Dependencies or libraries are pre-written code to help solve problems. In this video, we will introduce Pandas, a popular library for data analysis.

We can import the library or a dependency like Pandas using the following command.

Import pandas as pd

We start with the import command followed by the name of the library. We now have access to a large number of pre-built classes and functions. This assumes the library is installed. In our lab environment, all the necessary libraries are installed.

Let's say we would like to load a CSV file using the Pandas built-in function, read csv. A CSV is a typical file type used to store data. We simply typed the word Pandas, then a dot, and the name of the function with all the inputs.

Typing Pandas all the time may get tedious. We can use the as statement to shorten the name of the library. In this case, we use the standard abbreviation, pd. Now we type pd, and a dot, followed by the name of the function we would like to use. In this case, read\_csv.

We are not limited to the abbreviation pd. In this case, we use the term banana. We will stick with pd for the rest of this video. Let's examine this code more in-depth.

**Data Frame**

One way Pandas allows you to work with data is with the data frame. Let's go over the process to go from a CSV file to a data frame. This variable stores the path of the CSV. It is used as an argument to the read\_csv function.

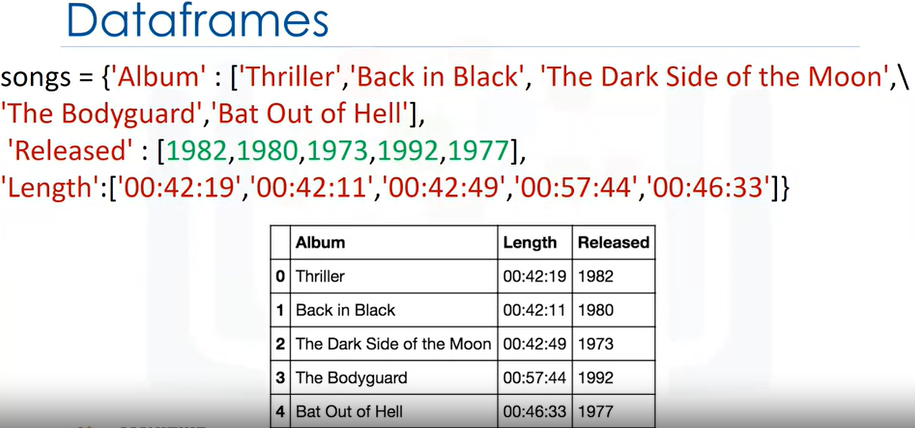
The result is stored to the variable df. This is short for data frame. Now that we have the data in a data frame, we can work with it.

**Head**

We can use the method head to examine the first five rows of a data frame. The process for loading an Excel file is similar. We use the path of the Excel file. The function reads Excel. The result is a data frame.

**Dictionary**

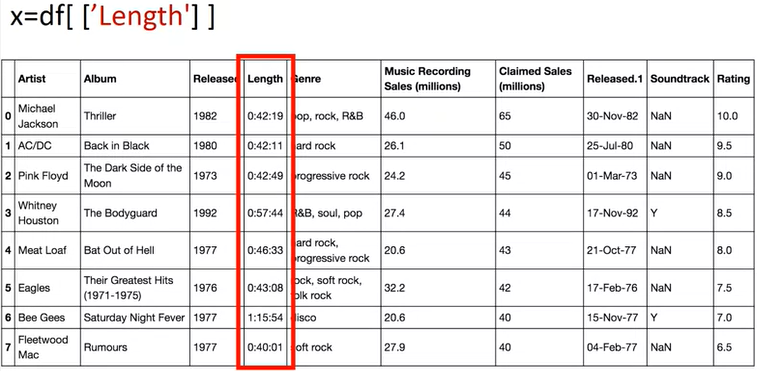
A data frame is comprised of rows and columns. We can create a data frame out of a dictionary. The keys correspond to the column labels. The values or lists corresponding to the rows. We then cast the dictionary to a data frame using the function data frame.



We can see the direct correspondence between the table. The keys correspond to the table headers.

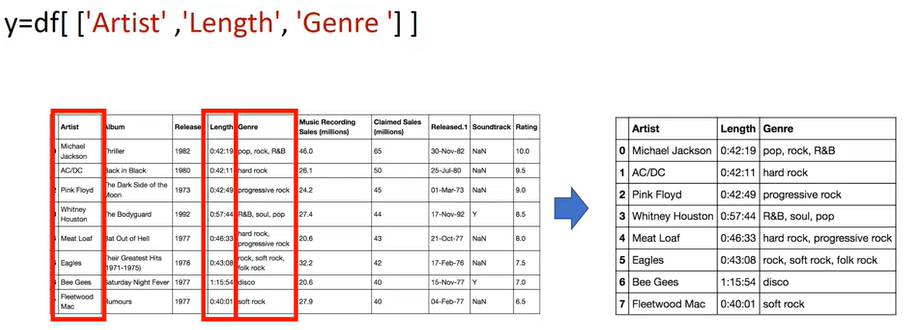


The values are lists corresponding to the rows. We can create a new data frame consisting of one column.



We just put the data frame name, in this case, df, and the name of the column header enclosed in double brackets. The result is a new data frame comprised of the original column.

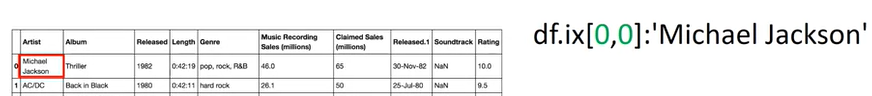
You can do the same thing for multiple columns. We just put the data frame name, in this case, df, and the name of the multiple column headers enclosed in double brackets. The result is a new data frame comprised of the specified columns.



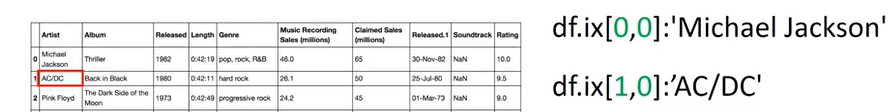
**“ix” method**

One way to access unique elements is the ix method.

You can access the first row and first column as follows.



You could access the second row and first column as follows.



You can access the first row, third column as follows.

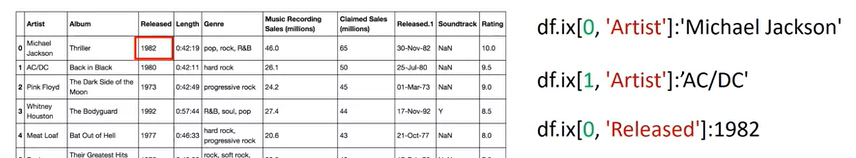


You can access the second row, third column as follows.

When you use the name of the column as well, you can access the first row of the column named Artist as follows.

Similarly, you can access the second row of the column named Artist. You can access the first row of the column named Released as follows.

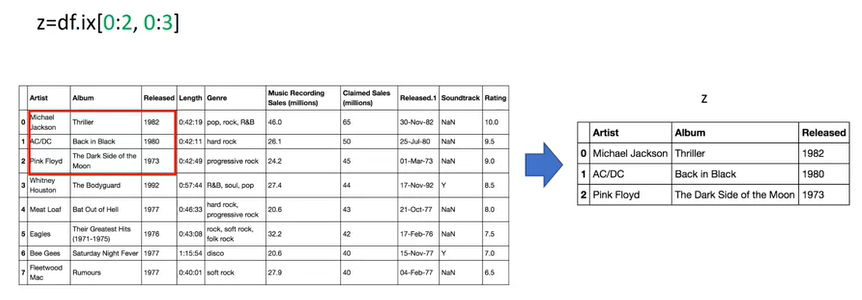
Finally, you can access the second row of the column named Released.



**Slicing Data**

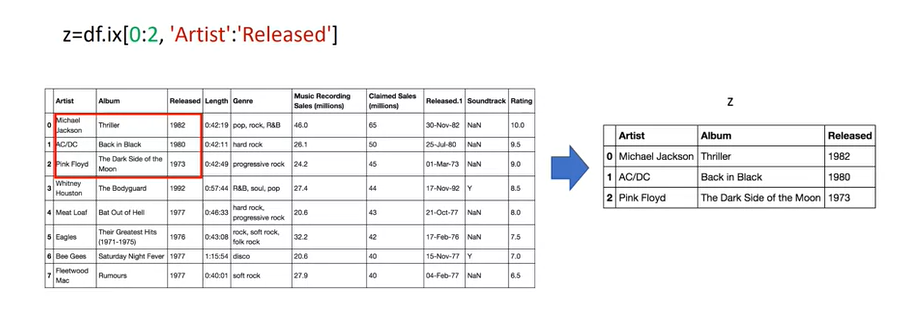
You can also slice data frames and **assign the values to a new data frame**.

We assign the first two rows and the first three columns to the variable Z.



The result is a data frame comprised of the selected rows and columns.

You can also slice data frames and assign the values to a new data frame using the column names.



The code assigns the first three rows and all columns in between to the columns named Artist and Released. The result is a new data frame Z with the corresponding values.