In [1]:
```
import pandas as pd
data = pd.read_csv("apples_and_oranges.csv")
data.head()
```

Out[1]:

	Weight	Size	Class
0	69	4.39	orange
1	69	4.21	orange
2	65	4.09	orange
3	72	5.85	apple
4	67	4.70	orange

In [2]:
```
from sklearn.model_selection import train_test_split
training_set, test_set = train_test_split(data, test_size = 0.2, random_s
```

In [3]:
```
X_train = training_set.iloc[:,0:2].values
Y_train = training_set.iloc[:,2].values
X_test = test_set.iloc[:,0:2].values
Y_test = test_set.iloc[:,2].values
```

In [4]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(X_train,Y_train)
```

Out[4]:
```
SVC(random_state=1)
```

In [5]:
```
Y_pred = classifier.predict(X_test)
```

In [6]:
```
test_set["Predictions"] = Y_pred
```

In [7]:
```
test_set
```

Out[7]:

	Weight	Size	Class	Predictions
2	65	4.09	orange	apple
31	66	4.68	orange	apple
3	72	5.85	apple	apple
21	70	4.83	orange	apple
27	70	4.22	orange	apple
29	71	5.26	apple	apple
22	69	4.61	orange	apple
39	73	5.03	apple	apple

Calculating the accuracy of the predictions

In [8]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy Of SVM For The Given Dataset : ", accuracy)
```

```
Accuracy Of SVM For The Given Dataset :  0.375
```

Visualizing the classifier

Before we visualize we might need to encode the classes ‘apple’ and ‘orange’ into numericals.We can achieve that using the label encoder.

In [9]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y_train = le.fit_transform(Y_train)
```

After encoding , fit the encoded data to the SVM

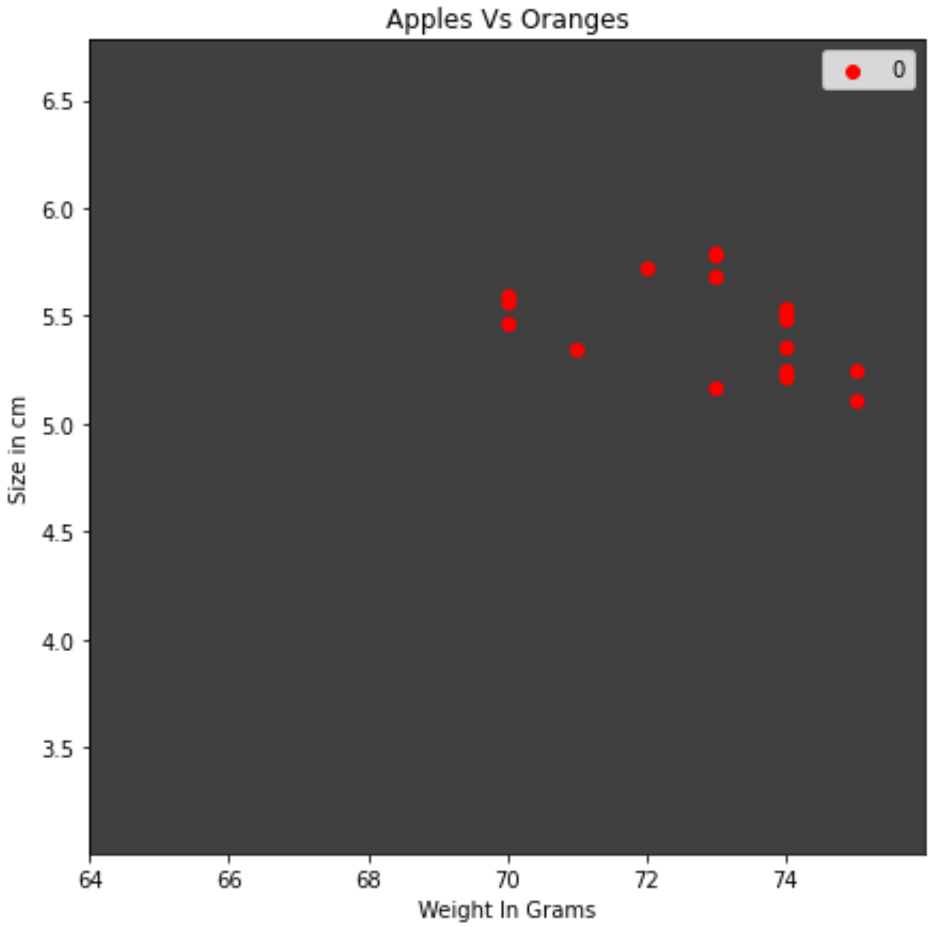
In [10]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(X_train,Y_train)
```

Out[10]:
```
SVC(random_state=1)
```

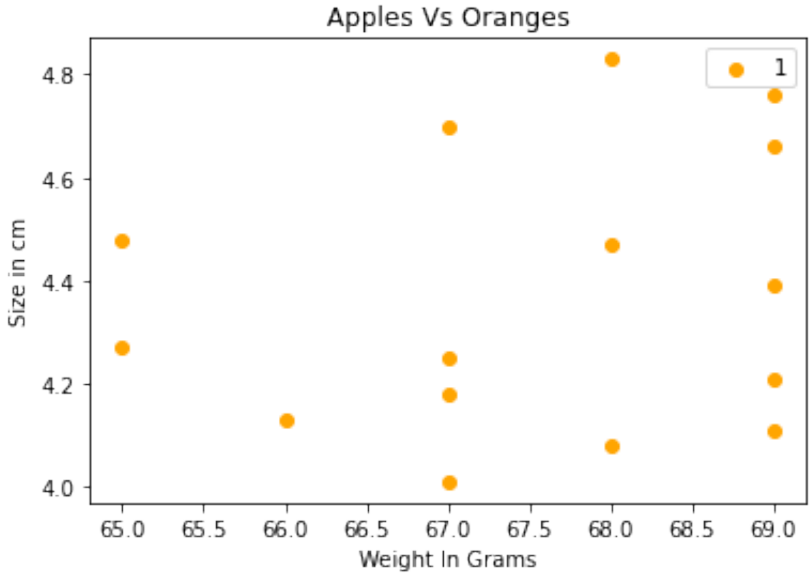
Let’s Visualize!

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
plt.figure(figsize = (7,7))
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_se
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedCol
    plt.title('Apples Vs Oranges')
    plt.xlabel('Weight In Grams')
    plt.ylabel('Size in cm')
    plt.legend()
    plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGB
A value for all points.
```



```
*c* argument looks like a single numeric RGB or RGBA sequence, which shou
ld be avoided as value-mapping will have precedence in case its length ma
tches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGB
A value for all points.
```


Apples Vs Oranges
● 1
Size in cm
Weight In Grams