```
 1 import pandas as pd
 2 from matplotlib import pyplot as plt
 3
 4 from sklearn.preprocessing import StandardScaler
 5 from sklearn.cluster import KMeans
 6
 7
 8 def readData():
 9     df = pd.read_csv('data3/data_assignment3.csv')
10     return df
11
12
13 def normalizeData(table):
14     table[['nPosition', 'Nphi', 'Npsi']] =
   StandardScaler().fit_transform(
15         table[['position', 'phi', 'psi']])
16     return table
17
18
19 def create2DHistogram(x_values, y_values):
20     plt.hist2d(x_values, y_values, bins=350, alpha=
   1, cmap='plasma')
21
22
23 def createPandaXYValues(table, colum1, colum2):
24     return table[colum1], table[colum2]
25
26
27 def initPlot():
28     plt.xlim(-180, 180)
29     plt.ylim(-180, 180)
30     plt.ylabel("Psi")
31     plt.xlabel("Phi")
32     plt.show()
33
34
35 def regularScatterPLot(x_values, y_values):
36     plt.scatter(x_values, y_values, marker="o", s=1
   )
37
38     if __name__ == '__main__':
```

```
39          x, y = createPandaXYValues(readData(), "phi
   ", "psi")
40
41      # print(readData())
42      # print(readData().describe())
43      # print(normalizeData(readData()))
44      create2DHistogram(x, y)
45      # regularScatterPLot(x, y)
46      initPlot()
47
48
```

```
 1 import pandas as pd
 2 from matplotlib import pyplot as plt
 3 from sklearn.preprocessing import StandardScaler
 4 from sklearn.cluster import KMeans
 5 import numpy as np
 6
 7 def readData():
 8     df = pd.read_csv('data3/data_assignment3.csv',
   usecols=['phi', 'psi'])
 9     return df
10
11 def normalizeData(table):
12     table[['Nphi', 'Npsi']] = StandardScaler().
   fit_transform(table[['phi', 'psi']])
13     return table
14
15 def create2DHistogram(x_values, y_values):
16     plt.hist2d(x_values, y_values, bins=350, alpha=
   1, cmap='plasma')
17
18 def createPandaXYValues(table, colum1, colum2):
19     return table[colum1], table[colum2]
20
21 def initPlot():
22     plt.xlim(-180, 180)
23     plt.ylim(-180, 180)
24     plt.ylabel("Psi")
25     plt.xlabel("Phi")
26     plt.show()
27
28 def regularScatterPLot(x_values, y_values):
29     plt.scatter(x_values, y_values, marker="o", s=1
   )
30
31
32 def elbowMethod(data):
33     #Creating empty list to store Within-Cluster
   Sum of Squares (WCSS) values
34     WCSS = []
35
36     #Creating for loop that runs kmeans with
```

```
36 different numbers of clusters 1-11
37     for i in range(1, 11):
38         kmeans = KMeans(n_clusters=i, init='k-means
   ++')
39         kmeans.fit(data)
40         #Appending the inertia value to the WCSS
   list
41         WCSS.append(kmeans.inertia_)
42 
43     #Plotting the elbow graph
44     plt.plot(range(1, 11), WCSS)
45     plt.title('The elbow method')
46     plt.xlabel('Number of clusters')
47     plt.ylabel('Within-Cluster Sum of Squares (WCSS
   )')
48     plt.show()
49 
50 if __name__ == '__main__':
51     x, y = createPandaXYValues(readData(), "phi", "
   psi")
52     # print(readData())
53     # print(readData().describe())
54     # print(normalizeData(readData()))
55     create2DHistogram(x, y)
56     # regularScatterPLot(x, y)
57     initPlot()
58     elbowMethod(readData())
```

```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
import seaborn as sns
from collections import Counter


def readData():
    df = pd.read_csv('data3/data_assignment3.csv')
    df.to_numpy()
    return df


def createPandaXYValues(table, colum1, colum2):
    return table[colum1], table[colum2]


def createPandaNPArray(dataFrame, column1, column2
):
    return dataFrame[[column1, column2]].to_numpy()


def createDBSCAN(epsilon, minsamples, dataFramed):
    return DBSCAN(eps=epsilon, min_samples=
minsamples).fit(dataFramed)


def plotINIT():
    plt.xlim(-180, 180)
    plt.ylim(-180, 180)
    plt.xlabel("Phi")
    plt.ylabel("Psi")
    plt.title("PRO")
    plt.show()
def calculateOptimizedEpsilon(dataset, neighbours):
    neighbors = NearestNeighbors(n_neighbors=
neighbours)
    neighbors_fit = neighbors.fit(dataset)
    distances, indices = neighbors_fit.kneighbors(
```

```
38 dataset)
39
40     distances = np.sort(distances, axis=0)
41     distances = distances[:, 1]
42     # plt.plot(distances)
43
44
45 def plot_outliers(data):
46     data['acids'] = list(dbscan.labels_)
47     outliers = data[data['acids'] == -1]
48     outliers['residue name'].value_counts().
   sort_values().plot.bar()
49
50
51
52 if __name__ == '__main__':
53     X = readData()
54     X = X.loc[X["residue name"] == "PRO"]
55     X = createPandaNPArray(X, "phi", "psi")
56
57     dbscan = createDBSCAN(13, 110, X)
58     snsPlot = sns.scatterplot(x=X[:, 0], y=X[:, 1
   ], hue= dbscan.labels_, legend="full", palette="
   deep")
59     sns.move_legend(snsPlot, "upper right",
   bbox_to_anchor=(1.13, 1.15), title='Clusters')
60     plt.show()
61
```