## - load data

In [17]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math

dataset = pd.read_csv('data-kmeans.csv')
data = dataset.values
```

## - init labels

In [18]:

```
label = []
for i in range(len(data)):
    label.append([i % 5 + 1])

label = np.array(label)
data = np.concatenate((data, label), axis=1)
init_data = data.copy()
```

## - Compute init Centroids

In [19]:

```
init_Centroids = [[0, 0], [0, 0], [0, 0], [0, 0], [0, 0]]

for i in range(data.shape[0]):
    if data[i][2] == 1:
        init_Centroids[0][0] += data[i][0] / 40
        init_Centroids[0][1] += data[i][1] / 40

    elif data[i][2] == 2:
        init_Centroids[1][0] += data[i][0] / 40
        init_Centroids[1][1] += data[i][1] / 40

    elif data[i][2] == 3:
        init_Centroids[2][0] += data[i][0] / 40
        init_Centroids[2][1] += data[i][1] / 40

    elif data[i][2] == 4:
        init_Centroids[3][0] += data[i][0] / 40
        init_Centroids[3][1] += data[i][1] / 40

    elif data[i][2] == 5:
        init_Centroids[4][0] += data[i][0] / 40
        init_Centroids[4][1] += data[i][1] / 40
```

## - Define functions

In [20]:

```
def compute_distance(a, b):
    # distance between a and b
    dist = math.sqrt((a[0] - b[0]) ** 2 + (a[1] - b[1]) ** 2)

    return dist
```

In [21]:

```
def compute_centroid(Z):
    center = [[0, 0] for i in range(5)]
    cluster_n = [0 for i in range(5)]

    for i in range(data.shape[0]):
        if Z[i][2] == 1:
            center[0][0] += data[i][0]
            center[0][1] += data[i][1]
            cluster_n[0] +=1

        elif Z[i][2] == 2:
            center[1][0] += data[i][0]
            center[1][1] += data[i][1]
            cluster_n[1] +=1

        elif Z[i][2] == 3:
            center[2][0] += data[i][0]
            center[2][1] += data[i][1]
            cluster_n[2] +=1

        elif Z[i][2] == 4:
            center[3][0] += data[i][0]
            center[3][1] += data[i][1]
            cluster_n[3] +=1

        elif Z[i][2] == 5:
            center[4][0] += data[i][0]
            center[4][1] += data[i][1]
            cluster_n[4] +=1

    for i in range(len(center)):
        center[i][0] = center[i][0] / cluster_n[i]
        center[i][1] = center[i][1] / cluster_n[i]

    return center
```

/

In [22]:

```
def compute_label(z, M):

    distance_list = [compute_distance(z, M[i]) for i in range(len(M))]
    label = distance_list.index(min(distance_list)) + 1

    return label
```

In [23]:

```
def compute_loss(C, M):
    loss = 0

    for i in range(C.shape[0]):
        loss += compute_distance(C[i], M[C[i][2] - 1])

    return loss / C.shape[0]
```

## - main code

In [24]:

```
loss_curve = []
centroid_curv = [[] for i in range(5)]
ephochs = 100
```

In [25]:

```
for run in range(ephochs):
    Centroids = compute_centroid(data)

    for i in range(data.shape[0]):
        data[i][2] = compute_label(data[i], Centroids)
    loss_curve.append(compute_loss(data, Centroids))

    for j in range(len(centroid_curv)):
        centroid_curv[j].append(compute_distance([0, 0], Centroids[j]))
```

/

## - Define color and label

In [26]:

```
color_list = ["green", "blue", "brown", "skyblue", "violet"]
name_list = ["Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4", "Cluster 5"]
```

## - Compute clusters

In [28]:

```
init_Cluster1 = []
init_Cluster2 = []
init_Cluster3 = []
init_Cluster4 = []
init_Cluster5 = []

for i in range(init_data.shape[0]):
    if init_data[i][2] == 1:
        init_Cluster1.append(init_data[i])

    elif init_data[i][2] == 2:
        init_Cluster2.append(init_data[i])


    elif init_data[i][2] == 3:
        init_Cluster3.append(init_data[i])


    elif init_data[i][2] == 4:
        init_Cluster4.append(init_data[i])


    elif init_data[i][2] == 5:
        init_Cluster5.append(init_data[i])
```

In [29]:

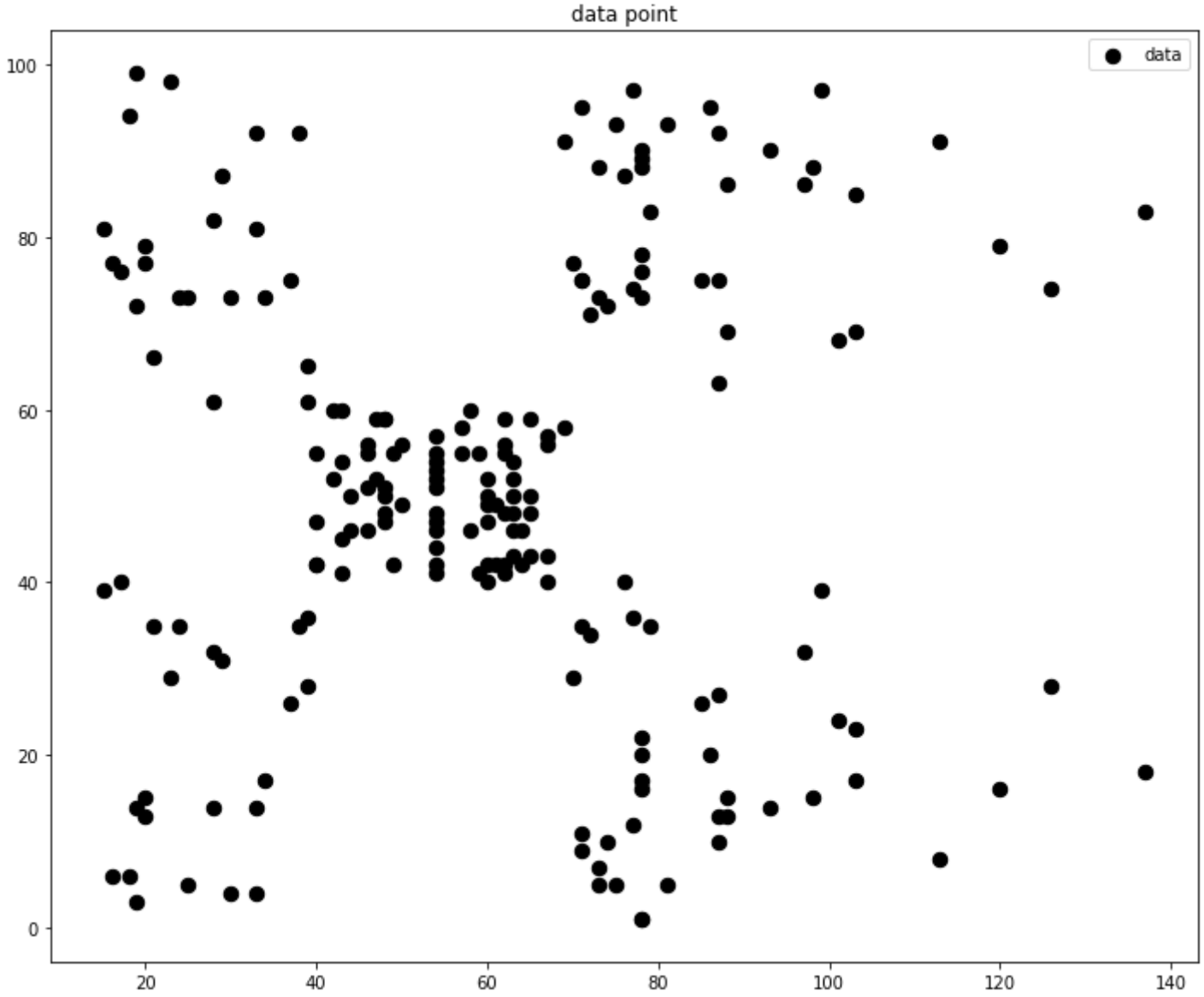
```
Cluster1 = []
Cluster2 = []
Cluster3 = []
Cluster4 = []
Cluster5 = []

for i in range(init_data.shape[0]):
    if data[i][2] == 1:
        Cluster1.append(data[i])

    elif data[i][2] == 2:
        Cluster2.append(data[i])


    elif data[i][2] == 3:
        Cluster3.append(data[i])


    elif data[i][2] == 4:
        Cluster4.append(data[i])


    elif data[i][2] == 5:
        Cluster5.append(data[i])
```

# 1. Plot the data points [1pt]

In [30]:

```
fig = plt.figure(figsize=(12,10))
plt.scatter(data[:,0], data[:,1], s =70, c = 'black', label='data')
plt.title("data point")
plt.legend(loc = 'upper right')
plt.show()
```

/



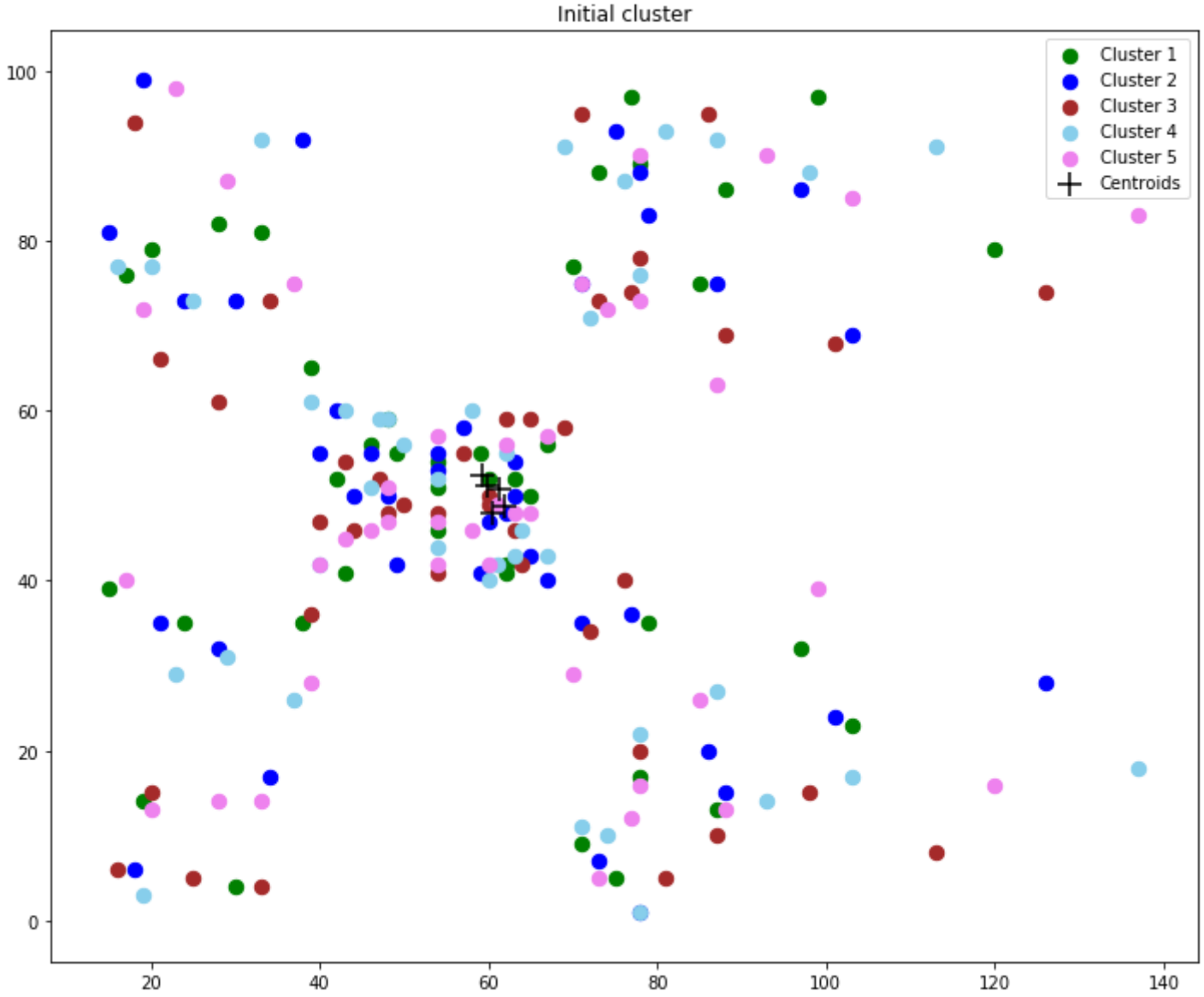

## 2. Visualise the initial condition of the point labels [1pt]

/

In [32]:

```
fig = plt.figure(figsize=(12,10))
plt.scatter(np.array(init_Cluster1)[:,0], np.array(init_Cluster1)[:,1], s = 70, c = color_list[0], label = name_list[0])
plt.scatter(np.array(init_Cluster2)[:,0], np.array(init_Cluster2)[:,1], s = 70, c = color_list[1], label = name_list[1])
plt.scatter(np.array(init_Cluster3)[:,0], np.array(init_Cluster3)[:,1], s = 70, c = color_list[2], label = name_list[2])
plt.scatter(np.array(init_Cluster4)[:,0], np.array(init_Cluster4)[:,1], s = 70, c = color_list[3], label = name_list[3])
plt.scatter(np.array(init_Cluster5)[:,0], np.array(init_Cluster5)[:,1], s = 70, c = color_list[4], label = name_list[4])
plt.scatter(np.array(init_Centroids)[:,0], np.array(init_Centroids)[:,1], s = 200, c = 'black', marker = '+', label = "Centroids")


plt.title("Initial cluster")
plt.legend(loc = 'upper right')
plt.show()
```

/

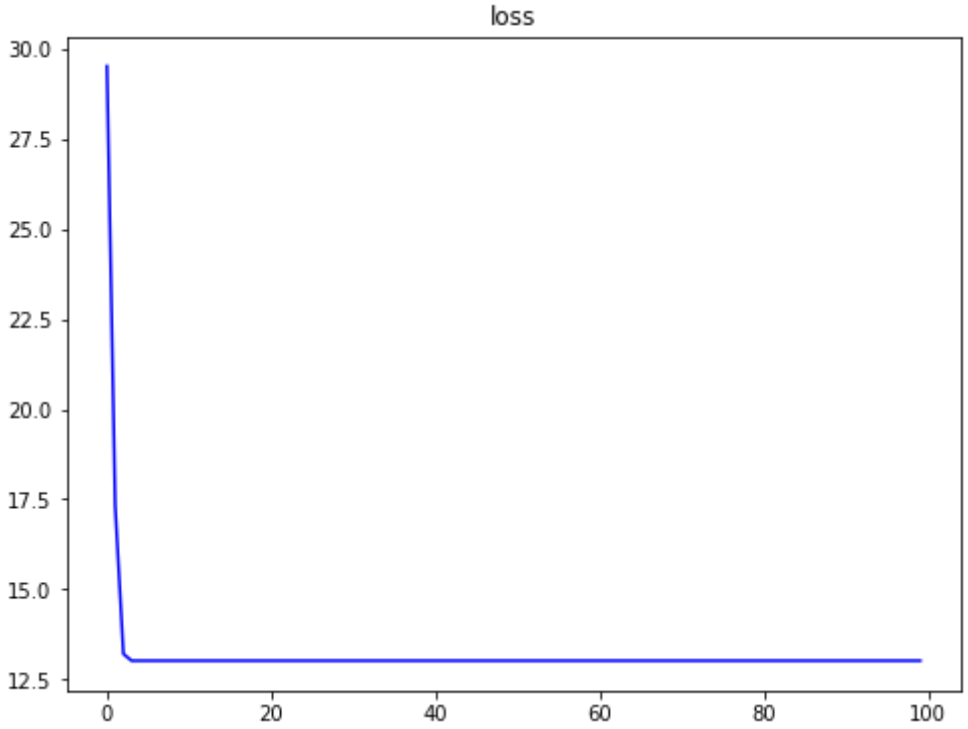

Initial cluster
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
100
80
60
40
20
0
20
40
60
80
100
120
140

/

## 3. Plot the loss curve [5pt]

In [33]:

```
fig = plt.figure(figsize=(8,6))
plt.plot([i for i in range(len(loss_curve))], loss_curve, c = 'b')
plt.title('loss')
plt.show()
```

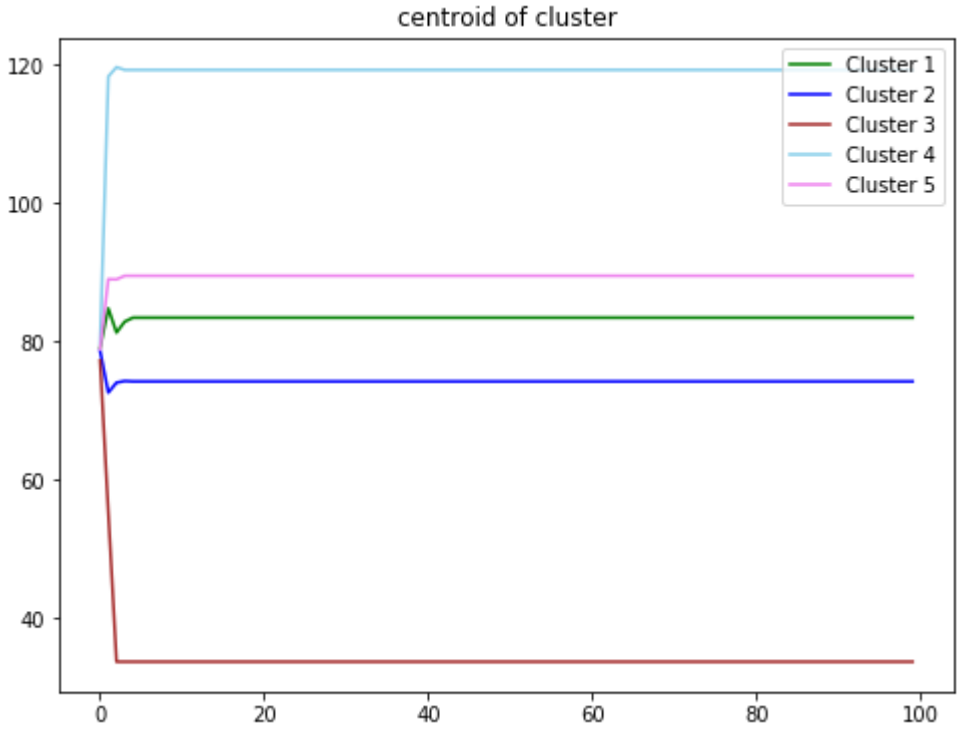



/

## 4. Plot the centroid of each clsuter [5pt]

In [34]:

```
fig = plt.figure(figsize=(8,6))

for i in range(len(centroid_curv)):
    plt.plot([i for i in range(len(centroid_curv[i]))], centroid_curv[i], c = color_list[i], label = name_list[i])

plt.title('centroid of cluster')
plt.legend(loc = 'upper right')
plt.show()
```

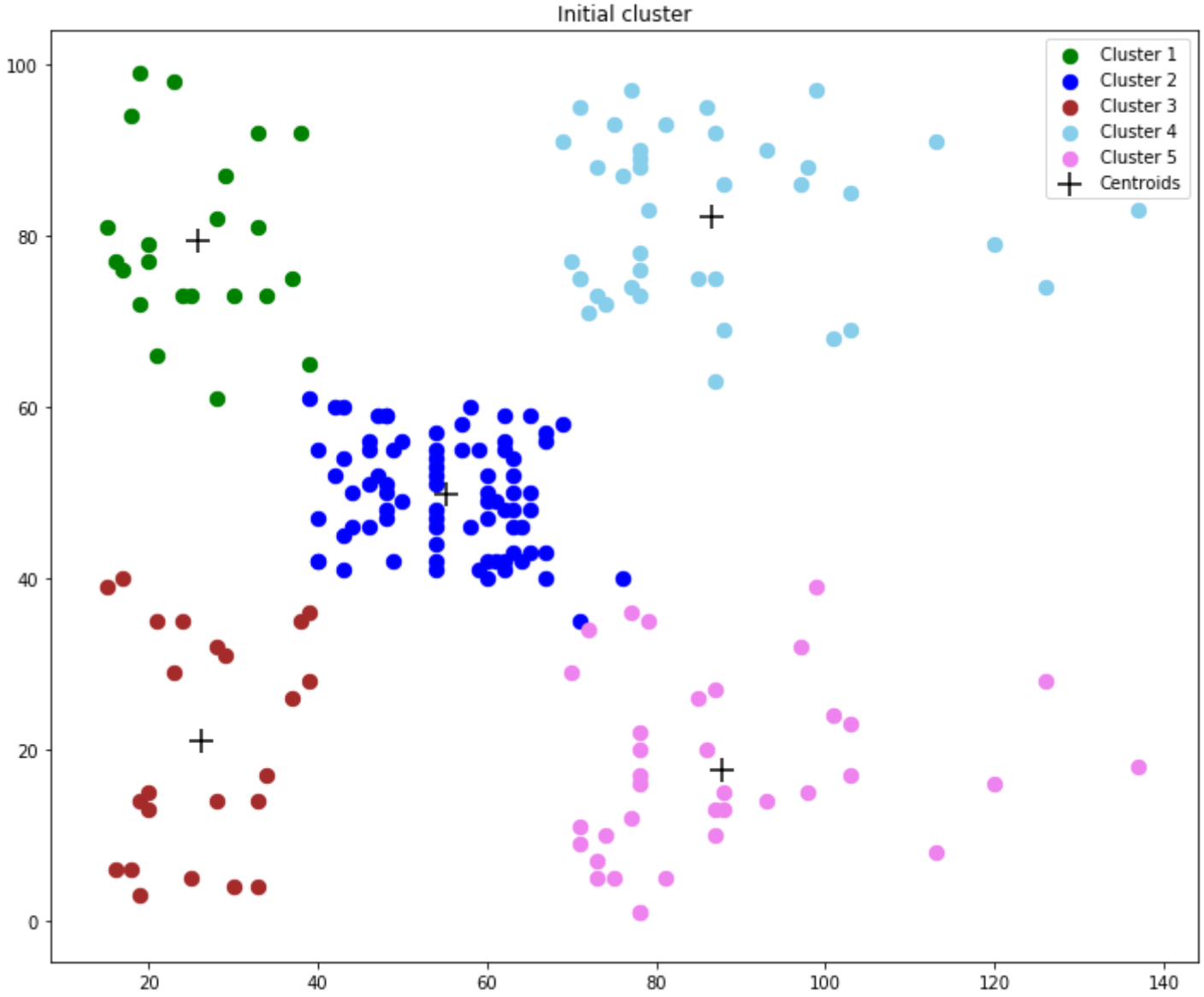



## 5. Plot the final clustering result [5pt]

In [35]:

```
fig = plt.figure(figsize=(12,10))

plt.scatter(np.array(Cluster1)[:,0], np.array(Cluster1)[:,1], s = 70, c = color_list[0], label = name_list[0])
plt.scatter(np.array(Cluster2)[:,0], np.array(Cluster2)[:,1], s = 70, c = color_list[1], label = name_list[1])
plt.scatter(np.array(Cluster3)[:,0], np.array(Cluster3)[:,1], s = 70, c = color_list[2], label = name_list[2])
plt.scatter(np.array(Cluster4)[:,0], np.array(Cluster4)[:,1], s = 70, c = color_list[3], label = name_list[3])
plt.scatter(np.array(Cluster5)[:,0], np.array(Cluster5)[:,1], s = 70, c = color_list[4], label = name_list[4])
plt.scatter(np.array(Centroids)[:,0], np.array(Centroids)[:,1], s = 200, c = 'black', marker = '+', label = "Centroids")


plt.title("Initial cluster")
plt.legend(loc = 'upper right')
plt.show()
```

/


Initial cluster
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
100
80
60
40
20
0
20
40
60
80
100
120
140

/

/