In [ ]:
```
Name - shivraj Pandurang Mane.
Class - BE Artificial Intelligence and Data Science.
Roll No. - 37
Practical No.08 - Implement DEAP (Distributed Evolutionary Algorithms) using P
```

In [2]:
```
# Import Required Libraries
```

In [3]:
```
import random
from deap import base, creator, tools, algorithms
import numpy as np
```

In [4]:
```
# Define the evaluation function (minimize a simple mathematical function)
def eval_func(individual):
# Example evaluation function (minimize a quadratic function)
    return sum(x ** 2 for x in individual),
```

In [5]:
```
# DEAP setup
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
```

In [6]:
```
toolbox = base.Toolbox()
```

In [7]:
```
# Define attributes and individuals
toolbox.register("attr_float", random.uniform, -5.0, 5.0)  # Example: Float va
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.a
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

In [8]:
```
# Evaluation function and genetic operators
toolbox.register("evaluate", eval_func)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)
```

In [9]:
```
# Create population
population = toolbox.population(n=50)
```

In [10]:
```
# Genetic Algorithm parameters
generations = 20
```

In [11]:
```
# Run the algorithm
for gen in range(generations):
    offspring = algorithms.varAnd(population, toolbox, cxpb=0.5, mutpb=0.1)

    fits = toolbox.map(toolbox.evaluate, offspring)
    for fit, ind in zip(fits, offspring):
        ind.fitness.values = fit

    population = toolbox.select(offspring, k=len(population))
```

In [12]:

```
# Get the best individual after generations
best_ind = tools.selBest(population, k=1)[0]
best_fitness = best_ind.fitness.values[0]

print("Best individual:", best_ind)
print("Best fitness:", best_fitness)
```

```
Best individual: [0.015090776125539648, -0.02898127643988327, 0.022542071782816
46]
Best fitness: 0.0015757909084177404
```

In [ ]: