In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df=pd.read_csv(r"C:\Users\DELL E5490\Downloads\Income.csv")
df
```

Out[2]:

	Gender	Age	Income($)
0	Male	19	15
1	Male	21	15
2	Female	20	16
3	Female	23	16
4	Female	31	17
...	...	...	...
195	Female	35	120
196	Female	45	126
197	Male	32	126
198	Male	32	137
199	Male	30	137

200 rows × 3 columns

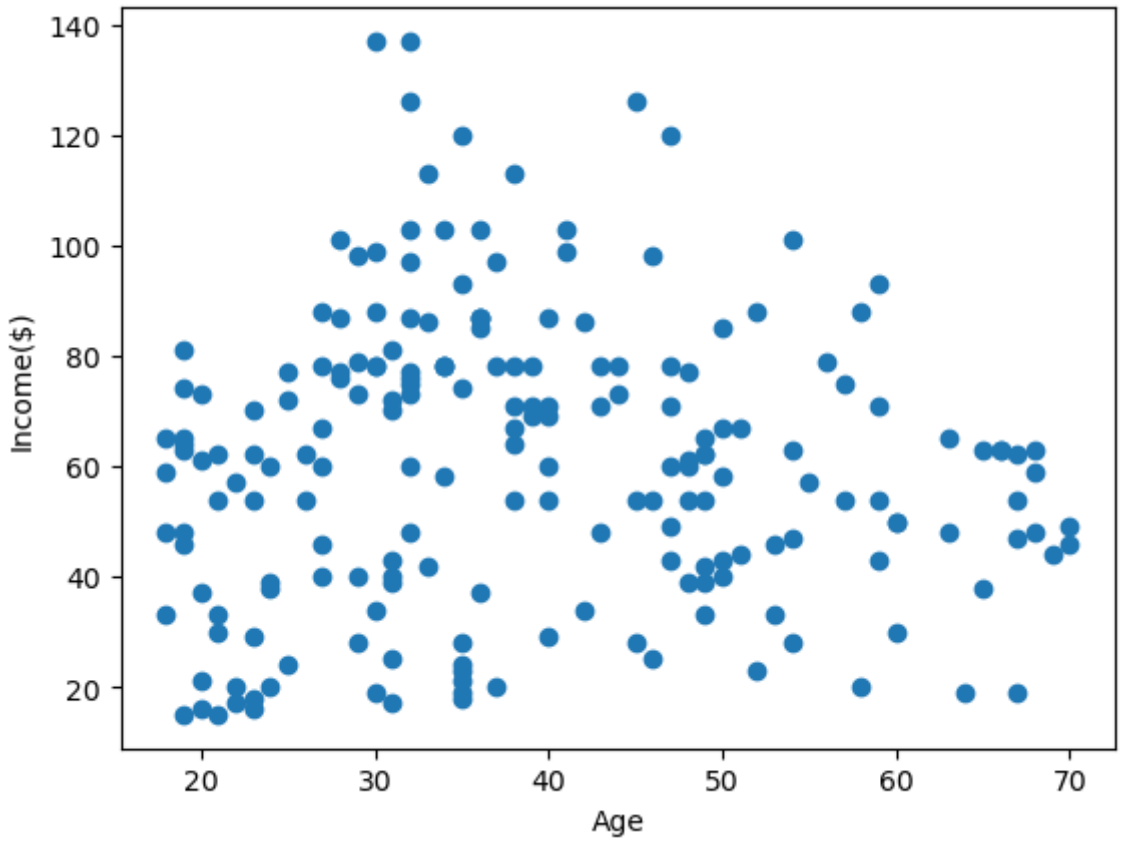
In [16]:
```
1 df.head()
```

Out[16]:

	Gender	Age	Income($)
0	Male	19	15
1	Male	21	15
2	Female	20	16
3	Female	23	16
4	Female	31	17

In [17]:
```
plt.scatter(df["Age"],df["Income($)"])
plt.xlabel("Age")
plt.ylabel("Income($)")
```

Out[17]: Text(0, 0.5, 'Income($)')



In [18]:
```
from sklearn.cluster import KMeans
km=KMeans()
km
```

Out[18]:
```
KMeans()
```

In [23]:
```
y_predicted=km.fit_predict(df[["Age","Income($)"]])
y_predicted
```

```
C:\Users\DELL E5490\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\DELL E5490\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting
the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[23]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 4, 2, 4, 2, 2, 2, 2, 2, 4, 2, 2, 2,
       4, 2, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2, 4, 2, 4, 2, 2, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 2, 2, 4, 1, 1, 4, 4, 4, 4, 0, 1, 4, 0, 1, 0, 4, 0, 1,
       4, 0, 1, 1, 0, 4, 0, 0, 0, 1, 5, 5, 1, 5, 0, 1, 0, 5, 1, 5, 0, 1,
       1, 5, 0, 1, 5, 5, 1, 1, 5, 1, 5, 1, 1, 5, 0, 1, 5, 1, 0, 5, 0, 0,
       0, 1, 5, 1, 1, 1, 0, 5, 5, 5, 7, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5,
       7, 7, 7, 7, 5, 7, 7, 7, 5, 7, 7, 7, 7, 7, 5, 7, 7, 7, 5, 7, 5, 7,
       5, 7, 7, 7, 7, 7, 5, 7, 7, 7, 3, 3, 3, 7, 3, 3, 3, 7, 3, 3, 3, 3,
       3, 7, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6, 6,
       6, 6])
```

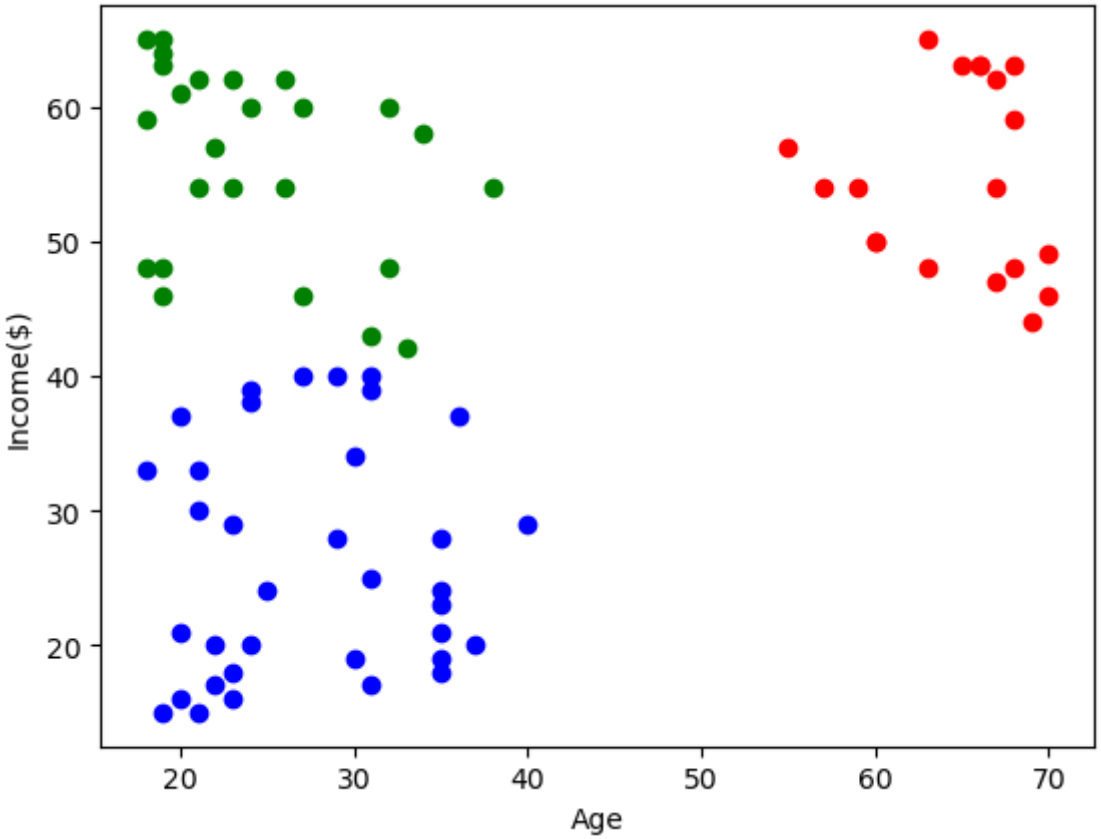
In [25]:
```
df["cluster"]=y_predicted
df.head()
```

Out[25]:

	Gender	Age	Income($)	cluster
0	Male	19	15	2
1	Male	21	15	2
2	Female	20	16	2
3	Female	23	16	2
4	Female	31	17	2

In [27]:
```
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]
plt.scatter(df1["Age"],df1["Income($)"],color="red")
plt.scatter(df2["Age"],df2["Income($)"],color="green")
plt.scatter(df3["Age"],df3["Income($)"],color="blue")
plt.xlabel("Age")
plt.ylabel("Income($)")
```

Out[27]: Text(0, 0.5, 'Income($)')



In [28]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(df[["Income($)"]])
df["Income($)"]=scaler.transform(df[["Income($)"]])
df.head()
```

Out[28]:

	Gender	Age	Income($)	cluster
0	Male	19	0.000000	2
1	Male	21	0.000000	2
2	Female	20	0.008197	2
3	Female	23	0.008197	2
4	Female	31	0.016393	2

In [29]:
```
scaler.fit(df[["Age"]])
df["Age"]=scaler.transform(df[["Age"]])
df.head()

```

Out[29]:

	Gender	Age	Income($)	cluster
0	Male	0.019231	0.000000	2
1	Male	0.057692	0.000000	2
2	Female	0.038462	0.008197	2
3	Female	0.096154	0.008197	2
4	Female	0.250000	0.016393	2

In [30]:
```
km=KMeans()
```

In [31]:
```
y_predicted=km.fit_predict(df[["Age","Income($)"]])
y_predicted
```

```
C:\Users\DELL E5490\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\DELL E5490\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting
the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[31]:
```
array([6, 6, 6, 6, 4, 6, 4, 6, 3, 4, 3, 4, 0, 6, 4, 6, 4, 6, 0, 4, 4, 6,
       0, 4, 0, 4, 0, 4, 4, 6, 3, 6, 0, 6, 0, 6, 0, 4, 4, 6, 3, 6, 0, 4,
       0, 6, 0, 4, 4, 4, 0, 4, 4, 3, 0, 0, 0, 3, 4, 0, 3, 2, 3, 0, 3, 2,
       0, 3, 2, 4, 3, 0, 3, 3, 3, 2, 0, 0, 2, 0, 3, 5, 3, 0, 2, 0, 7, 2,
       5, 7, 3, 2, 7, 5, 5, 2, 7, 2, 7, 2, 2, 7, 3, 2, 7, 2, 3, 7, 3, 3,
       3, 2, 5, 2, 2, 2, 3, 7, 7, 7, 2, 5, 5, 5, 2, 5, 7, 5, 7, 5, 7, 5,
       2, 5, 2, 5, 7, 5, 2, 5, 7, 5, 5, 5, 2, 5, 7, 5, 5, 5, 7, 5, 7, 5,
       7, 5, 5, 5, 5, 5, 7, 5, 2, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 5, 7, 5,
       7, 5, 7, 1, 1, 1, 1, 1, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

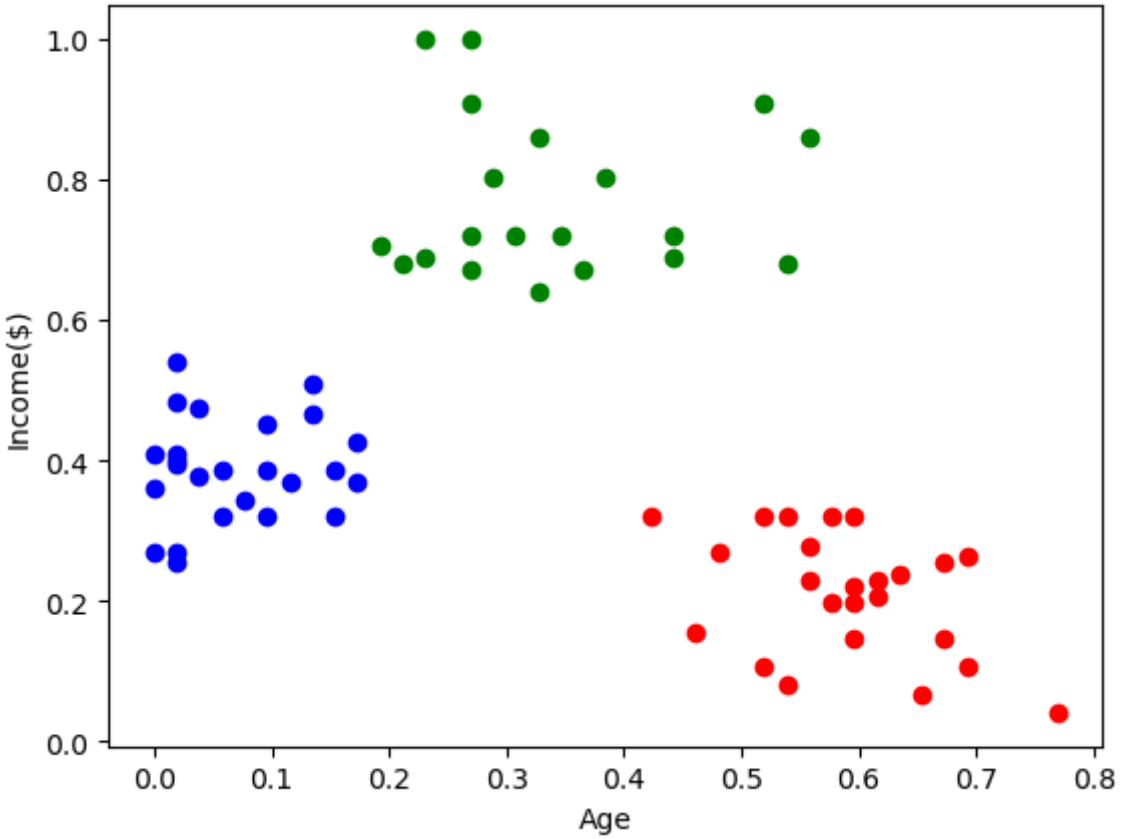
In [32]:
```
df["New Cluster"]=y_predicted
df.head()
```

Out[32]:

	Gender	Age	Income($)	cluster	New Cluster
0	Male	0.019231	0.000000	2	6
1	Male	0.057692	0.000000	2	6
2	Female	0.038462	0.008197	2	6
3	Female	0.096154	0.008197	2	6
4	Female	0.250000	0.016393	2	4

In [33]:
```
df1=df[df["New Cluster"]==0]
df2=df[df["New Cluster"]==1]
df3=df[df["New Cluster"]==2]
plt.scatter(df1["Age"],df1["Income($)"],color="red")
plt.scatter(df2["Age"],df2["Income($)"],color="green")
plt.scatter(df3["Age"],df3["Income($)"],color="blue")
plt.xlabel("Age")
plt.ylabel("Income($)")
```

Out[33]: Text(0, 0.5, 'Income($)')

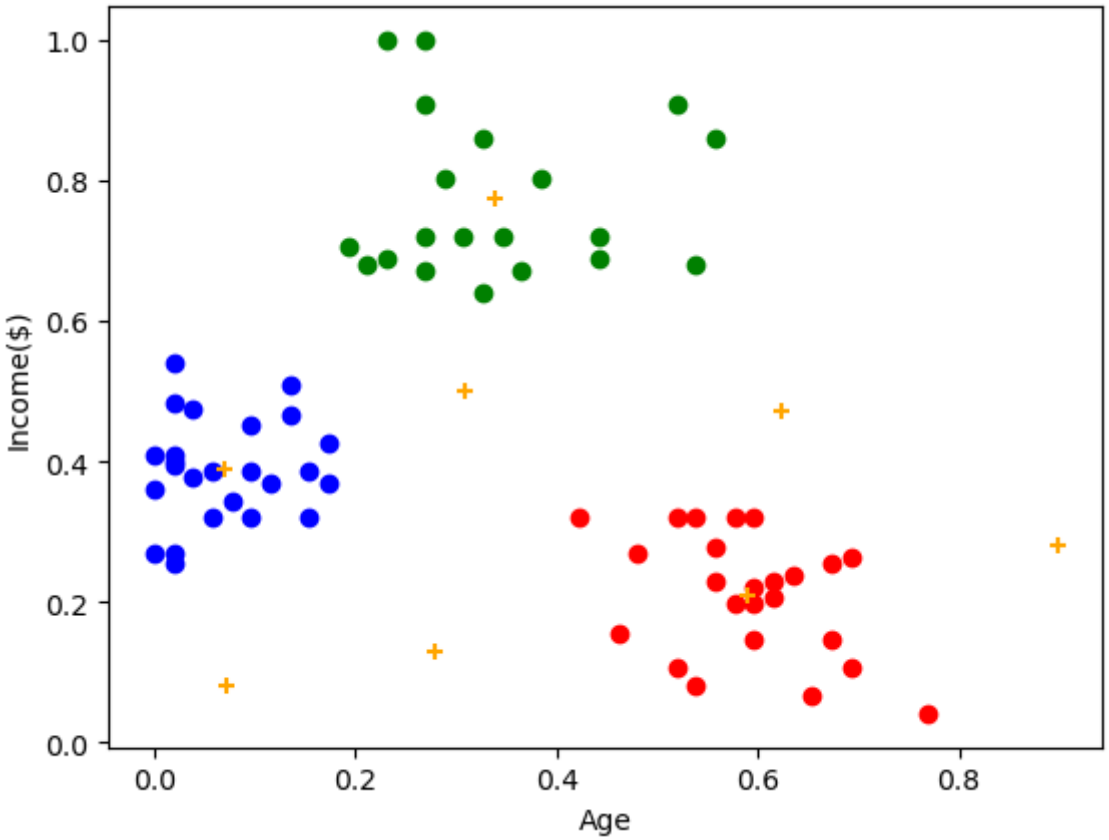


In [34]:
```
1 km.cluster_centers_
```

Out[34]:
```
array([[0.58974359, 0.20969945],
       [0.33942308, 0.77295082],
       [0.06923077, 0.38786885],
       [0.89799331, 0.28011404],
       [0.27884615, 0.13040238],
       [0.30903399, 0.50114373],
       [0.07239819, 0.08003857],
       [0.62352071, 0.47225725]])
```

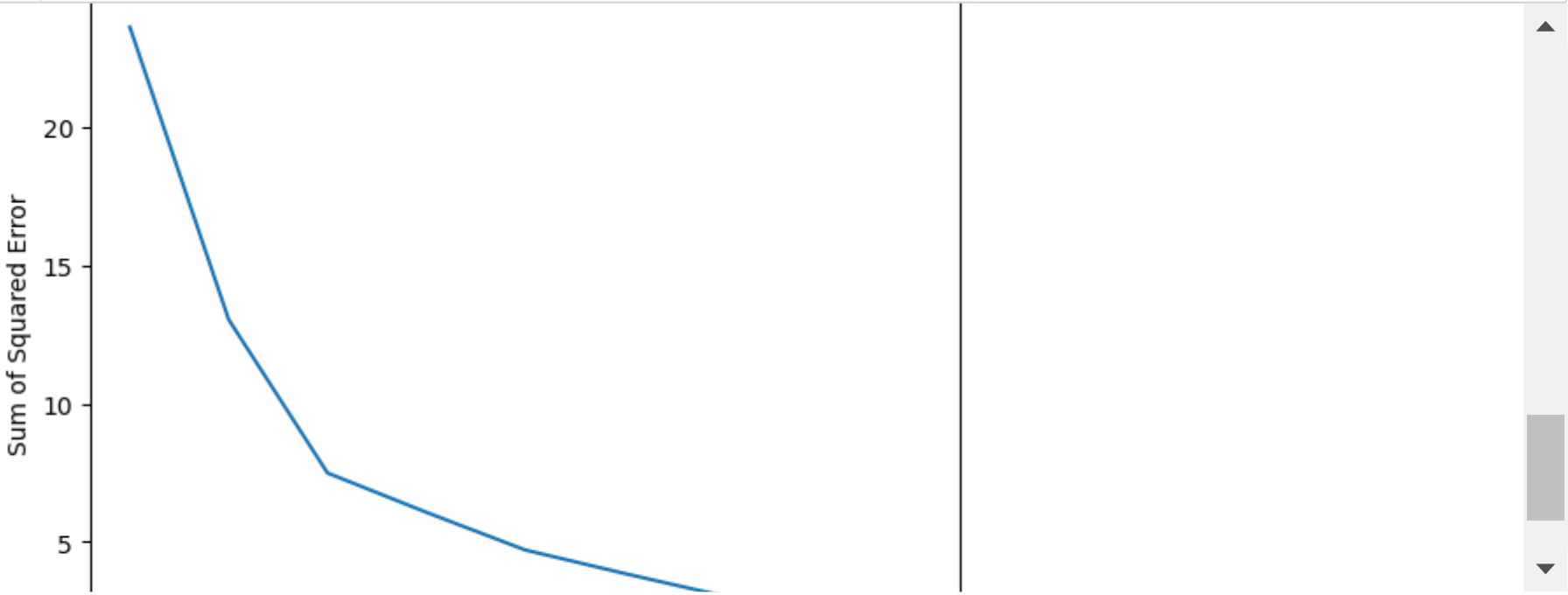
In [35]:
```
df1=df[df["New Cluster"]==0]
df2=df[df["New Cluster"]==1]
df3=df[df["New Cluster"]==2]
plt.scatter(df1["Age"],df1["Income($)"],color="red")
plt.scatter(df2["Age"],df2["Income($)"],color="green")
plt.scatter(df3["Age"],df3["Income($)"],color="blue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="orange",marker="+")
plt.xlabel("Age")
plt.ylabel("Income($)")
```

Out[35]: Text(0, 0.5, 'Income($)')



In [36]:
```
k_rng=range(1,10)
sse=[]
```

In [37]:
```
for k in k_rng:
 km=KMeans(n_clusters=k)
 km.fit(df[["Age","Income($)"]])
 sse.append(km.inertia_)
#km.inertia_ will give you the value of sum of square error
print(sse)
plt.plot(k_rng,sse)
plt.xlabel("K")
plt.ylabel("Sum of Squared Error")
```



In [ ]:
```
1
```