Import the packages we'll use throughout the course.

In [1]:
```
import numpy
import scipy
import pandas
import matplotlib.pyplot as plt
import sklearn
```

Read the data and check its contents using pandas. The csv file contains information about books from the British Library

In [2]:
```
df = pandas.read_csv('BL-Flickr-Images-Book.csv')
# df = pandas.excel('BL-Flickr-Images-Book.xlsx')  # if we had an excel file
df.head()
```

Out[2]:

	Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author
0	206	NaN	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.
1	216	NaN	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.
2	218	NaN	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.
3	472	NaN	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.
4	480	A new edition, revised, etc.	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.

Let's get rid of the columns with too many null/NaN values. First we need to check them using:

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8287 entries, 0 to 8286
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Identifier              8287 non-null   int64  
 1   Edition Statement       773 non-null    object 
 2   Place of Publication    8287 non-null   object 
 3   Date of Publication     8106 non-null   object 
 4   Publisher               4092 non-null   object 
 5   Title                   8287 non-null   object 
 6   Author                  6509 non-null   object 
 7   Contributors            8287 non-null   object 
 8   Corporate Author        0 non-null      float64
 9   Corporate Contributors  0 non-null      float64
 10  Former owner            1 non-null      object 
 11  Engraver                0 non-null      float64
 12  Issuance type           8287 non-null   object 
 13  Flickr URL              8287 non-null   object 
 14  Shelfmarks              8287 non-null   object 
dtypes: float64(3), int64(1), object(11)
memory usage: 971.3+ KB
```

From this output, we see that our dataframe consists of 8287 rows and 15 columns. We can also see the content of each column, getting rid of the useless columns:

In [4]:

```python
columns_to_remove = ['Edition Statement', 'Corporate Author',
'Corporate Contributors', 'Former owner', 'Engraver']
df.drop(columns_to_remove, inplace=True, axis=1)
df.head()
```

Out[4]:

	Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors
0	206	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.
1	216	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness
2	218	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of “All for Gr...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness
3	472	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	Appleyard, Ernest Silvanus.
4	480	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	BROOME, John Henry.

Now we'll visualize the publication dates of the books using matplotlib library. We see that publication date is of type 'object', which will become problematic since we want to have numbers (int or float).

We can deal with this issue by converting the column to numeric, resulting in non-numeric entries to be NaN, then filtering the rows with NaN values in that column. Remember to call inplace=True, otherwise the function doesn't overwrite the dataframe and just returns the output. It's useful in cases where we need to create 'views' of the dataframe while preserving the original data.

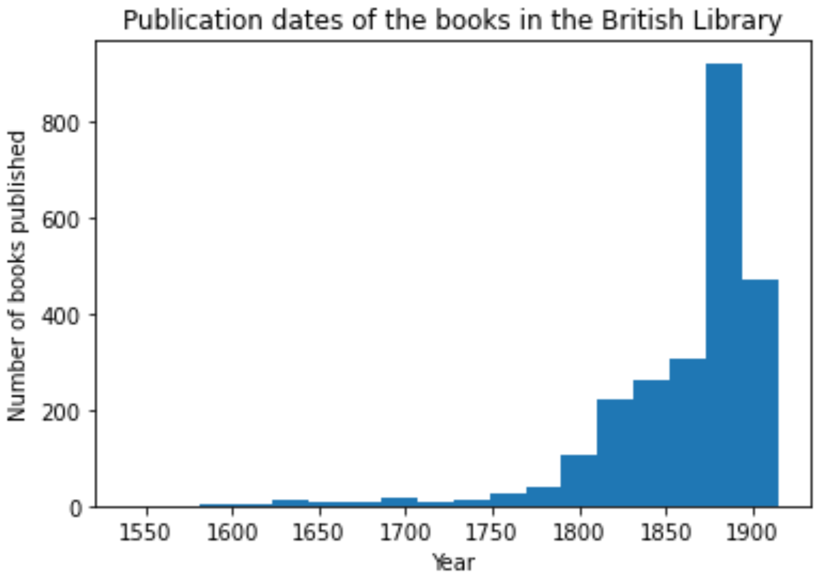
In [5]:
```
df['Date of Publication'] = pandas.to_numeric(df['Date of Publ
ication'], errors='coerce')
df.dropna(how='any', inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2440 entries, 1 to 8285
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Identifier            2440 non-null   int64
 1   Place of Publication  2440 non-null   object
 2   Date of Publication   2440 non-null   float64
 3   Publisher             2440 non-null   object
 4   Title                 2440 non-null   object
 5   Author                2440 non-null   object
 6   Contributors          2440 non-null   object
 7   Issuance type         2440 non-null   object
 8   Flickr URL            2440 non-null   object
 9   Shelfmarks            2440 non-null   object
dtypes: float64(1), int64(1), object(8)
memory usage: 209.7+ KB
```

Now we have filtered rows containing any NaN value and are left with a numeric date of publication column. Let's visualize:

```
In [6]: fig, ax = plt.subplots()
        ax.hist(df['Date of Publication'], bins=18)
        # beautify the plot
        ax.set_title('Publication dates of the books in the British Library')
        ax.set_xlabel('Year')
        ax.set_ylabel('Number of books published')
```

```
Out[6]: Text(0, 0.5, 'Number of books published')
```



Now let's consider the other dataset, olympics.csv:

```
In [7]: df = pandas.read_csv('olympics.csv', header=1)
        df.head()
```

Out[7]:

	Unnamed: 0	? Summer	01 !	02 !	03 !	Total	? Winter	01 !.1	02 !.1	03 !.1	Total.1	? Games	01 !.2
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0
1	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5
2	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18
3	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3

With header=1, we indicate reader to construct the column names using row 1 (instead of 0).

It contains the number of medals won by countries in the summer and winter olympics. We can clearly see that these column names won't work properly, so let's rename them:

In [8]:
```python
# new names dictionary, in the following format old_name: new_
name
new_names =  {'Unnamed: 0': 'Country',
              '? Summer': 'Summer Olympics',
              '01 !': 'Gold',
              '02 !': 'Silver',
              '03 !': 'Bronze',
              '? Winter': 'Winter Olympics',
              '01 !.1': 'Gold.1',
              '02 !.1': 'Silver.1',
              '03 !.1': 'Bronze.1',
              '? Games': '# Games',
              '01 !.2': 'Gold.2',
              '02 !.2': 'Silver.2',
              '03 !.2': 'Bronze.2'}
# renaming function
df.rename(columns=new_names, inplace=True)
# check if it works
df.head()
```

Out[8]:

	Country	Summer Olympics	Gold	Silver	Bronze	Total	Winter Olympics	Gold.1	Silver.1	Br
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	
1	Algeria (ALG)	12	5	2	8	15	3	0	0	
2	Argentina (ARG)	23	18	24	28	70	18	0	0	
3	Armenia (ARM)	5	1	2	9	12	6	0	0	
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	

With the columns renamed, let's find the outliers among the Bronze winners using z-scores, calculation using scipy:

$$Z = \frac{x - \mu}{\sigma}$$

In [10]:
```python
from scipy import stats
z_scores = stats.zscore(df['Gold'])
z_scores
```

Out[10]:
```
array([-1.61884505e-01, -1.49513419e-01, -1.17348593e-01, -1.
59410288e-01,
       -1.54461853e-01,  1.82031703e-01, -1.17348593e-01, -1.
47039201e-01,
       -1.49513419e-01, -1.61884505e-01, -1.61884505e-01, -1.
32193897e-01,
       -7.03384641e-02, -1.61884505e-01, -1.61884505e-01, -1.
61884505e-01,
       -1.04977507e-01, -1.61884505e-01, -3.56994215e-02, -1.
59410288e-01,
       -1.54461853e-01, -1.59056828e-02, -1.56936071e-01,  3.
35433178e-01,
       -1.56936071e-01, -1.59410288e-01, -1.61884505e-01, -1.
47039201e-01,
        1.62591424e-02, -1.61884505e-01, -1.27245463e-01, -4.
06478561e-02,
       -5.54931601e-02, -1.61884505e-01, -1.54461853e-01, -1.
59410288e-01,
       -1.44564984e-01, -1.61884505e-01, -1.39616549e-01, -1.
09925941e-01,
        8.80114450e-02,  3.37907395e-01, -1.61884505e-01, -1.
47039201e-01,
        2.68629310e-01, -9.26064200e-02,  2.16670746e-01, -2.
33283348e-02,
       -1.61884505e-01,  4.22030784e-01, -8.76579854e-02, -1.
59410288e-01,
       -1.61884505e-01, -1.61884505e-01, -1.61884505e-01, -1.
59410288e-01,
        2.51309789e-01, -1.61884505e-01, -1.39616549e-01, -1.
47039201e-01,
       -1.24771245e-01, -1.61884505e-01, -1.39616549e-01, -1.
59410288e-01,
        3.28010526e-01, -1.19822811e-01,  1.59763748e-01, -1.
22297028e-01,
       -1.00029072e-01, -1.27245463e-01,  3.85270984e-02, -1.
61884505e-01,
       -1.61884505e-01, -1.54461853e-01, -1.61884505e-01, -1.
61884505e-01,
       -1.47039201e-01, -1.59410288e-01, -1.61884505e-01, -1.
61884505e-01,
       -1.61884505e-01, -1.29719680e-01, -1.61884505e-01, -1.
56936071e-01,
       -1.61884505e-01, -1.47039201e-01, -1.59410288e-01, -1.
61884505e-01,
        2.86302291e-02, -1.61884505e-01, -5.79673774e-02, -1.
61884505e-01,
       -1.54461853e-01, -2.33283348e-02, -1.54461853e-01, -1.
59410288e-01,
```

```
        -1.61884505e-01, -1.59410288e-01, -1.61884505e-01, -3.
53459618e-03,
        -1.51987636e-01, -1.61884505e-01, -1.61884505e-01,  5.
58466197e-02,
         1.64712182e-01, -1.59410288e-01,  8.15431340e-01, -5.
05447254e-02,
        -1.61884505e-01, -1.61884505e-01, -1.59410288e-01, -1.
56936071e-01,
        -1.61884505e-01, -1.44564984e-01, -1.51987636e-01, -1.
04977507e-01,
        -7.03384641e-02, -1.61884505e-01, -1.61884505e-01, -1.
59410288e-01,
         1.91928573e-01, -4.55962908e-02, -1.59410288e-01, -1.
56936071e-01,
        -1.61884505e-01, -1.61884505e-01, -1.44564984e-01, -1.
61884505e-01,
        -1.61884505e-01, -1.56936071e-01, -1.54461853e-01, -6.
53900294e-02,
        -1.56936071e-01, -8.02353334e-02, -1.59410288e-01,  2.
25295161e+00,
        -1.56936071e-01, -1.49513419e-01, -1.56936071e-01, -1.
61884505e-01,
        -1.61884505e-01, -9.75548547e-02, -1.61884505e-01, -1.
61884505e-01,
        -1.54461853e-01, -1.42090767e-01,  1.17366266e+01])
```

Usually a z-score higher than 3 (or lower than -3) indicates an outlier, so let's find which rows satisfy that condition using numpy.where:

```
In [18]: outlier_indices = numpy.where(numpy.logical_or(z_scores > 3, z
         _scores < -3))
         outlier_indices
```

```
Out[18]: (array([146], dtype=int64),)
```

So row 146 is an outlier in terms of z-score, let's check its values:

```
In [23]: df.loc[outlier_indices[0], :]
```

```
Out[23]:
```

	Country	Summer Olympics	Gold	Silver	Bronze	Total	Winter Olympics	Gold.1	Silver.1	Bı
146	Totals	27	4809	4775	5130	14714	22	959	958	

As the totals row, it makes sense that it's an outlier. In an analysis, we would have to get rid of this row at the beginning, as follows:

In [37]:
```
outliers_dropped_df = df.drop(outlier_indices[0])
outliers_dropped_df
```

Out[37]:

	Country	Summer Olympics	Gold	Silver	Bronze	Total	Winter Olympics	Gold.1	Silver.1
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0
1	Algeria (ALG)	12	5	2	8	15	3	0	0
2	Argentina (ARG)	23	18	24	28	70	18	0	0
3	Armenia (ARM)	5	1	2	9	12	6	0	0
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0
...	...	...	...	...	...	...	...	...	...
141	Yugoslavia (YUG) [YUG]	16	26	29	28	83	14	0	3
142	Independent Olympic Participants (IOP) [IOP]	1	0	1	2	3	0	0	0
143	Zambia (ZAM) [ZAM]	12	0	1	1	2	0	0	0
144	Zimbabwe (ZIM) [ZIM]	12	3	4	1	8	1	0	0
145	Mixed team (ZZX) [ZZX]	3	8	5	4	17	0	0	0

146 rows × 16 columns