In [1]:
```python
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
```

In [3]:
```python
from sklearn.datasets import load_iris
iris =load_iris()
```

In [5]:
```python
dir(iris)
```

Out[5]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [7]:
```python
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

In [9]:
```python
iris_df
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

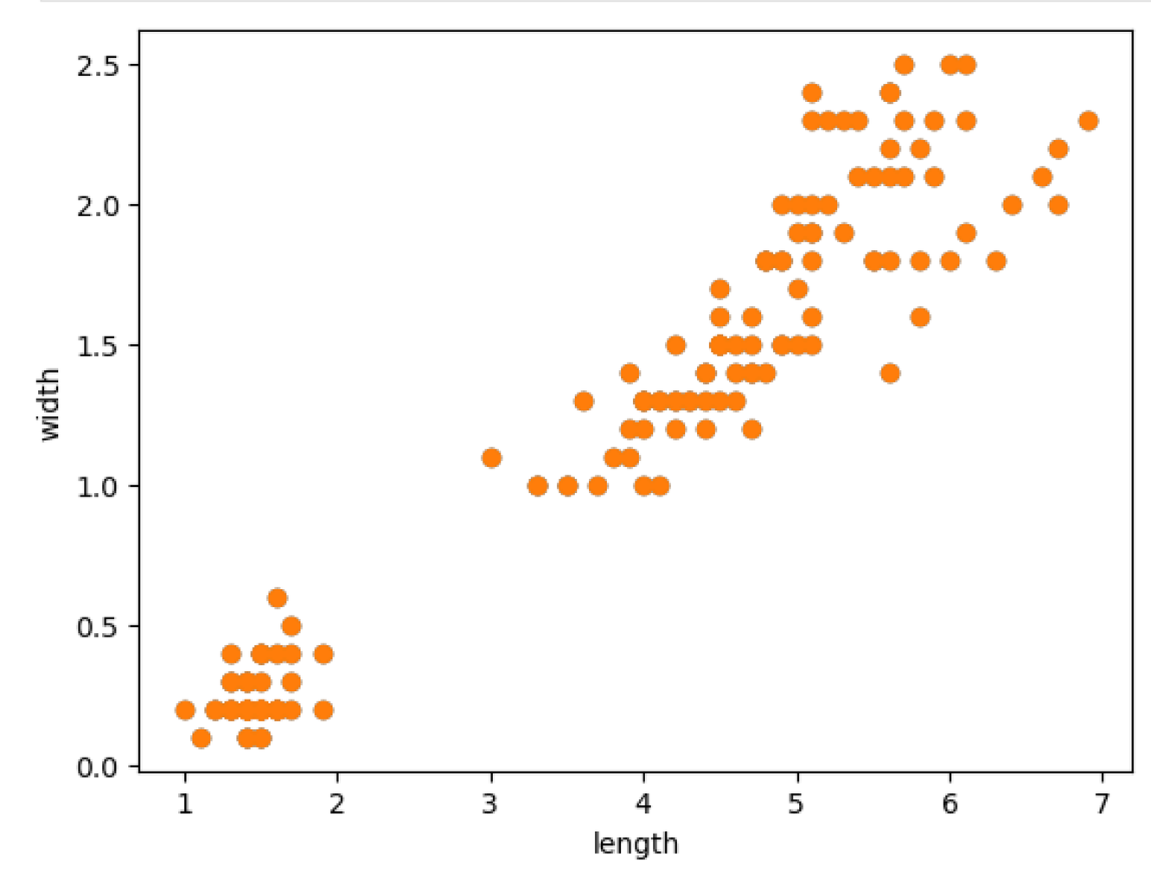
In [11]:
```python
iris_df1 = iris_df.drop(['sepal length (cm)', 'sepal width (cm)'], axis='columns')
iris_df1
```

Out[11]:

	petal length (cm)	petal width (cm)
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2
...	...	...
145	5.2	2.3
146	5.0	1.9
147	5.2	2.0
148	5.4	2.3
149	5.1	1.8

150 rows × 2 columns

In [15]:
```python
plt.scatter(iris_df1['petal length (cm)'], iris_df1['petal width (cm)'])
plt.xlabel("length")
plt.ylabel("width")
plt.show()
```



In [19]:
```python
km = KMeans(n_clusters=3)
y_predict = km.fit_predict(iris_df1[['petal length (cm)','petal width (cm)']])
y_predict
```

```
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[19]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

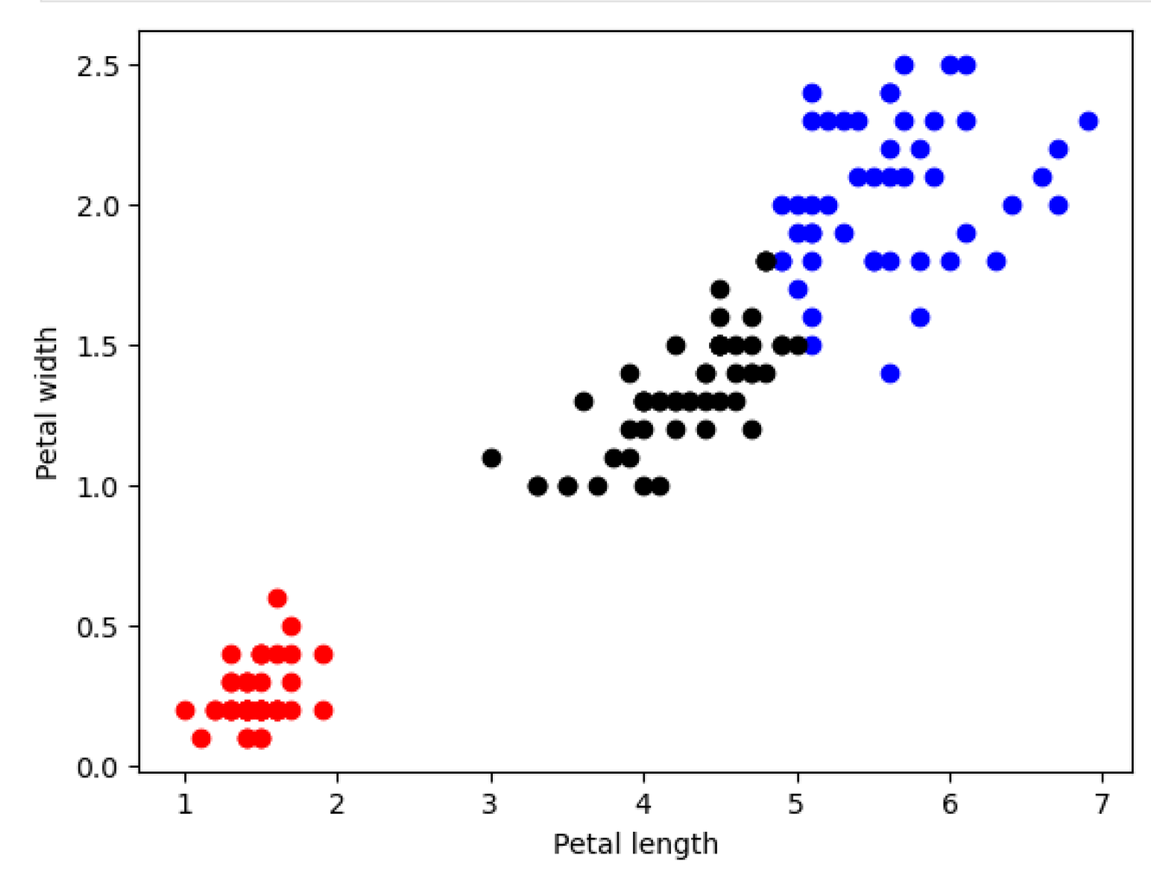
In [21]:
```python
iris_df1['cluster'] = y_predict
iris_df1
```

Out[21]:

	petal length (cm)	petal width (cm)	cluster
0	1.4	0.2	0
1	1.4	0.2	0
2	1.3	0.2	0
3	1.5	0.2	0
4	1.4	0.2	0
...	...	...	...
145	5.2	2.3	1
146	5.0	1.9	1
147	5.2	2.0	1
148	5.4	2.3	1
149	5.1	1.8	1

150 rows × 3 columns

In [23]:
```python
df1 = iris_df1[iris_df1.cluster==0]
df2 = iris_df1[iris_df1.cluster==1]
df3 = iris_df1[iris_df1.cluster==2]
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='red')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='blue')
plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='black')
plt.xlabel("Petal length")
plt.ylabel("Petal width")
plt.show()
```



In [25]:
```python
scaler = MinMaxScaler()
scaler.fit(iris_df1[['petal length (cm)']])
iris_df1['petal length (cm)'] = scaler.transform(iris_df1[['petal length (cm)']])
scaler.fit(iris_df1[['petal width (cm)']])
iris_df1['petal width (cm)'] = scaler.transform(iris_df1[['petal width (cm)']])
```

In [27]:
```python
iris_df1
```

Out[27]:

	petal length (cm)	petal width (cm)	cluster
0	0.067797	0.041667	0
1	0.067797	0.041667	0
2	0.050847	0.041667	0
3	0.084746	0.041667	0
4	0.067797	0.041667	0
...	...	...	...
145	0.711864	0.916667	1
146	0.677966	0.750000	1
147	0.711864	0.791667	1
148	0.745763	0.916667	1
149	0.694915	0.708333	1

150 rows × 3 columns

In [29]:
```python
km = KMeans(n_clusters=3)
y_predict = km.fit_predict(iris_df1[['petal length (cm)','petal width (cm)']])
y_predict
```

```
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[29]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [31]:
```python
iris_df1['cluster'] = y_predict
iris_df1
```

Out[31]:

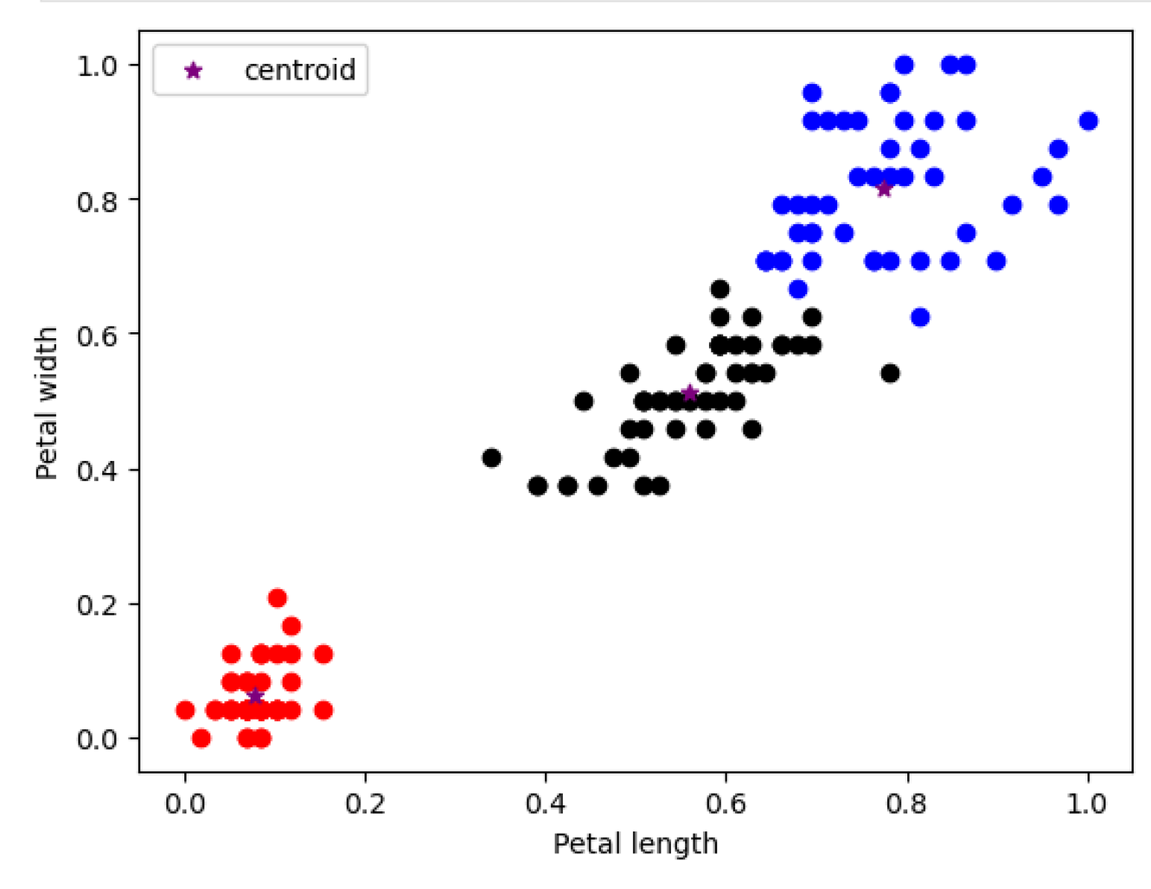
	petal length (cm)	petal width (cm)	cluster
0	0.067797	0.041667	0
1	0.067797	0.041667	0
2	0.050847	0.041667	0
3	0.084746	0.041667	0
4	0.067797	0.041667	0
...	...	...	...
145	0.711864	0.916667	1
146	0.677966	0.750000	1
147	0.711864	0.791667	1
148	0.745763	0.916667	1
149	0.694915	0.708333	1

150 rows × 3 columns

In [33]:
```python
km.cluster_centers_
```

Out[33]:
```
array([[0.07830508, 0.06083333],
       [0.7740113 , 0.81510417],
       [0.55867014, 0.51041667]])
```

In [37]:
```python
df1 = iris_df1[iris_df1.cluster==0]
df2 = iris_df1[iris_df1.cluster==1]
df3 = iris_df1[iris_df1.cluster==2]
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='red')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='blue')
plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],marker = '*',color='purple',label='centroid')
plt.xlabel("Petal length")
plt.ylabel("Petal width")
plt.legend()
plt.show()
```



In [43]:
```python
sse=[]
k_recg = range(1,30)
for k in k_recg:
    km = KMeans(n_clusters=k)
    km.fit(iris_df1[['petal length (cm)','petal width (cm)']])
    sse.append(km.inertia_)
```

```
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\thanu\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [45]:
```python
plt.xlabel('K')
plt.ylabel('Sum Of Squared error')
plt.plot(k_recg,sse)
```

Out[41]: [<matplotlib.lines.Line2D at 0x239805921e0>]

In [ ]: