

In [3]:

```
#####  
# Intro à la Manip de Données avec Pandas sous Python  
# Author : Rodrique KAFANDO  
# Destination : Master FD&IA - UV-BF  
#####
```

# Introduction to Data Analysis With PANDAS

## C'est quoi Pandas ?

- Pandas est une bibliothèque construite au-dessus du langage Python
- Chaque bibliothèque a sa propre orientation ou spécialité
- Pandas est spécialisé sur l'analyse des données
  - C'est une boîte à outils robuste : analyser, filtrer, manipuler, agréger, concatener, nettoyer, etc.
  - En gros, il s'agit de EXCEL de Python

## Quelques librairies pour le traitement de données

### 1. Apache Spark

- Stars: 27600, Commits: 28197, Contributors: 1638
- Apache Spark - A unified analytics engine for large-scale data processing

### 2. Pandas

- Stars: 26800, Commits: 24300, Contributors: 2126
- Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

### 3. Dask

- Stars: 7300, Commits: 6149, Contributors: 393
- Parallel computing with task scheduling

[https://www.kdnuggets.com/2020/11/top-python-libraries-data-science-data-visualization-machine-learning.html?utm\\_content=166184627&utm\\_medium=social&utm\\_source=linkedin&hss\\_channel=lcp-3740012](https://www.kdnuggets.com/2020/11/top-python-libraries-data-science-data-visualization-machine-learning.html?utm_content=166184627&utm_medium=social&utm_source=linkedin&hss_channel=lcp-3740012)

## Pré-requis

- Connaissances basiques sur le fonctionnement des tableurs
  - Notions de colonnes, lignes
- Connaissances sur le typage des données
  - String, integers, boolean, float, etc.


## Recommandations

- Connaissances avancées sur le fonctionnement des tableurs
  - Fonctions, pivoter, etc.
- Connaissances basiques sur le langage de programmation Python
  - Fonctions, variables, typage de variables, etc.
  - Nb: je vous recommanderai de faire un tour sur excel pour voir ces différentes fonctions avant, si vous n'avez aucune notion sur les tableurs

## À propos de ce cours

- Subdivisé en plusieurs modules
  - chaque module abordera une fonctionnalité spécifique
- Un ensemble de données, au format CSV (Comma-separated values) pour les différents cas d'étude
  - bien évidemment, on pourrait aborder le traitement de tout type de fichier, à votre demande

## Pour commencer

1. Python et Pandas doivent être installé sur votre PC
  - La plus facile des options, serait d'installer la distribution du paquet **Anaconda**
  - qui intègre non seulement python et pandas, mais aussi plus +100 autres librairies d'analyse de données
2. Un terminal/ligne de commande pour les mises à jours et l'installation de nouvelles librairies
3. editeur:  Jupyter-notebook

## Explorons Jupyter-notebook

- les différentes parties/sections
- séquences d'exécutions
- commentaires
- appel à de fonctions externes

## Basic Data Types

- Python est un langage orienté Objet
  - le paradigme orienté Objet: programme logiciel est une collection d'objets qui communiquent entre eux
  - C'est quoi un Objet ?

- type de donnée abstraite pouvant prendre en compte la notion de polymorphisme et de l'héritage
- structure numérique, un conteneur pouvant stocker une sorte d'informations
- tout est objet en Python: Object <=> structure de données sous Python

```
In [4]: # integer  
4
```

Out[4]: 4

```
In [5]: # floating point number  
2.5
```

Out[5]: 2.5

```
In [6]: # String  
"Michel"  
"3"  
""
```

Out[6]: ''

```
In [ ]:
```

```
In [7]: # boolean, toujours utilisés pour evaluer une situation donnée  
True  
False  
'M' in "Michel"  
2>8
```

Out[7]: False

```
In [8]: None # s'il n'y a rien à retourner/afficher
```

```
In [9]: print('')
```

## Opérateurs

```
In [10]: # example on integer objects  
3+3
```

Out[10]: 6

```
In [11]: # concatenation  
'FD' + 'IA'
```

```
#FD' - 'IA'  
#FD' * 'IA'  
x = 'FD ' * 10  
y = x.split(' ')  
y
```

Out[11]: ['FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', '']

In [12]: `[v for v in y if v]`

Out[12]: ['FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD']

In [13]: *# poids des opérateurs (PEMDAS => Parentheses Exponents  
Multiplication Division Addition Substraction)*  
`2 + 4 * 5`

Out[13]: 22

In [14]: *# division*  
`5/3`

Out[14]: 1.6666666666666667

In [15]: `10 // 5`

Out[15]: 2

In [16]: `5 % 3`

Out[16]: 2

In [17]: `14 / 3`

Out[17]: 4.666666666666667

In [18]: `round(14 / 3,2)`

Out[18]: 4.67

In [19]: `4.67 * 3`

Out[19]: 14.01

In [20]: *#equality*  
`"x" == 2`

Out[20]: False

```
In [21]: # Inequality
```

## Variables

- nom attribué à un objet dans un programme

## Built-in function

- procedure or sequence of step
- example : len(), str(), int(), float(), type(), etc.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

## Custom function

- custom function are build to answer a specific needs
- function name should describe what the function is doing

```
In [22]: # function that accepts temperature in Celcius and returns it in
Fahrenheit
def convert_to_fahrenheit(celcius_temp):
    fahrenheit = celcius_temp * 1.8 + 32
    return fahrenheit
# set default value
def convert_to_fahrenheit(celcius_temp = 0):
    fahrenheit = celcius_temp * 1.8 + 32
    return fahrenheit
```

```
In [23]: ##
# we can combine build-in with custom function
def addition(n):
    return n + n

# example
numbers = (1, 2, 3, 4)
result = map(addition, numbers)
print(list(result))
```

```
[2, 4, 6, 8]
```

```
In [24]: # using multi-build-in function
numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

result = map(lambda x, y: x + y, numbers1, numbers2)
result
list(result) # retourner la nouvelle donnée sous forme de list
```

```
Out[24]: [5, 7, 9]
```

```
In [ ]:
```

## String methods

- specific function that belong to a specific object
- a method is directly uppon on an object
- contary to a function, a method is directly associated to an object ### QUIZ
- difference between:
  - buit-in function
  - custom fonction
  - method

```
In [25]: # example
profession = " fd and ai student "
profession.capitalize()
profession.title()
profession.lstrip()
profession.rstrip()
profession.split()
```

```
Out[25]: ['fd', 'and', 'ai', 'student']
```

```
In [26]: age = 40
age.bit_length
```

```
Out[26]: <function int.bit_length(>
```

## Lists

- mutable
- to keep data in some ordered places

```
In [27]: ##
my_list = ['first', 'one', 'two', 'second']
```

```
In [28]: my_list.append('third')
         my_list
```

```
Out[28]: ['first', 'one', 'two', 'second', 'third']
```

```
In [29]: my_list.pop()
```

```
Out[29]: 'third'
```

```
In [30]: my_list
```

```
Out[30]: ['first', 'one', 'two', 'second']
```

```
In [ ]:
```

## NB

- remove() delete the matching element/object whereas del and pop removes the element at a specific index.
- del and pop deals with the index. The only difference between two is that- pop return deleted the value from the list and del does not return anything.
- Pop is only way that returns the object.
- Remove is the only one that searches object (not index). Which is the best way to delete the element in List?
- If you want to delete a specific object in the list, use remove method.
- If you want to delete the object at a specific location (index) in the list, you can either use del or pop.
- Use the pop, if you want to delete and get the object at the specific location.

## Index positions and slicing

```
In [31]: # string index
         name = 'boukary'
         len(name)
```

```
Out[31]: 7
```

```
In [32]: name[6]
         name[-1]
```

```
Out[32]: 'y'
```

```
In [33]: student_list = ['kabore', 'ilboudo', 'kafando', 'robgo']
```

```
In [34]: # first index position, and the last one that we want to stop. the
         first index will be include in the result
         student_list[1:3]
```

```
Out[34]: ['ilboudo', 'kafando']
```

```
In [35]: student_list[: -3]
```

```
Out[35]: ['kabore']
```

## dictionnary

- unordered collection of key-values pair
- association between two elements or linked-data

```
In [36]: # restaurant menu  
menu = {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000}
```

```
In [37]: menu
```

```
Out[37]: {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000}
```

```
In [38]: # add another speciallity, use pop() to delete  
menu['meat'] = 500
```

```
In [39]: menu
```

```
Out[39]: {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000, 'meat': 500}
```

```
In [40]: menu.values()
```

```
Out[40]: dict_values([300, 600, 6000, 7000, 500])
```

```
In [41]: menu.keys()
```

```
Out[41]: dict_keys(['rice', 'fish', 'pizza', 'salmon', 'meat'])
```

```
In [42]: menu.items()
```

```
Out[42]: dict_items([('rice', 300), ('fish', 600), ('pizza', 6000), ('salmon', 7000),  
('meat', 500)])
```

```
In [43]: l = [1, 2, 3]  
x = ['a', 'b', 'c']  
s = dict(zip(l, x))  
s
```

```
Out[43]: {1: 'a', 2: 'b', 3: 'c'}
```

```
In [ ]:
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [130..

In [3]:

In [4]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: