In [1]:
```
pip install pygad
```

```
Collecting pygad
  Downloading pygad-3.0.1-py3-none-any.whl (67 kB)
     ---------------------------------------- 68.0/68.0 kB 1.8 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in c:\users\jayadeep\anaconda3\lib\site-packages (from pygad) (3.5.2)
Requirement already satisfied: cloudpickle in c:\users\jayadeep\anaconda3\lib\site-packages (from pygad) (2.0.0)
Requirement already satisfied: numpy in c:\users\jayadeep\anaconda3\lib\site-packages (from pygad) (1.21.5)
Requirement already satisfied: packaging>=20.0 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->py
gad) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->pyga
d) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->
pygad) (1.4.2)
Requirement already satisfied: cycler>=0.10 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->pyga
d) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->p
ygad) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotli
b->pygad) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\jayadeep\anaconda3\lib\site-packages (from matplotlib->
pygad) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\users\jayadeep\anaconda3\lib\site-packages (from python-dateutil>=2.7-
>matplotlib->pygad) (1.16.0)
Installing collected packages: pygad
Successfully installed pygad-3.0.1
Note: you may need to restart the kernel to use updated packages.
```

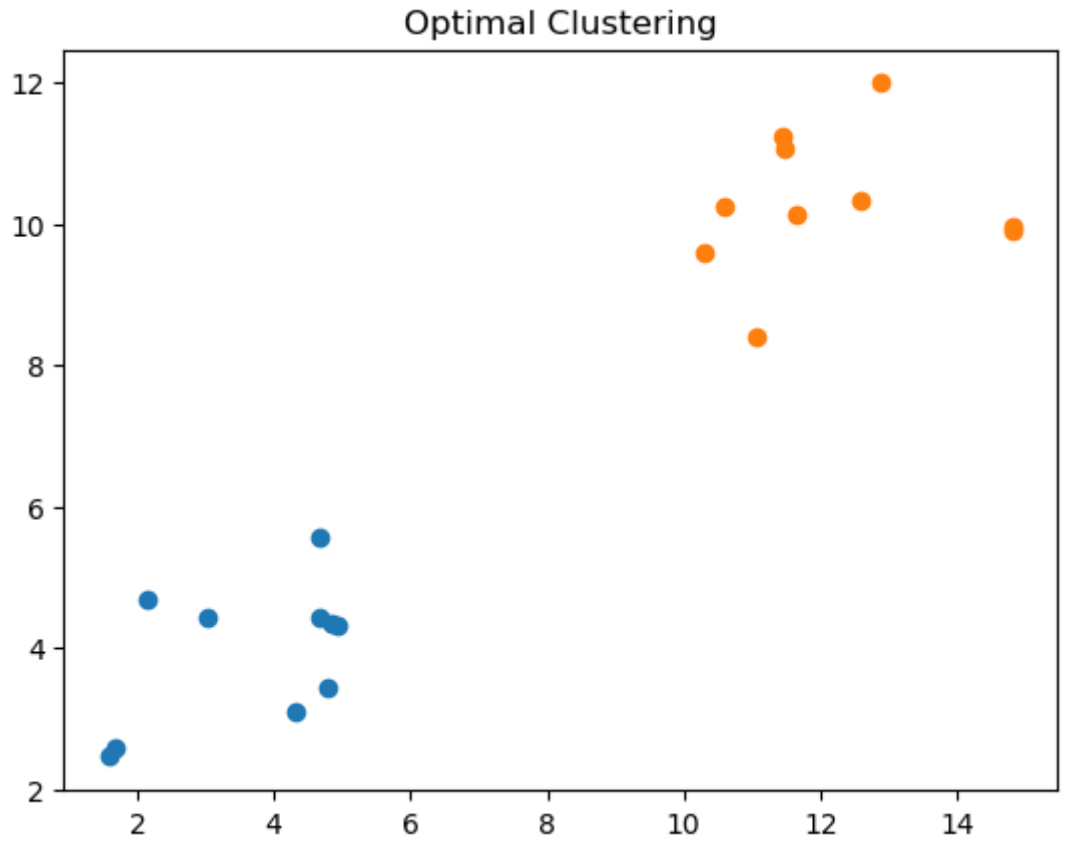
In [5]:
```
import numpy
import matplotlib.pyplot
import pygad
```

In [6]:
```
cluster1_num_samples = 10
cluster1_x1_start = 0
cluster1_x1_end = 5
cluster1_x2_start = 2
cluster1_x2_end = 6
cluster1_x1 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x1 = cluster1_x1 * (cluster1_x1_end - cluster1_x1_start) + cluster1_x1_start
cluster1_x2 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x2 = cluster1_x2 * (cluster1_x2_end - cluster1_x2_start) + cluster1_x2_start
cluster2_num_samples = 10
cluster2_x1_start = 10
cluster2_x1_end = 15
cluster2_x2_start = 8
cluster2_x2_end = 12
cluster2_x1 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x1 = cluster2_x1 * (cluster2_x1_end - cluster2_x1_start) + cluster2_x1_start
cluster2_x2 = numpy.random.random(size=(cluster1_num_samples))
cluster2_x2 = cluster2_x2 * (cluster2_x2_end - cluster2_x2_start) + cluster2_x2_start
```

In [7]:
```
c1 = numpy.array([cluster1_x1, cluster1_x2]).T
c2 = numpy.array([cluster2_x1, cluster2_x2]).T
data = numpy.concatenate((c1, c2), axis=0)
data
```

Out[7]:
```
array([[ 4.92664811,  4.32758969],
       [ 4.31803111,  3.09787853],
       [ 1.67525472,  2.57793639],
       [ 4.66750946,  4.43098494],
       [ 4.68315563,  5.56648397],
       [ 1.59130884,  2.46428823],
       [ 3.0202297 ,  4.44323675],
       [ 4.79628983,  3.45373858],
       [ 4.83475527,  4.34065429],
       [ 2.13954545,  4.6782799 ],
       [14.81758168,  9.90144897],
       [10.58263297, 10.25477129],
       [11.45991223, 11.23057995],
       [12.88194837, 11.98873751],
       [11.48559714, 11.07130778],
       [10.31549994,  9.60403836],
       [11.65398765, 10.12388349],
       [14.80915744,  9.96745259],
       [12.60360569, 10.32966396],
       [11.07939833,  8.40554038]])
```

In [11]:
```
matplotlib.pyplot.scatter(cluster1_x1, cluster1_x2)
matplotlib.pyplot.scatter(cluster2_x1, cluster2_x2)
matplotlib.pyplot.title("Optimal Clustering")
matplotlib.pyplot.show()
```



In [13]:
```
def euclidean_distance(X, Y):
    return numpy.sqrt(numpy.sum(numpy.power(X - Y, 2), axis=1))
```

In [14]:
```
def cluster_data(solution, solution_idx):
    global num_cluster, data
    feature_vector_length = data.shape[1]
    cluster_centers = []
    all_clusters_dists = []
    clusters = []
    clusters_sum_dist = []
    for clust_idx in range(num_clusters):
        cluster_centers.append(solution[feature_vector_length*clust_idx:feature_vector_length*(clust_idx+1)])
        cluster_center_dists = euclidean_distance(data, cluster_centers[clust_idx])
        all_clusters_dists.append(numpy.array(cluster_center_dists))
    cluster_centers = numpy.array(cluster_centers)
    all_clusters_dists = numpy.array(all_clusters_dists)
    cluster_indices = numpy.argmin(all_clusters_dists, axis=0)
    for clust_idx in range(num_clusters):
        clusters.append(numpy.where(cluster_indices == clust_idx)[0])
        if len(clusters[clust_idx]) == 0:
            clusters_sum_dist.append(0)
        else:
            clusters_sum_dist.append(numpy.sum(all_clusters_dists[clust_idx, clusters[clust_idx]]))
    clusters_sum_dist = numpy.array(clusters_sum_dist)
    return cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum_dist
```

In [17]:
```
def fitness_func(ga_instance,solution, solution_idx):
    _, _, _, _, clusters_sum_dist = cluster_data(solution, solution_idx)
    fitness = 1.0 / (numpy.sum(clusters_sum_dist) + 0.00000001)
    return fitness
```

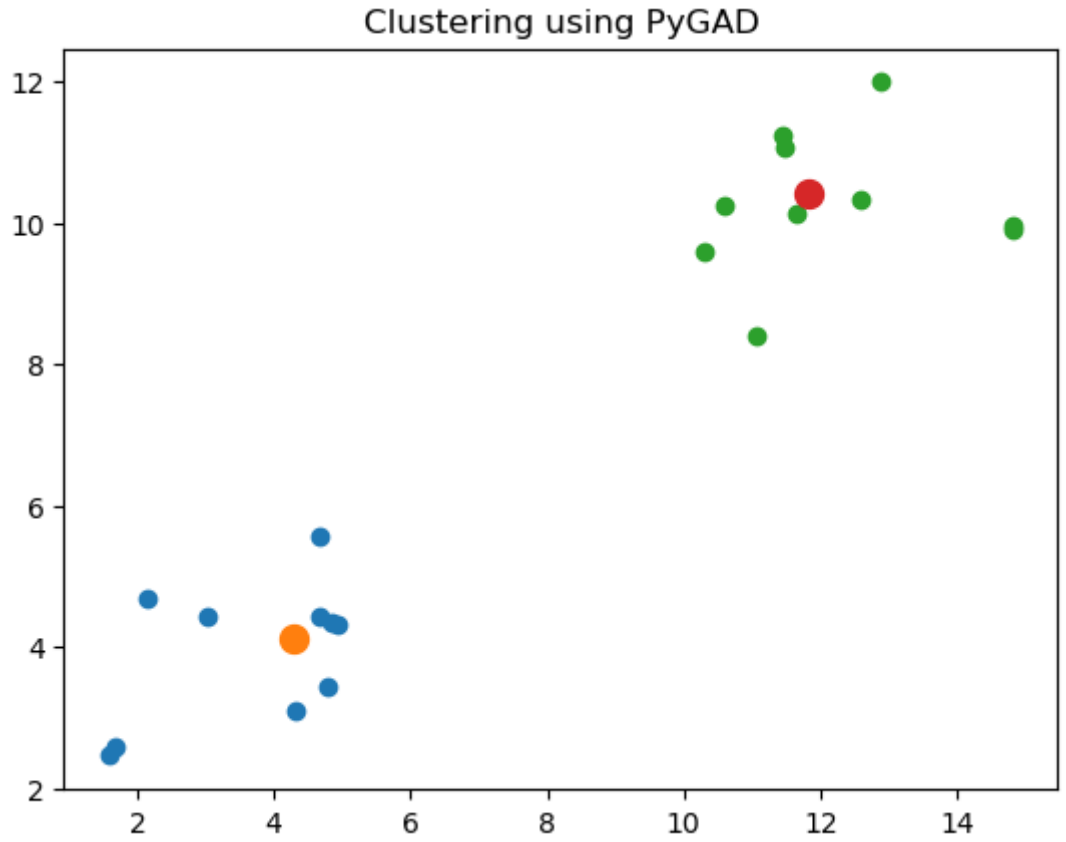
In [19]:
```
num_clusters = 2
num_genes = num_clusters * data.shape[1]
ga_instance = pygad.GA(num_generations=100,
                       sol_per_pop=10,
                       num_parents_mating=5,
                       init_range_low=-6,
                       init_range_high=20,
                       keep_parents=2,
                       num_genes=num_genes,
                       fitness_func=fitness_func,
                       suppress_warnings=True)
ga_instance.run()
```

In [20]:
```
best_solution, best_solution_fitness, best_solution_idx = ga_instance.best_solution()
print("Best solution is {bs}".format(bs=best_solution))
print("Fitness of the best solution is {bsf}".format(bsf=best_solution_fitness))
print("Best solution found after {gen} generations".format(gen=ga_instance.best_solution_generation))
```

```
Best solution is [ 4.29748073  4.11775849 11.82628985 10.40370041]
Fitness of the best solution is 0.03262645388658683
Best solution found after 99 generations
```

In [23]:
```
cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum_dist= cluster_data(best_solution, best_sc
```

In [24]:
```
for cluster_idx in range(num_clusters):
    cluster_x = data[clusters[cluster_idx], 0]
    cluster_y = data[clusters[cluster_idx], 1]
    matplotlib.pyplot.scatter(cluster_x, cluster_y)
    matplotlib.pyplot.scatter(cluster_centers[cluster_idx, 0], cluster_centers[cluster_idx, 1], linewidths=5)
matplotlib.pyplot.title("Clustering using PyGAD")
matplotlib.pyplot.show()
```



In [ ]:
```

```