initializing the libraries

Code

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



matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (12, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
df = pd.read_csv('dwm_salesdataset.csv')
dataset = df[['noofstudentsenrolled', 'discount']]


def init_centroids(k,dataset):
    centroids = []
    for i in range(0,k):
     point = []
     for col in dataset.columns:
        point.append(random.uniform(min(dataset[col]),max(dataset[col])))
     centroids.append(point)
    return centroids


def calcdist(dataset,cluster):
 dist = 0
 for idx in range(len(dataset.columns)-1):
  dist += (dataset[dataset.columns[idx]]-cluster[idx])**2
  dist = dist**(1/2)
 return dist


def kmeans(k,dataset):
  centroids = init_centroids(k,dataset)
  dataset['Cluster'] = 0
  original = dataset['Cluster']
  while True:
   dist = pd.Series([math.inf] * len(dataset))
   for idx in range(len(centroids)):
     point = centroids[idx]
     dataset.loc[calcdist(dataset,point)<=dist,['Cluster']] = idx
     dist = pd.concat([dist, calcdist(dataset,point)], axis=1).min(axis=1)
   for idx in range(len(centroids)):
    centroids[idx] = list(dataset[dataset['Cluster']==idx][dataset.columns[0:-1]].mean(axi
    if dataset['Cluster'].eq(original, axis=0).all():
     return dataset,centroids
    else:
```

```
        original = dataset['Cluster']
        ct = init_centroids(k,dataset)
```

```
k = int(input('Enter number of clusters : '))
result,centroids = kmeans(k,dataset);
result.head(20)
```

```
Enter number of clusters : 4
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
result.Cluster.unique()
```

```
array([3, 2, 0])
```

```
  self._setitem_single_block(indexer, value, name)
```

```
def getX(lst):
  return [item[0] for item in lst]
```

```
  iloc._setitem_with_indexer(indexer, value, self.name)
```

```
def getY(lst):
    return [item[1] for item in lst]


color = ['#F9C74F', '#90BE6D', '#43AA8B', '#577590',
'#6D597A','#003F88', '#F94144', '#F3722C', '#F8961E', '#FDC500']
```

Output

```
for i in range(k):
  if(i < 2):
    plt.scatter(result[result['Cluster'] == i][result.columns[0]],
    result[result['Cluster'] == i][result.columns[1]], s = 100, c = color[i], label = f'Cl
  elif(k > i):
    plt.scatter(result[result['Cluster'] == i][result.columns[0]],
    result[result['Cluster'] == i][result.columns[1]], s = 100, c = color[i], label =f'Clu
plt.scatter(getX(centroids), getY(centroids), s = 300, c = 'yellow', label ='Centroids')
plt.title('Graph')
plt.xlabel('No of students enrolled ')
plt.ylabel('discount')
plt.legend()
plt.show()
```