Pandas

In this notebook, we introduce you to Pandas which provides data structures and analysis tools for Data Science. Much of what we'll cover here involves the DataFrame data structure. As before, refer to documentation as and when you need to: https://pandas.pydata.org/ (https://pandas.pydata.org/)

Getting started with Pandas

Start by loading the `pandas` library (with alias `pd`).

In [1]:

```
# import the library
import pandas as pd
```

Load the provided dataset `data/airfoil.csv` using the `pd.read_csv()` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) function; call the DataFrame you load `df` .

In [2]:

```
# load the dataframe, use "head" to have a look
df = pd.read_csv("data/airfoil.csv")
```

Note that this `read_csv()` function is very flexible and can accomodate all sorts of file. You will do this in much more details in module 2. For now, we're giving you a nicely formatted dataset that directly works well with Pandas.

`df` is a pandas DataFrame object. DataFrame objects have many methods and attributes which make data science easier! Use the `.head()` method to show what `df` looks like.

In [3]:

```
# Use .head() on df
df.head()
```

Out[3]:

	Frequency [Hz]	Angle [deg]	Chord length [m]	FS velocity [m/s]	SSD thickness [m]	Sound pressure [dB]
0	800	0.0	0.3048	71.3	0.002663	126.201
1	1000	0.0	0.3048	71.3	0.002663	125.201
2	1250	0.0	0.3048	71.3	0.002663	125.951
3	1600	0.0	0.3048	71.3	0.002663	127.591
4	2000	0.0	0.3048	71.3	0.002663	127.461

Let's load another dataset into another dataframe and call it `smalldf` . The data is located at `data/smalldata.csv` .

First, have a look at the data yourself. If you're on a unix system, you can do this from the command line like this:

In [4]:

```
# Edit this if you're on Windows
!cat data/smalldata.csv  # note that this cell is runnable inside the notebook!
```

```
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

See what happens if you just run pd.read_csv() on it

In [5]:

```
# read it with pandas (don't save it, just allow it to print)
pd.read_csv('data/smalldata.csv')
```

Out[5]:

	F1	F2	F3	ID
0	15	70	M	Bob
1	18	50	F	Alice
2	15	55	F	Maria
3	12	85	M	John
4	14	68	M	Kevin
5	20	110	M	Donald

DataFrames have column names and an index. Import the csv and save it as a DataFrame called `smalldf` . Check the documentation (or google!) and:

- make the column names: `['Age', 'Weight', 'Gender', 'Name']`
- specify `name` as the index

Display `smalldf` in the notebook and check it against what you did above.

In [6]:

```
# add your code here to load smalldf
smalldf = pd.read_csv("data/smalldata.csv",
                      names=['Age', 'Weight', 'Gender', 'Name'],
                      index_col='Name',
                      skiprows=1)
smalldf
```

Out[6]:

	Age	Weight	Gender
Name			
Bob	15	70	M
Alice	18	50	F
Maria	15	55	F
John	12	85	M
Kevin	14	68	M
Donald	20	110	M

Retrieving some basic information

Now that you have a DataFrame object you can explore it (by typing `df.<TAB>` in a cell, you'll see all the methods and attributes)!

Examples of useful attributes:

- `shape` stores the dimensions of the data frame
- `columns` stores the names of the columns
- `index` stores the names of the rows, by default pandas uses a range from 0 to the number of rows
- `dtypes` stores the `dtype` of each column

Show all of those, check it matches what you expected versus the output of `head` used earlier.

You can also use `df.describe()` to get a "description" of your data (per column, a number of standard statistics such as the mean, variance etc).

In [7]:

```
# add your code here to explore df's attributes
print(">> Shape attr:\n{}".format(df.shape))
print("\n>> Columns:\n{}".format(df.columns))
print("\n>> Index:\n{}".format(df.index))
print("\n>> Dtypes:\n{}".format(df.dtypes))
```

```
>> Shape attr:
(1503, 6)

>> Columns:
Index(['Frequency [Hz]', 'Angle [deg]', 'Chord length [m]',
       'FS velocity [m/s]', 'SSD thickness [m]', 'Sound pressure [dB]'],
      dtype='object')

>> Index:
RangeIndex(start=0, stop=1503, step=1)

>> Dtypes:
Frequency [Hz]           int64
Angle [deg]            float64
Chord length [m]       float64
FS velocity [m/s]      float64
SSD thickness [m]      float64
Sound pressure [dB]    float64
dtype: object
```

In [8]:

```
# use `df.describe()`
df.describe()
```

Out[8]:

	Frequency [Hz]	Angle [deg]	Chord length [m]	FS velocity [m/s]	SSD thickness [m]	Sound pressure [dB]
count	1503.000000	1503.000000	1503.000000	1503.000000	1503.000000	1503.000000
mean	2886.380572	6.782302	0.136548	50.860745	0.011140	124.835943
std	3152.573137	5.918128	0.093541	15.572784	0.013150	6.898657
min	200.000000	0.000000	0.025400	31.700000	0.000401	103.380000
25%	800.000000	2.000000	0.050800	39.600000	0.002535	120.191000
50%	1600.000000	5.400000	0.101600	39.600000	0.004957	125.721000
75%	4000.000000	9.900000	0.228600	71.300000	0.015576	129.995500
max	20000.000000	22.200000	0.304800	71.300000	0.058411	140.987000

Accessing elements in a dataframe

Let's get the value of the 1st column (`Age`) of the 3rd row (`Maria`) of `smalldf` . This can be done in many ways, the most convenient to you will depend on situation:

1. using `iloc`
2. using `loc`
3. each column of a `DataFrame` is a `Series` object: you can first get this then access relevant element

Try each of these methods below.

Note: remember that indexing in Python starts at 0.

In [9]:

```
# add your code here
print("1. {}".format(smalldf.iloc[2, 0]))
print("2. {}".format(smalldf.loc['Maria', 'Age']))
print("3. {}".format(smalldf['Age'][2]))
```

```
1. 15
2. 15
3. 15
```

Using loc for fancy selections

Using `.loc` , can you retrieve the sub-dataframe of `df` with all the columns whose name has strictly more than 15 characters? Call this `df2` .

In [10]:

```
# add your code here
df2 = df.loc[:, [e for e in df.columns if len(e)>15]]
df2.head()
```

Out[10]:

	Chord length [m]	FS velocity [m/s]	SSD thickness [m]	Sound pressure [dB]
0	0.3048	71.3	0.002663	126.201
1	0.3048	71.3	0.002663	125.201
2	0.3048	71.3	0.002663	125.951
3	0.3048	71.3	0.002663	127.591
4	0.3048	71.3	0.002663	127.461

Using `to_csv` , output `df2` as a tab separated file (not comma) and call the file `airfoil_2.dat` . (Open the file in an editor to check it matches what you expect).

In [11]:

```
df2.to_csv("airfoil_2.dat", sep='\t')
```

Working with a pd.Series

Retrieve the series corresponding to the sound pressure from the dataframe, display

- show the name of the series (it should be `Sound pressure [dB]`)
- show the shape of the series (it should be `(1503,)`)
- the mean and the median (resp. `124.84` and `125.72`)
- the mean of the squared values (it should be `15631.57`)
- Check [the documentation (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html) and try some other methods and attributes

In [12]:

```
# add your code here
sp = df['Sound pressure [dB]']
print("Name: {}".format(sp.name))
print("Shape: {}".format(sp.shape))
print("Mean: {0:.2f}, Median: {1:.2f}".format(sp.mean(), sp.median()))
print("Mean of sq. vals: {0:.2f}".format((sp**2).mean()))
```

```
Name: Sound pressure [dB]
Shape: (1503,)
Mean: 124.84, Median: 125.72
Mean of sq. vals: 15631.57
```

Getting the raw values

It can sometimes be useful to access the elements of a dataframe as a raw numpy array. For this you simply need to call the `.values` attribute on a series or attribute.

- retrieve a raw array containing the sound pressures
- retrieve a raw array containing the frequencies and the sound pressures (watch the type inference!)

In [13]:

```
# add your code here
print(df['Sound pressure [dB]'].values)

print(df[['Frequency [Hz]', 'Sound pressure [dB]']].values)
```

```
[126.201 125.201 125.951 ... 106.604 106.224 104.204]
[[ 800.     126.201]
 [1000.     125.201]
 [1250.     125.951]
 ...
 [4000.     106.604]
 [5000.     106.224]
 [6300.     104.204]]
```

(Bonus) Joining dataframes

Check the pandas documentation on the [.join method (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) to work out how to join `smalldf` with the following data:

```
{'Salary': [100, 150, 110, 90, 105, 500], 'Education': [5, 10, 7, 3, 4, 0]}
```

In [14]:

```
# add your code here
# the index *must* be specified
otherdf = pd.DataFrame({'Salary': [100, 150, 110, 90, 105, 500],
                        'Education': [5, 10, 7, 3, 4, 0]},
                        index=smalldf.index)

completedf = smalldf.join(otherdf)
completedf.head()
```

Out[14]:

	Age	Weight	Gender	Salary	Education
Name					
Bob	15	70	M	100	5
Alice	18	50	F	150	10
Maria	15	55	F	110	7
John	12	85	M	90	3
Kevin	14	68	M	105	4