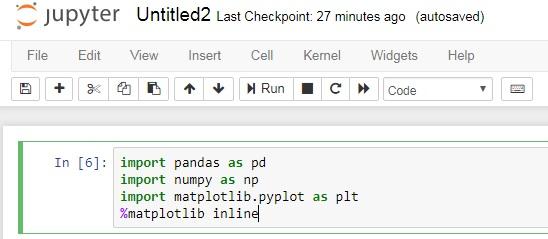
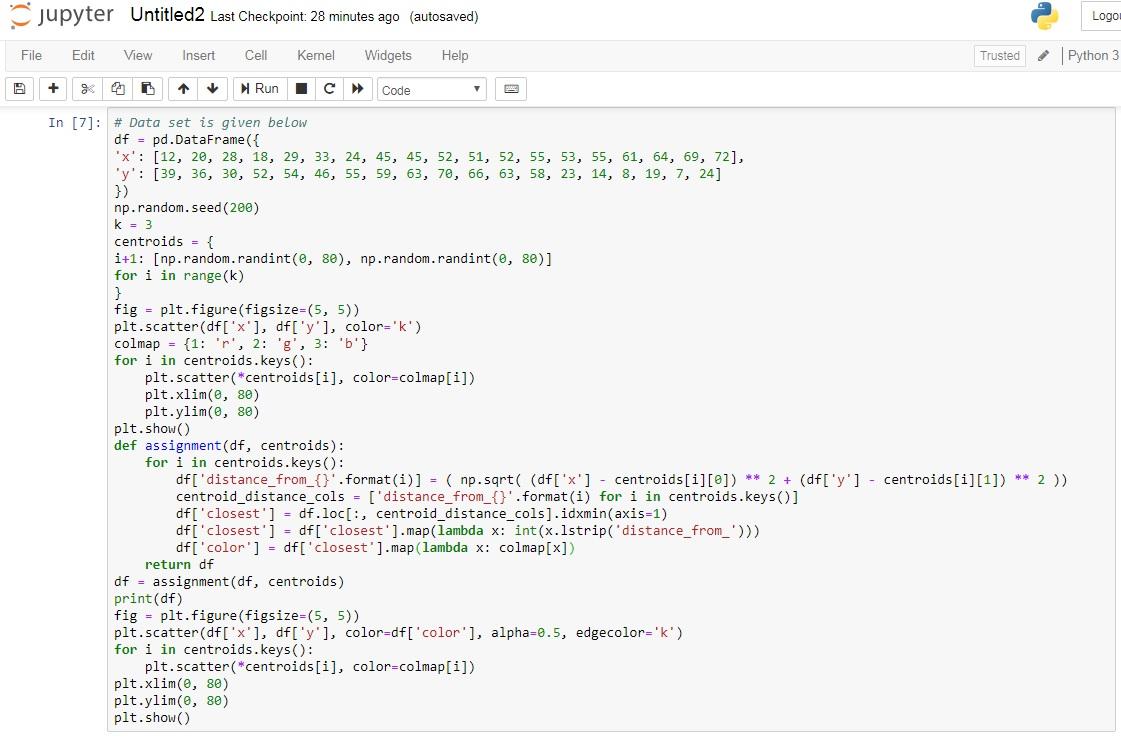
FOLLOW THE INDENTATIONS:-

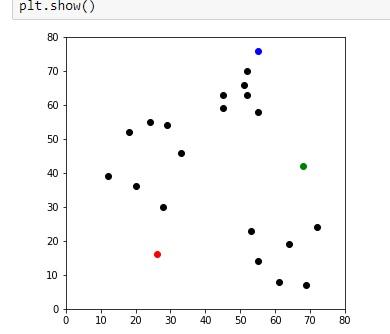
**In[1]:**

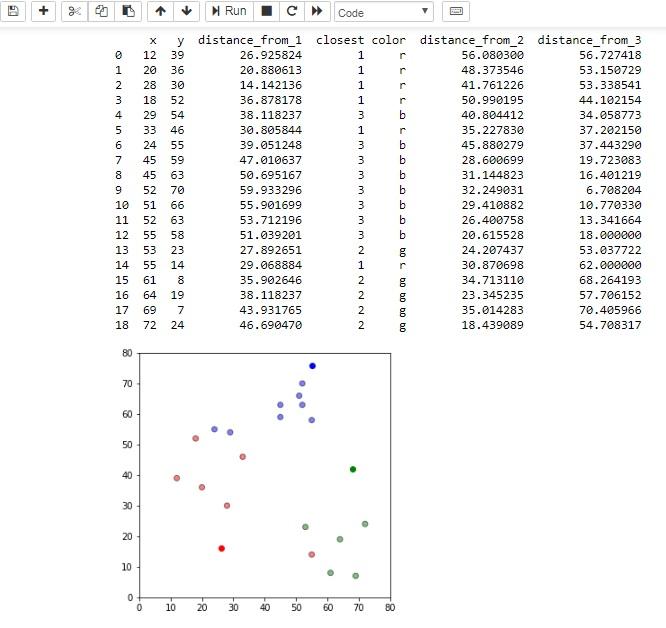


**In[2]:**

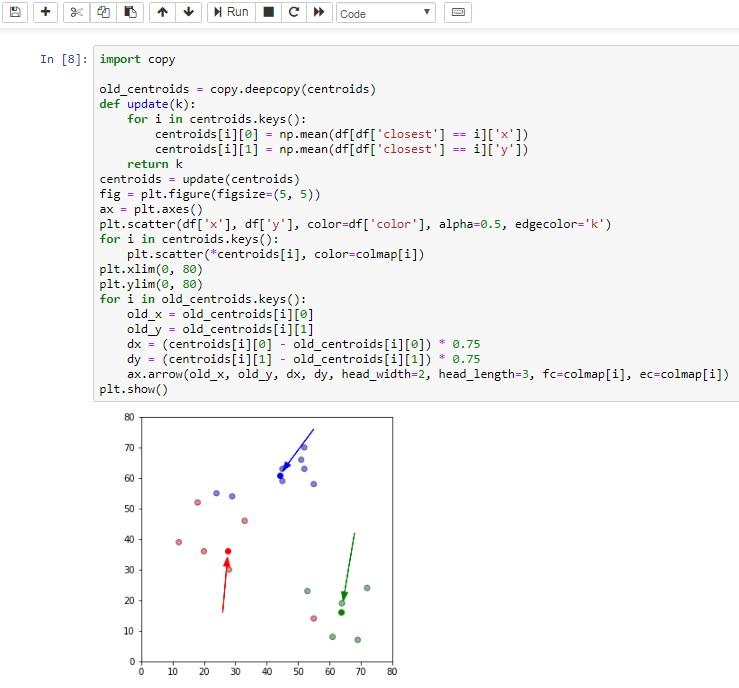


**OUTPUT:**

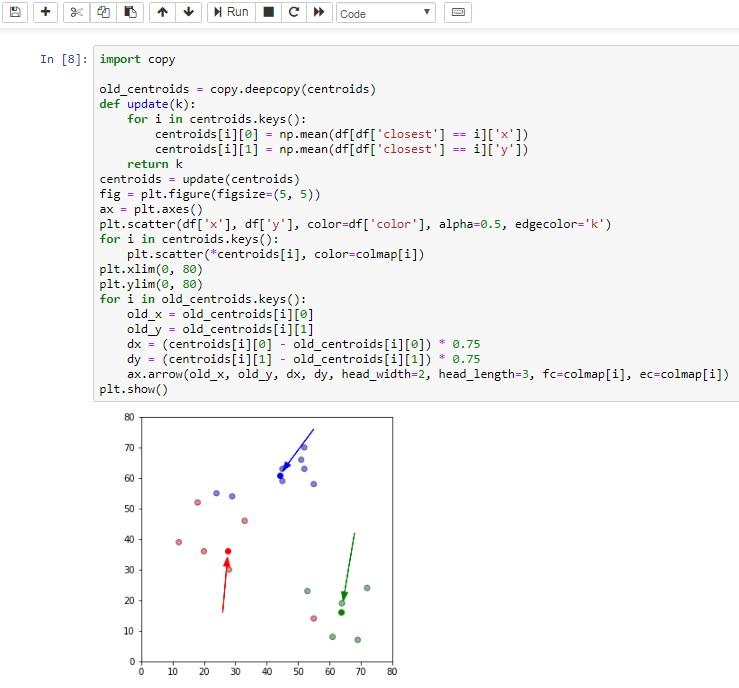
****



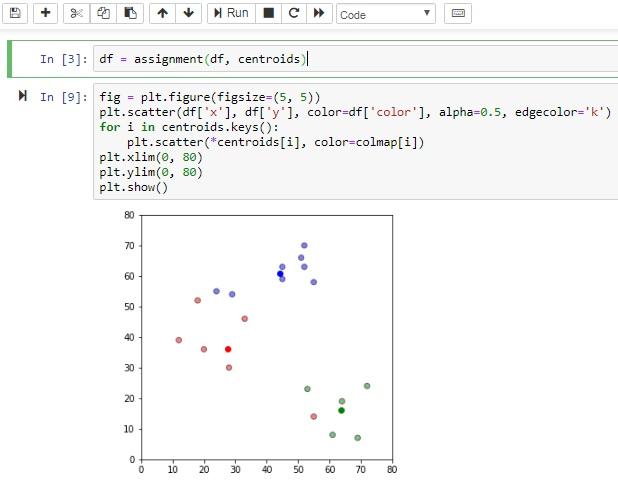
In[3]:



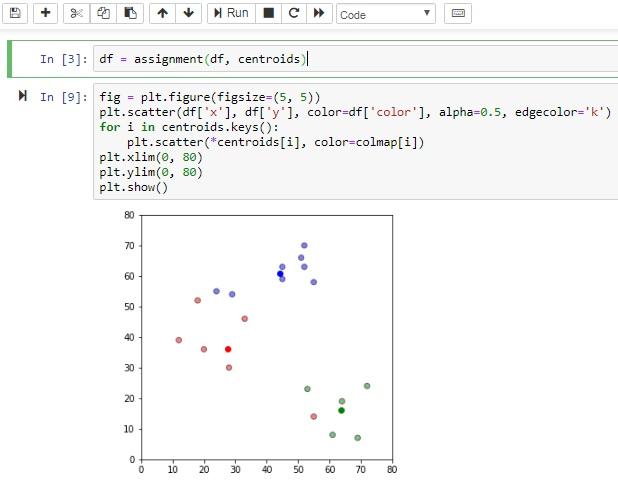
**OUTPUT:**

****

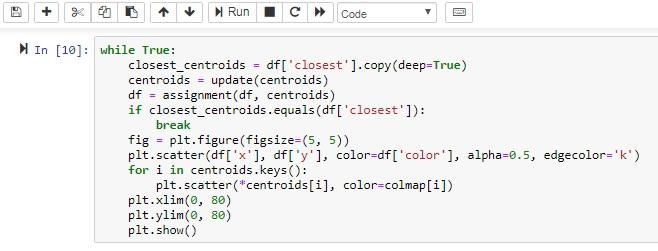
**In[4]: & In[5]:**



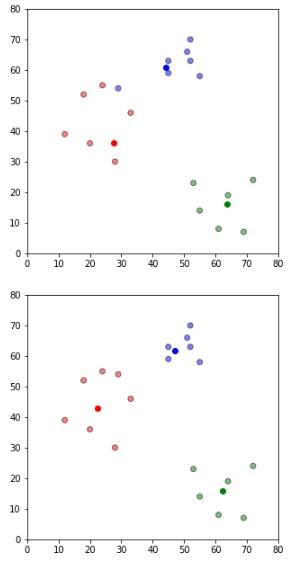
**OUTPUT:**

****

**Ln[6]:**

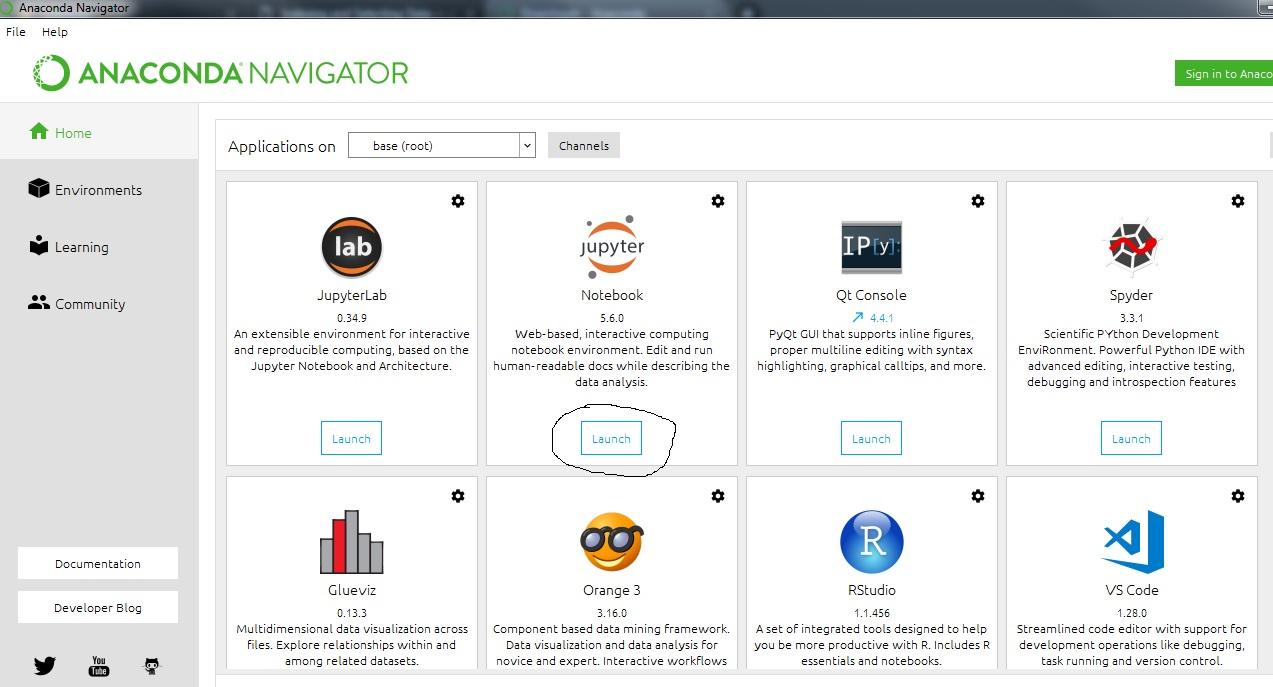
****

**OUTPUT:**

****

Download anaconda from - <https://www.anaconda.com/download/> 64bit/32bit select Python 3.7 version

Launch anaconda from start menu then click on jupyter notebook -> Launch.



Chrome/Mozilla will launch->Click on New Python 3.



|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  %matplotlib inline |
|  |
| df = pd.DataFrame({  'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],  'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]  })  np.random.seed(200)  k = 3  centroids = {  i+1: [np.random.randint(0, 80), np.random.randint(0, 80)]  for i in range(k)  }  fig = plt.figure(figsize=(5, 5))  plt.scatter(df['x'], df['y'], color='k')  colmap = {1: 'r', 2: 'g', 3: 'b'}  for i in centroids.keys():  plt.scatter(\*centroids[i], color=colmap[i])  plt.xlim(0, 80)  plt.ylim(0, 80)  plt.show()  def assignment(df, centroids):  for i in centroids.keys():  df['distance\_from\_{}'.format(i)] = ( np.sqrt( (df['x'] - centroids[i][0]) \*\* 2 + (df['y'] - centroids[i][1]) \*\* 2 ))  centroid\_distance\_cols = ['distance\_from\_{}'.format(i) for i in centroids.keys()]  df['closest'] = df.loc[:, centroid\_distance\_cols].idxmin(axis=1)  df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance\_from\_')))  df['color'] = df['closest'].map(lambda x: colmap[x])  return df  df = assignment(df, centroids)  print(df)  fig = plt.figure(figsize=(5, 5))  plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')  for i in centroids.keys():  plt.scatter(\*centroids[i], color=colmap[i])  plt.xlim(0, 80)  plt.ylim(0, 80)  plt.show() |

|  |
| --- |
| import copy  old\_centroids = copy.deepcopy(centroids)  def update(k):  for i in centroids.keys():  centroids[i][0] = np.mean(df[df['closest'] == i]['x'])  centroids[i][1] = np.mean(df[df['closest'] == i]['y'])  return k  centroids = update(centroids)  fig = plt.figure(figsize=(5, 5))  ax = plt.axes()  plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')  for i in centroids.keys():  plt.scatter(\*centroids[i], color=colmap[i])  plt.xlim(0, 80)  plt.ylim(0, 80)  for i in old\_centroids.keys():  old\_x = old\_centroids[i][0]  old\_y = old\_centroids[i][1]  dx = (centroids[i][0] - old\_centroids[i][0]) \* 0.75  dy = (centroids[i][1] - old\_centroids[i][1]) \* 0.75  ax.arrow(old\_x, old\_y, dx, dy, head\_width=2, head\_length=3, fc=colmap[i], ec=colmap[i])  plt.show() |
|  |
| df = assignment(df, centroids) |
|  |
| fig = plt.figure(figsize=(5, 5))  plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')  for i in centroids.keys():  plt.scatter(\*centroids[i], color=colmap[i])  plt.xlim(0, 80)  plt.ylim(0, 80)  plt.show() |
|  |
| while True:  closest\_centroids = df['closest'].copy(deep=True)  centroids = update(centroids)  df = assignment(df, centroids)  if closest\_centroids.equals(df['closest']):  break  fig = plt.figure(figsize=(5, 5))  plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')  for i in centroids.keys():  plt.scatter(\*centroids[i], color=colmap[i])  plt.xlim(0, 80)  plt.ylim(0, 80)  plt.show() |