

In [5]:

```
#####  
# L'Essentiel sur Pandas  
# Author : Rodrique KAFANDO  
# Destination : CITADEL  
#####
```

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

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 [6]: # integer  
4
```

Out[6]: 4

```
In [7]: # floating point number  
2.5
```

Out[7]: 2.5

```
In [8]: # String  
"Michel"  
"3"  
""
```

Out[8]: ''

```
In [ ]:
```

```
In [9]: # boolean, toujours utilisés pour evaluer une situation donnée  
True  
False  
'M' in "Michel"  
2>8
```

Out[9]: False

```
In [10]: None # s'il n'y a rien à retourner/afficher
```

```
In [11]: print('')
```

Opérateurs

```
In [12]: # example on integer objects  
3+3
```

Out[12]: 6

```
In [13]: # concatenation
```

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

Out[13]: ['FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', '']

```
In [14]: [v for v in y if v]
```

Out[14]: ['FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD', 'FD']

```
In [15]: # poids des opérateurs (PEMDAS => Parentheses Exponents
          Multiplication Division Addition Substraction)
          2 + 4 * 5
```

Out[15]: 22

```
In [16]: a = "Mr ilboudo"
          a.split()
```

Out[16]: ['Mr', 'ilboudo']

```
In [17]: # division
          5/3
```

Out[17]: 1.6666666666666667

```
In [18]: 10 // 5
```

Out[18]: 2

```
In [19]: 5 % 3
```

Out[19]: 2

```
In [20]: 14 / 3
```

Out[20]: 4.666666666666667

```
In [21]: round(14 / 3, 2)
```

Out[21]: 4.67

```
In [22]: 4.67 * 3
```

Out[22]: 14.01

```
In [23]: #equality
"x" == 2
```

Out[23]: False

```
In [24]: # 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.

Custom function

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

```
In [25]: # 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 [26]: ##
# 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 [27]:
```

```
##Exrcise
def unit(string):
    first_el = string[0]
    return first_el
```

```
In [28]: # unit('UVBF')
list_of_string = ["Nom", 'prenom', 'activity']
result = map(unit, list_of_string)
list(result)
```

Out[28]: ['N', 'p', 'a']

In []:

```
In [29]: # 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[29]: [5, 7, 9]

```
In [30]: list_1 = ["a", "b", "c", "d" ]
list_2 = [10,23,50,300]
my_dic=dict(zip(list_1,list_2))
print(my_dic)
```

{'a': 10, 'b': 23, 'c': 50, 'd': 300}

```
In [31]: for i,y in my_dic.items():
    print(i,">",y)
```

```
a > 10
b > 23
c > 50
d > 300
```

In []:

String methods

- specific function that belong to a specific object
- a method is directly upon on an object
- contrary to a function, a method is directly associated to an object ### QUIZ

- difference between:
 - built-in function
 - custom function
 - method

```
In [33]: # example
profession = " Data Scientis at Meta "
profession.capitalize()
profession.title()
profession.lstrip()
profession.rstrip()
profession.split()
```

```
Out[33]: ['Data', 'Scientis', 'at', 'Meta']
```

```
In [34]: age = 40
age.bit_length
```

```
Out[34]: <function int.bit_length()>
```

Lists

- mutable
- to keep data in some ordered places

```
In [35]: ##
my_list = ['first', 'one', 'two', 'second']
```

```
In [36]: my_list.append('third')
my_list
```

```
Out[36]: ['first', 'one', 'two', 'second', 'third']
```

```
In [37]: my_list.pop()
```

```
Out[37]: 'third'
```

```
In [38]: my_list
```

```
Out[38]: ['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 [39]: # string index
name = 'boukary'
len(name)
```

Out[39]: 7

```
In [40]: name[6]
name[-1]
```

Out[40]: 'y'

```
In [41]: student_list = ['kabore', 'ilboudo', 'kafando', 'robgo']
```

```
In [42]: # 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[42]: ['ilboudo', 'kafando']

```
In [43]: student_list[: -3]
```

Out[43]: ['kabore']

dictionary

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

```
In [44]: # restaurant menu
menu = {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000}
```

```
In [45]: menu
```



```
Out[45]: {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000}
```

```
In [46]: # add another speciallity,use pop() to delete  
menu['meat'] = 500
```

```
In [47]: menu
```

```
Out[47]: {'rice': 300, 'fish': 600, 'pizza': 6000, 'salmon': 7000, 'meat': 500}
```

```
In [48]: menu.values()
```

```
Out[48]: dict_values([300, 600, 6000, 7000, 500])
```

```
In [49]: menu.keys()
```

```
Out[49]: dict_keys(['rice', 'fish', 'pizza', 'salmon', 'meat'])
```

```
In [50]: menu.items()
```

```
Out[50]: dict_items([('rice', 300), ('fish', 600), ('pizza', 6000), ('salmon', 7000),  
('meat', 500)])
```

```
In [51]: l = [1,2,3]  
x = ['a','b','c']  
s = dict(zip(l,x))  
s
```

```
Out[51]: {1: 'a', 2: 'b', 3: 'c'}
```

```
In [52]: x = "Mr Ilboudo Mr Ilboudo Mr Ilboudo Mr Ilboudo"
```

```
In [55]: string_val = ("Mr Ilboudo"*4).split()  
string_val[0]
```

```
Out[55]: 'Mr'
```

```
In [ ]:
```