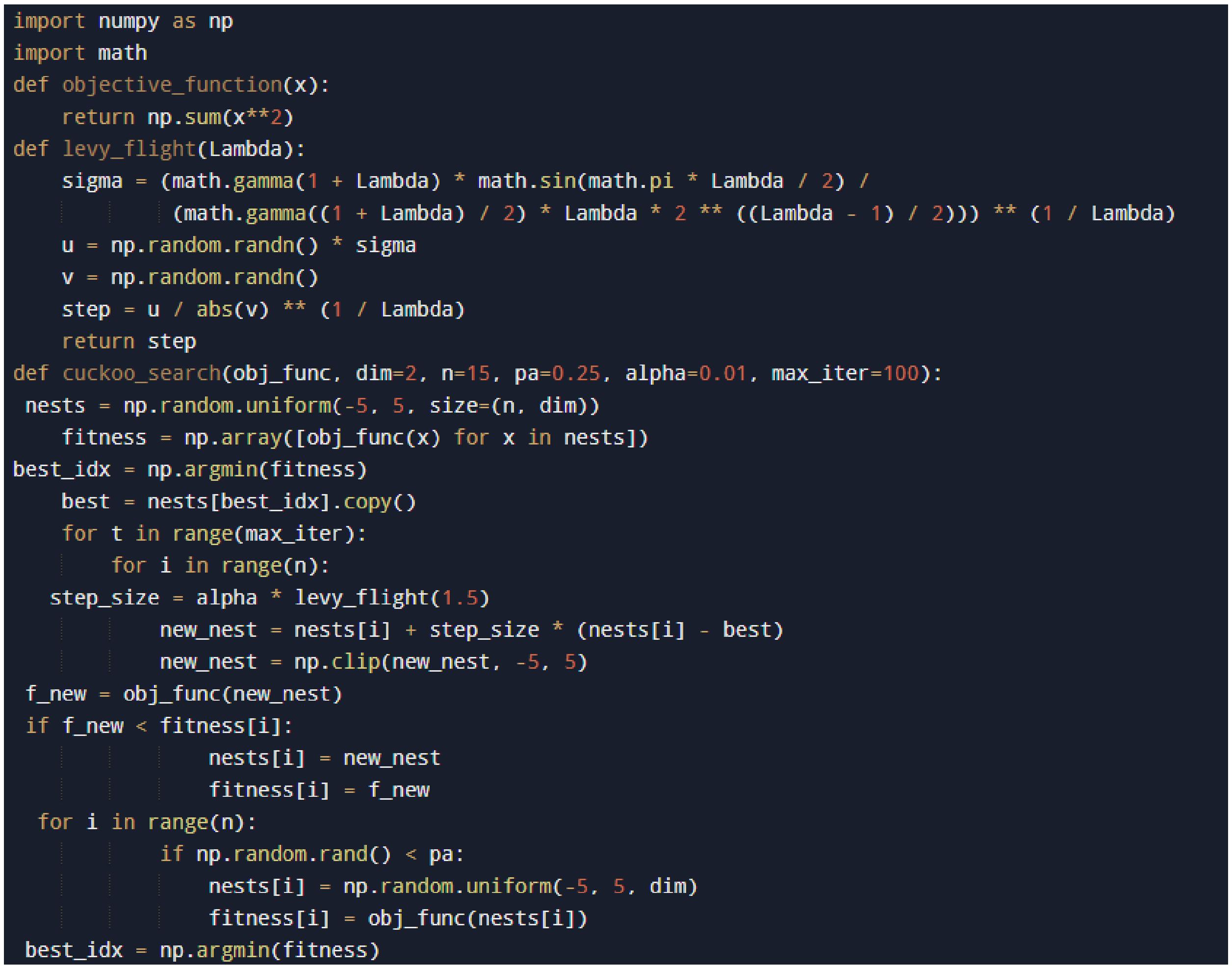
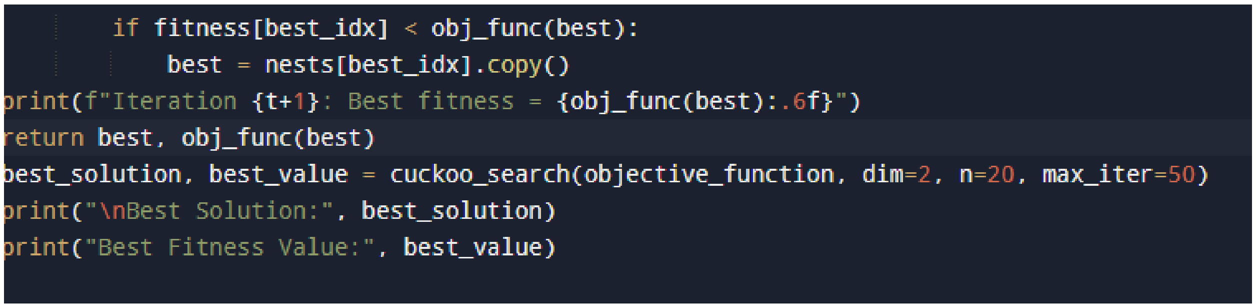
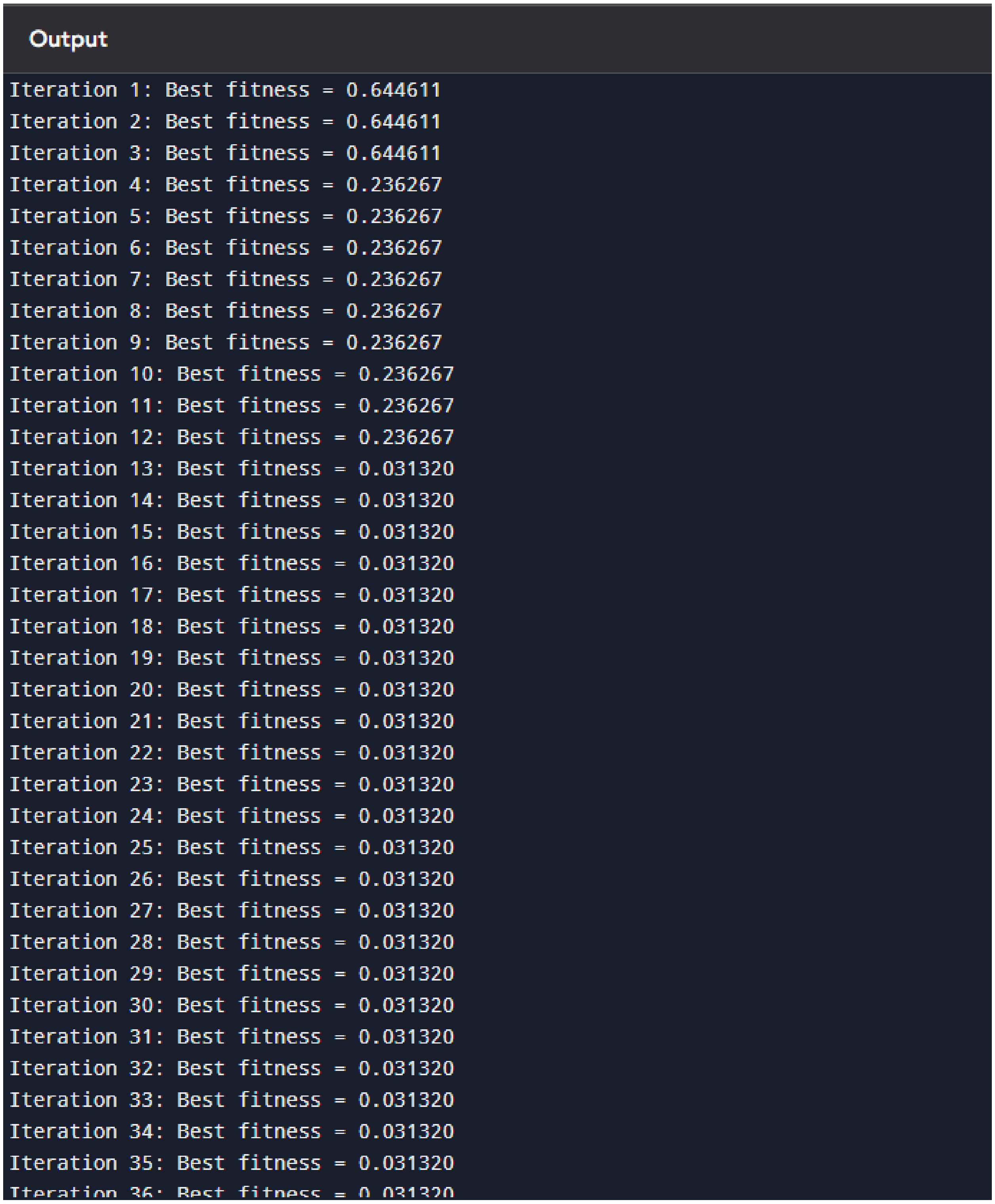
Cuckoo Search (CS):



```python
import numpy as np
import math
def objective_function(x):
    return np.sum(x**2)
def levy_flight(Lambda):
    sigma = (math.gamma(1 + Lambda) * math.sin(math.pi * Lambda / 2) /
             (math.gamma((1 + Lambda) / 2) * Lambda * 2 ** ((Lambda - 1) / 2))) ** (1 / Lambda)
    u = np.random.randn() * sigma
    v = np.random.randn()
    step = u / abs(v) ** (1 / Lambda)
    return step
def cuckoo_search(obj_func, dim=2, n=15, pa=0.25, alpha=0.01, max_iter=100):
 nests = np.random.uniform(-5, 5, size=(n, dim))
    fitness = np.array([obj_func(x) for x in nests])
best_idx = np.argmin(fitness)
    best = nests[best_idx].copy()
    for t in range(max_iter):
        for i in range(n):
   step_size = alpha * levy_flight(1.5)
            new_nest = nests[i] + step_size * (nests[i] - best)
            new_nest = np.clip(new_nest, -5, 5)
 f_new = obj_func(new_nest)
 if f_new < fitness[i]:
                nests[i] = new_nest
                fitness[i] = f_new
  for i in range(n):
            if np.random.rand() < pa:
                nests[i] = np.random.uniform(-5, 5, dim)
                fitness[i] = obj_func(nests[i])
 best_idx = np.argmin(fitness)
```



```python
        if fitness[best_idx] < obj_func(best):
            best = nests[best_idx].copy()
print(f"Iteration {t+1}: Best fitness = {obj_func(best):.6f}")
return best, obj_func(best)
best_solution, best_value = cuckoo_search(objective_function, dim=2, n=20, max_iter=50)
print("\nBest Solution:", best_solution)
print("Best Fitness Value:", best_value)
```



```
Iteration 1: Best fitness = 0.644611
Iteration 2: Best fitness = 0.644611
Iteration 3: Best fitness = 0.644611
Iteration 4: Best fitness = 0.236267
Iteration 5: Best fitness = 0.236267
Iteration 6: Best fitness = 0.236267
Iteration 7: Best fitness = 0.236267
Iteration 8: Best fitness = 0.236267
Iteration 9: Best fitness = 0.236267
Iteration 10: Best fitness = 0.236267
Iteration 11: Best fitness = 0.236267
Iteration 12: Best fitness = 0.236267
Iteration 13: Best fitness = 0.031320
Iteration 14: Best fitness = 0.031320
Iteration 15: Best fitness = 0.031320
Iteration 16: Best fitness = 0.031320
Iteration 17: Best fitness = 0.031320
Iteration 18: Best fitness = 0.031320
Iteration 19: Best fitness = 0.031320
Iteration 20: Best fitness = 0.031320
Iteration 21: Best fitness = 0.031320
Iteration 22: Best fitness = 0.031320
Iteration 23: Best fitness = 0.031320
Iteration 24: Best fitness = 0.031320
Iteration 25: Best fitness = 0.031320
Iteration 26: Best fitness = 0.031320
Iteration 27: Best fitness = 0.031320
Iteration 28: Best fitness = 0.031320
Iteration 29: Best fitness = 0.031320
Iteration 30: Best fitness = 0.031320
Iteration 31: Best fitness = 0.031320
Iteration 32: Best fitness = 0.031320
Iteration 33: Best fitness = 0.031320
Iteration 34: Best fitness = 0.031320
Iteration 35: Best fitness = 0.031320
Iteration 36: Best fitness = 0.031320
```