Exercise 13: Different linkage, different hierarchical clustering!

Now, perform a hierarchical clustering of the voting countries with `'single'` linkage, and compare the resulting dendrogram with the one in the video. Different linkage, different hierarchical clustering!

First, we need to do a little pre-processing to account for one of the Eurovision rules: countries are not allowed to vote for themselves.

Step 1: Load the DataFrame *(written for you)*

In [14]:
```
import pandas as pd

scores_df = pd.read_csv('eurovision-2016-televoting.csv', index_col=0)
country_names = list(scores_df.index)
```

Step 2: Display the DataFrame, and have a look. Each row represents a country that *voted*, while each column represents a country that *performed*.

Notice the NaN ("not-a-number") values. These correspond to missing scores in the original CSV file. These scores are missing because countries that performed were not allowed to vote for themselves.

```
In [15]: scores_df
```

```
Out[15]:
```

	Armenia	Australia	Austria	Azerbaijan	Belgium	Bulgaria	Croatia	Cyprus
From country								
Albania	2.0	12.0	0.0	0.0	0.0	8.0	0.0	0.0
Armenia	NaN	0.0	4.0	0.0	0.0	0.0	0.0	6.0
Australia	0.0	NaN	3.0	0.0	12.0	10.0	0.0	0.0
Austria	0.0	3.0	NaN	0.0	0.0	5.0	0.0	0.0
Azerbaijan	0.0	2.0	0.0	NaN	0.0	8.0	0.0	0.0
Belarus	7.0	1.0	0.0	8.0	0.0	4.0	0.0	0.0
Belgium	7.0	4.0	3.0	0.0	NaN	5.0	0.0	0.0
Bosnia & Herzegovina	0.0	3.0	5.0	8.0	0.0	2.0	10.0	0.0
Bulgaria	8.0	5.0	4.0	1.0	0.0	NaN	0.0	7.0
Croatia	0.0	5.0	6.0	0.0	0.0	1.0	NaN	0.0
Cyprus	8.0	5.0	0.0	0.0	0.0	12.0	0.0	NaN
Czech Republic	8.0	1.0	4.0	6.0	0.0	5.0	0.0	0.0
Denmark	0.0	10.0	1.0	0.0	8.0	0.0	0.0	0.0
Estonia	0.0	4.0	6.0	0.0	0.0	0.0	0.0	3.0
F.Y.R. Macedonia	7.0	3.0	0.0	0.0	4.0	10.0	5.0	0.0
Finland	0.0	7.0	6.0	0.0	0.0	4.0	0.0	1.0
France	12.0	0.0	8.0	0.0	4.0	5.0	0.0	0.0
Georgia	12.0	1.0	0.0	7.0	0.0	3.0	0.0	0.0
Germany	2.0	5.0	7.0	0.0	0.0	4.0	0.0	0.0
Greece	8.0	5.0	1.0	0.0	0.0	7.0	0.0	12.0
Hungary	0.0	3.0	6.0	0.0	0.0	4.0	0.0	5.0
Iceland	0.0	8.0	2.0	1.0	3.0	0.0	0.0	0.0
Ireland	0.0	6.0	1.0	0.0	0.0	5.0	0.0	0.0
Israel	6.0	5.0	3.0	2.0	1.0	7.0	0.0	0.0
Italy	1.0	0.0	0.0	0.0	0.0	7.0	0.0	6.0
Latvia	0.0	6.0	4.0	3.0	0.0	1.0	0.0	0.0
Lithuania	0.0	5.0	1.0	0.0	0.0	0.0	0.0	0.0

	Armenia	Australia	Austria	Azerbaijan	Belgium	Bulgaria	Croatia	Cyprus
From country								
Malta	0.0	12.0	0.0	6.0	0.0	8.0	0.0	0.0
Moldova	7.0	5.0	4.0	8.0	0.0	2.0	0.0	0.0
Montenegro	0.0	0.0	0.0	7.0	0.0	5.0	6.0	0.0
Norway	0.0	8.0	0.0	0.0	2.0	5.0	0.0	0.0
Poland	2.0	7.0	4.0	0.0	0.0	3.0	0.0	1.0
Russia	12.0	4.0	8.0	6.0	0.0	0.0	0.0	7.0
San Marino	2.0	5.0	0.0	0.0	0.0	3.0	0.0	0.0
Serbia	2.0	6.0	0.0	0.0	5.0	8.0	4.0	3.0
Slovenia	0.0	3.0	6.0	0.0	0.0	2.0	8.0	0.0
Spain	6.0	4.0	2.0	0.0	0.0	12.0	0.0	0.0
Sweden	0.0	12.0	5.0	0.0	0.0	4.0	0.0	0.0
Switzerland	0.0	1.0	10.0	0.0	0.0	0.0	0.0	0.0
The Netherlands	8.0	5.0	6.0	0.0	12.0	1.0	0.0	0.0
Ukraine	7.0	4.0	0.0	10.0	0.0	2.0	0.0	0.0
United Kingdom	0.0	6.0	0.0	0.0	0.0	8.0	0.0	2.0

42 rows × 26 columns

Step 3: Fill in the NaNs with the highest possible score (12) - we are assuming that countries would vote for themselves, if they had been allowed to do so. *(This bit written for you).*

In [16]:
```
scores_df=scores_df.fillna(12)
```

Step 4: Import the `normalize` function from `sklearn.preprocessing`.

In [17]:
```
from sklearn.preprocessing import normalize
```

Step 5: Apply the normalize function to `scores_df.values`, assigning the result to `samples`.

(Why do we need to normalize? Because now that the missing values are filled with 12 points, some countries (those that performed) given a greater total number of points when voting. The `normalize` function corrects for this.)

In [18]:
```
samples= normalize(scores_df)
```

Step 6: Import:

- `linkage` and `dendrogram` from `scipy.cluster.hierarchy`.
- `matplotlib.pyplot` as `plt`.

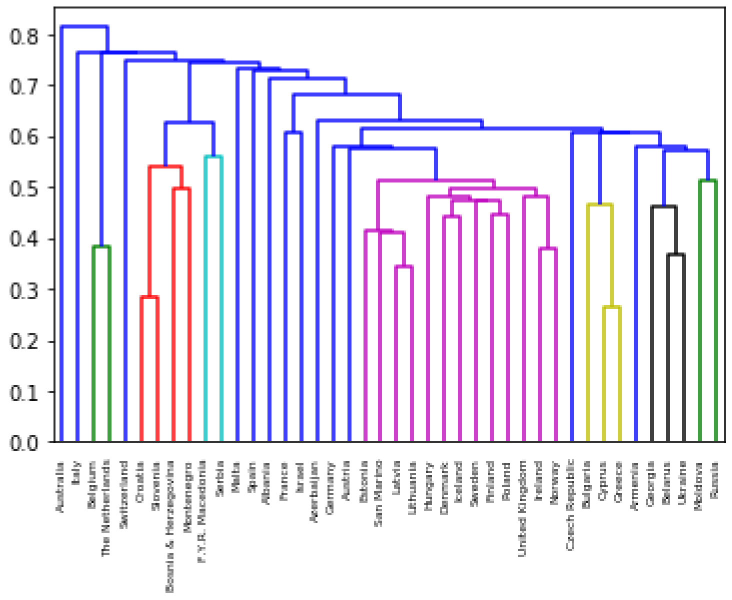
In [19]:
```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import linkage, dendrogram
```

Step 7: Perform hierarchical clustering on `samples` using the `linkage()` function with the `method='single'` keyword argument. Assign the result to `mergings`.

In [20]:
```
mergings=linkage(samples,method='single')
```

Step 8: Plot a dendrogram of the hierarchical clustering, using the list `country_names` as the `labels`. In addition, specify the `leaf_rotation=90`, and `leaf_font_size=6` keyword arguments as you have done earlier.

```
In [21]: dendrogram(mergings,labels=country_names,leaf_rotation=90,leaf_font_size=6)
         plt.show()
```



Step 9: Compare your dendrogram above to the one in the slides and notice that different linkage functions give different hierarchical clusterings.

Both the linkage functions we've considered, "complete" and "single", have advantages and disadvantages. In practice, just try both out, and see which dendrogram seems more sensible.