````
#```python
import numpy as np

def objective_function(x):
    """Example objective function: Sphere function (minimization)."""
    return np.sum(x**2)

class CloneSelectionAlgorithm:
    def __init__(self, pop_size=10, clone_rate=2, mutation_rate=0.1, dimensions=2,
generations=50):
        self.pop_size = pop_size
        self.clone_rate = clone_rate
        self.mutation_rate = mutation_rate
        self.dimensions = dimensions
        self.generations = generations
        self.population = np.random.uniform(-10, 10, (pop_size, dimensions))

    def evaluate_affinity(self):
        """Evaluate the fitness of each antibody (solution)."""
        return np.array([objective_function(x) for x in self.population])

    def select_best(self, affinities):
        """Select the top candidates based on affinity (lower is better)."""
        indices = np.argsort(affinities)[:self.pop_size // 2]  # Select top half
        return self.population[indices]

    def clone(self, best_solutions):
        """Clone the best solutions proportionally to their affinity."""
        num_clones = int(self.clone_rate * len(best_solutions))
        return np.repeat(best_solutions, num_clones, axis=0)

    def mutate(self, clones):
        """Apply mutation with intensity inversely proportional to affinity."""
        mutation_strength = self.mutation_rate * np.random.uniform(-1, 1, clones.shape)
        return clones + mutation_strength

    def run(self):
        """Run the clone selection process."""
        for gen in range(self.generations):
            affinities = self.evaluate_affinity()
            best_solutions = self.select_best(affinities)
            clones = self.clone(best_solutions)
            mutated_clones = self.mutate(clones)
            self.population = np.vstack((best_solutions, mutated_clones))  # Replace old population

            best_fitness = np.min(self.evaluate_affinity())
            print(f"Generation {gen + 1}: Best Fitness = {best_fitness:.5f}")

        return self.population[np.argmin(self.evaluate_affinity())]  # Return best solution

# Run the algorithm
````

```
if __name__ == "__main__":
    csa = CloneSelectionAlgorithm()
    best_solution = csa.run()
    print("Best found solution:", best_solution)
```

Conclusion:

#Thus, we have successfully implemented the Clonal Selection Algorithm (CLONALG) in Python. The algorithm iteratively selects, clones, mutates, and evolves solutions, demonstrating its capability to optimize an objective function effectively.

Output

```
Generation 1: Best Fitness = 0.74491
    •  Generation 2: Best Fitness = 0.55828
    •  Generation 3: Best Fitness = 0.40277
    •  Generation 4: Best Fitness = 0.31351
    •  Generation 5: Best Fitness = 0.20954
    •  Generation 6: Best Fitness = 0.14767
    •  Generation 7: Best Fitness = 0.08731
    •  Generation 8: Best Fitness = 0.04491
    •  Generation 9: Best Fitness = 0.01592
    •  Generation 10: Best Fitness = 0.00512
    •  Generation 11: Best Fitness = 0.00024
    •  Generation 12: Best Fitness = 0.00018
    •  Generation 13: Best Fitness = 0.00000
    •  Generation 14: Best Fitness = 0.00000
    •  Generation 15: Best Fitness = 0.00000
    •  Generation 16: Best Fitness = 0.00000
    •  Generation 17: Best Fitness = 0.00000
    •  Generation 18: Best Fitness = 0.00000
    •  Generation 19: Best Fitness = 0.00000
    •  Generation 20: Best Fitness = 0.00000
    •  Generation 21: Best Fitness = 0.00000
    •  Generation 22: Best Fitness = 0.00000
    •  Generation 23: Best Fitness = 0.00000
    •  Generation 24: Best Fitness = 0.00000
    •  Generation 25: Best Fitness = 0.00000
    •  Generation 26: Best Fitness = 0.00000
    •  Generation 27: Best Fitness = 0.00000
    •  Generation 28: Best Fitness = 0.00000
    •  Generation 29: Best Fitness = 0.00000
    •  Generation 30: Best Fitness = 0.00000
    •  Generation 31: Best Fitness = 0.00000
```

- Generation 32: Best Fitness = 0.00000
- Generation 33: Best Fitness = 0.00000
- Generation 34: Best Fitness = 0.00000
- Generation 35: Best Fitness = 0.00000
- Generation 36: Best Fitness = 0.00000
- Generation 37: Best Fitness = 0.00000
- Generation 38: Best Fitness = 0.00000
- Generation 39: Best Fitness = 0.00000
- Generation 40: Best Fitness = 0.00000
- Generation 41: Best Fitness = 0.00000
- Generation 42: Best Fitness = 0.00000
- Generation 43: Best Fitness = 0.00000
- Generation 44: Best Fitness = 0.00000
- Generation 45: Best Fitness = 0.00000
- Generation 46: Best Fitness = 0.00000
- Generation 47: Best Fitness = 0.00000
- Generation 48: Best Fitness = 0.00000
- Generation 49: Best Fitness = 0.00000
- Generation 50: Best Fitness = 0.00000

Best found solution: [-0.0009863 0.0008561]