In [2]:

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

In [3]:

```
df= pd.read_csv('element_data.csv')
```

In [4]:

```
df
```

Out[4]:

	atomic_number	atomic_volume	boiling_point	en_ghosh	evaporation_heat	heat_of_formation	lattice_constant	specific_he
0	47	10.30	2435.15	0.147217	254.1	284.90	4.09	0.2
1	13	10.00	2792.15	0.150078	284.1	330.90	4.05	0.8
2	79	10.20	3109.15	0.261370	340.0	368.20	4.08	0.1
3	29	7.10	2833.15	0.151172	304.6	337.40	3.61	0.3
4	77	8.54	4701.15	0.251060	604.0	669.00	3.84	0.1
5	28	6.60	3186.15	0.147207	378.6	430.10	3.52	0.4
6	82	18.30	2022.15	0.177911	177.8	195.20	4.95	0.1
7	46	8.90	3236.15	0.144028	372.4	376.60	3.89	0.2
8	78	9.10	4098.15	0.256910	470.0	565.70	3.92	0.1
9	45	8.30	3968.15	0.140838	494.0	556.00	3.80	0.2
10	90	19.80	5058.15	0.102770	513.7	602.00	5.08	0.1
11	70	24.80	1469.15	0.221190	159.0	155.60	5.49	0.1
12	56	39.00	2118.15	0.158679	142.0	179.10	5.02	0.2
13	24	7.23	2944.15	0.131305	342.0	397.48	2.88	0.4
14	63	28.90	1802.15	0.189935	176.0	177.40	4.61	0.1
15	26	7.10	3134.15	0.139253	340.0	415.50	2.87	0.4
16	3	13.10	1615.15	0.105093	148.0	159.30	3.49	3.5
17	25	7.39	2334.15	0.135284	221.0	283.30	8.89	0.4
18	42	9.40	4912.15	0.131267	590.0	658.98	3.15	0.2
19	11	23.70	1156.09	0.093214	97.9	107.50	4.23	1.2
20	41	10.80	5014.15	0.128078	680.0	733.00	3.30	0.2
21	73	10.90	5728.15	0.234581	758.0	782.00	3.31	0.1
22	23	8.35	3680.15	0.127334	460.0	515.50	3.02	0.4
23	74	9.53	5828.15	0.239050	824.0	851.00	3.16	0.1
24	4	5.00	2741.15	0.144986	309.0	324.00	2.29	1.8
25	20	29.90	1757.15	0.115412	153.6	177.80	5.58	0.6
26	48	13.10	1040.15	0.150407	59.1	111.80	2.98	0.2
27	27	6.70	3200.15	0.143236	389.1	426.70	2.51	0.4
28	66	19.00	2840.15	0.203330	291.0	290.40	3.59	0.1
29	68	18.40	3141.15	0.212261	317.0	316.40	3.56	0.1
30	64	19.90	3546.15	0.194400	398.0	397.50	3.64	0.2
31	72	13.60	4873.15	0.229987	575.0	618.40	3.20	0.1

	atomic_number	atomic_volume	boiling_point	en_ghosh	evaporation_heat	heat_of_formation	lattice_constant	specific_he
32	67	18.70	2973.15	0.207795	301.0	300.60	3.58	0.1
33	71	17.80	3675.15	0.225650	414.0	427.60	3.51	0.1
34	12	14.00	1363.15	0.121644	131.8	147.10	3.21	1.0
35	75	8.85	5863.15	0.243516	704.0	774.00	2.76	0.1
36	21	15.00	3109.15	0.119383	332.7	377.80	3.31	0.5
37	65	19.20	3503.15	0.198863	389.0	388.70	3.60	0.1
38	22	10.60	3560.15	0.123364	422.6	473.00	2.95	0.5
39	81	17.20	1746.15	0.173447	162.4	182.20	3.46	0.1
40	69	18.10	2223.15	0.216724	232.0	232.20	3.54	0.1
41	39	19.80	3618.15	0.121699	367.0	424.70	3.65	0.2
42	30	9.20	1180.15	0.155152	114.8	130.40	2.66	0.3

In [ ]:

```
data1 = pd.read_csv('tut11')
```

In [5]:

```
df =df[['atomic_number','youngs_modulus']]
```

In [26]:

```
df0= pd.DataFrame(df)
```

In [7]:

```
df_sorted = df.sort_values(by=['youngs_modulus'])
```

In [8]:

```
df_sorted
```

Out[8]:

	atomic_number	youngs_modulus
16	3	4.9
39	81	8.0
19	11	10.0
12	56	13.0
6	82	16.0
14	63	18.0
25	20	20.0
11	70	24.0
34	12	45.0
26	48	50.0
30	64	55.0
37	65	56.0
28	66	61.0
41	39	64.0
32	67	65.0
33	71	69.0
29	68	70.0

	atomic_number	youngs_modulus
1	12	70.0
36	21	74.0
40	69	74.0
31	72	78.0
2	79	78.0
10	90	79.0
0	47	83.0
20	41	105.0
42	30	108.0
38	22	116.0
7	46	121.0
22	23	128.0
3	29	130.0
8	78	168.0
21	73	186.0
17	25	198.0
5	28	200.0
27	27	209.0
15	26	211.0
9	45	275.0
13	24	279.0
24	4	287.0
18	42	329.0
23	74	411.0
35	75	463.0
4	77	528.0

In [10]:

```
len(df_sorted)
```

Out[10]:

43

In [27]:

```
#Group 1 shuffled
df1 = df0.iloc[0:7]

x1 = df1.sample(frac=1)
x1
```

Out[27]:

	youngs_modulus
5	200.0
4	528.0
2	78.0
3	130.0
1	70.0
0	83.0

	youngs_modulus
6	16.0

In [ ]:

```
```

In [28]:

```
#Group 3 shuffled
df3 = df0.iloc[21:28]
x3 = df3.sample(frac=1)
x3
```

Out[28]:

	youngs_modulus
25	20.0
21	186.0
27	209.0
26	50.0
22	128.0
24	287.0
23	411.0

In [ ]:

```
```

In [29]:

```
#Group 5 shuffled
df5 = df0.iloc[31:38]
x5 = df5.sample(frac=1)
x5
```

Out[29]:

	youngs_modulus
33	69.0
31	78.0
32	65.0
36	74.0
34	45.0
37	56.0
35	463.0

In [ ]:

```
```

In [30]:

```
#Group 3 shuffled
df7 = df0.iloc[38:43]
x7 = df7.sample(frac=1)
x7
```

Out[30]:

	youngs_modulus

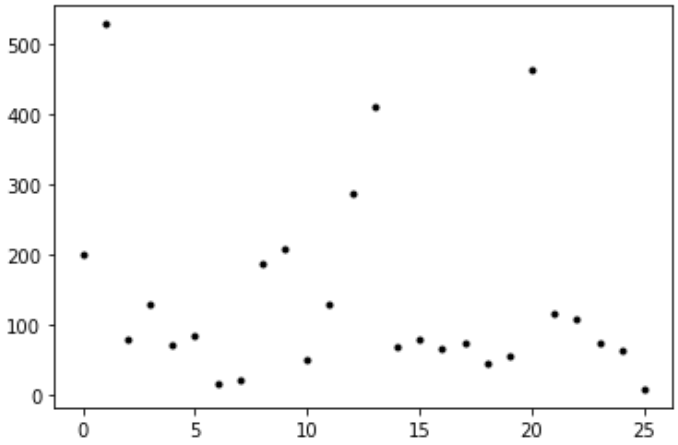
	youngs_modulus
38	116.0
42	108.0
40	74.0
41	64.0
39	8.0

In [ ]:

```
```

In [31]:

```
X = np.concatenate((x1,x3,x5,x7),axis=0)
plt.plot(X[:,0],'k.')
#plt.plot(X[:,0],X[:,1],'k.')
plt.show()
```

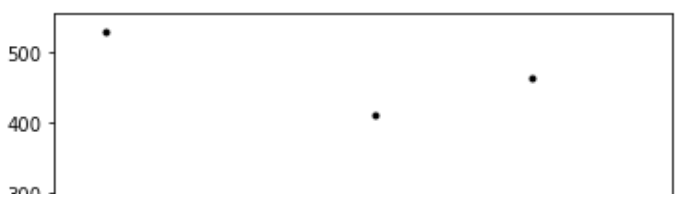


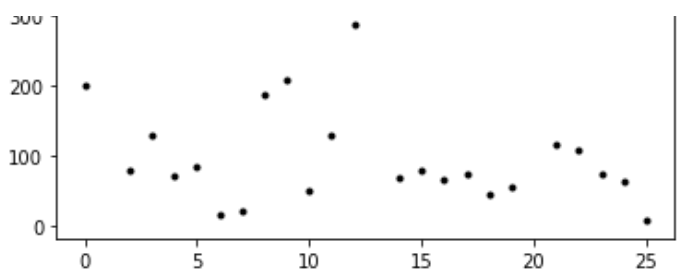
In [ ]:

```
```

In [36]:

```
from sklearn.cluster import KMeans


n = 4
k_means = KMeans(n_clusters=n)
k_means.fit(X)



KMeans(n_clusters=4)

centroids = k_means.cluster_centers_
labels= k_means.labels_


plt.plot(X[labels==0,0],'r.', label='cluster 1')
plt.plot(X[labels==1,0],'b.', label='cluster 2')
plt.plot(X[labels==2,0],'g.', label='cluster 3')
plt.plot(X[labels==3,0],'k.', label='cluster 4')
plt.plot(centroids[:,0],'mo',markersize=8, label='centroids')

plt.legend(loc='best')
plt.show()
```





In [ ]:

Q.2 Apply k-Means method on this data and reveal how many groups it has. Dataset - Micrographs of Metals. The final results should include (a) elbow plot and (b) graphics of clusters or 'blobs' in PC space.

In [ ]:

In [39]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.utils import shuffle
from sklearn.metrics import pairwise_distances_argmin
from PIL import Image
import os
```

In [40]:

```
def compress_image(image_path, k):
    # Load the image
    img = Image.open(image_path)

    # Convert the image to a numpy array
    img_array = np.asarray(img)

    # Reshape the image array to 2D
    w, h, d = tuple(img_array.shape)
    image_array = np.reshape(img_array, (w * h, d))

    # Create the k-means model with k clusters
    kmeans = KMeans(n_clusters=k)

    # Fit the model to the image array
    kmeans.fit(image_array)

    # Replace each pixel's RGB value with its nearest centroid
    compressed_image = kmeans.cluster_centers_[kmeans.predict(image_array)]

    # Reshape the compressed image array back to 3D
    compressed_image = np.reshape(compressed_image, (w, h, d))

    # Convert the compressed image array to an image and save it
    compressed_image = Image.fromarray(np.uint8(compressed_image))
    compressed_image.save("compressed_" + os.path.basename(image_path))
```

In [ ]:

In [ ]:

```
# Set the directory path
```

```
directory = "images/"

# Set the number of clusters (k) for k-means clustering
k = 16

# Loop through all the images in the directory
for filename in os.listdir(directory):
    if filename.endswith(".jpg"):
        # Get the full path of the image file
        image_path = os.path.join(directory, filename)

        # Apply k-means clustering and save the compressed image
        compress_image(image_path, k)
```

In [ ]:

```

```

In [46]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```