เริ่มต้นด้วยการ import

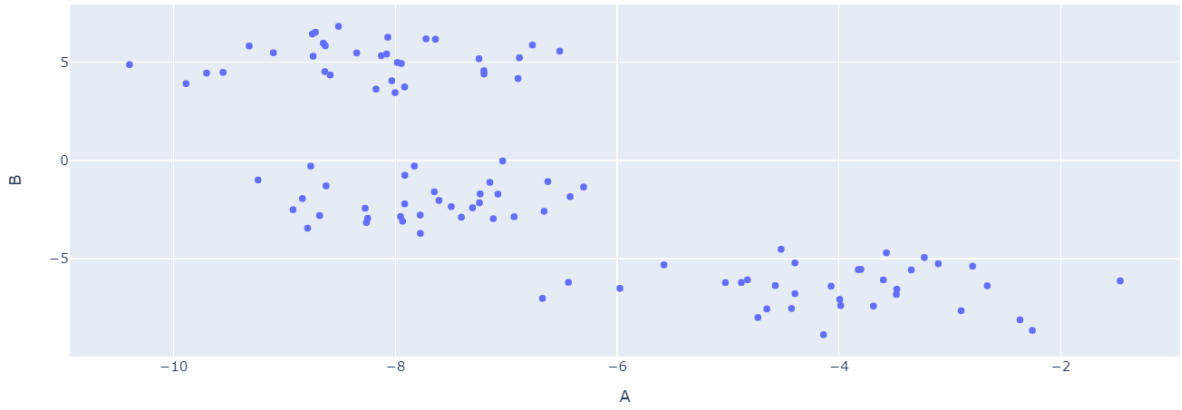
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial import distance_matrix
```

```
data = pd.read_csv("data/data2Dset1.csv",header=None)
data.columns = ["A","B"]
data.head()
```

	A	B
0	-4.575007	-6.364897
1	-7.202692	4.560245
2	-7.148368	-1.115191
3	-7.915773	-0.757674
4	-7.118251	-2.965019

2) Plot จุดข้อมูล data1

```
import plotly.express as px
fig = px.scatter(data1, x="A", y="B")
fig.show()
```



3) Data2Dset1 เริ่มด้วยการเขียนโปรแกรม plot จุดข้อมูลโดยใช้วิธี kmeans

K=1

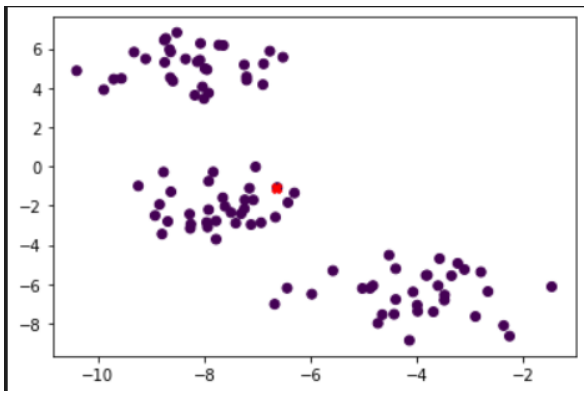
```
model_kmeans = cluster.KMeans(n_clusters=1, max_iter=50, random_state=1)
```

```
model_kmeans.fit(data1)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)


  [[-6.63872652 -1.18919951]]
```

```
plt.scatter(data1['A'],data1['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=1

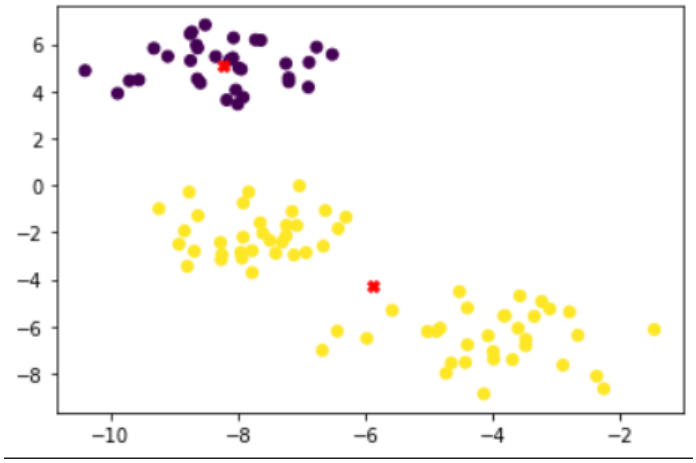


K=2

```
model_kmeans = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1)
model_kmeans.fit(data1)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers
print(centroids)
   [[-5.84938602 -4.29998479  1.01492537]
    [-8.24132694  5.12663729  1.        ]]
```

```
plt.scatter(data1['A'],data1['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=2



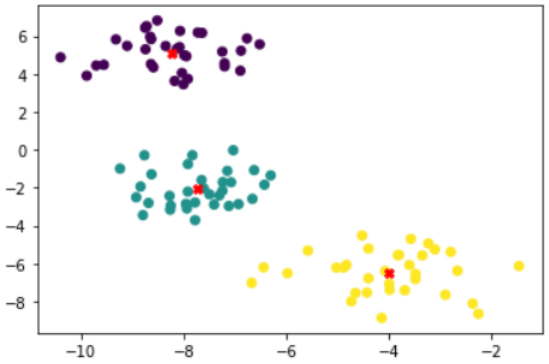
K=3

```
model_kmeans = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
model_kmeans.fit(data1)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-7.72806305e+00 -2.06698658e+00 -6.66133815e-16]
 [-8.24132694e+00  5.12663729e+00  1.00000000e+00]
 [-4.02596420e+00 -6.46730659e+00  2.58823529e+00]]
```

```
plt.scatter(data1['A'],data1['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=3



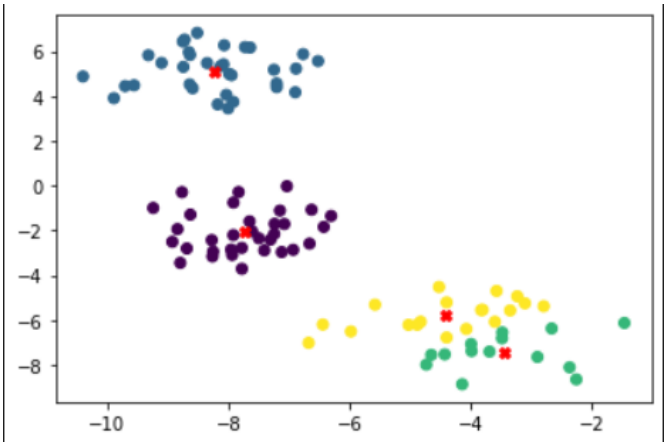
K=4

```
model_kmeans = cluster.KMeans(n_clusters=4, max_iter=50, random_state=1)
model_kmeans.fit(data1)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-7.72806305 -2.06698658  0.        ]
 [-8.24132694  5.12663729  0.        ]
 [-3.44525215 -7.43309475  0.        ]
 [-4.43246263 -5.79125489  0.        ]]
```

```
plt.scatter(data1['A'],data1['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์K=4



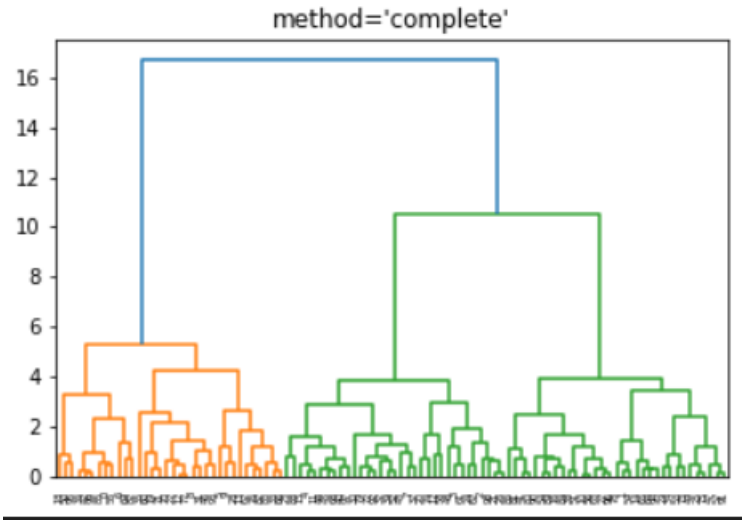
จากผลลัพธ์สรุปได้ว่า K=3 ดีที่สุดสำหรับ Data2Dset1 เพราะจะข้อมูลจะเกาะกลุ่มกันเป็นกลุ่มใหญ่ๆอย่างเห็นได้ชัดสามกลุ่ม ถ้าเป็นK4 จะมีตัวข้อมูลที่เกาะกลุ่มกันมากและถ้าเป็นK1,K2 จะเกิดข้อมูลที่ไม่แน่ชัด

4) เขียนโปแกรมจัดกลุ่มชุดข้อมูลที่อ่านเข้ามา โดยใช้วิธี Hierarchical Clustering

4.1) ให้เลือกใช้method ที่ต่างกัน 3 แบบ แสดง dendrogram ที่ได้แต่ละแบบ

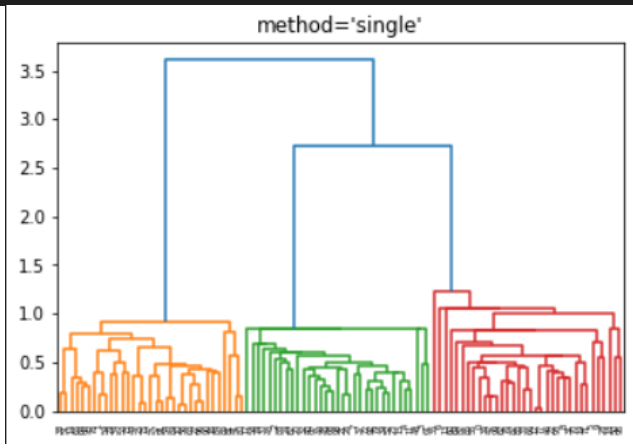
Complete:

```
linkage_data = linkage(data1, method='complete' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='complete'")
plt.show()
```

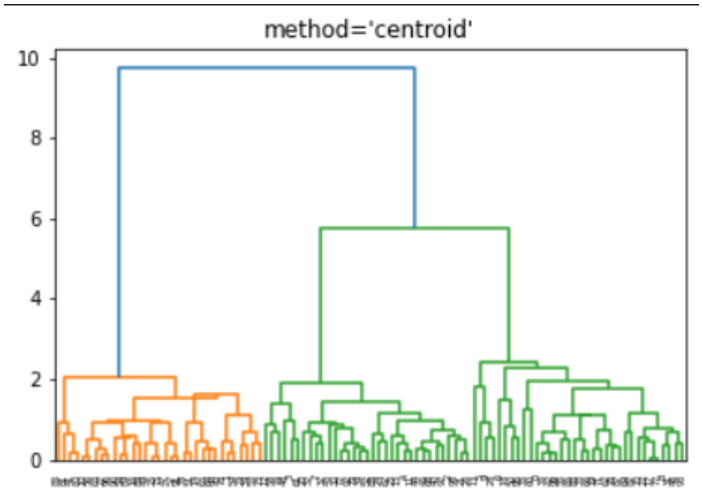


Single:

```
linkage_data = linkage(data1, method='single' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='single'")
plt.show()
```



Centroid:



4.2) เลือก cut-off โดยกำหนด criterion='distance' และให้นักศึกษาเลือกระบุค่า t ที่คิดว่า เหมาะสม สำหรับแต่ละ dendrogram ที่ได้ในข้อ 4.1)

Complete: ค่า t ที่เหมาะที่สุดคือ 6

```
cluster_id = fcluster(linkage_data,t=6,criterion='distance')
plt.scatter(data1["A"],data1["B"],c=cluster_id)
plt.show()
```

Single: ค่า t ที่เหมาะที่สุดคือ 2

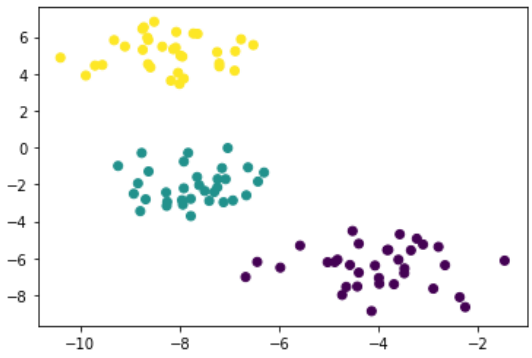
```
cluster_id = fcluster(linkage_data,t=2,criterion='distance')
plt.scatter(data1["A"],data1["B"],c=cluster_id)
plt.show()
```

Centroid: ค่า t ที่เหมาะที่สุดคือ 3 ถึง 5

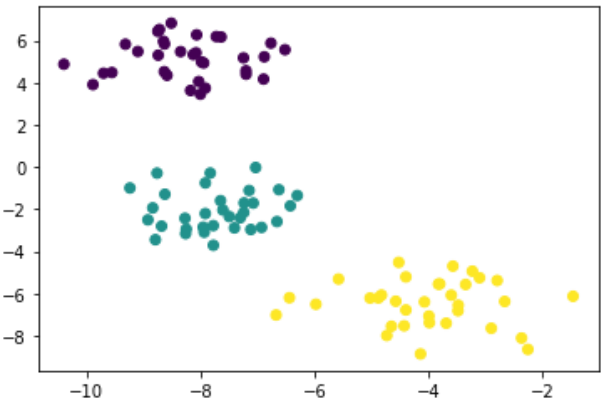
```
cluster_id = fcluster(linkage_data,t=5,criterion='distance')
plt.scatter(data1["A"],data1["B"],c=cluster_id)
plt.show()
```

4.3) Plot ผลการจัดกลุ่ม ที่ได้แต่ละแบบในข้อ 4.2)

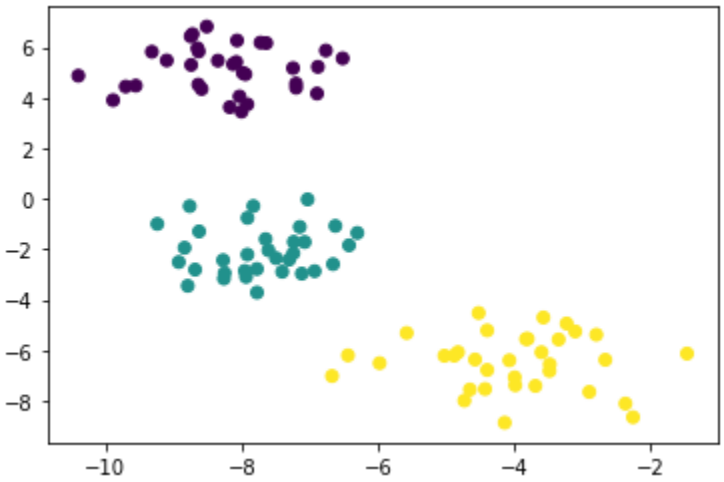
Complete:



Single:



Centroid:



6. เขียนบรรยายสรุปผลการทดลอง แสดงความคิดเห็น วิธีใด เหมาะกับ ชุดข้อมูลแบบไหน แต่ละวิธีมีข้อดี/ ข้อเสีย อย่างไร

- Data2DSet1 จากการทดลองพบว่า วิธี k-Means สามารถทำได้ง่ายกว่า สามารถแบ่งกลุ่มข้อมูลได้ง่ายกว่า และตัวข้อมูลมีดูมีความละเอียดแบ่งเป็นกลุ่มๆอย่างเห็นได้ชัด

เริ่มต้นด้วยการ import

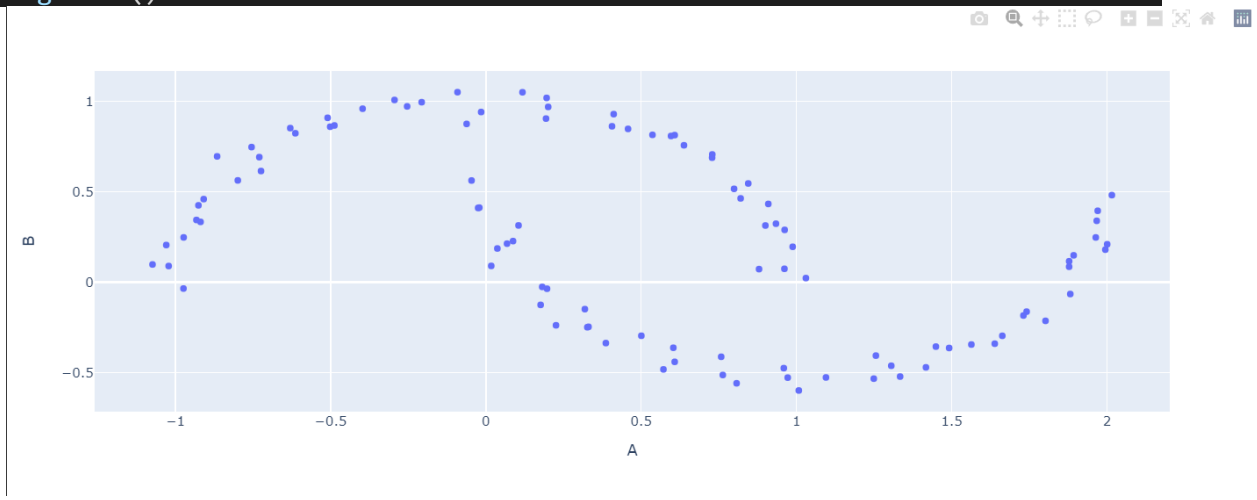
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial import distance_matrix
```

```
data = pd.read_csv("data/data2Dset2.csv")
data.columns = ["A","B"]
data.head()
```

	A	B
0	1.967099	0.339064
1	0.762843	-0.513650
2	-1.029709	0.205156
3	0.637710	0.756872
4	2.000786	0.209418

2) Plot จุดข้อมูล data1

```
1) import plotly.express as px
2) fig = px.scatter(data, x="A", y="B")
3) fig.show()
```



3) Data2Dset2 เริ่มด้วยการเขียนโปรแกรม plot จุดข้อมูลโดยใช้วิธี kmeans

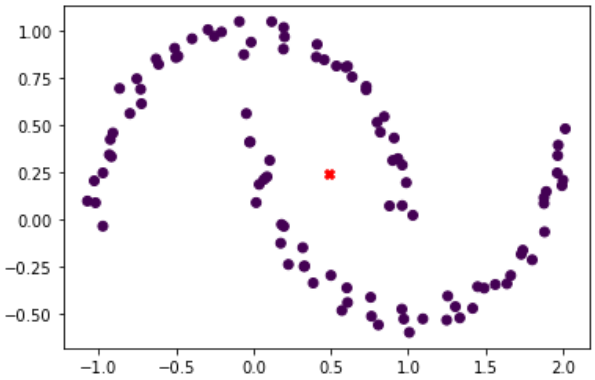
K=1

```
model_kmeans = cluster.KMeans(n_clusters=1, max_iter=50, random_state=1)
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[0.49237282 0.24273502 0.54      ]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=1



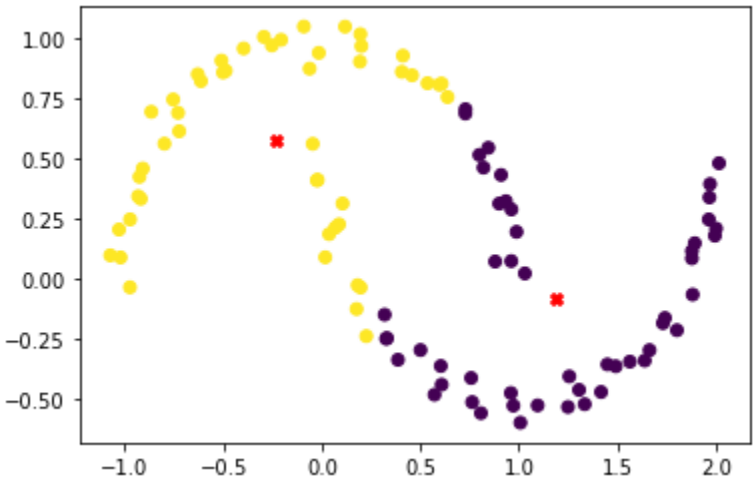
K=2

```
model_kmeans = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1)
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[ 1.26403634 -0.10476722  0.5       ]
 [-0.16497018  0.53875545  1.37037037]]
```

```
plt.scatter(data['A'],data['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=2



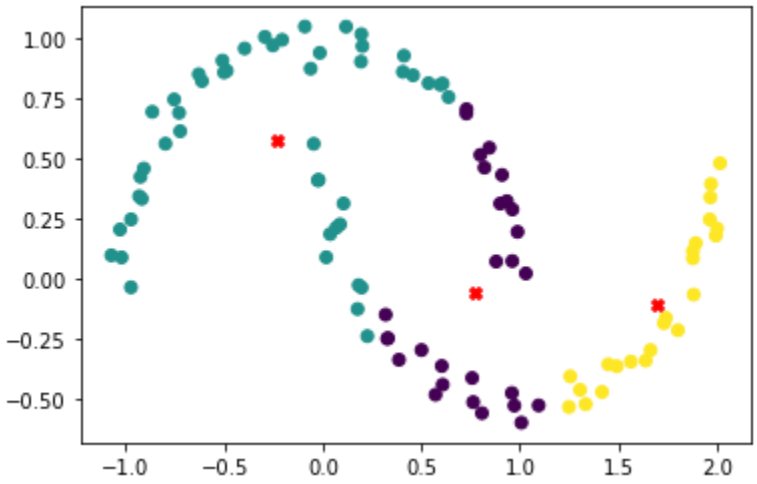
K=3

```
model_kmeans = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[ 1.69940443 -0.10991557  2.        ]
 [ 0.45486035  0.28170959  0.52631579]
 [-0.78880298  0.5372057   3.        ]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=3



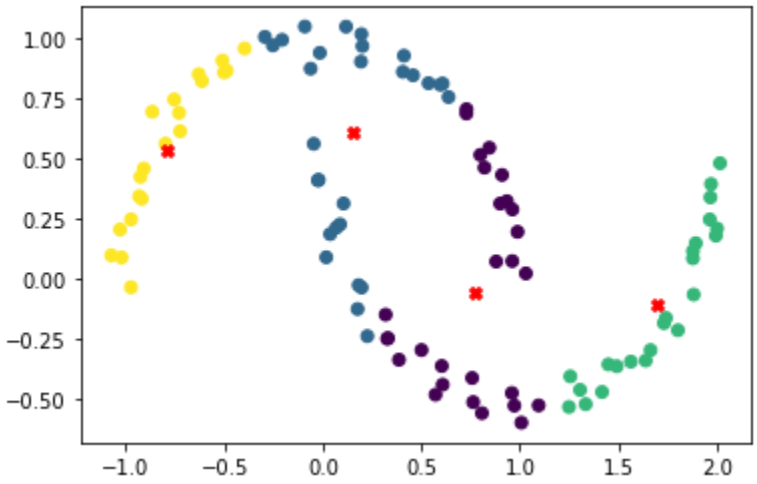
K=4

```
model_kmeans = cluster.KMeans(n_clusters=4, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[ 0.67963398 -0.18109054  0.        ]
 [ 0.25256409  0.6982297   0.        ]
 [ 1.69940443 -0.10991557  0.        ]
 [-0.78880298  0.5372057   0.        ]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์K=4



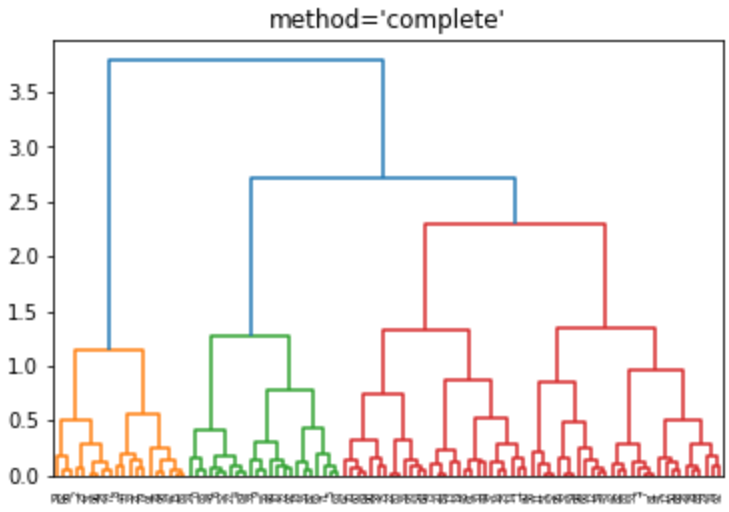
จากผลลัพธ์สรุปได้ว่า K=4 ดีที่สุดสำหรับ Data2Dset2 เพราะข้อมูลจะดูละเอียดอ่อนมากขึ้นถ้าเทียบกับKอื่นๆ ตัวข้อมูลจะถูกแบ่งเป็น4กลุ่มซึ่งจะแยกกันไปตามช่วงประมาณ1

4) เขียนโปแกรมจัดกลุ่มชุดข้อมูลที่อ่านเข้ามา โดยใช้วิธี Hierarchical Clustering

4.1) ให้เลือกใช้method ที่ต่างกัน 3 แบบ แสดง dendrogram ที่ได้แต่ละแบบ

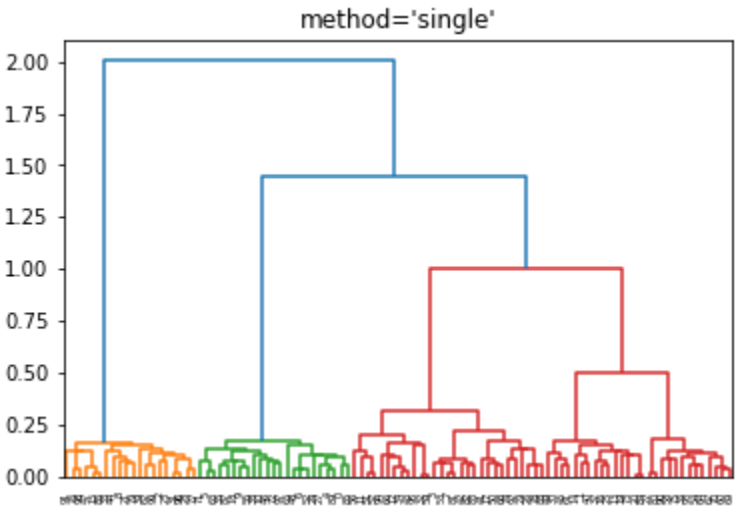
Complete:

```
linkage_data = linkage(data, method='complete' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='complete'")
plt.show()
```

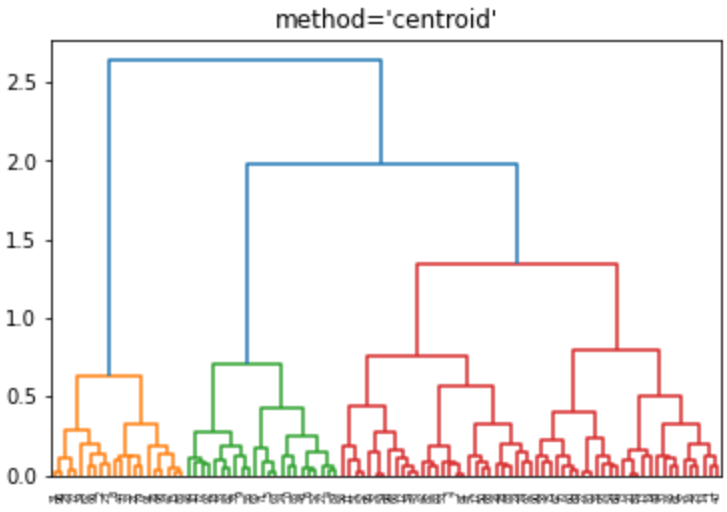


Single:

```
linkage_data = linkage(data, method='single' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='single'")
plt.show()
```



Centroid:



4.2) เลือก cut-off โดยกำหนด criterion='distance' และให้นักศึกษาเลือกระบุค่า t ที่คิดว่า เหมาะสม สำหรับแต่ละ dendrogram ที่ได้ในข้อ 4.1)

Complete: ค่า t ที่เหมาะที่สุดคือ 2

```
cluster_id = fcluster(linkage_data,t=2,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Single: ค่า t ที่เหมาะที่สุดคือ 2

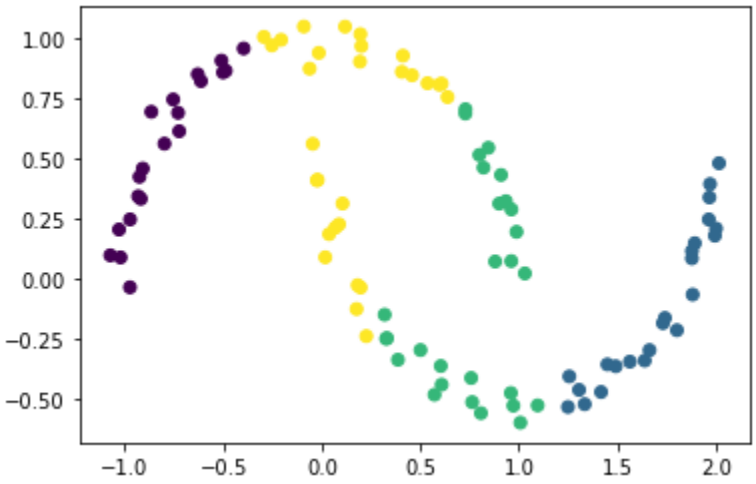
```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Centroid: ค่า t ที่เหมาะที่สุดคือ 0.8

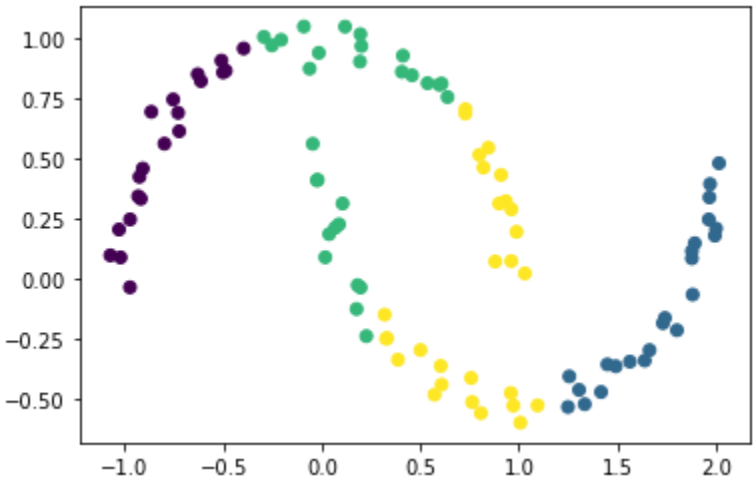
```
cluster_id = fcluster(linkage_data,t=0.8,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

4.3) Plot ผลการจัดกลุ่ม ที่ได้แต่ละแบบในข้อ 4.2)

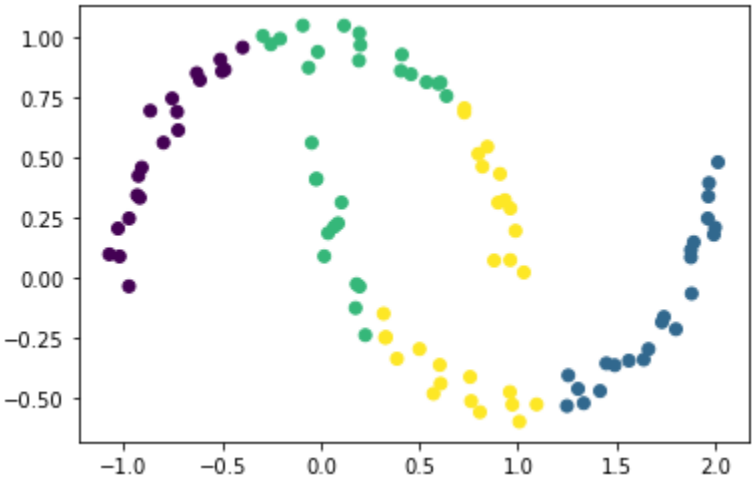
Complete:



Single:



Centroid:



6. เขียนบรรยายสรุปผลการทดลอง แสดงความคิดเห็น วิธีใด เหมาะกับ ชุดข้อมูลแบบไหน แต่ละวิธีมีข้อดี/ ข้อเสียอย่างไร

- Data2DSet2 จากการทดลองพบว่า วิธี Hierarchical Clusterings เพราะ ตัวข้อมูลมีความชิดกันมากจึงไม่สามารถกำหนดค่า K-mean ได้เนื่องจากข้อมูลมีความใกล้กันมาก

เริ่มต้นด้วยการ import

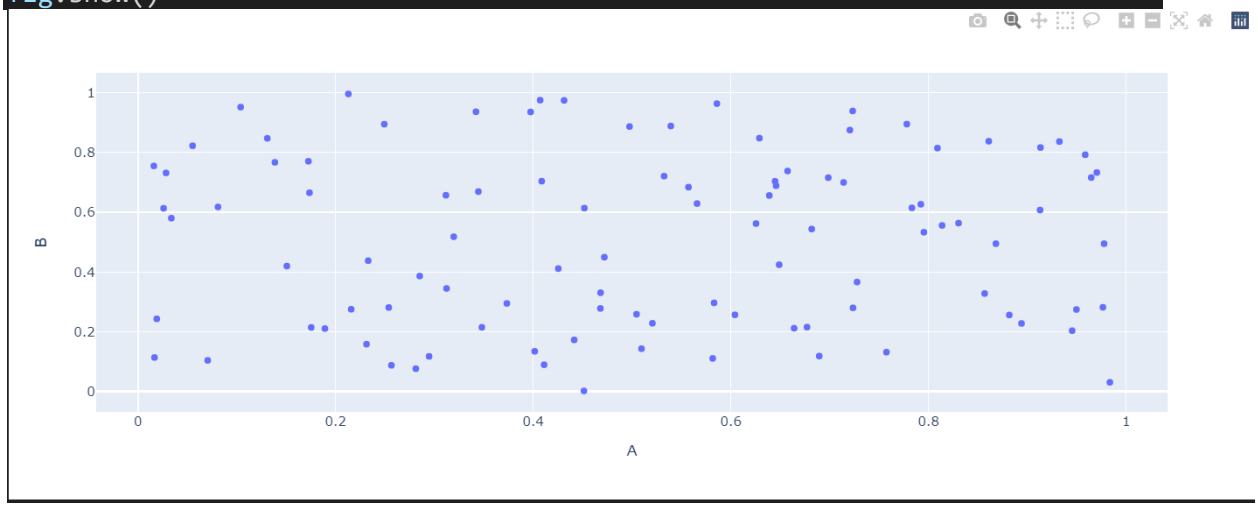
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial import distance_matrix
```

```
data = pd.read_csv("data/data2Dset3.csv",header=None)
data.columns = ["A","B"]
data.head()
```

	A	B
0	0.028405	0.731484
1	0.471940	0.449512
2	0.970259	0.732859
3	0.070531	0.104092
4	0.539139	0.888368

2) Plot จุดข้อมูล data1

```
import plotly.express as px
fig = px.scatter(data, x="A", y="B")
fig.show()
```



3) Data2Dset3 เริ่มด้วยการเขียนโปรแกรม plot จุดข้อมูลโดยใช้วิธี kmeans

K=1

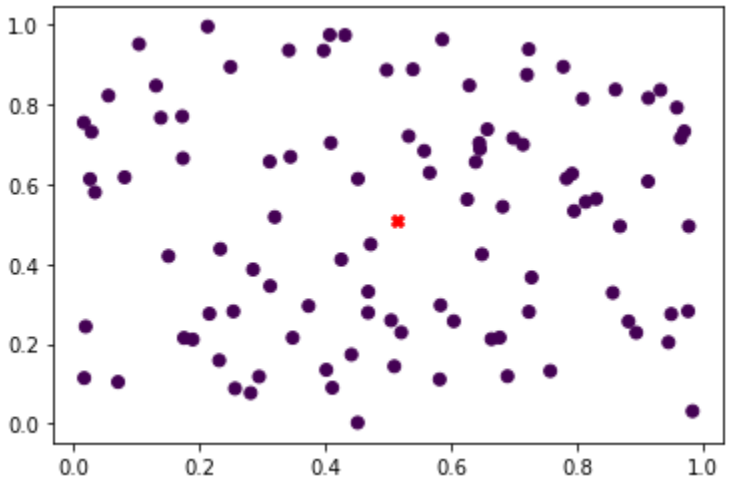
```
model_kmeans = cluster.KMeans(n_clusters=1, max_iter=50, random_state=1)
```

```
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[0.51374147 0.51161133]]
```

```
plt.scatter(data['A'],data['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=1



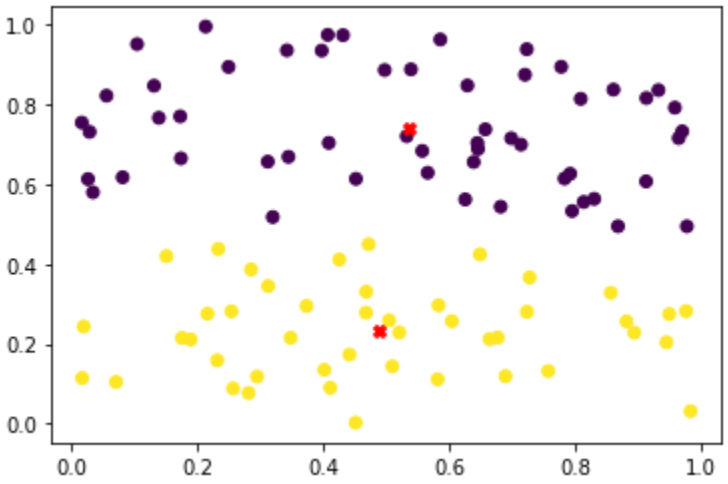
K=2

```
model_kmeans = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1)
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[0.53548368 0.73937577 0.        ]
 [0.48716765 0.23323258 0.        ]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=2



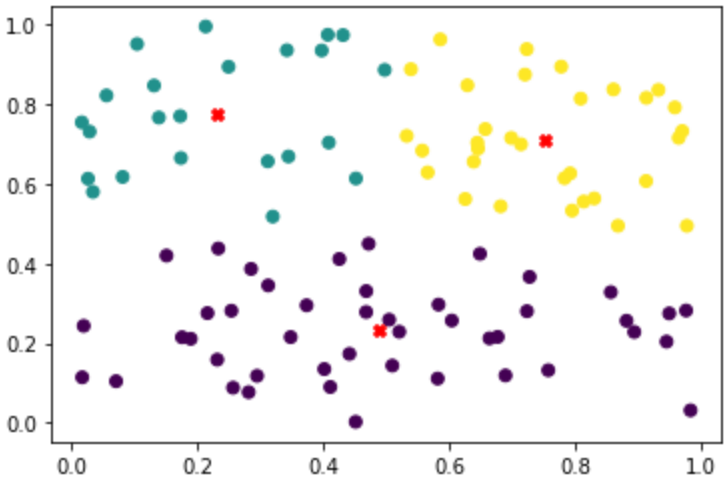
K=3

```
model_kmeans = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[4.87167648e-01 2.33232575e-01 1.00000000e+00]
 [2.31849764e-01 7.77406137e-01 1.11022302e-16]
 [7.53720560e-01 7.12041439e-01 2.77555756e-16]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=3

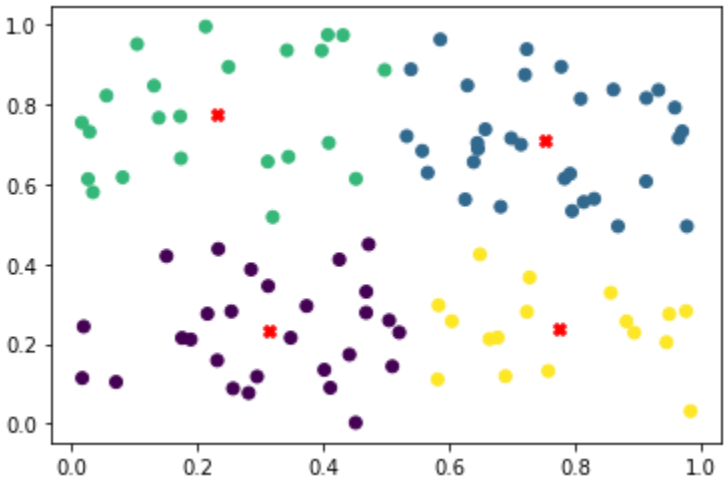


K=4

```
model_kmeans = cluster.KMeans(n_clusters=4, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
 [[ 3.13529957e-01  2.31446910e-01 -1.11022302e-16]
  [ 7.53720560e-01  7.12041439e-01  2.00000000e+00]
  [ 2.31849764e-01  7.77406137e-01  1.00000000e+00]
  [ 7.73159140e-01  2.36173670e-01  3.33066907e-16]]


plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์K=4



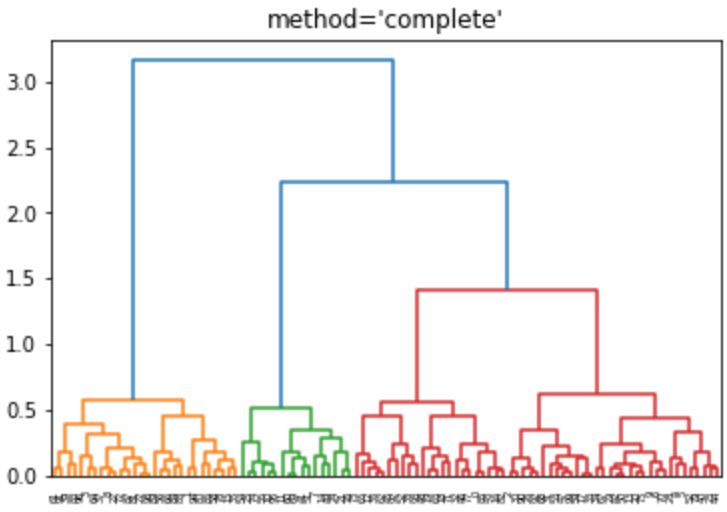
จากผลลัพธ์สรุปได้ว่า K=4 ดีที่สุดสำหรับ Data2Dset2 เพราะข้อมูลจะดูละเอียดอ่อนมากขึ้นถ้าเทียบกับKอื่นๆ ตัวข้อมูลจะถูกแบ่งเป็น 4กลุ่มใหญ่ๆอย่างเห็นได้ชัด

4) เขียนโปแกรมจัดกลุ่มชุดข้อมูลที่อ่านเข้ามา โดยใช้วิธี Hierarchical Clustering

4.1) ให้เลือกใช้method ที่ต่างกัน 3 แบบ แสดง dendrogram ที่ได้แต่ละแบบ

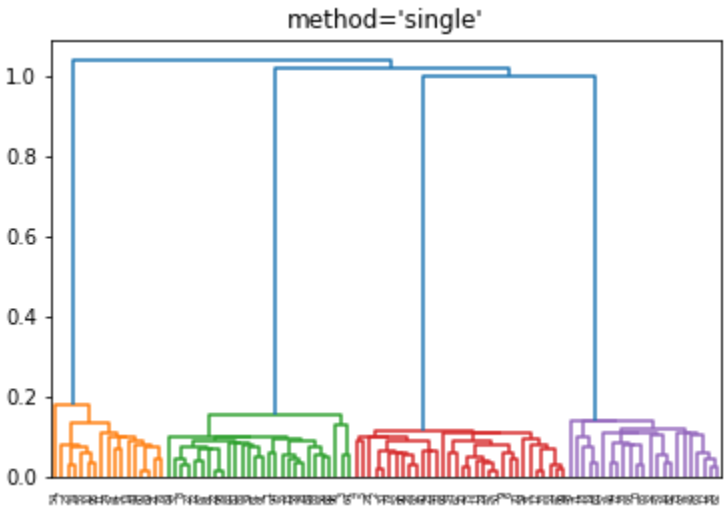
Complete:

```
linkage_data = linkage(data, method='complete' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='complete'")
plt.show()
```



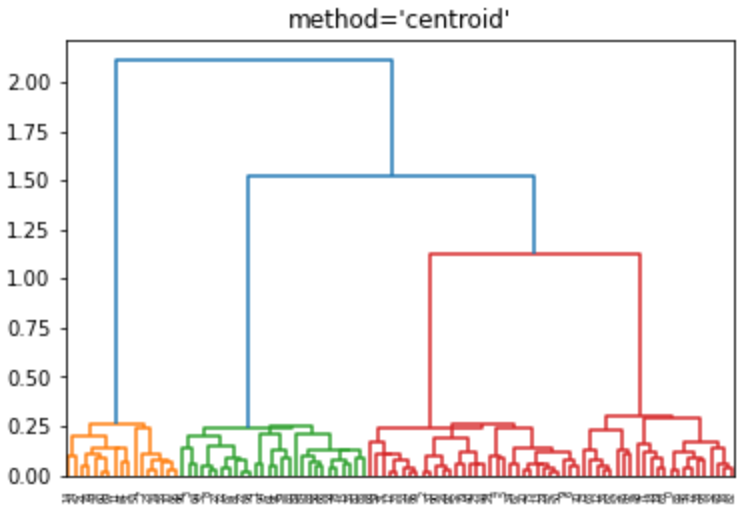
Single:

```
linkage_data = linkage(data, method='single' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='single'")
plt.show()
```



Centroid:

```
linkage_data = linkage(data, method='centroid' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='centroid'")
plt.show()
```



4.2) เลือก cut-off โดยกำหนด criterion='distance' และให้นักศึกษาเลือกระบุค่า t ที่คิดว่า เหมาะสม สำหรับแต่ละ dendrogram ที่ได้ในข้อ 4.1)

Complete: ค่า t ที่เหมาะที่สุดคือ 1

```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Single: ค่า t ที่เหมาะที่สุดคือ 1

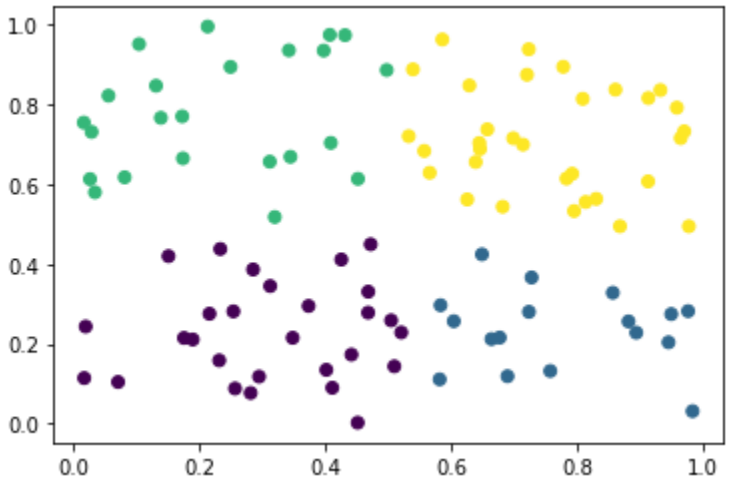
```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Centroid: ค่า t ที่เหมาะที่สุดคือ 1

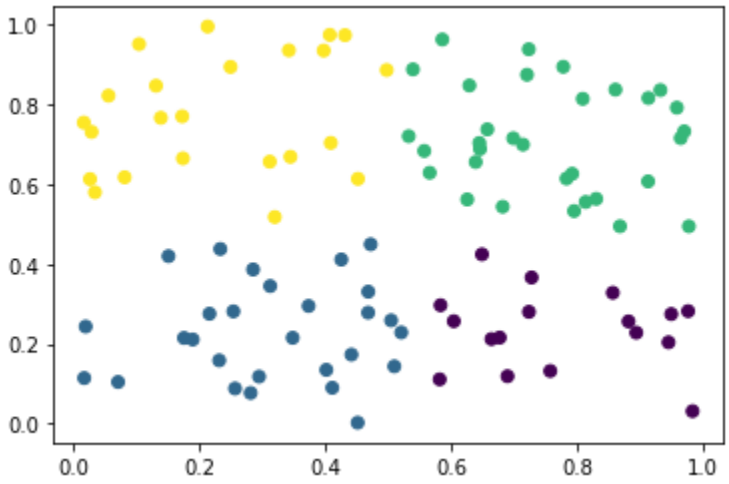
```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

4.3) Plot ผลการจัดกลุ่ม ที่ได้แต่ละแบบในข้อ 4.2)

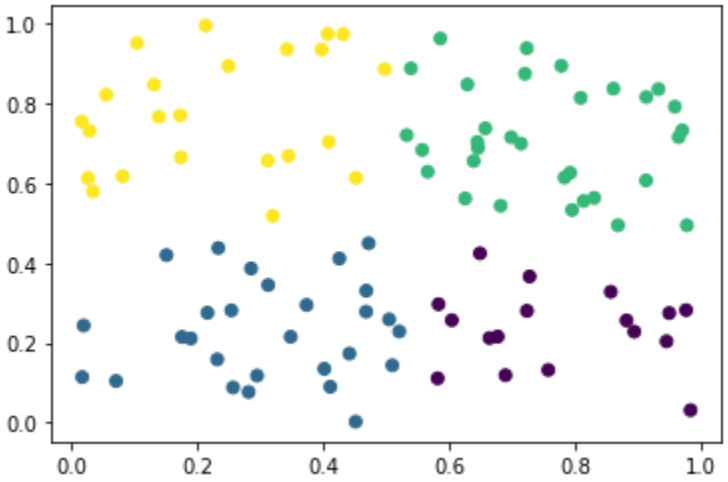
Complete:



Single:



Centroid:



6. เขียนบรรยายสรุปผลการทดลอง แสดงความคิดเห็น วิธีใด เหมาะกับ ชุดข้อมูลแบบไหน แต่ละวิธีมีข้อดี/ ข้อเสีย อย่างไร

- Data2DSet2 จากการทดลองพบว่า วิธี Hierarchical Clusterings เพราะจะมีความละเอียดมากกว่าและจากข้อมูลเราไม่สามารถแยกข้อมูลด้วยตาเปล่าได้ การใช้ kmeans จึงค่อนข้างยาก

เริ่มต้นด้วยการ import

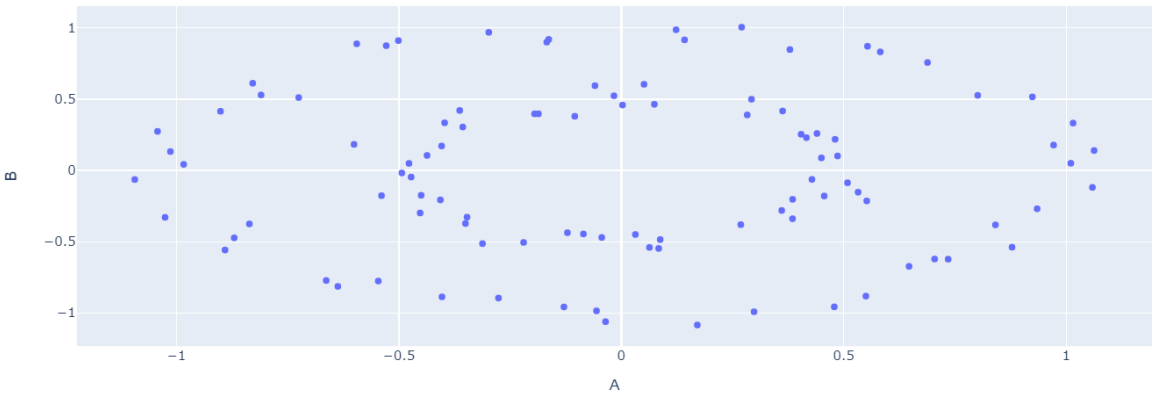
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial import distance_matrix
```

```
data = pd.read_csv("data/data2Dset4.csv",header=None)
data.columns = ["A","B"]
data.head()
```

	A	B
0	-0.016537	0.523408
1	0.971732	0.177843
2	-0.403983	0.171210
3	-0.406715	-0.207174
4	-0.055831	-0.985840

2) Plot จุดข้อมูล data1

```
import plotly.express as px
fig = px.scatter(data, x="A", y="B")
fig.show()
```



3) Data2Dset4 เริ่มด้วยการเขียนโปรแกรม plot จุดข้อมูลโดยใช้วิธี kmeans

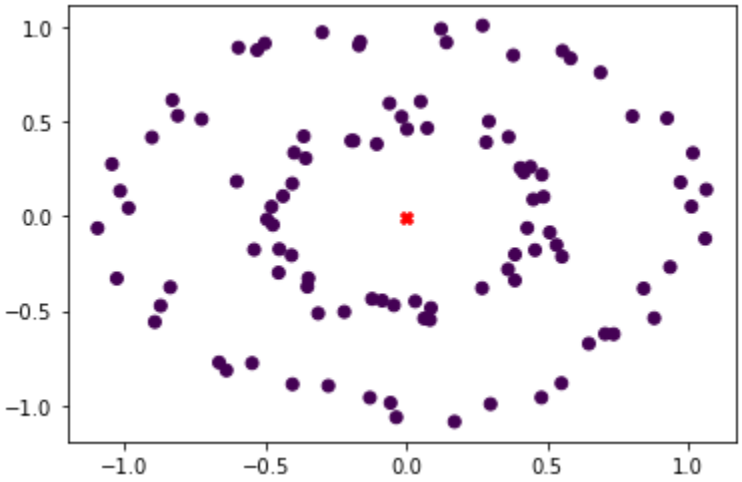
K=1

```
model_kmeans = cluster.KMeans(n_clusters=1, max_iter=50, random_state=1)
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-0.0020635  -0.00248395  0.        ]]
```

```
plt.scatter(data['A'],data['B'], c=data1['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=1



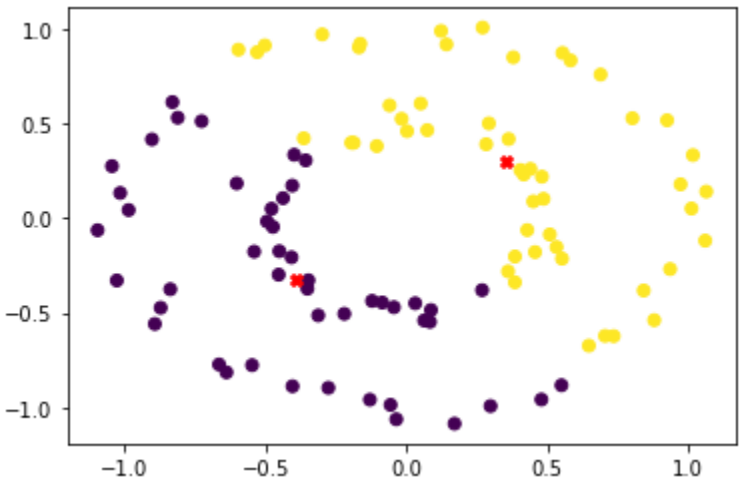
K=2

```
model_kmeans = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1)
model_kmeans.fit(data)
data1['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-0.38961084 -0.3252557   0.        ]
 [ 0.35567251  0.2954592   0.        ]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=2



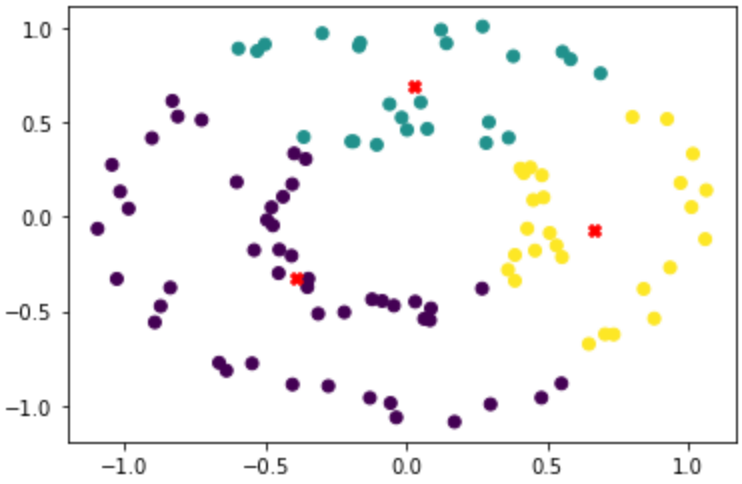
K=3

```
model_kmeans = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-3.89610837e-01 -3.25255700e-01  4.44089210e-16]
 [ 2.50569961e-02  6.88911462e-01  1.00000000e+00]
 [ 6.61797986e-01 -6.88484582e-02  1.00000000e+00]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์ K=3



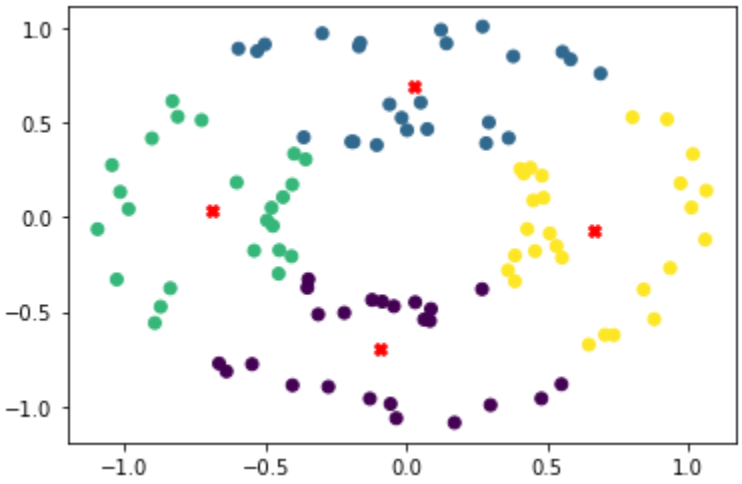
K=4

```
model_kmeans = cluster.KMeans(n_clusters=4, max_iter=50, random_state=1)
model_kmeans.fit(data)
data['cluster_id'] = model_kmeans.labels_
centroids = model_kmeans.cluster_centers_
print(centroids)
```

```
[[-9.13911135e-02 -6.89520810e-01  4.44089210e-16]
 [ 6.61797986e-01 -6.88484582e-02  2.00000000e+00]
 [ 2.50569961e-02  6.88911462e-01  1.00000000e+00]
 [-6.87830561e-01  3.90094112e-02  4.44089210e-16]]
```

```
plt.scatter(data['A'],data['B'], c=data['cluster_id'])
plt.scatter(centroids[:,0],centroids[:,1],marker='X',c='r')
plt.show()
```

ผลลัพธ์K=4



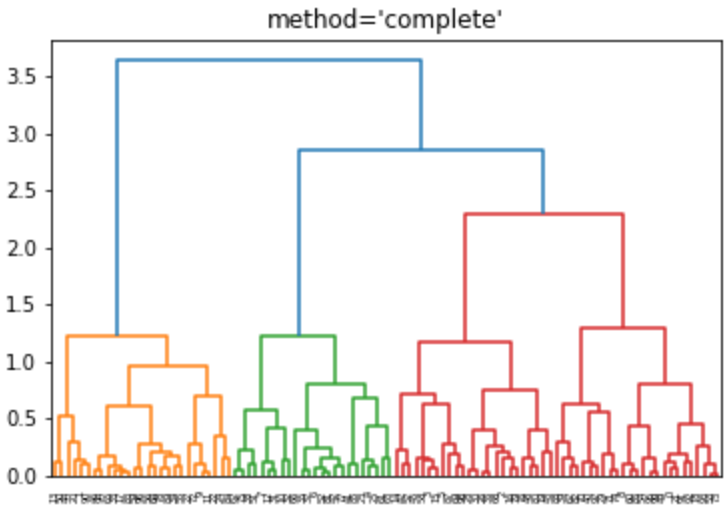
จากผลลัพธ์สรุปได้ว่า K=4 ดีที่สุดสำหรับ Data2Dset2 เพราะข้อมูลจะดูละเอียดอ่อนมากขึ้นถ้าเทียบกับKอื่นๆ ตัวข้อมูลจะถูกแบ่งเป็น 4กลุ่มใหญ่ๆอย่างเห็นได้ชัด

4) เขียนโปแกรมจัดกลุ่มชุดข้อมูลที่อ่านเข้ามา โดยใช้วิธี Hierarchical Clustering

4.1) ให้เลือกใช้method ที่ต่างกัน 3 แบบ แสดง dendrogram ที่ได้แต่ละแบบ

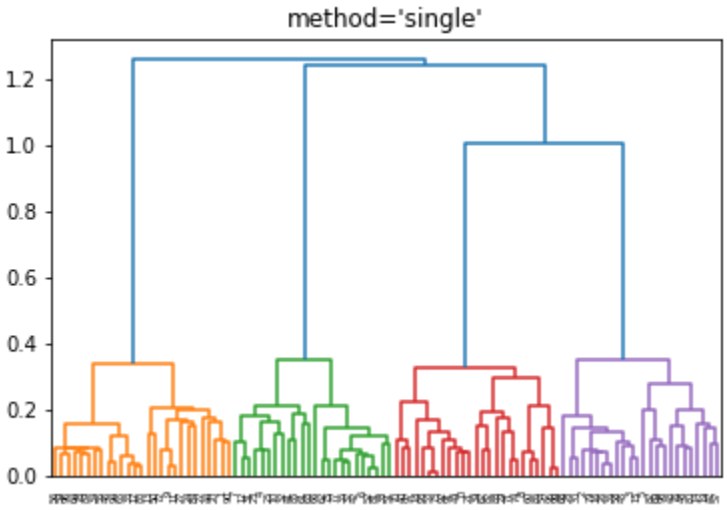
Complete:

```
linkage_data = linkage(data, method='complete' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='complete'")
plt.show()
```



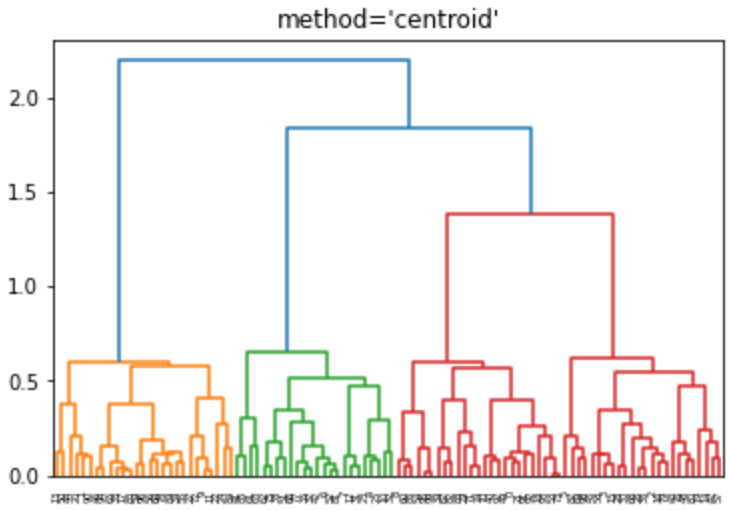
Single:

```
linkage_data = linkage(data, method='single' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='single'")
plt.show()
```



Centroid:

```
linkage_data = linkage(data, method='centroid' , metric='euclidean')
dendrogram(linkage_data)
plt.title("method='centroid'")
plt.show()
```



4.2) เลือก cut-off โดยกำหนด criterion='distance' และให้นักศึกษาเลือกระบุค่า t ที่คิดว่า เหมาะสม สำหรับแต่ละ dendrogram ที่ได้ในข้อ 4.1)

Complete: ค่า t ที่เหมาะที่สุดคือ 1.5

```
cluster_id = fcluster(linkage_data,t=1.5,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Single: ค่า t ที่เหมาะที่สุดคือ 1

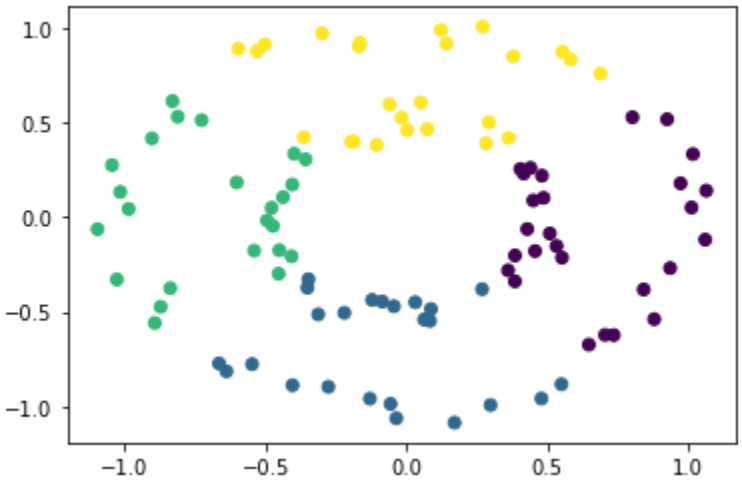
```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

Centroid: ค่า t ที่เหมาะที่สุดคือ 1

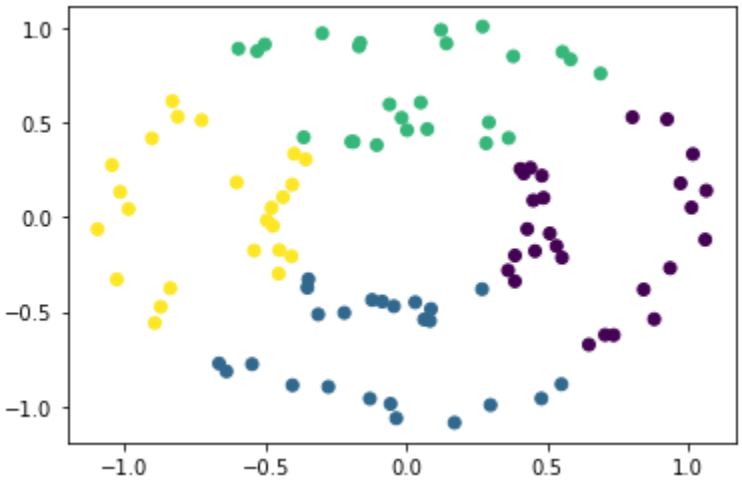
```
cluster_id = fcluster(linkage_data,t=1,criterion='distance')
plt.scatter(data["A"],data["B"],c=cluster_id)
plt.show()
```

4.3) Plot ผลการจัดกลุ่ม ที่ได้แต่ละแบบในข้อ 4.2)

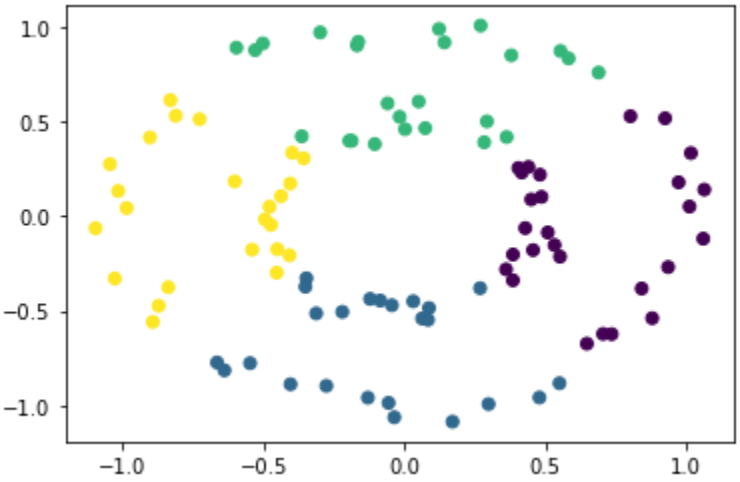
Complete:



Single:



Centroid:



6. เขียนบรรยายสรุปผลการทดลอง แสดงความคิดเห็น วิธีใด เหมาะกับ ชุดข้อมูลแบบไหน แต่ละวิธีมีข้อดี/ ข้อเสีย อย่างไร

- Data2DSet2 จากการทดลองพบว่า วิธี Hierarchical Clusterings เพราะจะมีความละเอียดมากกว่าและจากข้อมูลเราไม่สามารถแยกข้อมูลด้วยตาเปล่าได้ การใช้ kmeans จึงค่อนข้างยาก