Because a robotic arm's goal is to move its end effector to a designed spot in space, I must first determine its starting point in order to map the end effector's route to target

process of calculating the joint angles that achieve a desired position of the end effector. Starting with the current position of the end effector is essential for solving inverse kinematics problems.

it's important to know where the end effector is to avoid collisions with objects in the environment or the robot itself.

Calculating the end effector position is crucial for accurate visualization.

start of the code or a task sequence, verifying the initial position of the end effector can serve as a consistency check to ensure that the robot is starting from the expected configuration.

If the task involves optimizing the path of the end effector, knowing the starting position is necessary to compute the optimal trajectory or to use optimization algorithms like gradient descent.

In your specific case, calculating the end effector's position at the beginning of the code serves as a basis for further computations, particularly for optimization problems where you need to minimize the distance between the end effector's current position and the target position. It establishes the initial state from which improvements are made through techniques like gradient descent.

The function **end\_effector\_position\_deg** calculates the position of the end effector of a two-jointed arm in a 2D plane. Each joint has an arm of a certain length, and each arm can rotate around its joint. The position of the end effector (the tip of the arm assembly) is determined by the angles of these joints.

'''With each iteration of gradient descent;

xk is updated in each iteration to get closer to the minimum

based on gradient of the function at the current point and step size,

thats why I initialize xk so that it allows this updates''

This initializes a counter, niterations, to keep track of the number of iterations (steps) taken in the gradient descent process. It starts at 0 and will be incremented with each iteration. This is important for terminating the algorithm either when it converges (based on the tolerance) or when it reaches the maximum number of iterations specified by maxiterations.

!!!!!!

The Role of the Gradient in Optimization:

The gradient, calculated by g(xk), is a vector that points in the direction of the steepest increase of the function f at the point xk.

In mathematical terms, if f is a function of two variables, f(x, y), then the gradient of f is a vector of partial derivatives [∂f/∂x, ∂f/∂y]. These partial derivatives give the rate of change of f with respect to x and y, respectively.

In gradient descent, this gradient vector is used to guide the search for the minimum of the function: by moving in the direction opposite to the gradient, you move towards the point where f decreases most rapidly.

!!!!!!!!

In a two-jointed robotic arm, the position of the end effector is determined by the positions of both joints. However, the computation of the position is hierarchical, meaning that you have to know where the first joint (and therefore the end of the first arm segment) is before you can determine where the second joint leads the end effector. Here's why the computation is done this way:

1. \*\*Hierarchy of Movement\*\*:

- The first joint is attached to the base, which is a fixed point. Its position solely depends on its own angle, theta1.

- The second joint, however, is not attached to a fixed point but rather to the end of the first arm segment. Therefore, its position depends on both its angle, theta2, and the position of the first joint.

2. \*\*Sequential Calculation\*\*:

- The calculation must be sequential. First, you calculate where the first joint ends because it is the reference point for the second arm segment.

- Once you know where the first arm segment ends (which is where the second starts), you can calculate the position of the end of the second arm segment, which is the end effector.

3. \*\*Position of First Joint (x1, y1)\*\*:

- The first joint rotates around the base of the arm. Thus, its position in Cartesian coordinates can be directly calculated from the base using the length of the first arm segment, l1, and the angle theta1.

- This is a straightforward polar to Cartesian coordinate transformation.

4. \*\*Position of End Effector (x2, y2)\*\*:

- The end effector's position is influenced by two rotations - the first and second joints.

- To get its position, you need to account for the total effect of both joints. This is why you need a function like `end\_effector\_position\_deg`, which considers the cumulative effect of both theta1 and theta2 to find the final (x2, y2) coordinates.

5. \*\*Why Not Calculate the Second Joint Directly\*\*:

- While you could calculate the second joint's position as an intermediate step, what you're usually interested in for a robotic arm is the position of the end effector since that's the part of the arm that interacts with objects (e.g., for grasping or manipulation tasks).

- The end effector's position is the critical piece of information, and calculating the intermediate position of the second joint is not necessary unless you specifically need it for some reason.

In summary, the first joint's position is calculated directly from the base because it's a simple single rotation. The end effector's position is the ultimate goal and is calculated from the cumulative effect of both joints. The second joint's position could be calculated but is typically not needed unless the problem specifically requires it.