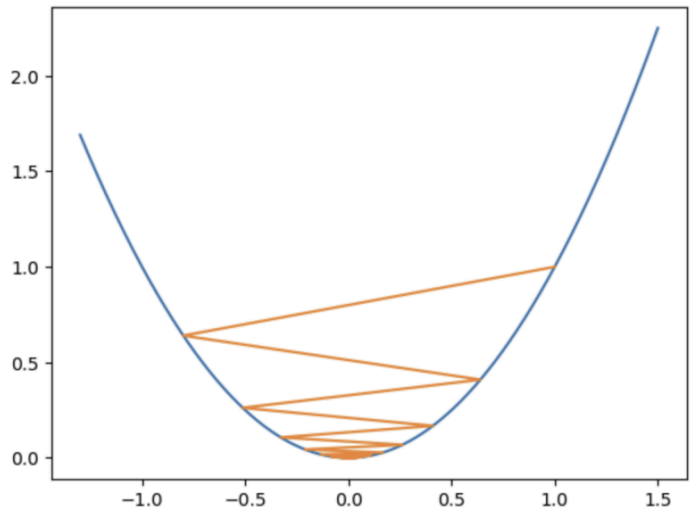
```
import matplotlib.pyplot as plt
import numpy as np

def gradient_descent(f, learning_rate, initial_point):

    def deriv(f, base_point): #estimate the derivative of function f at base_point
        return (f(base_point+10**(-9))-f(base_point))/(10**(-9))

    x_coords=[initial_point] #This list is where you will store the x_n's
    y_coords=[f(initial_point)] #This list is where you will store the y_n's

    i=0 #initialize a counter so there is an index in our while loop.

    while abs(deriv(f, x_coords[i]))>0.0000000001 or i<100000: # while loop stops if the derivative is very small or more than 100000 times
        x_coords.append(x_coords[i]-learning_rate*deriv(f,x_coords[i])) #here the next x-coordinate is added given by formula x_{i+1} & depends on slope at x_i
        y_coords.append(f(x_coords[i+1])) #the next y-coordinate is just our function applied to the x-coordinate we just created.
        i=i+1 #increment the counter for the next iteration

    plot_range=np.linspace(min(x_coords)-0.5, max(x_coords)+0.5,10000)
    function_range=[f(i) for i in plot_range]
    plt.plot(plot_range, function_range) # plots function
    plt.plot(x_coords, y_coords) #plots points

    return round(x_coords[-1],3), round(y_coords[-1],3) #returns your last x_n and y_n rounded to three decimal places.
```

```
def func1(x):
    return x**2
```

```
gradient_descent(func1,0.9,1)
```

```
(-0.0, 0.0)
```



```
def func2(x):
    return -3*(x**3)+10*(x**2)-(5*x)-3

gradient_descent(func2, 0.1,1)
```

```
(0.287, -3.682)
```