TASK 5:

```
import numpy as np

# Given distance matrix between cities
d = np.array([[0, 10, 12, 11, 14],
        [10, 0, 13, 15, 8],
        [12, 13, 0, 9, 14],
        [11, 15, 9, 0, 16],
        [14, 8, 14, 16, 0]])

iteration = 100
n_ants = 5
n_cities = 5

# Parameters
e = 0.5       # evaporation rate
alpha = 1      # pheromone factor
beta = 2       # visibility factor

# Visibility (heuristic info): 1/distance (avoid division by zero)
visibility = np.zeros_like(d, dtype=float)
with np.errstate(divide='ignore'):
  visibility = 1 / d
visibility[d == 0] = 0  # no visibility for same city

# Initialize pheromone levels
pheromone = 0.1 * np.ones((n_cities, n_cities))

# Route array to hold paths for all ants (n_ants x n_cities+1)
route = np.zeros((n_ants, n_cities + 1), dtype=int)
```

```
for ite in range(iteration):
    # Start all ants from city 0 (index-based)
    route[:, 0] = 0

    for i in range(n_ants):
        visited = set([0])  # mark starting city visited

        for step in range(1, n_cities):
            cur_city = route[i, step - 1]

            # Compute probabilities for next city selection
            pheromone_row = pheromone[cur_city]
            visibility_row = visibility[cur_city]

            # Mask visited cities to zero out their attractiveness
            mask = np.array([city not in visited for city in range(n_cities)])
            pheromone_row = pheromone_row * mask
            visibility_row = visibility_row * mask

            # Calculate probabilities (pheromone^alpha * visibility^beta)
            probs = (pheromone_row ** alpha) * (visibility_row ** beta)

            total = np.sum(probs)
            if total == 0:
                # No available city, pick any unvisited city randomly
                candidates = [city for city in range(n_cities) if city not in visited]
                next_city = np.random.choice(candidates)
            else:
                probs /= total
                next_city = np.random.choice(n_cities, p=probs)
```

```
            route[i, step] = next_city
            visited.add(next_city)

        # Return to start city
        route[i, -1] = 0

    # Calculate distances for all ants
    dist_cost = np.zeros(n_ants)
    for i in range(n_ants):
        s = 0
        for j in range(n_cities):
            s += d[route[i, j], route[i, j + 1]]
        dist_cost[i] = s

    # Find best ant in this iteration
    dist_min_loc = np.argmin(dist_cost)
    dist_min_cost = dist_cost[dist_min_loc]
    best_route = route[dist_min_loc]

    # Evaporate pheromone
    pheromone *= (1 - e)

    # Update pheromone by ants' routes
    for i in range(n_ants):
        contribution = 1 / dist_cost[i]
        for j in range(n_cities):
            from_city = route[i, j]
            to_city = route[i, j + 1]
            pheromone[from_city, to_city] += contribution
            pheromone[to_city, from_city] += contribution  # undirected
```

```
print('Route of all ants:')

print(route)

print()

print('Best path (0-based cities):', best_route)

print('Cost of the best path:', dist_min_cost)
```

OUTPUT:

```
Output
Route of all ants:
[[0 3 2 1 4 0]
 [0 3 2 1 4 0]
 [0 4 1 2 3 0]
 [0 3 2 1 4 0]
 [0 3 2 1 4 0]]

Best path (0-based cities): [0 3 2 1 4 0]
Cost of the best path: 55.0

=== Code Execution Successful ===
```