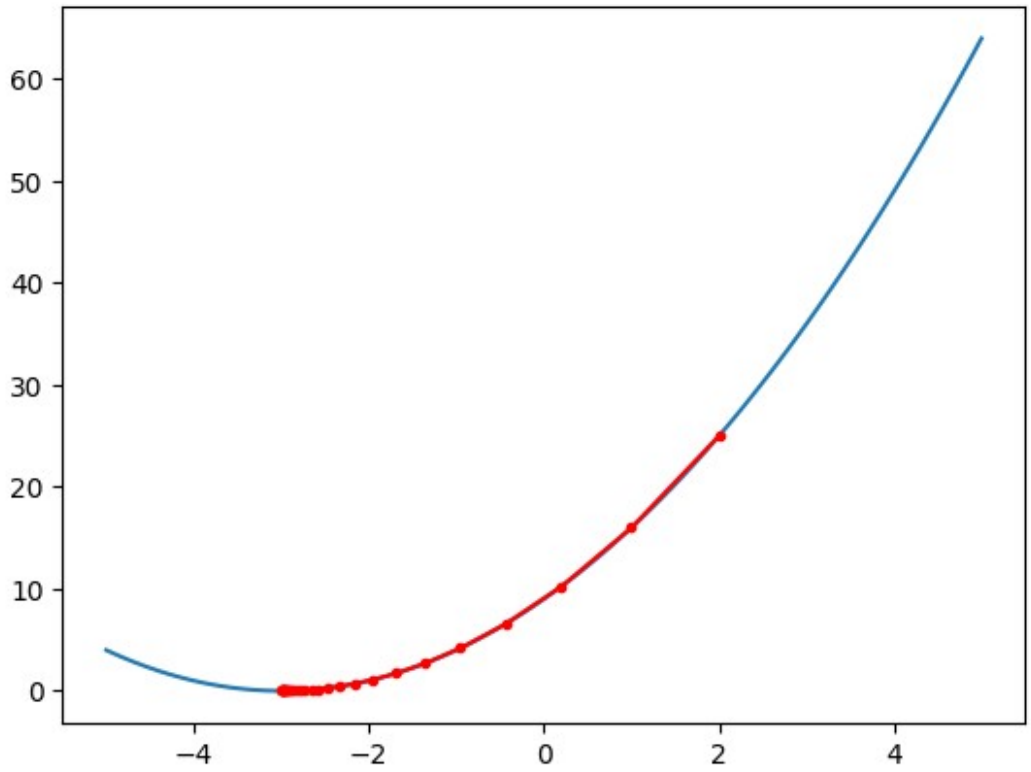
```
def gradient_descent1(expr,alpha, start, max_iter):
  x_list = list()
  x = sym.symbols('x')
  grad = sym.Derivative(expr,x).doit()
  x_val= start;
  x_list.append(x_val)
  for i in range(max_iter):
    gradient = grad.subs(x,x_val);
    x_val = x_val - (alpha*gradient);
    x_list.append(x_val);
  return x_list

alpha = 0.1       #Step_size
start = 2         #Starting point
max_iter = 30     #Limit on iterations
x = sym.symbols('x')
expr = (x+3)**2;   #target function
```

```
x_cordinate = np.linspace(-15,15,100)
pyplot.plot(x_cordinate,objective(x_cordinate))
pyplot.plot(2,objective(2),'ro')
```

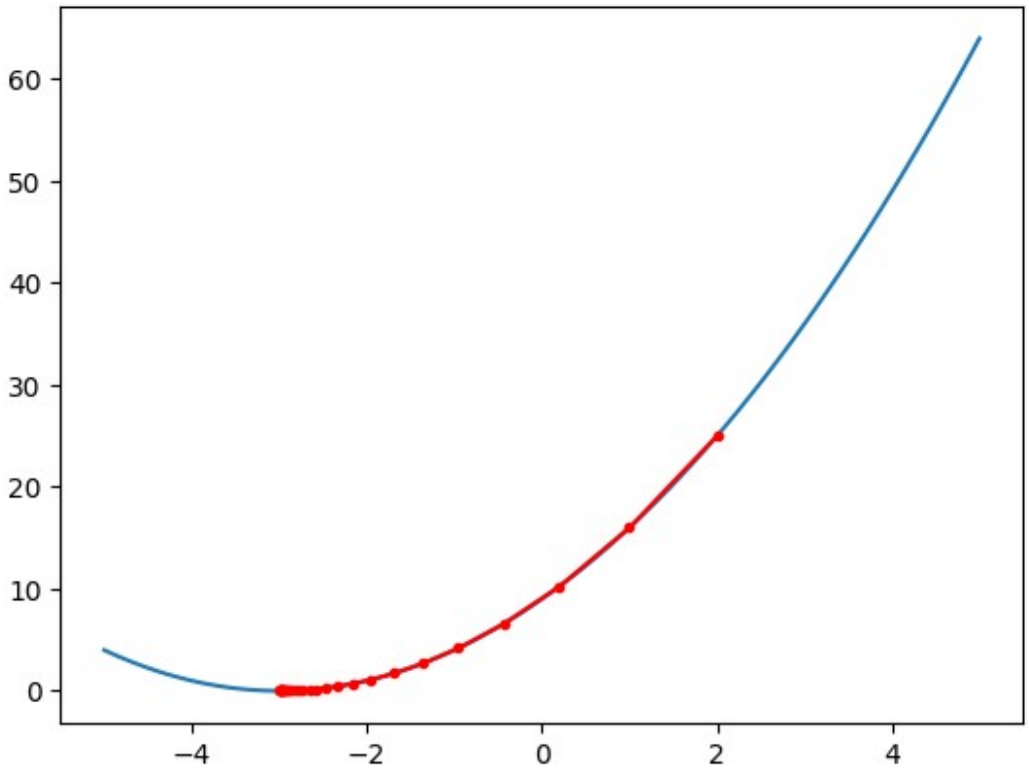
```
[<matplotlib.lines.Line2D at 0x25f42a4ed10>]
```




```
X = gradient_descent(alpha,start,max_iter)

x_cordinate = np.linspace(-5,5,100)
pyplot.plot(x_cordinate,objective(x_cordinate))

X_arr = np.array(X)
pyplot.plot(X_arr, objective(X_arr), '.-', color='red')
pyplot.show()
```




```
X= gradient_descent1(expr,alpha,start,max_iter)
X_arr = np.array(X)

x_cordinate = np.linspace(-5,5,100)
pyplot.plot(x_cordinate,objective(x_cordinate))

X_arr = np.array(X)
pyplot.plot(X_arr, objective(X_arr), '.-', color='red')
pyplot.show()
```


60
50
40
30
20
10
0
−4
−2
0
2
4