Python et Excel

avec le module Pandas

Introduction

Excel permet l'organisation et le traitement d'un grand nombre de données. On peut également automatiser le traitement de plusieurs documents similaires sous *Excel,* ceux-ci devront alors être normalisés. Afin d'automatiser ces taches, le traitement des fichiers *Excel* sous *Python* peut être très efficace.

La bibliothèque *Pandas* vise à intégrer les fonctionnalités de *NumPy* et *matplotlib*, pour donner un outil pratique pour analyser et visualiser les données. *Pandas* (http://pandas.pydata.org/, https://pandas.pydata.org/pandas-docs/dev/user_guide/10min.html) est une bibliothèque Python permettant de manipuler simplement et efficacement des données structurées dont les fichiers *Excel*. Cette librairie permet de lire et d'écrire des fichiers *.xls* (*Excel* avant 2007) ou *.xlsx* (*Excel* après 2007), grâce à des dépendances : *xlrd* (https://pypi.org/project/xlrd/) pour la lecture et xlwt (https://pypi.org/project/xlwt/) pour l'écriture.

Il existe beaucoup de documentation pour la lecture des fichiers *Excel* grâce à la librairie *xlrd* : http://python-simple.com/python-autres-modules-non-standards/xlrd.php, https://riptutorial.com/fr/python/example/23044/lisez-les-donnees-excel-avec-le-module-xlrd .

La librairie *Pandas* utilise principalement une structure de *DataFrame* pour stocker les données, cette structure est très proche des tableaux et dictionnaires lors d'un traitement classique en *Python*. De nombreuses fonctionnalités sont fournies pour manipuler ces *DataFrame* de façon très poussée.

Nous allons réaliser quelques exercices simples pour découvrir les fonctionnalités de la bibliothèque *pandas.* Les principales instructions à utiliser sont décrites dans la documentation "*Essentiel Pandas*" disponible sur Arche.

Découverte de la bibliothèque Pandas

Pandas est une librairie python qui permet de manipuler facilement des données à analyser.

La bibliothèque *Pandas* permet très facilement par exemple de :

- Calculer des statistiques, la moyenne, la médiane, le maximum ou le minimum de chaque colonne…
- Nettoyer les données en supprimant les valeurs manquantes et en filtrant les lignes ou les colonnes selon certains critères
- Visualiser les données avec l'aide de *Matplotlib*, tracer des barres, des lignes, des histogrammes, des bulles, etc.
- Stocker les données nettoyées et transformées dans un CSV, un autre fichier ou une base de données

Avec *Pandas* on manipule principalement deux types d'objets les *Séries* et les *DataFrames.*

Les séries

Un objet *Série* est comme un tableau de dimension 1[1].

- C'est est un objet de données *mutable*.
- Il peut contenir des données de tout type.

Une *Série* peut être crée à partir d'une liste, d'un dictionnaire …

Une *Série* peut être assimilée à un dictionnaire tel qu'on les utilise en *Python*. Par défaut, à la création d'une *série*, les index sont les nombres à partir de 0 jusqu'à la longueur de la colonne. Mais il est possible également de fixer d'autres index, soit à la création soit a posteriori.

	pays
0	'United kingdom'
1	'France
2	'Canada'

symbole	pays
'uk'	'United kingdom'
'fr'	'France
'ca'	'Canada'

Les DataFrames

Un *DataFrame* est un tableau avec des lignes et des colonnes, comme une feuille de calcul Excel. La table peut stocker en mémoire des données dans des formats différents.

symbole	pays
uk	United kingdom
fr	France
ca	Canada

+

symbole	capitale
uk	Londres
fr	Paris
ca	Ottawa

=

symbole	pays	capitale
uk	United kingdom	Londres
fr	France	Paris
ca	Canada	Ottawa

Un *DataFrame* peut être utilisé de façon similaire à un dictionnaire dans lequel les clefs sont utilisées pour représenter les noms des colonnes et les valeurs sont représentés par des *Series*[2].

Les *DataFrames* et les *Series* sont assez similaires dans la mesure où de nombreuses opérations que vous pouvez effectuer avec l'une peuvent être effectuées avec l'autre, telles que le remplissage de valeurs nulles et le calcul de la moyenne.

On peut facilement lire et écrire ces *DataFrames* à partir ou vers un fichier tabulé (classeur ou fichier csv).

1) D'un département à sa région

On dispose d'un classeur *Excel* nommé *fichier_departements_regions.xlsx* qui associe à chaque département le nom de la région dans laquelle il est localisé.

Pour avoir un aperçu de l'organisation du fichier nous en avons extrait les premières lignes :

Numéro	***Nom***	***Région***	***Ancienne Région***	***Population***
01	*Ain*	*Auvergne-Rhône-Alpes*	*Rhône-Alpes*	*656 955*
02	*Aisne*	*Hauts de France*	*Picardie*	*526 050*
03	*Allier*	*Auvergne-Rhône-Alpes*	*Auvergne*	*331 315*
04	*Alpes de Haute-Provence*	*Provence-Alpes-Côte d'Azur*	*Provence-Alpes-Côte d'Azur*	*165 197*

Pour se familiariser avec la structure d'un *DataFrame* et la syntaxe *Pandas*, réalisons au préalable les étapes suivantes :

---

[1] https://www.datasciencemadesimple.com/create-series-in-python-pandas/

[2] https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/

1) Récupérer *le data frame* du fichier *fichier_departements_regions.xlsx* en précisant que le **numéro** du département servira d'index pour chaque série du *DataFrame.* Pour ce faire, il faut ouvrir le fichier *Excel* et charger le *dataframe*.

   Afficher la liste des noms des feuilles du classeur

2) Afficher les noms des colonnes du *DataFrame*
3) Afficher les valeurs des index du *DataFrame*
4) Afficher les 5 premières lignes
5) Afficher les statistiques sur les valeurs (nombres de valeurs, moyenne, écart-type, ...)
6) Afficher les informations du département de clé "88"
7) Afficher le nom de la région du département "54"
8) Afficher les informations sur le département stocké à la ligne 24 du dataframe. Attention, les numéros de lignes du dataframe ne correspondent pas aux numéros de ligne du fichier Excel (elles commencent à 0 et ne prennent pas en compte l'entête).
9) Afficher la population de chaque département
10) Créer un *DataFrame grand_est* avec les départements de la région Grand Est
11) Afficher les noms de ces départements
12) Créer un *DataFrame gros_departements* avec les départements dont la population est supérieure à 1 million d'habitants
13) Trier le *DataFrame gros_departements* par population décroissante
14) Afficher la population de chacun de ces départements