In [2]:
```
1 import pandas as pd
```

In [3]:
```
1 import matplotlib.pyplot as plt
```

In [4]:
```
1 # Import the CSV file
2 df = pd.read_csv("medals.csv")
```

In [5]:
```
1 df
```

Out[5]:

ıder	medal_type	participant_type	participant_title	athlete_url	athlete_full_name	country_name	country_code	country_
xed	GOLD	GameTeam	Italy	https://olympics.com/en/athletes/stefania-cons...	Stefania CONSTANTINI	Italy	IT	
xed	GOLD	GameTeam	Italy	https://olympics.com/en/athletes/amos-mosaner	Amos MOSANER	Italy	IT	
xed	SILVER	GameTeam	Norway	https://olympics.com/en/athletes/kristin-skaslien	Kristin SKASLIEN	Norway	NO	
xed	SILVER	GameTeam	Norway	https://olympics.com/en/athletes/magnus-nedreg...	Magnus NEDREGOTTEN	Norway	NO	
xed	BRONZE	GameTeam	Sweden	https://olympics.com/en/athletes/almida-de-val	Almida DE VAL	Sweden	SE	
...	...	...	...	...	...	...	...	
ıen	SILVER	Athlete	NaN	https://olympics.com/en/athletes/viggo-jensen	Viggo JENSEN	Denmark	DK	
ıen	BRONZE	Athlete	NaN	NaN	Alexandros Nikolopoulos	Greece	GR	
ıen	GOLD	Athlete	NaN	https://olympics.com/en/athletes/viggo-jensen	Viggo JENSEN	Denmark	DK	
ıen	SILVER	Athlete	NaN	https://olympics.com/en/athletes/launceston-el...	Launceston ELLIOT	Great Britain	GB	
ıen	BRONZE	Athlete	NaN	https://olympics.com/en/athletes/sotirios-versis	Sotirios VERSIS	Greece	GR	

In [6]:

```
# Analyze the discipline_title column
print(df["discipline_title"].value_counts())
```

```
discipline_title
Athletics            3080
Swimming             1763
Wrestling            1356
Rowing               1072
Boxing                996
                     ...
Jeu de Paume            3
Water Motorsports       3
Roque                   3
Cricket                 2
Basque Pelota           2
Name: count, Length: 86, dtype: int64
```

In [7]:

```
# Analyze the slug_game column
print(df["slug_game"].value_counts())
```

slug_game	
tokyo-2020	1188
rio-2016	1063
beijing-2008	1047
london-2012	1044
sydney-2000	1023
athens-2004	1022
atlanta-1996	917
barcelona-1992	887
seoul-1988	797
los-angeles-1984	730
moscow-1980	670
montreal-1976	652
munich-1972	630
mexico-city-1968	551
tokyo-1964	528
rome-1960	484
melbourne-1956	478
helsinki-1952	465
antwerp-1920	457
london-1948	438
paris-1924	395
berlin-1936	388
los-angeles-1932	358
beijing-2022	355
london-1908	343
amsterdam-1928	333
pyeongchang-2018	331
stockholm-1912	323
sochi-2014	314
paris-1900	292
st-louis-1904	290
vancouver-2010	279
turin-2006	273
salt-lake-city-2002	249
nagano-1998	217
lillehammer-1994	195
albertville-1992	183
calgary-1988	150
sarajevo-1984	129
lake-placid-1980	127

```
athens-1896                     126
innsbruck-1976                  123
grenoble-1968                   115
innsbruck-1964                  114
sapporo-1972                    114
squaw-valley-1960                84
cortina-d-ampezzo-1956           78
st-moritz-1948                   74
oslo-1952                        73
garmisch-partenkirchen-1936      57
chamonix-1924                    52
lake-placid-1932                 48
st-moritz-1928                   44
Name: count, dtype: int64
```

In [8]:

```python
# Analyze the event_title column
print(df["event_title"].value_counts())

```

```
event_title
Individual men                           215
individual mixed                         192
team mixed                               176
doubles men                              168
1500m men                                162
                                        ...
8m rating 1907 mixed                       1
open class A men                           1
12m rating 1919 mixed                      1
fixed bird target large birds teams men    1
12m rating 1907 mixed                      1
Name: count, Length: 1436, dtype: int64
```

In [9]:
```
# Analyze the event_gender column
print(df["event_gender"].value_counts())
```

```
event_gender
Men      13932
Women     6323
Open       998
Mixed      444
Name: count, dtype: int64
```

In [10]:
```
# Analyze the medal_type column
print(df["medal_type"].value_counts())

```

```
medal_type
BRONZE    7529
GOLD      7109
SILVER    7059
Name: count, dtype: int64
```

In [11]:
```
# Analyze the participant_type column
print(df["participant_type"].value_counts())
```

```
participant_type
Athlete     15113
GameTeam     6584
Name: count, dtype: int64
```

In [12]:

```
# Analyze the participant_title column
print(df["participant_title"].value_counts())

```

```
participant_title
United States team        523
Germany team              337
Great Britain team        284
Soviet Union team         282
France team               247
                         ...
Eleda                       1
Latvia team                 1
Belarus                     1
Kitty #1                    1
New College, Oxford #2      1
Name: count, Length: 493, dtype: int64
```

In [13]:

```
# Analyze the athlete_url column
print(df["athlete_url"].value_counts())
```

```
athlete_url
https://olympics.com/en/athletes/michael-phelps-ii (https://olympics.com/en/athletes/michael-phelps-ii)    16
https://olympics.com/en/athletes/larisa-latynina (https://olympics.com/en/athletes/larisa-latynina)      14
https://olympics.com/en/athletes/nikolay-andrianov (https://olympics.com/en/athletes/nikolay-andrianov)    12
https://olympics.com/en/athletes/marit-bjoergen (https://olympics.com/en/athletes/marit-bjoergen)       12
https://olympics.com/en/athletes/ireen-wust (https://olympics.com/en/athletes/ireen-wust)           10
                                                                                                    ..
https://olympics.com/en/athletes/alberto-costa (https://olympics.com/en/athletes/alberto-costa)         1
https://olympics.com/en/athletes/kristie-boogert (https://olympics.com/en/athletes/kristie-boogert)      1
https://olympics.com/en/athletes/miriam-oremans (https://olympics.com/en/athletes/miriam-oremans)        1
https://olympics.com/en/athletes/els-callens (https://olympics.com/en/athletes/els-callens)           1
https://olympics.com/en/athletes/paul-ereng (https://olympics.com/en/athletes/paul-ereng)             1
Name: count, Length: 12116, dtype: int64
```

In [14]:
```
# Analyze the athlete_full_name column
print(df["athlete_full_name"].value_counts())
```

```
athlete_full_name
Michael PHELPS             16
Larisa LATYNINA            14
Marit BJOERGEN             12
Nikolay ANDRIANOV          12
Boris SHAKHLIN             10
                           ..
Seon-Yong Jeong             1
Marisabel LOMBA             1
Djamel BOURAS               1
Soso LIPARTELIANI           1
Alexandros Nikolopoulos     1
Name: count, Length: 12895, dtype: int64
```

In [15]:
```
# Analyze the country_name column
print(df["country_name"].value_counts())
```

```
country_name
United States of America    3094
Soviet Union                1272
Germany                     1167
Great Britain               1045
France                       952
                            ...
Eritrea                       1
Paraguay                      1
Burkina Faso                  1
Tonga                         1
Turkmenistan                  1
Name: count, Length: 154, dtype: int64
```

In [16]:

```
# Analyze the country_code column
print(df["country_code"].value_counts())
```

```
country_code
US    3094
DE    1433
GB    1045
FR     952
CN     807
      ...
MU       1
SD       1
PY       1
WS       1
ME       1
Name: count, Length: 143, dtype: int64
```

In [17]:

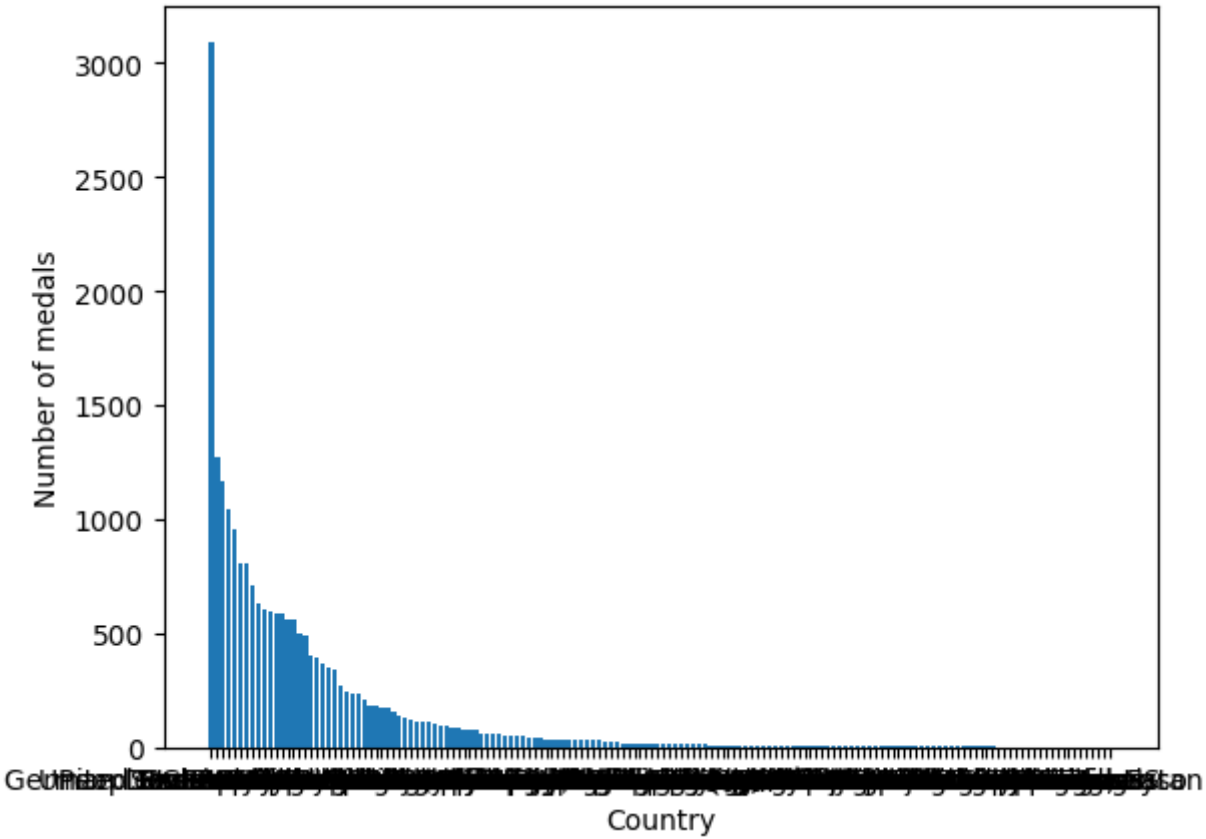
```
# Analyze the country_3_letter_code column
print(df["country_3_letter_code"].value_counts())

```

```
country_3_letter_code
USA    3094
URS    1272
GER    1167
GBR    1045
FRA     952
       ...
ERI       1
PAR       1
BUR       1
TGA       1
TKM       1
Name: count, Length: 154, dtype: int64
```

In [18]:

```
# Plot a bar chart of the number of medals won by each country
plt.bar(df["country_name"].value_counts().index, df["country_name"].value_counts())
plt.xlabel("Country")
plt.ylabel("Number of medals")
plt.show()
```



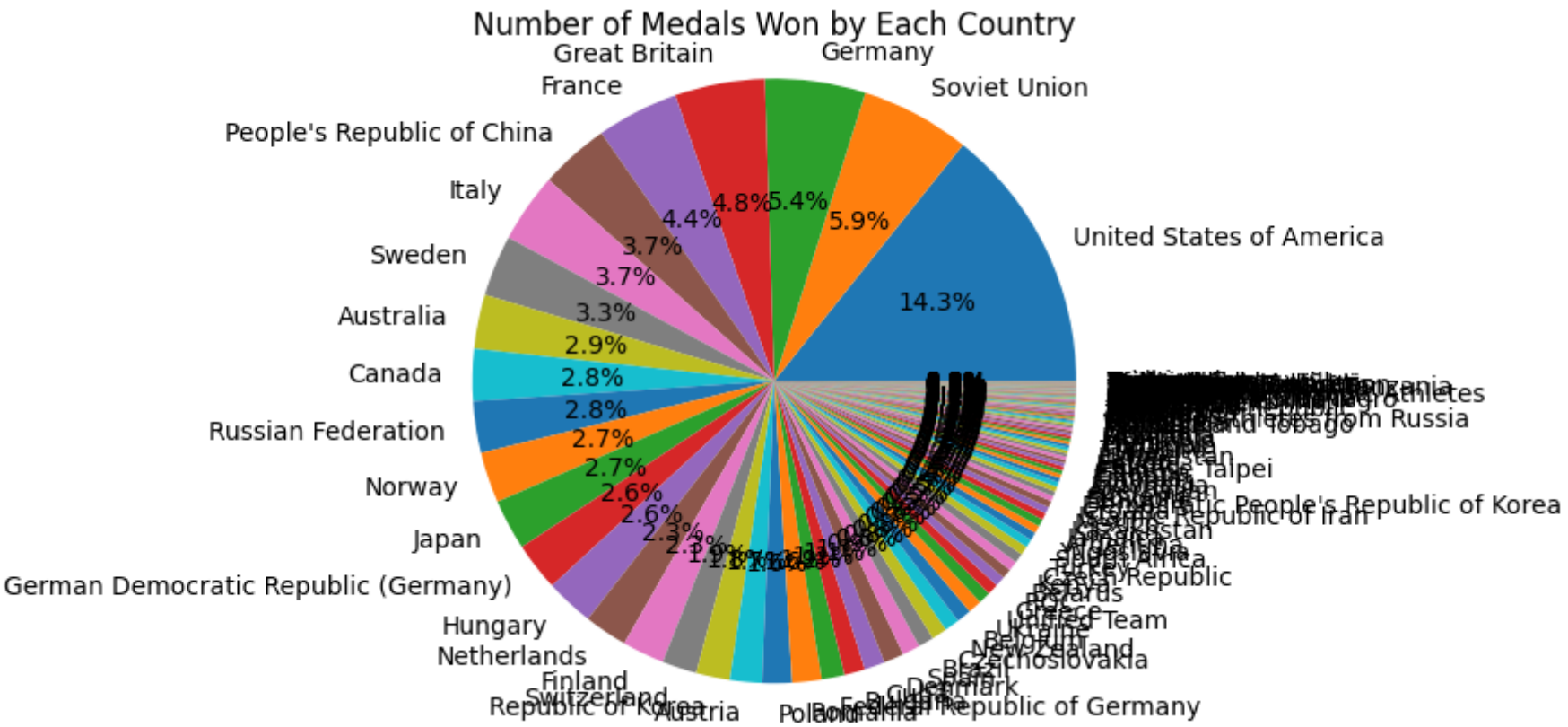
In [19]:

```python
import matplotlib.pyplot as plt

# Calculate the number of medals won by each country
medal_counts = df["country_name"].value_counts()

# Plot a pie chart of the number of medals won by each country
plt.pie(medal_counts, labels=medal_counts.index, autopct='%1.1f%%')
plt.title("Number of Medals Won by Each Country")
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()

```



In [25]:

```
1 pip install scikit-learn
```

```
0.0/9.2 MB 10.2 kB/s eta 0:14:55
0.0/9.2 MB 10.2 kB/s eta 0:14:55
0.0/9.2 MB 13.0 kB/s eta 0:11:43
0.1/9.2 MB 21.0 kB/s eta 0:07:15
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 24.7 kB/s eta 0:06:09
0.1/9.2 MB 40.1 kB/s eta 0:03:47
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
0.1/9.2 MB 46.6 kB/s eta 0:03:15
```

In [26]:

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Read the CSV file
data = pd.read_csv('medals.csv')

# Split the data into input features (X) and target variable (y)
X = data[['medal_type']]
y = data['country_name']

# Convert categorical variables to numerical using one-hot encoding
X = pd.get_dummies(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

```

```
Accuracy: 0.14147465437788018
```

```
C:\Users\rosha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```