# Bootcamp
# Python

42 ARTIFICIAL INTELLIGENCE

## Module04
## Pandas

# Module04 - Pandas

Today you will learn how to use a Python library that will allow you to manipulate dataframes.

## Notions of the module

Pandas! And Bamboos! (DataFrames, Series, slicing, mask . . . )

## General rules

- Use the Pandas Library.
- The version of Python recommended to use is 3.7, you can check the version of Python with the following command: `python -V`
- The norm: during this bootcamp you will follow the PEP 8 standards. You can install pycodestyle which is a tool to check your Python code.
- The function eval is never allowed.
- The exercises are ordered from the easiest to the hardest.
- Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
- Your manual is the internet.
- You can also ask questions in the `#bootcamps` channel in the 42 AI Slack: 42-ai.slack.com.
- If you find any issue or mistakes in the subject please create an issue on our bootcamp python repository on Github.

## Helper

For this module you will use the dataset `athlete_events.csv` provided in the `resources` folder.

You will need to install pandas library:

```
pip install pandas
```

Ensure that you have the right Python version.

```
> which python
/goinfre/miniconda/bin/python
> python -V
Python 3.7.*
> which pip
/goinfre/miniconda/bin/pip
```

\clearpage

# Exercise 00 - FileLoader

| | |
|---|---|
| Turn-in directory: | ex00/ |
| Files to turn in: | FileLoader.py |
| Allowed libraries: | Pandas |
| Remarks: | Be as lazy as possible... |

## Objective:

Basic usage of Pandas to retrieve a dataset and store it in a `PandaDataFrame`.

## Instructions:

Write a class named `FileLoader` which implements the following methods:

- `load(path)` : takes as an argument the file path of the dataset to load, displays a message specifying the dimensions of the dataset (e.g. 340 x 500) and returns the dataset loaded as a pandas.DataFrame,
- `display(df, n)` : takes a pandas.DataFrame and an integer as arguments, displays the first $n$ rows of the dataset if $n$ is positive, or the last n rows if $n$ is negative.

`FileLoader` object should not raise any exceptions (i.e. you should handle wrong path, file does not exist, parameters different than the expected type ...).

## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load("../data/adult_data.csv")
Loading dataset of dimensions 32561 x 15
>>> loader.display(data, 12)
age         workclass  fnlwgt  ... hours-per-week  native-country salary
0    39         State-gov   77516  ...             40   United-States  <=50K
1    50  Self-emp-not-inc   83311  ...             13   United-States  <=50K
2    38           Private  215646  ...             40   United-States  <=50K
3    53           Private  234721  ...             40   United-States  <=50K
4    28           Private  338409  ...             40            Cuba  <=50K
5    37           Private  284582  ...             40   United-States  <=50K
6    49           Private  160187  ...             16         Jamaica  <=50K
7    52  Self-emp-not-inc  209642  ...             45   United-States   >50K
```

```
8    31           Private    45781  ...           50   United-States    >50K
9    42           Private   159449  ...           40   United-States    >50K
10   37           Private   280464  ...           80   United-States    >50K
11   30         State-gov   141297  ...           40           India    >50K

[12 rows x 15 columns]
```

***NB***: Your terminal may display more columns if the window is wider.

\clearpage

# Exercise 01 - YoungestFellah

| | |
|---|---|
| Turn-in directory: | ex01/ |
| Files to turn in: | FileLoader.py, YoungestFellah.py |
| Allowed libraries: | Pandas |
| Remarks: | n/a |

## Objective:

Discover of the concept of slicing and indexing in `Pandas.DataFrame`.

## Instructions:

This exercise uses the following dataset: `athlete_events.csv`

Write a function `youngestFellah` that takes two arguments as describe by the function's signature:

```
def youngestFellah(df, year):
    """
    Get the name of the youngest woman and man for the given year.
    Args:
    -----
        df: pandas.DataFrame object containing the dataset.
        year: integer corresponding to a year.
    Return:
    -------
        dct: dictionary with 2 keys for female and male athlete.
    Raises:
    -------
        This function should not raise any Exceptions.
    """
    ... your code ...
```

The function returns a dictionary containing the age of the youngest woman and man who took part in the Olympics on that year. The name of the dictionary's keys is up to you, but it must be self-explanatory.

## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../ressources/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15
```

```
>>> from YoungestFellah import youngestFellah
>>> youngestFellah(data, 2004)
# Output
{'f': 13.0, 'm': 14.0}

>>> youngestFellah(data, 1991)
# Output
{'f': 'nan', 'm': 'nan'}
```

\clearpage

# Exercise 02 - ProportionBySport

| | |
|---|---|
| Turn-in directory: | ex02/ |
| Files to turn in: | FileLoader.py, ProportionBySport.py |
| Allowed libraries: | Pandas |
| Remarks: | n/a |

## Objective:

Practise of the concept of slicing and indexing in `Pandas.DataFrame`.

## Insctructions:

This exercise uses the dataset `athlete_events.csv` Write a function `proportionBySport` that takes four arguments as describe by the function's signature:

```
def proportionBySport(df, year, sport, gender):
    """
    Calculates the percentage of participants of specific gender for a specific sport among
↪   all the participants of the same gender for the given year.
    Args:
    -----
        df: pandas.DataFrame object containing the dataset.
        year: integer corresponding to a year.
        sport: string corresponding to a sport.
        gender: string corresponding to a gender.
    Return:
    -------
        dct: dictionary with 2 keys for female and male athlete.
    Raises:
    -------
        This function should not raise any Exceptions.
    """
    ... your code ...
```

The function returns a float corresponding to the proportion (percentage) of participants who played the given sport among the participants of the given gender.

The function answers questions like the following : "What was the percentage of female basketball players among all the female participants of the 2016 Olympics?"

***Hint***: Here and further, if needed, drop duplicated sports people to count only unique ones. Beware to call the dropping function at the right moment and with the right parameters, in order to not omit any individuals.

## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

>>> from ProportionBySport import proportionBySport
>>> proportionBySport(data, 2004, 'Tennis', 'F')
# Output
0.01935634328358209
```

We assume that we are always using appropriate arguments as input, and thus do not need to handle input errors.

\clearpage

# Exercise 3 - HowManyMedals

| | |
|---|---|
| Turn-in directory: | ex03/ |
| Files to turn in: | FileLoader.py, HowManyMedals.py |
| Allowed libraries: | Pandas |
| Remarks: | n/a |

## Objective:

Deepen the manipulation of the concept of slicing and indexing in `Pandas.DataFrame`.

## Instructions:

This exercise uses the dataset: `athlete_events.csv`

Write a function `howManyMedals` that takes two arguments, as describe by the function signature:

```
def proportionBySport(df, name):
    """
    Calculates the percentage of participants of specific gender for a specific sport among
↪   all the participants of the same gender for the given year.
    Args:
    -----
        df: pandas.DataFrame object containing the dataset.
        name: string corresponding to the name of a participant.
    Return:
    -------
        dct: nested dictionary {'year_1': {'G':nb_1G, 'S':nb_1S, 'B':n_1B}, 'year_2':
↪   {'G':nb_2G, 'S':nb_2S, 'B':n_2B}}.
    Raises:
    -------
        This function should not raise any Exceptions.
    """
    ... your code ...
```

The function returns a dictionary of dictionaries giving the number and type of medals for each year during which the participant won medals. The keys of the main dictionary are the Olympic games years. In each year's dictionary, the keys are 'G', 'S', 'B' corresponding to the type of medals won (gold, silver, bronze). The innermost values correspond to the number of medals of a given type won for a given year.

## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../ressources/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

>>> from HowManyMedals import howManyMedals
>>> howManyMedals(data, 'Kjetil Andr Aamodt')
# Output
{1992: {'G': 1, 'S': 0, 'B': 1},
 1994: {'G': 0, 'S': 2, 'B': 1},
 1998: {'G': 0, 'S': 0, 'B': 0},
 2002: {'G': 2, 'S': 0, 'B': 0},
 2006: {'G': 1, 'S': 0, 'B': 0}}
```

\clearpage

# Exercise 04 - SpatioTemporalData

| | |
|---:|:---|
| Turn-in directory: | ex04/ |
| Files to turn in: | FileLoader.py, SpatioTemporalData.py |
| Allowed libraries: | Pandas |
| Remarks: | Take a look to the methods of DataFrame |

## Objective:

The goal of this exercise is to deepen your manipulation of `pandas.DataFrame` and it methods, especially the methods for slicing and indexing.

## Instructions:

You will use the dataset `athlete_events.csv`

You have to write the class `SpatioTemporalData` which takes a dataset (`pandas.DataFrame`) as argument in its constructor and have the two following methods:

- `when(self, location)` : takes a location as an argument and returns a list containing the years where games were held in the given location.
- `where(self, date)` : takes a date as an argument and returns the location where the Olympics took place in the given year.

No exceptions should be risen by `where` and `when`. ## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../ressources/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

>>> from SpatioTemporalData import SpatioTemporalData
>>> sp = SpatioTemporalData(data)
>>> sp.where(1896)
# Output
['Athina']
```

```
>>> sp.where(2016)
# Output
['Rio de Janeiro']

>>> sp.when('Athina')
# Output
[2004, 1906, 1896]

>>> sp.when('Paris')
# Output
[1900, 1924]
```

\clearpage

# Exercise 05 - HowManyMedalsByCountry

| | |
|---:|:---|
| Turn-in directory: | ex05/ |
| Files to turn in: | FileLoader.py, HowManyMedalsByCountry.py |
| Allowed libraries: | Pandas |
| Remarks: | n/a |

## Objective:

The goal of this exercise is to deepen your manipulation of `pandas.DataFrame` and it methods, especially the methods for slicing and indexing.

## Instructions:

You will use the following dataset: `athlete_events.csv`

You have to write a function `howManyMedalsByCountry` that takes two arguments:

- a pandas.DataFrame which contains the dataset
- a country name.

The function returns a nested dictionary which gives the number and type of medal for each competition where the country delegation earned medals.
The keys of the main dictionary are the Olympic games' years. In each year's dictionary, the key are 'G', 'S', 'B' corresponding to the type of medals won.

Duplicated medals per team games should be handled and not counted twice. Plus, function should not raise any exceptions.

## Examples:

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

>>> from HowManyMedalsByCountry import howManyMedalsByCountry
>>> howManyMedalsByCountry(data, 'Martian Federation')
# Output
{2192: {'G': 17, 'S': 14, 'B': 23}, 2196: {'G': 8, 'S': 21, 'B': 19}, 2200: {'G': 26, 'S':
↪  19, 'B': 7}}
```

You probably guessed but it is not a real examples.

If you want real examples, you can easily look online. Beware that some medals might be awarded or removed years after the games are over, for example if a previous medallist was found to have cheated and is sanctioned. The `athlete_events.csv` dataset might not always take these posterior changes into account.

\clearpage

# Exercise 06 - MyPlotLib

| | |
|---|---|
| Turn-in directory: | ex06/ |
| Files to turn in: | MyPlotLib.py |
| Allowed libraries: | Pandas, Matplotlib, Seaborn, Scipy |
| Remarks: | The less work you do, the better!<br>You don't necessarily need all those libraries to complete the exercise. |

## Objective:

The goal of the exercise is to introduce plotting methods among the different graphical libraries (or graphical modules within libraries) `Pandas`, `Matplotlib`, `Seaborn`, `Scipy` ...

## Instructions:

You will use the dataset: `athlete_events.csv`.

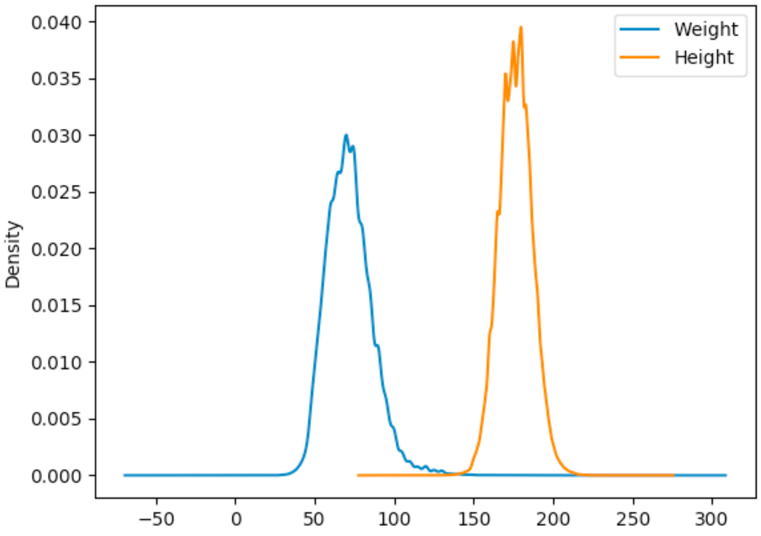
You have to write a class called `MyPlotLib`. This class implements different plotting methods, each taking two arguments:

- a pandas.DataFrame which contains the dataset
- a list of feature (also called independant variable) names.

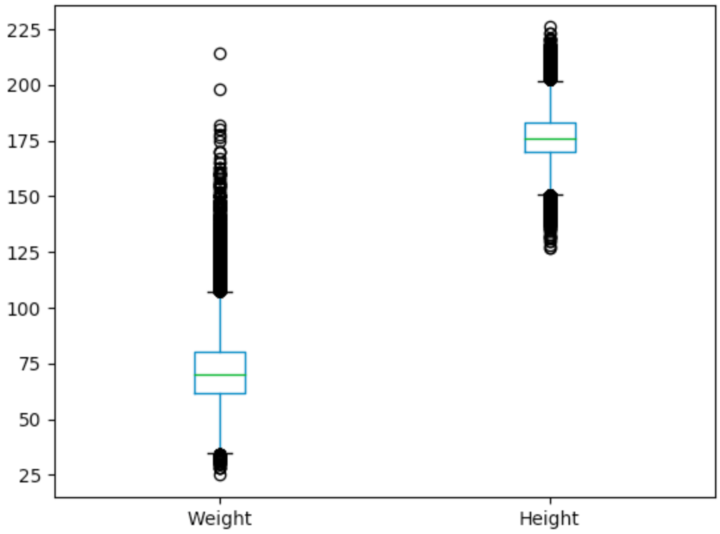
Here the list of methods you will implement:

- `histogram(data, features)`: plots one histogram for each numerical feature in the list.
- `density(data, features)`: plots the density curve of each numerical feature in the list.
- `pair_plot(data, features)`: plots a matrix of subplots (also called scatter plot matrix). On each subplot shows a scatter plot of one numerical variable against another one. The main diagonal of this matrix shows simple histograms.
- `box_plot(data, features)`: displays a box plot for each numerical variable in the dataset.

## Examples:



histogram



pair_plot

\clearpage

# Exercise 07 - Komparator

| | |
|---|---|
| Turn-in directory: | ex07/ |
| Files to turn in: | Komparator.py, MyPlotLib.py (optional) |
| Allowed libraries: | Pandas, Matplotlib, Seaborn, Scipy |
| Remarks: | The less work you do, the better!<br>You don't necessarily need all those libraries to complete the exercise. |

## Objective:

The goal of the exercise is to introduce plotting methods among the different graphical libraries `Pandas`, `Matplotlib`, `Seaborn`, `Scipy` ...

## Instructions:

You will use the dataset: `athlete_events.csv`

You have to write a class called `Komparator` whose constructor takes `pandas.DataFrame` as argument which contains the dataset. The class must implement the following methods, which take as input two variable names:

- `compare_box_plots(categorical_var, numerical_var)` : displays box plot with several boxes which allow to compare the distribution of the numerical variable according to the categorical values of categorical_var. There should be as many box as categories. For example, with Sex and Height, we would compare the height distributions of men vs. women via two boxes.
- `density(categorical_var, numerical_var)` : displays the density of the numerical variable. Each subpopulation should be represented by a separate curve on the graph.
- `compare_histograms(categorical_var, numerical_var)` : plots the numerical variable in a separate histogram for each category. As a bonus, you can use overlapping histograms with a color code.

***Bonus***: Your functions can also accept a list of numerical variables (instead of just one), and output a comparison plot for each variable in the list.

\clearpage