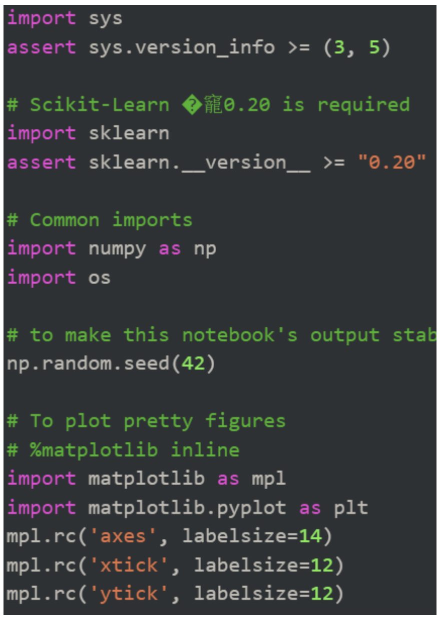
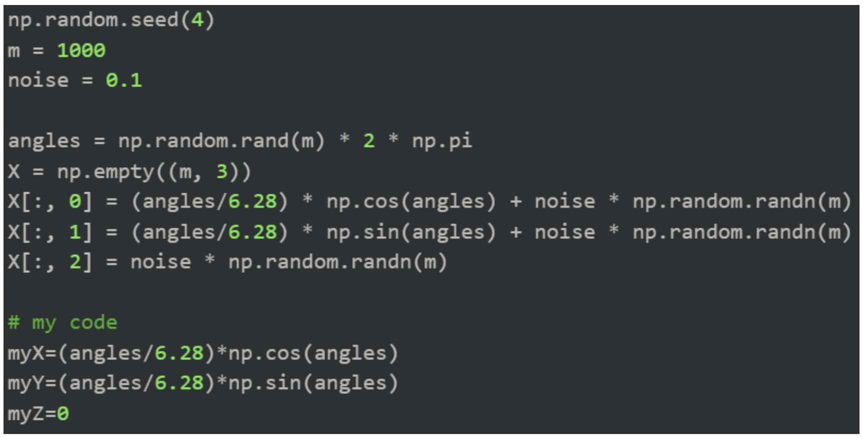
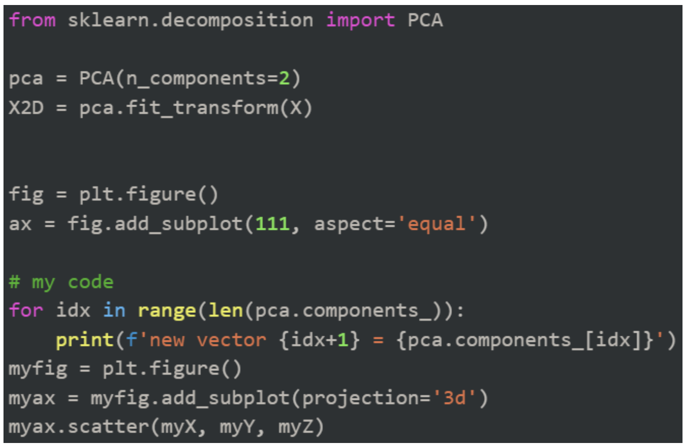
1a.



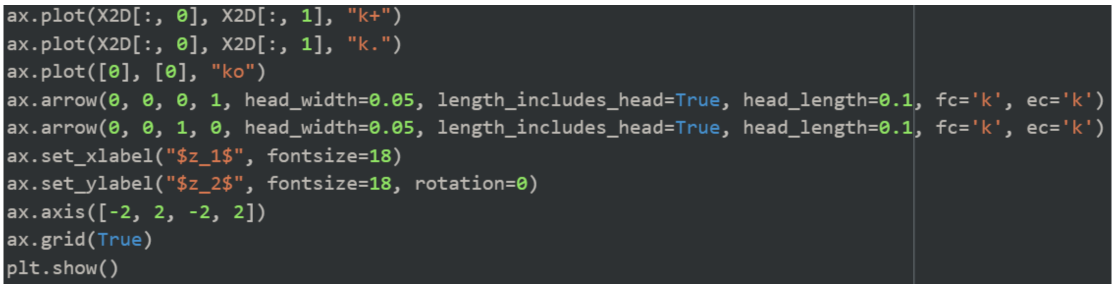
```
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn �寵0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stab
np.random.seed(42)

# To plot pretty figures
# %matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
```



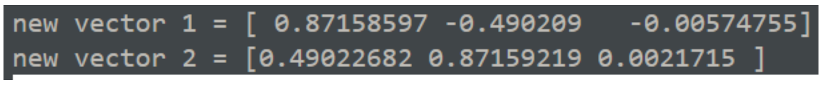
```
np.random.seed(4)
m = 1000
noise = 0.1

angles = np.random.rand(m) * 2 * np.pi
X = np.empty((m, 3))
X[:, 0] = (angles/6.28) * np.cos(angles) + noise * np.random.randn(m)
X[:, 1] = (angles/6.28) * np.sin(angles) + noise * np.random.randn(m)
X[:, 2] = noise * np.random.randn(m)

# my code
myX=(angles/6.28)*np.cos(angles)
myY=(angles/6.28)*np.sin(angles)
myZ=0
```



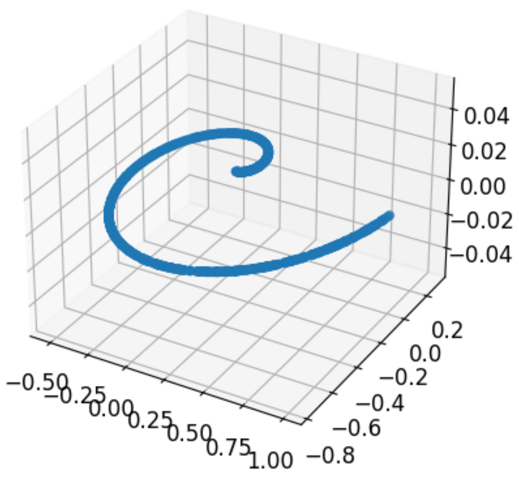
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X2D = pca.fit_transform(X)


fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal')

# my code
for idx in range(len(pca.components_)):
    print(f'new vector {idx+1} = {pca.components_[idx]}')
myfig = plt.figure()
myax = myfig.add_subplot(projection='3d')
myax.scatter(myX, myY, myZ)
```

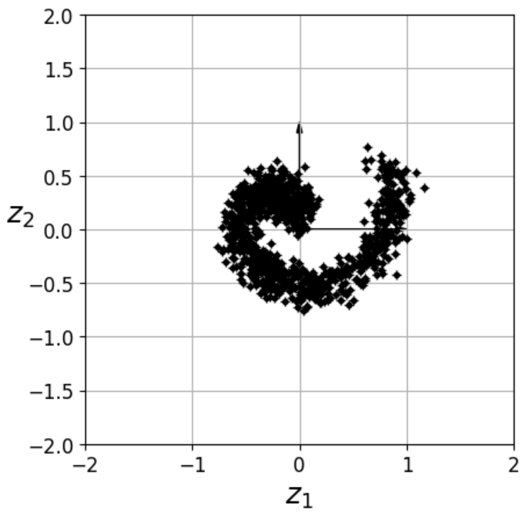


```
ax.plot(X2D[:, 0], X2D[:, 1], "k+")
ax.plot(X2D[:, 0], X2D[:, 1], "k.")
ax.plot([0], [0], "ko")
ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.set_xlabel("$z_1$", fontsize=18)
ax.set_ylabel("$z_2$", fontsize=18, rotation=0)
ax.axis([-2, 2, -2, 2])
ax.grid(True)
plt.show()
```

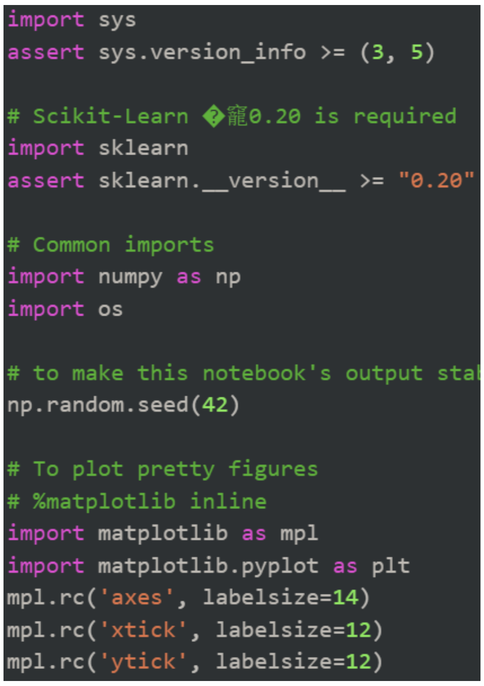


```
new vector 1 = [ 0.87158597 -0.490209   -0.00574755]
new vector 2 = [0.49022682 0.87159219 0.0021715 ]
```





1b.



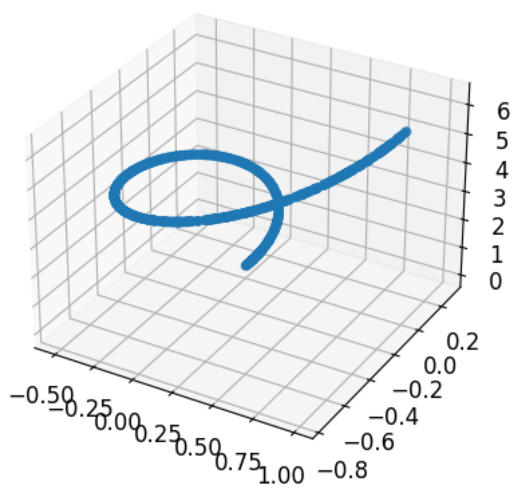
```
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn �寵0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# to make this notebook's output stat
np.random.seed(42)

# To plot pretty figures
# %matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
```

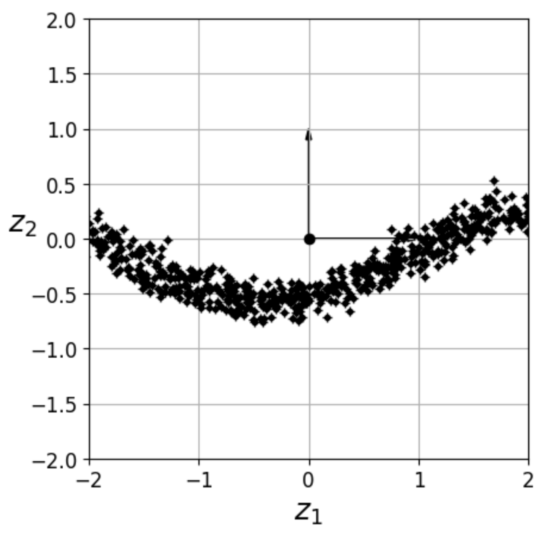
```python
np.random.seed(4)
m = 1000
noise = 0.1

angles = np.random.rand(m) * 2 * np.pi
X = np.empty((m, 3))
X[:, 0] = (angles/6.28) * np.cos(angles) + noise * np.random.randn(m)
X[:, 1] = (angles/6.28) * np.sin(angles) + noise * np.random.randn(m)
X[:, 2] = angles + noise * np.random.randn(m)

# my code
myX=(angles/6.28)*np.cos(angles)
myY=(angles/6.28)*np.sin(angles)
myZ=angles
```

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X2D = pca.fit_transform(X)


fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal')

# my code
for idx in range(len(pca.components_)):
    print(f'new vector {idx+1} = {pca.components_[idx]}')
myfig = plt.figure()
myax = myfig.add_subplot(projection='3d')
myax.scatter(myX, myY, myZ)
```

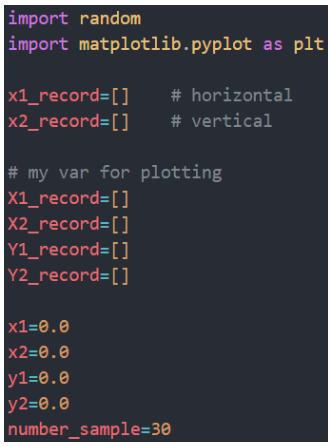
```python
ax.plot(X2D[:, 0], X2D[:, 1], "k+")
ax.plot(X2D[:, 0], X2D[:, 1], "k.")
ax.plot([0], [0], "ko")
ax.arrow(0, 0, 0, 1, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.arrow(0, 0, 1, 0, head_width=0.05, length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.set_xlabel("$z_1$", fontsize=18)
ax.set_ylabel("$z_2$", fontsize=18, rotation=0)
ax.axis([-2, 2, -2, 2])
ax.grid(True)
plt.show()
```

```
new vector 1 = [-0.10392065  0.15097154 -0.98306057]
new vector 2 = [ 0.98988417  0.11169138 -0.08748921]
```

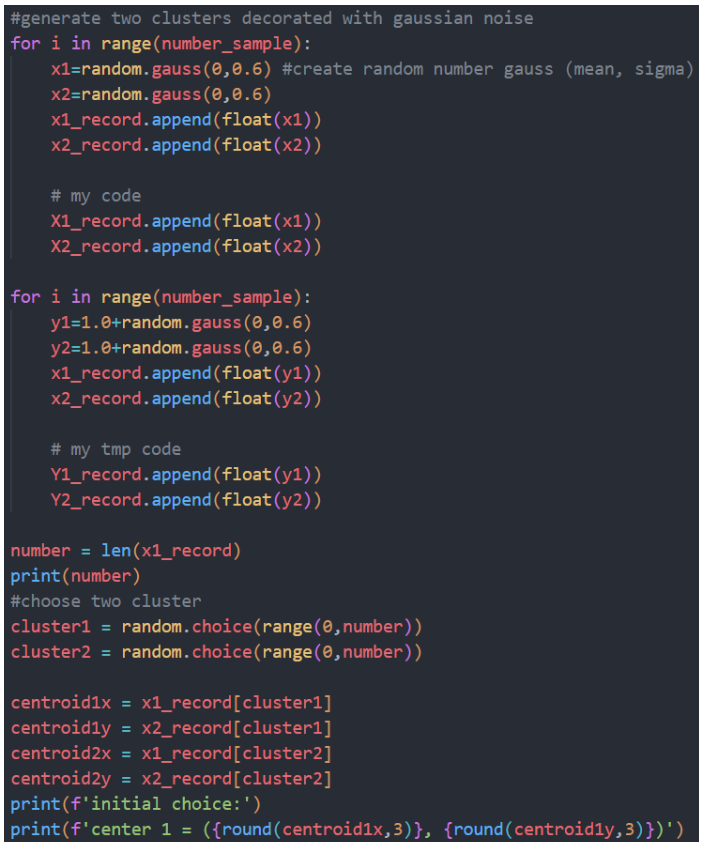




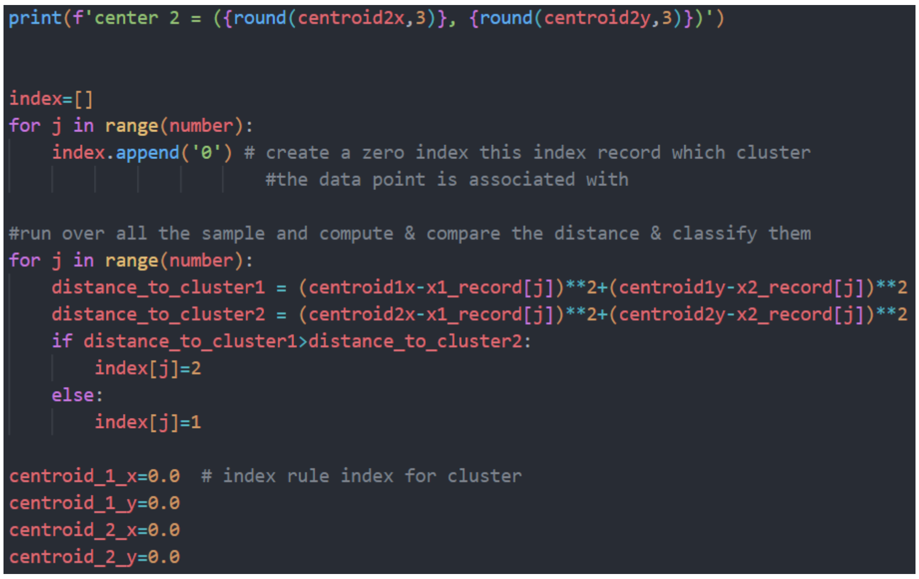
2.



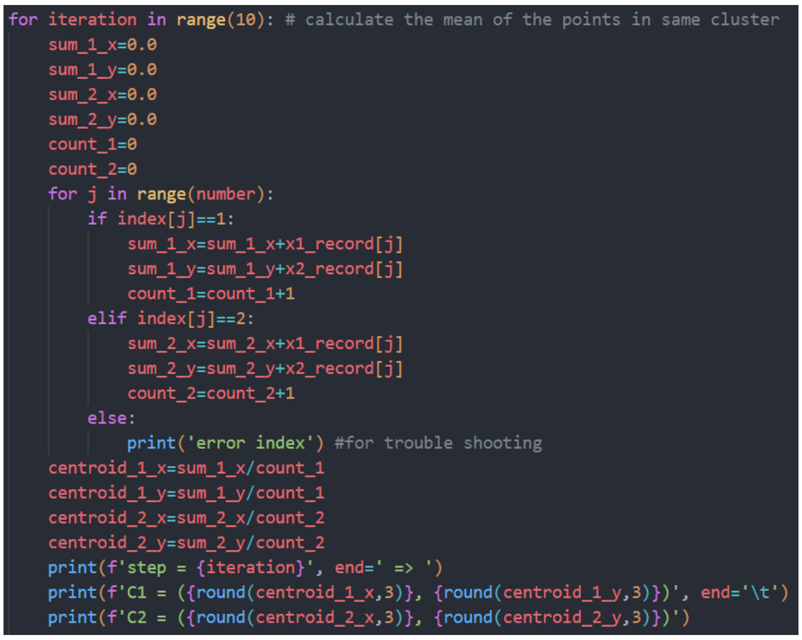
```python
import random
import matplotlib.pyplot as plt

x1_record=[]    # horizontal
x2_record=[]    # vertical

# my var for plotting
X1_record=[]
X2_record=[]
Y1_record=[]
Y2_record=[]

x1=0.0
x2=0.0
y1=0.0
y2=0.0
number_sample=30
```



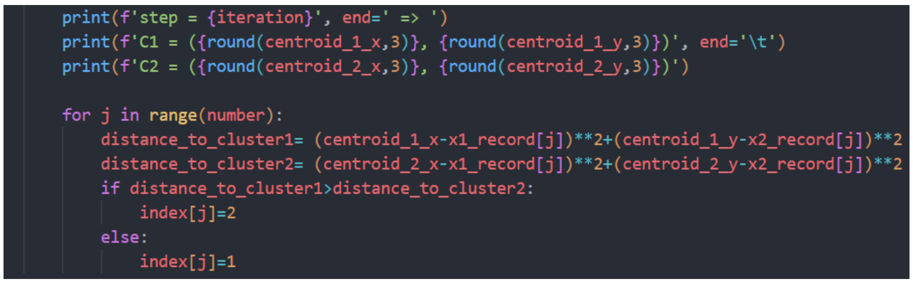
```python
#generate two clusters decorated with gaussian noise
for i in range(number_sample):
    x1=random.gauss(0,0.6) #create random number gauss (mean, sigma)
    x2=random.gauss(0,0.6)
    x1_record.append(float(x1))
    x2_record.append(float(x2))

    # my code
    X1_record.append(float(x1))
    X2_record.append(float(x2))

for i in range(number_sample):
    y1=1.0+random.gauss(0,0.6)
    y2=1.0+random.gauss(0,0.6)
    x1_record.append(float(y1))
    x2_record.append(float(y2))

    # my tmp code
    Y1_record.append(float(y1))
    Y2_record.append(float(y2))

number = len(x1_record)
print(number)
#choose two cluster
cluster1 = random.choice(range(0,number))
cluster2 = random.choice(range(0,number))

centroid1x = x1_record[cluster1]
centroid1y = x2_record[cluster1]
centroid2x = x1_record[cluster2]
centroid2y = x2_record[cluster2]
print(f'initial choice:')
print(f'center 1 = ({round(centroid1x,3)}, {round(centroid1y,3)})')
```



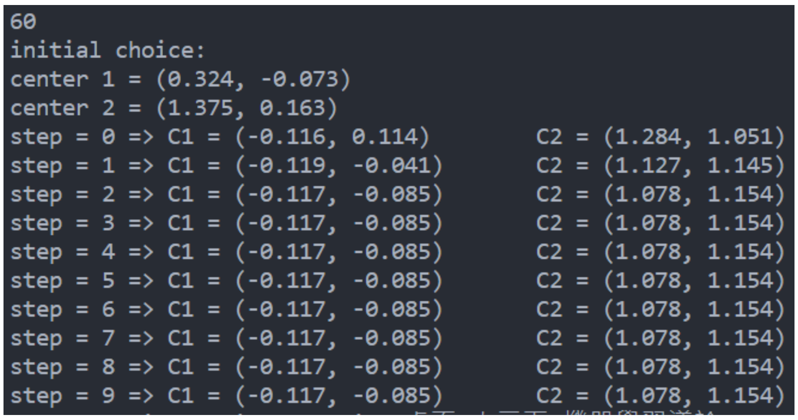
```python
print(f'center 2 = ({round(centroid2x,3)}, {round(centroid2y,3)})')


index=[]
for j in range(number):
    index.append('0') # create a zero index this index record which cluster
                      #the data point is associated with

#run over all the sample and compute & compare the distance & classify them
for j in range(number):
    distance_to_cluster1 = (centroid1x-x1_record[j])**2+(centroid1y-x2_record[j])**2
    distance_to_cluster2 = (centroid2x-x1_record[j])**2+(centroid2y-x2_record[j])**2
    if distance_to_cluster1>distance_to_cluster2:
        index[j]=2
    else:
        index[j]=1

centroid_1_x=0.0  # index rule index for cluster
centroid_1_y=0.0
centroid_2_x=0.0
centroid_2_y=0.0
```



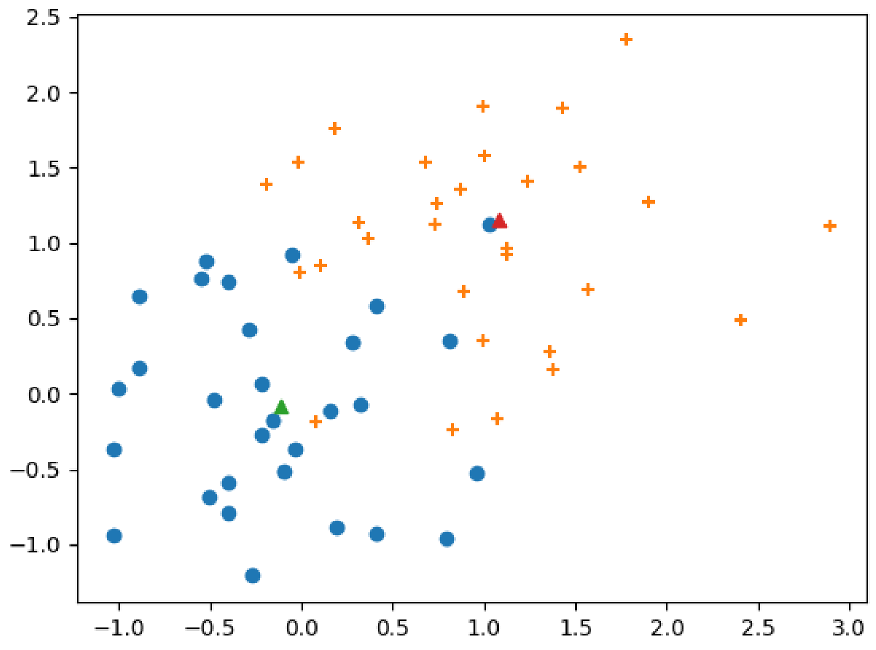
```python
for iteration in range(10): # calculate the mean of the points in same cluster
    sum_1_x=0.0
    sum_1_y=0.0
    sum_2_x=0.0
    sum_2_y=0.0
    count_1=0
    count_2=0
    for j in range(number):
        if index[j]==1:
            sum_1_x=sum_1_x+x1_record[j]
            sum_1_y=sum_1_y+x2_record[j]
            count_1=count_1+1
        elif index[j]==2:
            sum_2_x=sum_2_x+x1_record[j]
            sum_2_y=sum_2_y+x2_record[j]
            count_2=count_2+1
        else:
            print('error index') #for trouble shooting
    centroid_1_x=sum_1_x/count_1
    centroid_1_y=sum_1_y/count_1
    centroid_2_x=sum_2_x/count_2
    centroid_2_y=sum_2_y/count_2
    print(f'step = {iteration}', end=' => ')
    print(f'C1 = ({round(centroid_1_x,3)}, {round(centroid_1_y,3)})', end='\t')
    print(f'C2 = ({round(centroid_2_x,3)}, {round(centroid_2_y,3)})')
```



```python
    print(f'step = {iteration}', end=' => ')
    print(f'C1 = ({round(centroid_1_x,3)}, {round(centroid_1_y,3)})', end='\t')
    print(f'C2 = ({round(centroid_2_x,3)}, {round(centroid_2_y,3)})')

    for j in range(number):
        distance_to_cluster1= (centroid_1_x-x1_record[j])**2+(centroid_1_y-x2_record[j])**2
        distance_to_cluster2= (centroid_2_x-x1_record[j])**2+(centroid_2_y-x2_record[j])**2
        if distance_to_cluster1>distance_to_cluster2:
            index[j]=2
        else:
            index[j]=1
```



```
60
initial choice:
center 1 = (0.324, -0.073)
center 2 = (1.375, 0.163)
step = 0 => C1 = (-0.116, 0.114)        C2 = (1.284, 1.051)
step = 1 => C1 = (-0.119, -0.041)       C2 = (1.127, 1.145)
step = 2 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 3 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 4 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 5 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 6 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 7 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 8 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
step = 9 => C1 = (-0.117, -0.085)       C2 = (1.078, 1.154)
```



2b.

3.