Convergence in Gradient Descent:

In the context of gradient descent, convergence refers to the algorithm successfully finding a point that is close enough to the minimum of the function it is trying to optimize.

The "closeness" is usually determined by how small the gradient (rate of change) of the function is at the current point, indicating that further movements will not significantly decrease the function value.

When the condition niterations < maxiterations is true, it implies the following:

Early Stopping: The algorithm stopped iterating before it reached the maximum allowed number of iterations.

Likely Convergence: This early stopping usually happens because the algorithm has met its convergence criteria (like the gradient becoming sufficiently small), indicating that it has likely found a point near the function's minimum.

In optimization, and specifically in gradient descent, convergence refers to the process of approaching the minimum (or maximum, in case of maximization problems) of a function. For gradient descent, which is a method to minimize a function, convergence means that the algorithm is successfully getting closer to the point where the function has its lowest value.

How is Convergence Achieved?

In gradient descent, convergence is typically achieved by:

Starting at an Initial Point: You begin at a chosen point in the function's domain.

Iteratively Updating the Point: In each iteration, you move this point in the direction opposite to the gradient (the direction of steepest ascent) at that point. This is because the gradient points towards the direction of the greatest increase, and you want to minimize the function.

Using a Step Size (\lambda): The amount by which you move the point in each iteration is determined by the step size or learning rate (\lambda). If \lambda is too large, you might overshoot the minimum; if it's too small, the convergence might be very slow or stall before reaching the minimum.

How is Convergence Checked in Your Code?

In your code, convergence is determined based on two factors:

The Number of Iterations (niterations): This is the count of how many times the loop in the gradient\_descent function has run.

Maximum Allowed Iterations (maxiterations): This is a threshold you set to prevent the algorithm from running indefinitely, especially if it's not converging.

The logic you use is:

• If niterations < maxiterations: It means the algorithm stopped iterating before hitting the maximum iterations limit. This suggests that the algorithm likely found a point close to the minimum (i.e., it converged).

• If niterations == maxiterations: It means the algorithm ran for the maximum allowed iterations without stopping early. This could suggest that it did not find a point close enough to the minimum (i.e., it did not converge).

Convergence in the Context of Your Code

• Testing Different \lambda Values: By running the gradient descent with various \lambda values, you are essentially trying to find out which step sizes allow the algorithm to converge.

• Storing Results: Whether the algorithm converges for a particular \lambda value is stored in convergence\_results. This helps you identify which step sizes are more effective for the given optimization problem.

In summary, convergence in your code is about whether the gradient descent algorithm can find the minimum of the function within a reasonable number of steps, and this is influenced by the choice of the step size (\lambda). Your approach tests various \lambda values to see which ones lead to successful convergence.