Practical No 3

Title: Implement Gradient Descent Algorithm to find the local minima of a function.

For example, find the local minima of the function $y=(x+3)^2$ starting from the point $x=2$.

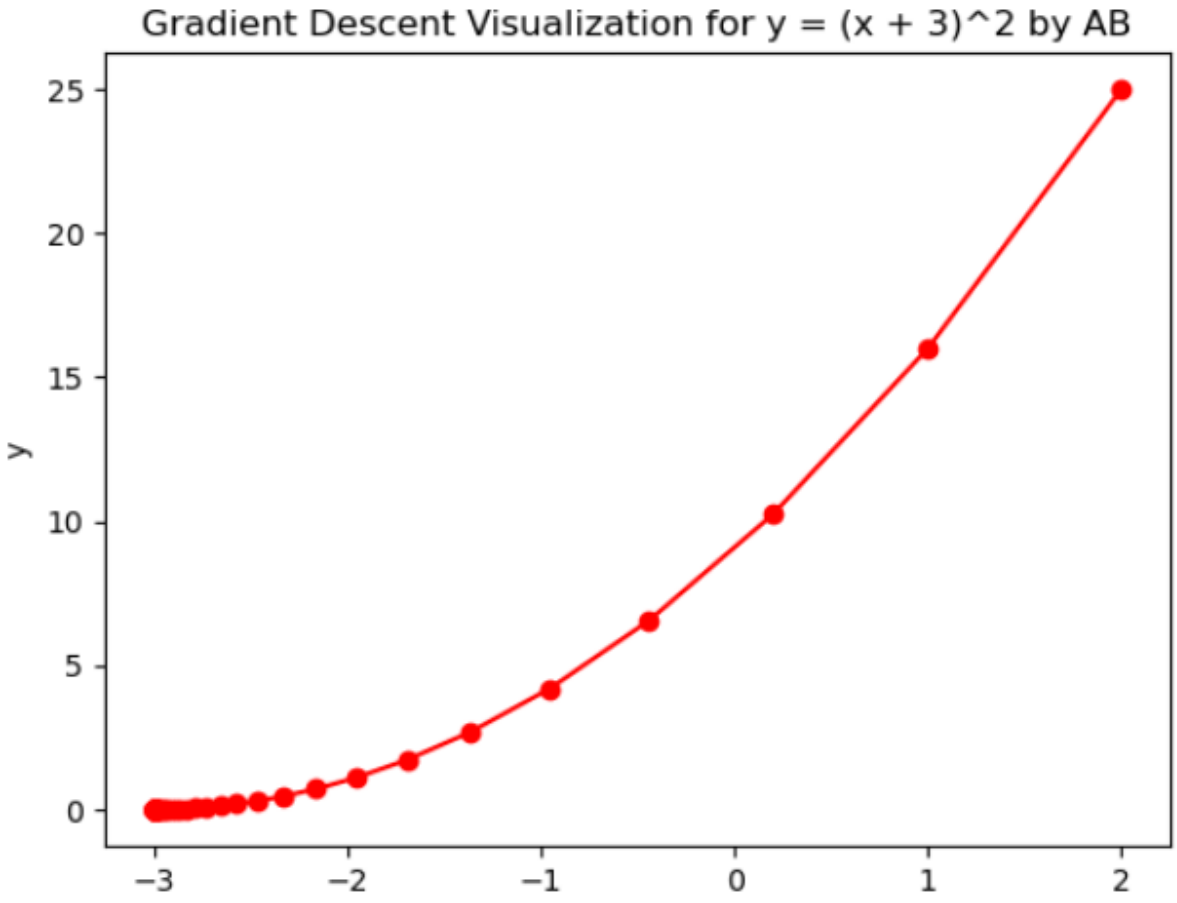
[1]:
```
import matplotlib.pyplot as plt
```

[2]:
```
def cost_function(x):
    # ithe given function yenar
    return (x + 3) ** 2
```

[3]:
```
def gradient(x):
    # ithe derivate of given function yenar
    return 2 * (x + 3)
```

[4]:
```
learning_rate = 0.1
initial_x = 2.0
num_iterations = 100
```

```
Iteration 1: x = 1.0, Cost = 16.0
Iteration 2: x = 0.19999999999999996, Cost = 10.240000000000002
Iteration 3: x = -0.44000000000000017, Cost = 6.553599999999998
Iteration 4: x = -0.9520000000000001, Cost = 4.194304
Iteration 5: x = -1.3616000000000001, Cost = 2.6843545599999996
Iteration 6: x = -1.6892800000000001, Cost = 1.7179869183999996
Iteration 7: x = -1.951424, Cost = 1.099511627776
Iteration 8: x = -2.1611392, Cost = 0.7036874417766399
Iteration 9: x = -2.32891136, Cost = 0.4503599627370493
Iteration 10: x = -2.463129088, Cost = 0.28823037615171165
Iteration 11: x = -2.5705032704, Cost = 0.1844674407370954
Iteration 12: x = -2.6564026163200003, Cost = 0.11805916207174093
Iteration 13: x = -2.725122093056, Cost = 0.07555786372591429
Iteration 14: x = -2.7800976744448, Cost = 0.04835703278458515
Iteration 15: x = -2.82407813955584, Cost = 0.030948500982134555
Iteration 16: x = -2.8592625116446717, Cost = 0.019807040628566166
Iteration 17: x = -2.8874100093157375, Cost = 0.012676506002282305
Iteration 18: x = -2.90992800745259, Cost = 0.008112963841460692
Iteration 19: x = -2.927942405962072, Cost = 0.005192296858534868
Iteration 20: x = -2.9423539247696575, Cost = 0.0033230699894623056
Iteration 21: x = -2.953883139815726, Cost = 0.002126764793255884
Iteration 22: x = -2.9631065118525806, Cost = 0.0013611294676837786
Iteration 23: x = -2.9704852094820646, Cost = 0.0008711228593176078
```


Gradient Descent Visualization for y = (x + 3)^2 by AB
y
25
20
15
10
5
0
−3
−2
−1
0
1
2