In [4]:

```
# Importing the Necessary libraries


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
plt.style.use('ggplot')
```

In [5]:

```
# Loading and preparing the dataset load_iris which is an inbuilt dataset for classificatio
from sklearn.datasets import load_iris
```

In [6]:

```
iris=load_iris()
```

In [7]:

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

In [92]:

```
df=df[['sepal length (cm)','petal length (cm)']]
df
```

Out[92]:

	sepal length (cm)	petal length (cm)
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5
4	5.0	1.4
...	...	...
145	6.7	5.2
146	6.3	5.0
147	6.5	5.2
148	6.2	5.4
149	5.9	5.1

150 rows × 2 columns

In [93]:

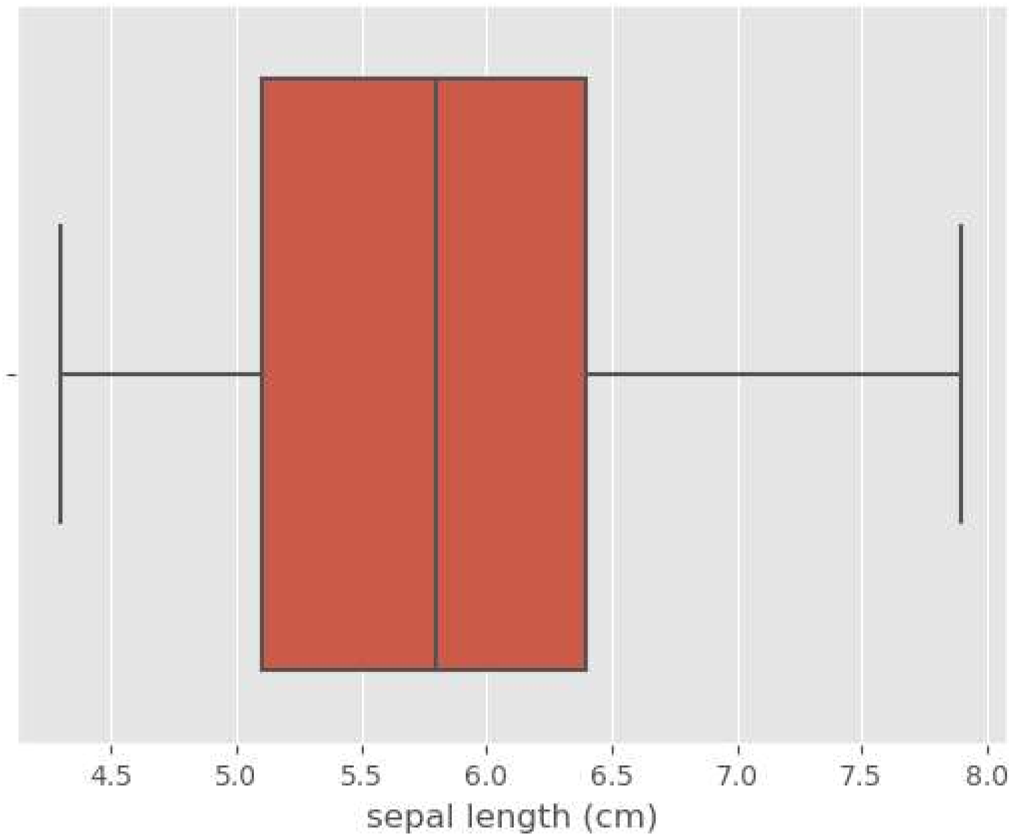
```
X=df.values
```

In [94]:

```
# To check if any outlier is present by boxplot method

sns.boxplot(x='sepal length (cm)',data=df )
```

Out[94]:

```
<AxesSubplot:xlabel='sepal length (cm)'>
```



In [95]:

```
from sklearn.cluster import KMeans
```

In [96]:

```
# Type-1 --Training the model and checking all the by hyperparameter tuning of n-clusters ar


model=KMeans(n_clusters=2,random_state=20,init='k-means++')
model.fit(X)
print('Inertia of the model=',model.inertia_)
```

```
Inertia of the model= 112.99207175925925
```

In [97]:

```
model=KMeans(n_clusters=3,random_state=20,init='k-means++')
model.fit(X)
print('Inertia of the model=',model.inertia_)
```

```
Inertia of the model= 53.80997864410694
```

In [98]:

```
model=KMeans(n_clusters=4,random_state=20,init='k-means++')
model.fit(X)
print('Inertia of the model=',model.inertia_)
```

```
Inertia of the model= 34.31702077922079
```

In [99]:

```
model=KMeans(n_clusters=5,random_state=20,init='k-means++')
model.fit(X)
print('Inertia of the model=',model.inertia_)
```

```
Inertia of the model= 25.634064509564507
```

In [100]:

```
model=KMeans(n_clusters=6,random_state=20,init='k-means++')
model.fit(X)
print('Inertia of the model=',model.inertia_)
```

```
Inertia of the model= 21.797231176231172
```

In [101]:

```
print(model.cluster_centers_)
```

```
[[6.19142857 4.74285714]
 [4.67       1.415     ]
 [6.56153846 5.48461538]
 [5.23       1.49333333]
 [5.52592593 3.94074074]
 [7.475      6.3       ]]
```

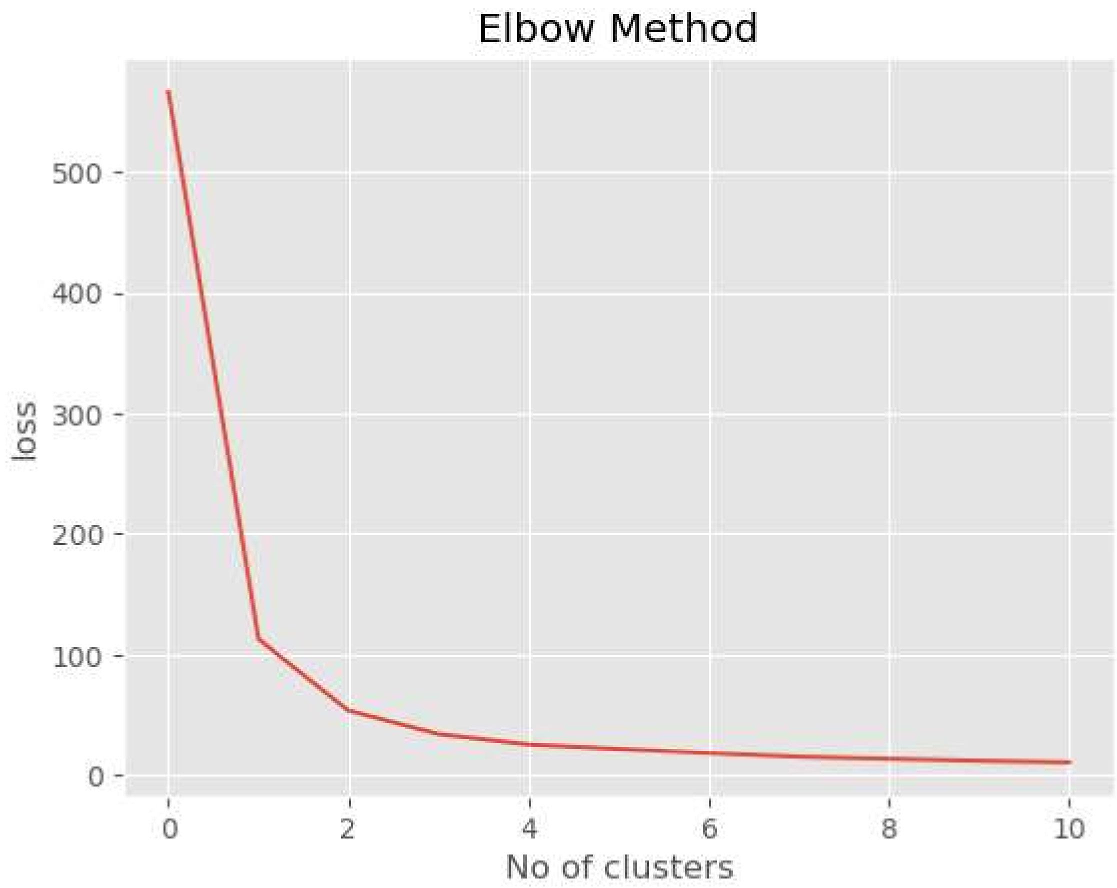
In [102]:

```
# 2nd method to check the inertia direct in range of 1to10 directly by Elbow Method

import warnings
warnings.filterwarnings('ignore')

loss=[]

for i in range(1,12):
    kmeans=KMeans(n_clusters=i,random_state=20,init='k-means++')
    kmeans.fit(X)
    loss.append(kmeans.inertia_)
    print('Loss of the model=',loss)
```

```
Loss of the model= [566.4937333333332]
Loss of the model= [566.4937333333332, 112.99207175925925]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172, 18.29513382680431]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172, 18.29513382680431,
15.1963170995671]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172, 18.29513382680431,
15.1963170995671, 13.423398629148638]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172, 18.29513382680431,
15.1963170995671, 13.423398629148638, 11.788615282000716]
Loss of the model= [566.4937333333332, 112.99207175925925, 53.80997864410694,
34.31702077922079, 25.634064509564507, 21.797231176231172, 18.29513382680431,
15.1963170995671, 13.423398629148638, 11.788615282000716, 10.490218276735302]
```

In [104]:

```
plt.plot(loss)
plt.xlabel('No of clusters')
plt.ylabel('loss')
plt.title('Elbow Method')
plt.show()
```



In [109]:

```
# Now let us finalize the Value of K would be 3 as elbow id bent on it

model=KMeans(n_clusters=3,random_state=20,init='k-means++')
model.fit(X)
```

Out[109]:

```
KMeans(n_clusters=3, random_state=20)
```

In [106]:

```
centroid=model.cluster_centers_
```

In [107]:

```
model.labels_
```

Out[107]:

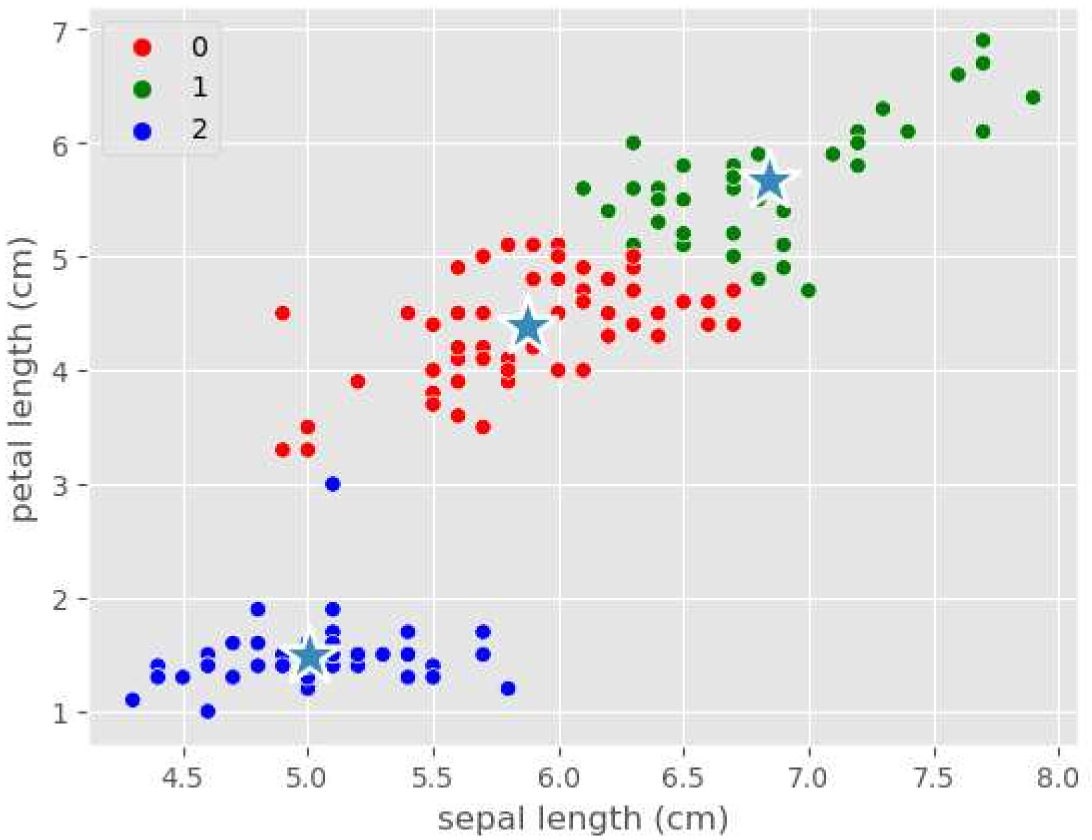
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0])
```

In [108]:

```
# Visualising the Kmeans clusters

sns.scatterplot(x='sepal length (cm)' ,y= 'petal length (cm)',data=df,hue=model.labels_,pal
sns.scatterplot(x=centroid[:,0],y=centroid[:,1],marker='*',s=500)
```

Out[108]:

<AxesSubplot:xlabel='sepal length (cm)', ylabel='petal length (cm)'>



In [ ]: