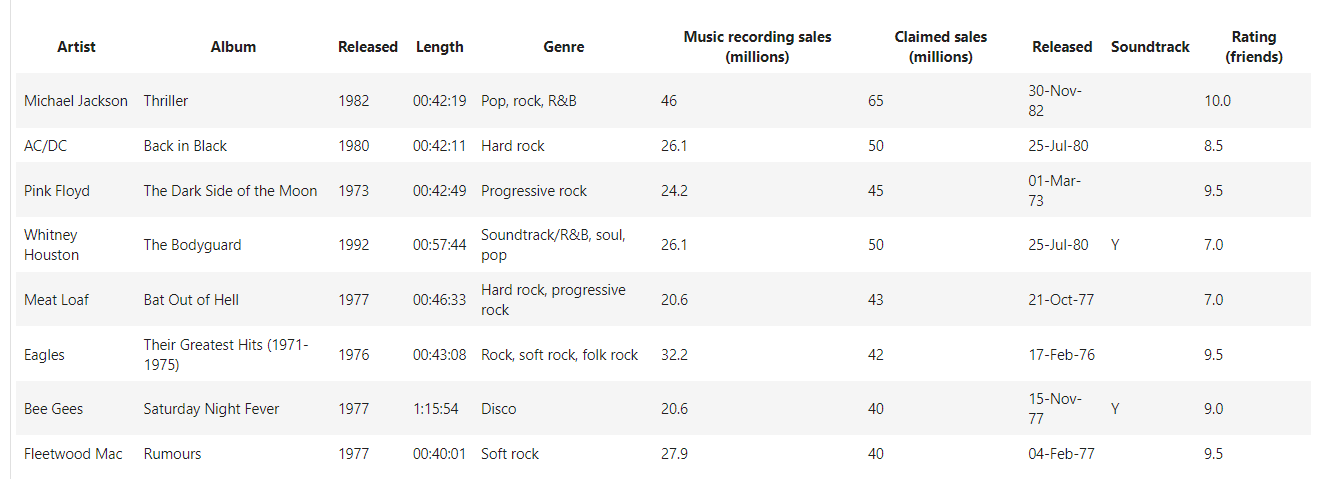
**Introduction to Pandas in Python**

## About the Dataset

The table has one row for each album and several columns.

* **artist**: Name of the artist
* **album**: Name of the album
* **released\_year**: Year the album was released
* **length\_min\_sec**: Length of the album (hours,minutes,seconds)
* **genre**: Genre of the album
* **music\_recording\_sales\_millions**: Music recording sales (millions in USD) on [[SONG://DATABASE]](http://www.song-database.com/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0101ENSkillsNetwork1005-2022-01-01)
* **claimed\_sales\_millions**: Album's claimed sales (millions in USD) on [[SONG://DATABASE]](http://www.song-database.com/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0101ENSkillsNetwork1005-2022-01-01)
* **date\_released**: Date on which the album was released
* **soundtrack**: Indicates if the album is the movie soundtrack (Y) or (N)
* **rating\_of\_friends**: Indicates the rating from your friends from 1 to 10

You can see the dataset here:



## Introduction of Pandas

*# Dependency needed to install file*

!pip install xlrd

!pip install openpyxl

Requirement already satisfied: xlrd in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.2.0)

Collecting openpyxl

Downloading openpyxl-3.1.2-py2.py3-none-any.whl (249 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250.0/250.0 kB 36.0 MB/s eta 0:00:00

Collecting et-xmlfile

Downloading et\_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)

Installing collected packages: et-xmlfile, openpyxl

Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.2

# Import required library

import pandas as pd

After the import command, 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. One way pandas allows you to work with data is a dataframe. Let's go through the process to go from a comma separated values (**.csv**) file to a dataframe. This variable csv\_path stores the path of the **.csv**, that is used as an argument to the read\_csv function. The result is stored in the object df, this is a common short form used for a variable referring to a Pandas dataframe.

*# Read data from CSV file*

csv\_path = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Module%204/data/TopSellingAlbums.csv'

df = pd.read\_csv(csv\_path)

We can use the method head() to examine the first five rows of a dataframe:

*# Print first five rows of the dataframe*

df.head()

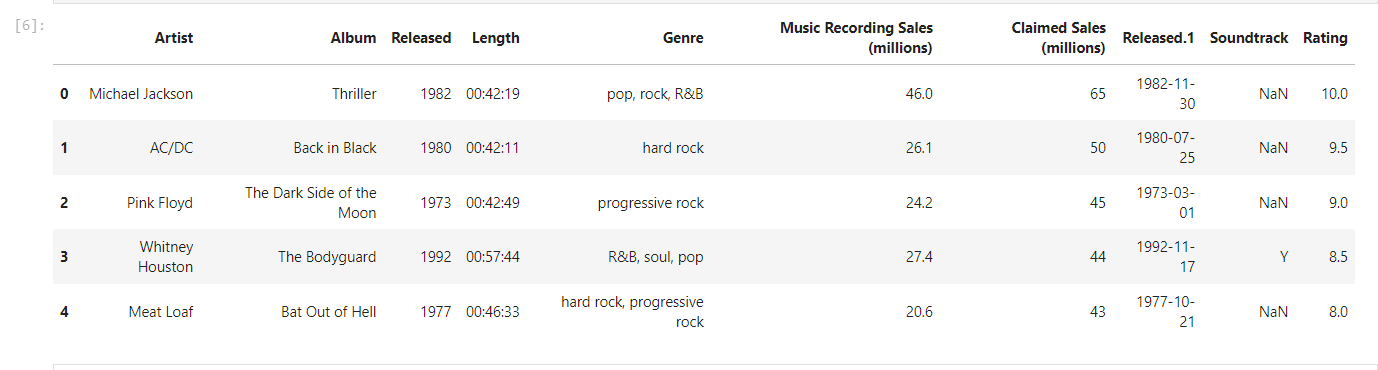
We use the path of the excel file and the function read\_excel. The result is a data frame as before:

*# Read data from Excel File and print the first five rows*

xlsx\_path = 'https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/PY0101EN/Chapter%204/Datasets/TopSellingAlbums.xlsx'

df = pd.read\_excel(xlsx\_path)

df.head()



We can access the column **Length** and assign it a new dataframe **x**:

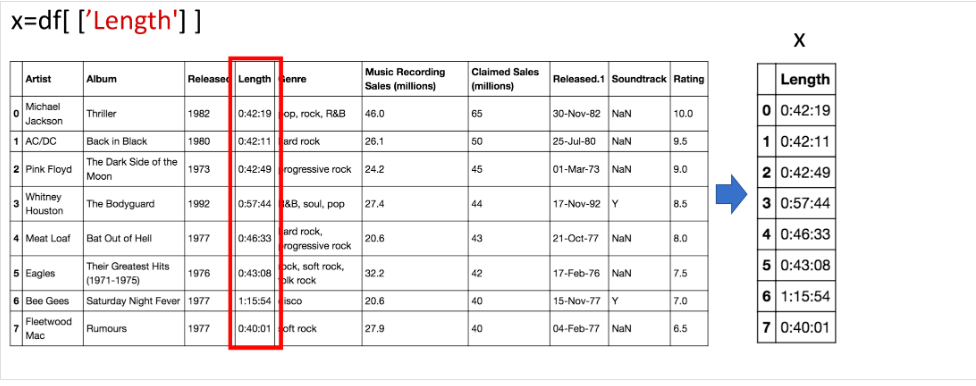
*# Access to the column Length*

x = df[['Length']]

x



The process is shown in the figure:



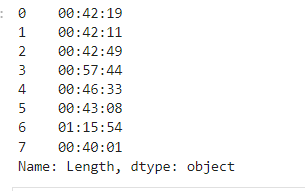
## Viewing Data and Accessing Data

You can also get a column as a series. You can think of a Pandas series as a 1-D dataframe. Just use one bracket:

*# Get the column as a series*

x = df['Length']

x



You can also get a column as a dataframe. For example, we can assign the column **Artist**:

*# Get the column as a dataframe*

x = df[['Artist']]

type(x)

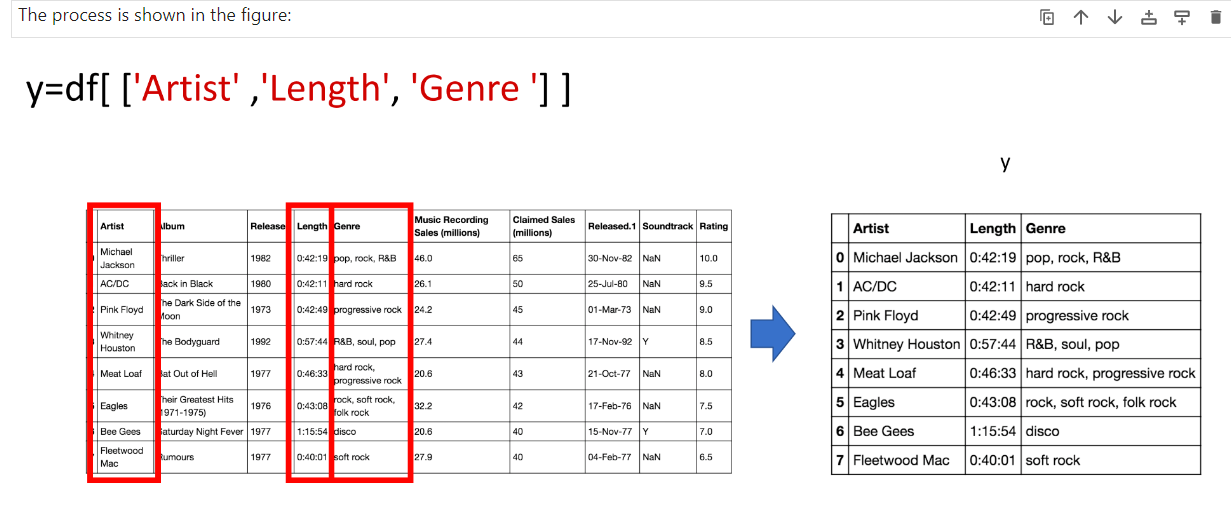
You can do the same thing for multiple columns; we just put the dataframe name, in this case, df, and the name of the multiple column headers enclosed in double brackets. The result is a new dataframe comprised of the specified columns:

*# Access to multiple columns*

y = df[['Artist','Length','Genre']]

y





One way to access unique elements is the iloc method, where you can access the 1st row and the 1st column as follows:

*# Access the value on the first row and the first column*

df.iloc[0, 0]

'Michael Jackson'

You can access the 2nd row and the 1st column as follows:

*# Access the value on the second row and the first column*

df.iloc[1,0]

'AC/DC'

You can access the 1st row and the 3rd column as follows:

*# Access the value on the first row and the third column*

df.iloc[0,2]

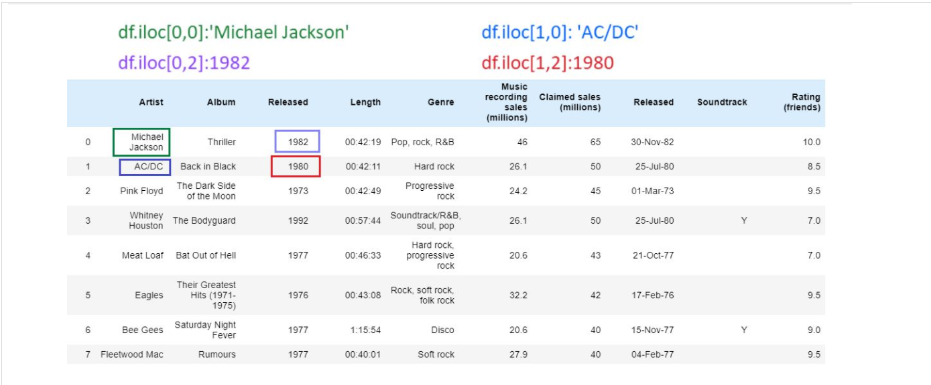
1982

*# Access the value on the second row and the third column*

df.iloc[1,2]

1980

This is shown in the following image



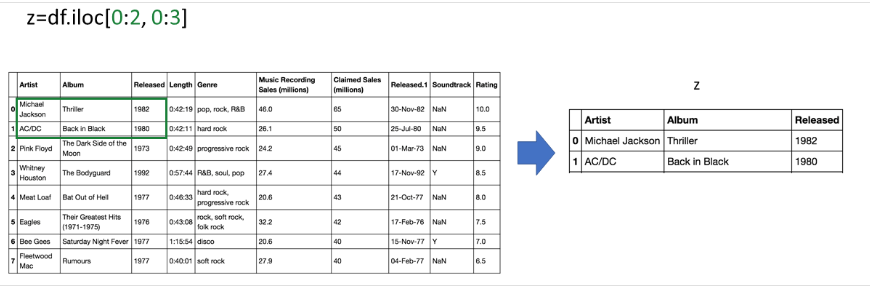
You can access the column using the name as well, the following are the same as above:



You can perform slicing using both the index and the name of the column:

*# Slicing the dataframe*

df.iloc[0:2, 0:3]



*# Slicing the dataframe using name*

df.loc[0:2, 'Artist':'Released']

