TO MAKE DBSCAN CLUSTERING

In [102]:
```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
print(__doc__)
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
```

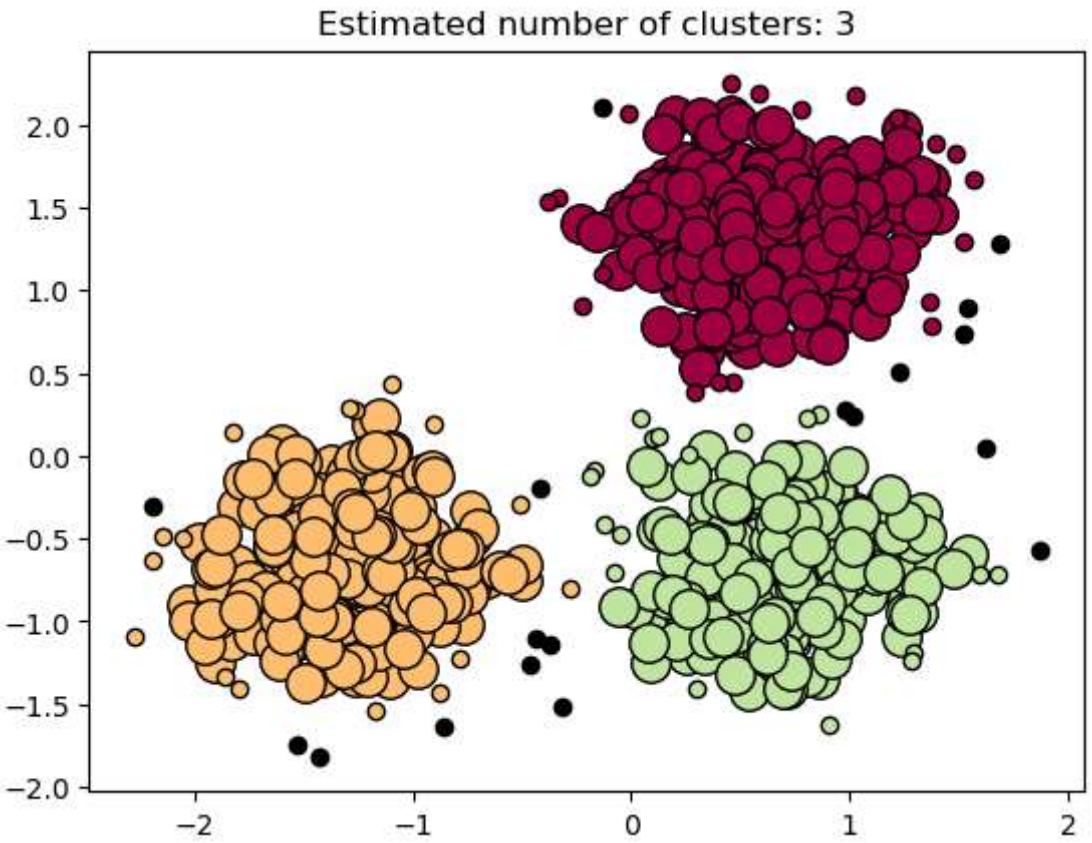
```
Automatically created module for IPython interactive environment
```

In [103]:
```
# Generate sample data
centers = [[1,1],[-1, -1],[1, -1]]
x, labels_true = make_blobs(n_samples  = 750, centers = centers, cluster_std = 0.4, random_state = 0)
x = StandardScaler().fit_transform(x)
```

In [104]:
```
# To Compute DBSCAN
db = DBSCAN(eps =0.3, min_samples = 10).fit(x)
core_samples_mask = np.zeros_like(db.labels_, dtype = bool)

core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, Ignoring noise if present:
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print('Estimated number of clusters: %d' %n_clusters_)
print('Estimated number of noise points: %d' %n_noise_)
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
print("Adjusted Rand Index: %0.3f" % metrics.adjusted_mutual_info_score(labels_true, labels))
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(x,labels))
```

```
Estimated number of clusters: 3
Estimated number of noise points: 18
Homogeneity: 0.953
Completeness: 0.883
V-measure: 0.917
Adjusted Rand Index: 0.916
Silhouette Coefficient: 0.626
```

In [105]:
```
# To plot the dataset: Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise:
        col = [0, 0, 0, 1]
    class_member_mask = (labels == k)
    xy = x[class_member_mask & core_samples_mask]
    plt.plot(xy[:,0], xy[:, 1], 'o', markerfacecolor = tuple(col),
             markeredgecolor = 'k', markersize = 14)
    xy = x[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor = tuple(col),
              markeredgecolor = 'k', markersize = 6)
plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

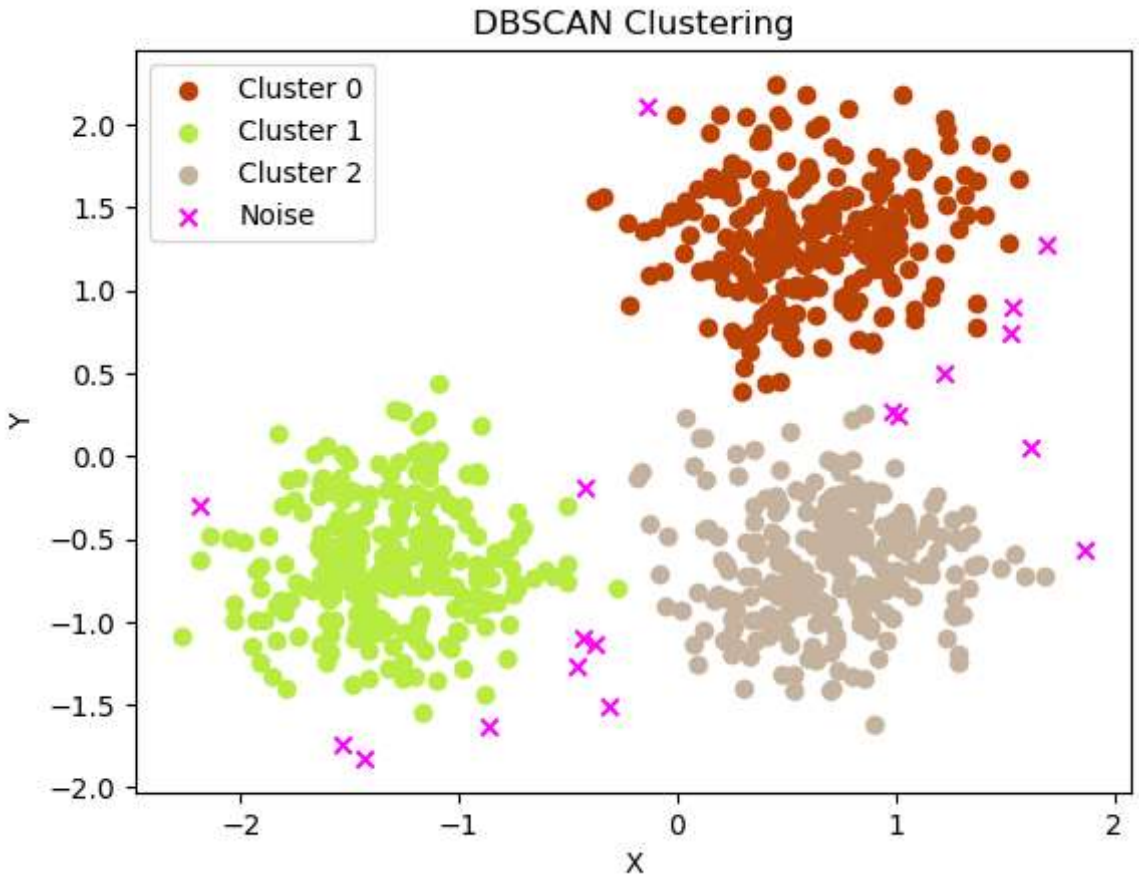


In [106]:
```
# To do DBSCAN clustering in different method
dbscan = DBSCAN(eps = 0.3, min_samples = 10)
labels = dbscan.fit_predict(x)
```

In [107]:
```
unique_labels = np.unique(labels)
unique_labels = unique_labels[unique_labels != -1]
num_clusters = len(unique_labels)
num_clusters
```

Out[107]: 3

In [108]:
```
# To Generate random colors for the clusters

colors = np.random.rand(num_clusters, 3)
# To plot the clusters with random colors
for label, color in zip(unique_labels, colors):
    cluster_data = x [labels == label]
    plt.scatter(cluster_data[:, 0 ], cluster_data[:, 1], c = [color], label = f'Cluster {label}')
# Ploting noise points as magenta X' s
noise_data = x[labels == -1]
plt.scatter(noise_data[:, 0], noise_data[:, 1], marker = 'x', color = 'magenta',label = 'Noise')

plt.title('DBSCAN Clustering')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: