Assignment 5 Analysis

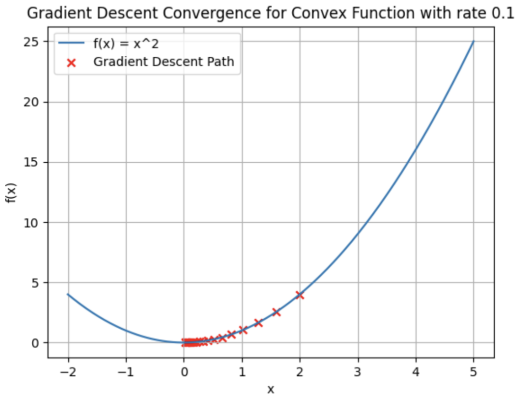
```
import numpy as np
import matplotlib.pyplot as plt

"""
    Let f : R → R be differentiable.  To minimize f, consider the gradient descent
    method x_(n+1) = x_n − η*f'(x_n), where x_1 ∈ R and η > 0 (learning rate).
    (a) Take a convex f and show that for small η the method converges to the
        minimum off.
"""


def f(x):
    return x ** 2


def f_derivative(x):
    return 2 * x


# non-convex function f(x) = x^3
def f_non_convex(x):
    return x ** 3


# Define the gradient of the function
def f_non_convex_derivative(x):
    return x ** 2 * 3


def gradient_descent(rate, initial_x, iter):
    x = initial_x
    trajectory = [x]

    for i in range(iter):
        # x_(n+1) = x_n − η*f'(x_n)
        x = x - rate * f_derivative(x)
        trajectory.append(x)

    return x, trajectory


def gradient_descent_non_convex(rate, initial_x, iter):
    x = initial_x
    trajectory = [x]

    for i in range(iter):
        # x_(n+1) = x_n − η*f'(x_n)
        x = x - rate * f_non_convex_derivative(x)
        trajectory.append(x)

    return x, trajectory


def graph_funct_non_convex(minimizer, trajectory, rate):
    x_values = np.linspace(0, 4, 400)
    y_values = f_non_convex(x_values)
```

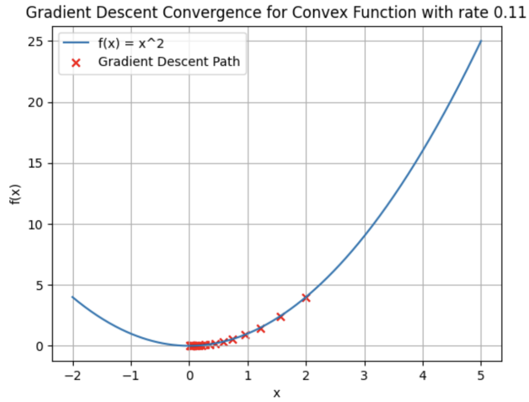
```python
    plt.plot(x_values, y_values, label='f(x) = x^3')
    trajectory_x = np.array(trajectory)
    trajectory_y = f_non_convex(trajectory_x)
    plt.scatter(trajectory_x, trajectory_y, color='red', label='Gradient Descent Path', marker='x')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.title('Gradient Descent Getting Stuck in Local Minimum for Non-Convex Function')
    plt.grid(True)
    plt.show()


def graph_function(minimizer, trajectory, rate):
    values_x = np.linspace(-2, 5, 400)
    values_y = f(values_x)
    plt.plot(values_x, values_y, label='f(x) = x^2')

    trajectory_x = np.array(trajectory)
    trajectory_y = f(trajectory_x)
    plt.scatter(trajectory_x, trajectory_y, color='red', label='Gradient Descent Path', marker='x')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.title(f'Gradient Descent Convergence for Convex Function with rate {rate}')
    plt.grid(True)
    plt.show()

    print(f"Minimum found by Gradient Descent using the rate {rate}: {minimizer}")


"""
    (b) Show that by increasing η the method can converge faster (in fewer steps).
"""

"""
    (c) Show that taking η too large might lead to the divergence of the method.
"""

"""
    (d) Take a non-convex f and show that the method can get stuck in a local minimum.
"""


if __name__ == "__main__":
    minimizer, trajectory = gradient_descent(0.1, 2.0, 20)
    graph_function(minimizer, trajectory, 0.1)

    minimizer, trajectory = gradient_descent(0.11, 2.0, 20)
    graph_function(minimizer, trajectory, 0.11)

    minimizer, trajectory = gradient_descent(1.0, 2.0, 20)
    graph_function(minimizer, trajectory, 1.0)

    minimizer, trajectory = gradient_descent_non_convex(0.1, 2.0, 20)
    graph_funct_non_convex(minimizer, trajectory, 0.1)
```

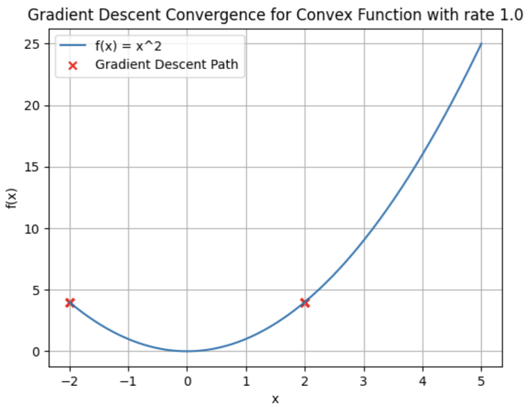
An easy convex function is $f(x) = x^2$, for which the global minimum is 0

The gradient is just a fancy word for derivative, or the rate of change of a function => $f'(x) = 2x$

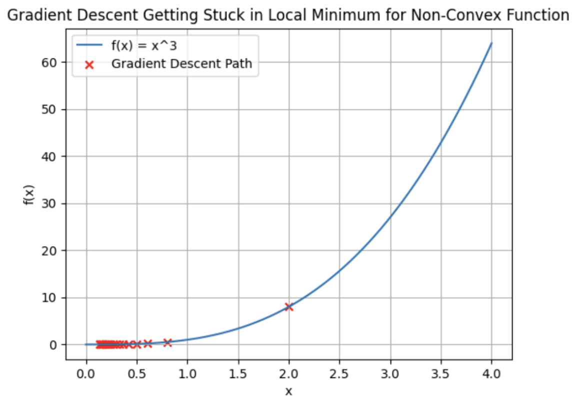
```
Minimum found by Gradient Descent using the rate 0.1: 0.023058430092136940
Minimum found by Gradient Descent using the rate 0.11: 0.013897031741724304
Minimum found by Gradient Descent using the rate 1.0: 2.0
```







2 and -2 are the values found



the function doesn't have a minimum → goes to -∞, but the **gradient decent** is stuck to the **local minimum**