In [5]:

```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
```

Cady Stringer and David Aarhus

1. K Means

Once you've chosen random cluster centers to begin with, K-means has two main steps:

1. calculate which center each data point is closest to.
2. use those cluster assignments to recalculate the center of each cluster.

Write two functions: howFar(centers, points), and calculateCenters(points, assignments) which do 1 and 2 respectively. See below for an example of what centers, points and assignments will look like. (assume data points have 2 features, and that you're using euclidean distance, but assume that centers and points could be of variable lengths).

In [98]:

```
#1. calculate which center each data point is closest to.

def howFar (df1, df2):
    dist_list = []
    assignments = []
    #points
    for i in range(len(df2)):
        point_dist_list = []
        point1 = df2.loc[i, : ]
        x1 = point1[0]
        y1 = point1[1]
        cluster_num = i
        centers_dist_list = []
        #centers
        for int in range(len(df1)):
            point2 = df1.loc[int, : ]
            x2 = point2[0]
            y2 = point2[1]
            dist = np.sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - 
y2))
            centers_dist_list.append(dist)
        cluster_assgnment = centers_dist_list.index(np.amin(cent
ers_dist_list))
        assignments.append(cluster_assgnment)
    return assignments
```

In [100]:

```
assignments = howFar(centers1,points1)
print(assignments)
```

```
[2, 2, 2, 0, 1, 0, 2, 1]
```

In [124]:

```
#2. use those cluster assignments to recalculate the center of each cluster.

def calculateCenters(df2, distances):
    distances = np.array(distances)
    unique_list = np.unique(distances)
    centers = []
    for i in unique_list:
        cluster_point_list_x = []
        cluster_point_list_y = []
        for row in range(len(df2)):
             if (distances[row]==i):
                point = df2.loc[row, : ]
                cluster_point_list_x.append(point[0])
                cluster_point_list_y.append(point[1])
        cluster_point_list_x = np.array(cluster_point_list_x)
        cluster_point_list_y = np.array(cluster_point_list_y)
        avg_x = cluster_point_list_x.mean()
        avg_y = cluster_point_list_y.mean()
        centers.append(avg_x)
        centers.append(avg_y)
    return centers
```

In [125]:

```
calculateCenters(points,assignments)
```

Out[125]:

```
[6.475, 13.315, 3.495, 3.575, 10.5225, 4.789999999999999]
```

In [104]:

```
centers = pd.DataFrame([(10,15), (2,3), (7,7)], columns = ["X", "Y"])
points = pd.DataFrame([(9.09,6.93),(10.79,8.76),(9.07,2.25),
                        (7.91,12.59),(3.22,5.61),(5.04,14.04),(13.14,1.22),(3.77,1.54)], columns = ["X", "Y"])
```

In [ ]:

```
# check your answer
howFar(centers,points) == [2, 2, 2, 0, 1, 0, 2, 1]
calculateCenters(points,assignments) == pd.DataFrame([(6.475, 13
.315), (3.495, 3.575), (10.5225, 4.7899999999999999)])
```

3. Hierarchical Agglomeretive Clustering

In Hierarchical Agglomeretive Clustering, we progressively merge clusters together by determining which two clusters are the closest/most similar. We learned about a few different types of linkage criteria. Write three functions: single(cluster1, cluster2), complete(cluster1, cluster2), and average(cluster1, cluster2) that compute and return the distance between two clusters using single, complete, and average linkage respectively. Assume you're using euclidean distance for all 3 functions.

- Single Linkage: smallest distance between two points (one from cluster1, one from cluster2)
- Complete Linkage: largest distance between two points (one from cluster1, one from cluster2)
- Average Linkage: average distance between all pairs of two points (one from cluster1, one from cluster2)

See below for an example of what cluster1 and cluster2 will look like (each row is a data point. but assume they can have any number of data points).

In [9]:

```
cluster1 = pd.DataFrame({"X": [6.3,2.9,1.1,2.3,1.9],
                        "Y": [4.2,-0.9,-2.6,1.5,-1]})
cluster2 = pd.DataFrame({"X": [-1.2,-2.8,1.4,-2,-1.9],
                        "Y": [-1.5,1.9,0.9,-0.3,1.5]})
```

Out[9]:

	X	Y
0	6.3	4.2
1	2.9	-0.9
2	1.1	-2.6
3	2.3	1.5
4	1.9	-1.0

In [39]:

```
def single (df1, df2):
    dist_list = []
    for i in range(len(df1)):
        point1 = df1.loc[i, : ]
        x1 = point1[0]
        y1 = point1[1]
        for int in range(len(df2)):
            point2 = df2.loc[int, : ]
            x2 = point2[0]
            y2 = point2[1]
            dist = np.sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 -
y2))
            dist_list.append(dist)
    dist_list = np.array(dist_list)
    location = np.where(dist_list == np.amin(dist_list))
    best_dist = dist_list[location]
    return best_dist
```

In [40]:

```
a = single(cluster1,cluster2)
print(a)
```

```
[1.08166538]
```

In [41]:

```
def complete (df1, df2):
    dist_list = []
    for i in range(len(df1)):
        point1 = df1.loc[i, : ]
        x1 = point1[0]
        y1 = point1[1]
        for int in range(len(df2)):
            point2 = df2.loc[int, : ]
            x2 = point2[0]
            y2 = point2[1]
            dist = np.sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - 
y2))
            dist_list.append(dist)
    dist_list = np.array(dist_list)
    location = np.where(dist_list == np.amax(dist_list))
    best_dist = dist_list[location]
    return best_dist
```

In [42]:

```
b = complete(cluster1,cluster2)
print(b)
```

```
[9.4413982]
```

In [49]:

```
def average (df1, df2):
    dist_list = []
    for i in range(len(df1)):
        point1 = df1.loc[i, : ]
        x1 = point1[0]
        y1 = point1[1]
        for int in range(len(df2)):
            point2 = df2.loc[int, : ]
            x2 = point2[0]
            y2 = point2[1]
            dist = np.sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 -
y2))
            dist_list.append(dist)
    dist_list = np.array(dist_list)
    best_dist = dist_list.mean()
    return best_dist
```

In [50]:

```
c = average(cluster1,cluster2)
print(c)
```

```
5.027741968109063
```