Brief Explanation of the Code

Libraries Used:

- `sympy` (imported as `sp`): A symbolic mathematics library used for differentiation, simplification, and symbolic computation.

Function: verify_kkt(f, g, x_star, lambda_star)

This function checks the **Karush-Kuhn-Tucker (KKT) conditions** for optimality in constrained optimization problems.

1. **Stationarity**: Computes the gradient of the objective function (∇f) and constraints (∇g), then verifies if `∇f + λ * ∇g = 0` at `x_star`.
2. **Primal Feasibility**: Checks if all constraints are satisfied (`g(x*) ≤ 0`).
3. **Dual Feasibility**: Ensures that all Lagrange multipliers (λ) are non-negative.
4. **Complementary Slackness**: Verifies that `λ_i * g_i(x*) = 0`, meaning that active constraints have nonzero multipliers.

Each condition is evaluated using symbolic manipulation (`subs()`, `simplify()`, `evalf()`) to ensure numerical correctness. Finally, the function prints whether each condition holds and returns `True` if all are satisfied.

Example Usage:

The script tests KKT conditions on a simple quadratic minimization problem (`f = x1² + x2²`) subject to a constraint (`x1 + x2 − 1 ≤ 0`). It verifies whether `(x1=0.5, x2=0.5)` is an optimal solution.

Results:

Stationarity holds: False
Primal Feasibility holds: True
Dual Feasibility holds: True
Complementary Slackness holds: True