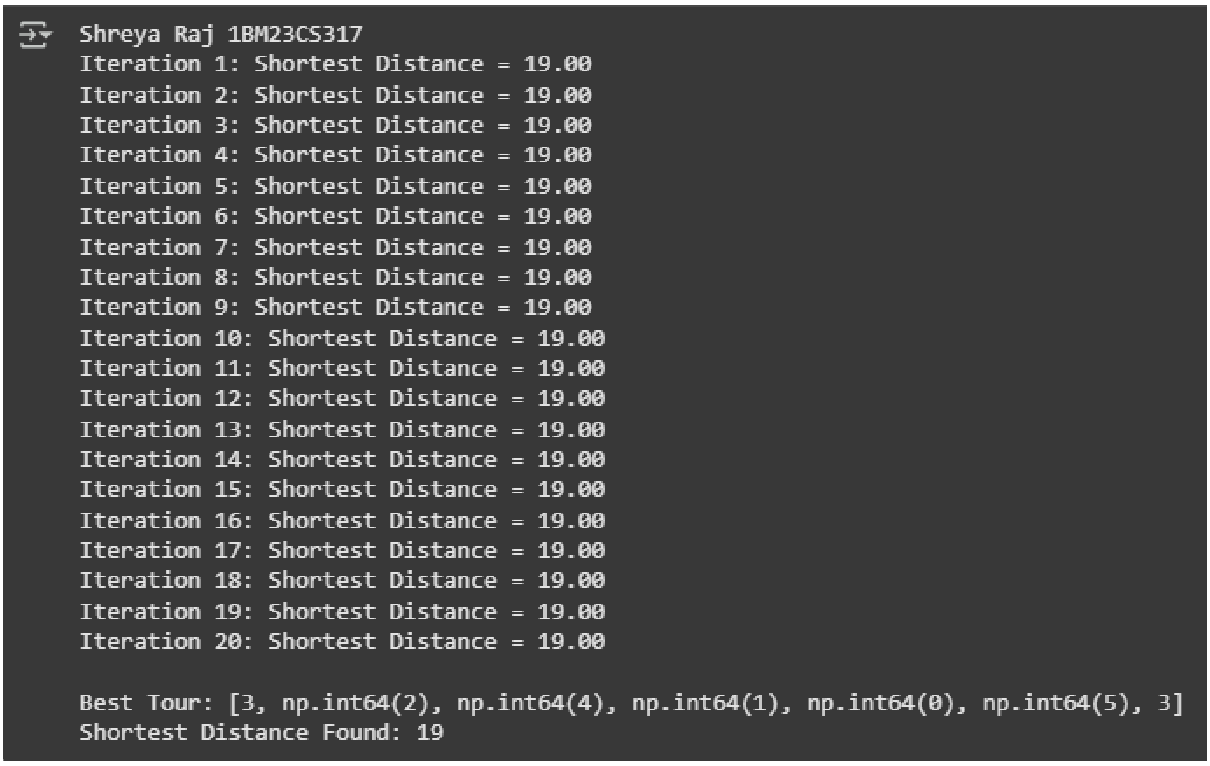
Ant Colony Optimization

Code:

```
import numpy as np

import random

print('Shreya Raj 1BM23CS317')

class ACO_TSP:

    def __init__(self, distances, n_ants=10, n_iterations=50, alpha=1, beta=3, rho=0.5, Q=100):

        self.distances = distances

        self.num_cities = distances.shape[0]

        self.n_ants = n_ants

        self.n_iterations = n_iterations

        self.alpha = alpha   # Influence of pheromone

        self.beta = beta     # Influence of visibility (1/distance)

        self.rho = rho       # Evaporation rate

        self.Q = Q           # Pheromone deposit factor

        self.pheromone = np.ones((self.num_cities, self.num_cities))

        self.visibility = 1 / (distances + np.eye(self.num_cities))  # Avoid divide by zero


    def run(self):

        best_distance = np.inf

        best_tour = None


        for iteration in range(self.n_iterations):

            all_tours = []

            all_distances = []


            for _ in range(self.n_ants):

                tour = self.construct_tour()

                distance = self.calculate_distance(tour)
                all_tours.append(tour)
                all_distances.append(distance)

            # Update pheromones based on all ants
            self.update_pheromones(all_tours, all_distances)

            # Track the best tour
            min_distance = min(all_distances)
            if min_distance < best_distance:
                best_distance = min_distance
                best_tour = all_tours[np.argmin(all_distances)]

            print(f"Iteration {iteration+1}: Shortest Distance = {min_distance:.2f}")

        print("\nBest Tour:", best_tour)
        print("Shortest Distance Found:", best_distance)
        return best_tour, best_distance

    def construct_tour(self):
        start = random.randint(0, self.num_cities - 1)
        tour = [start]
        visited = set(tour)

        for _ in range(self.num_cities - 1):
            current = tour[-1]
            next_city = self.select_next_city(current, visited)
            tour.append(next_city)
            visited.add(next_city)

        tour.append(tour[0])  # Return to start
        return tour
    def select_next_city(self, current, visited):
        probabilities = []
        pheromone = np.copy(self.pheromone[current])
        visibility = np.copy(self.visibility[current])

        for city in range(self.num_cities):
            if city not in visited:
                probabilities.append((pheromone[city] ** self.alpha) * (visibility[city] ** self.beta))
            else:
                probabilities.append(0)

        probabilities = np.array(probabilities)
        probabilities = probabilities / probabilities.sum()
        return np.random.choice(range(self.num_cities), p=probabilities)

    def calculate_distance(self, tour):
        distance = 0
        for i in range(len(tour) - 1):
            distance += self.distances[tour[i], tour[i+1]]
        return distance

    def update_pheromones(self, all_tours, all_distances):
        self.pheromone *= (1 - self.rho)
        for tour, dist in zip(all_tours, all_distances):
            for i in range(len(tour) - 1):
                self.pheromone[tour[i], tour[i+1]] += self.Q / dist

# Example: Distance matrix for 6 cities
if __name__ == "__main__":
    distance_matrix = np.array([
    [0, 2, 9, 10, 7, 3],
    [2, 0, 6, 4, 3, 8],
    [9, 6, 0, 5, 2, 7],
    [10, 4, 5, 0, 6, 4],
    [7, 3, 2, 6, 0, 5],
    [3, 8, 7, 4, 5, 0]
])

aco = ACO_TSP(distance_matrix, n_ants=8, n_iterations=20, alpha=1, beta=3, rho=0.4)
best_tour, best_distance = aco.run()
```

Output:



```
Shreya Raj 1BM23CS317
Iteration 1: Shortest Distance = 19.00
Iteration 2: Shortest Distance = 19.00
Iteration 3: Shortest Distance = 19.00
Iteration 4: Shortest Distance = 19.00
Iteration 5: Shortest Distance = 19.00
Iteration 6: Shortest Distance = 19.00
Iteration 7: Shortest Distance = 19.00
Iteration 8: Shortest Distance = 19.00
Iteration 9: Shortest Distance = 19.00
Iteration 10: Shortest Distance = 19.00
Iteration 11: Shortest Distance = 19.00
Iteration 12: Shortest Distance = 19.00
Iteration 13: Shortest Distance = 19.00
Iteration 14: Shortest Distance = 19.00
Iteration 15: Shortest Distance = 19.00
Iteration 16: Shortest Distance = 19.00
Iteration 17: Shortest Distance = 19.00
Iteration 18: Shortest Distance = 19.00
Iteration 19: Shortest Distance = 19.00
Iteration 20: Shortest Distance = 19.00

Best Tour: [3, np.int64(2), np.int64(4), np.int64(1), np.int64(0), np.int64(5), 3]
Shortest Distance Found: 19
```