

# MANIPULATION DE DONNÉES ET STATISTIQUES DESCRIPTIVES EN PYTHON

## PRÉSENTATION DES DONNÉES

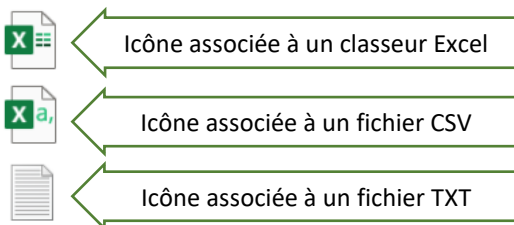
Les données du classeur Excel « [Données Etudiants anonymisées - avec poursuites études.xlsx](#) » utilisé en introduction étaient réparties dans trois feuilles, « Poursuites d'études », « Résultat » et « Profil entrant ».

Nous souhaitons travailler sur ces données en Python.

Même s'il est possible pour Python de lire directement des fichiers Excel, nous utiliserons une méthode plus classique qui consiste à passer par un fichier texte pour importer des données.

Nous avons pour cela enregistré chacune des feuilles du classeur dans des **fichiers texte en CSV\*** (comma separated values) qui, dans nos environnements non anglophones correspondent à des données séparées, non pas par des « , », mais par des « ; ».

\* Remarque sur les CVS : dans un environnement Windows, il arrive souvent que le l'icône associée à un fichier CSV évoque un format Excel :



Une icône « Excel » est associée aux fichiers CSV car, en général, Excel va lire correctement ce type de fichiers texte qui représentent des données tabulaires avec des champs (colonnes) séparées par un « ; ». **Il n'en demeure pas moins que le fichier CSV est un fichier texte, lisible par n'importe quel éditeur de texte.**

## EXTRAIT DES FICHIERS OBTENUS

### Fichier ProfilEntrant.csv

```
Num;Groupe Candidat;Classement recrutement (dans le groupe);Type De Formation;Serie Diplome Code;Mention Diplome;Entree
140;Bac etranger;;Terminale;;;2017
18;Bac général-S;79;Terminale;S;Admis sans mention;2017
196;Bac général-S;;Terminale;S;;2017
83;Bac général-S;119;Terminale;S;Admis sans mention;2017
30;Bac Technologique (STI,STI2D);3;Terminale;STI2D;Admis mention Très Bien;2016
40;Bac Technologique (STI,STI2D);61;Terminale;STI2D;Admis mention Assez Bien;2018
195;Bac général-S (ISN);83;Terminale;S;Admis sans mention;2017
```

### Fichier Resultats.csv

```
Num;UE 21;UE 22;Moyenne S2;Décision S2;annee S2;UE 11;UE 12;Moyenne;Décision;annee S1
30;14;12,86;13,47;Passage semestre suivant;16-17;14,2;12,22;13,34;Passage semestre suivant;16-17
97;13,38;12,68;13,05;Passage semestre suivant;16-17;13,58;13,28;13,45;Passage semestre suivant;16-17
143;12,67;11,88;12,3;Passage semestre suivant;16-17;13,74;11,44;12,75;Passage semestre suivant;16-17
102;10,56;10,04;10,32;Passage semestre suivant;16-17;11,2;11,23;11,21;Passage semestre suivant;16-17
82;7,99;9,61;8,93;Passage semestre suivant;16-17;9,94;11,37;10,72;Passage semestre suivant;16-17
65;12,35;11,76;12,08;Passage semestre suivant;16-17;12,44;11,77;12,15;Passage semestre suivant;16-17
219;11,49;10,39;10,98;Passage semestre suivant;16-17;10,49;11,33;10,86;Passage semestre suivant;16-17
184;8,53;11,61;9,97;Redoublement semestre précédent;16-17;8,36;10,89;9,46;Passage semestre suivant;16-17
```

### Fichier Poursuite.csv

```
code;Situation;Alternance;Niveau formation;complement;promo;
106;Etudes longues;En contrat d'apprentissage;L3, Ecole d'ingénieur;Ecole Privée;2019;
149;Etudes longues;En formation initiale;L3, Ecole d'ingénieur;Univ Publique;2019;
102;Année sabbatique;;;2019;
181;Etudes longues;En formation initiale;L2, année spéciale,...;Univ Publique;2019;
63;Etudes longues;En contrat d'apprentissage;L3, Ecole d'ingénieur;Ecole Ingénieur Publique;2019;
186;Emploi / Recherche d'emploi;;;2019;
105;Etudes longues;En contrat de professionnalisation;L3, Ecole d'ingénieur;Ecole Privée;2019;
123;Etudes longues;En contrat d'apprentissage;L3, Ecole d'ingénieur;Ecole Ingénieur Publique;2019;
87;Etudes longues;En contrat d'apprentissage;L3, Ecole d'ingénieur;Ecole Privée;2019;
```

## LA BIBLIOTHÈQUE PANDAS ET LE DATAFRAME

Pandas est une bibliothèque Python offrant de nombreuses possibilités de manipulations de données. La structure de base de cette bibliothèque est le **DataFrame** qui correspond au tableau **individus x variables** vu dans la première partie du cours : individus en lignes et variables en colonnes.

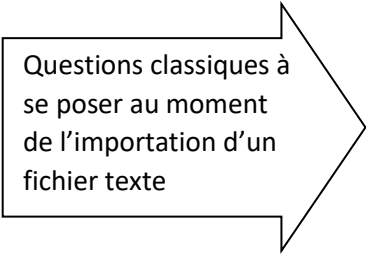
Nous verrons quelques manipulations des DataFrame par la suite. Dans un premier temps, nous allons importer nos données CSV dans des DataFrame.

### IMPORTATION D'UN FICHIER TEXTE DANS UN DATAFRAME

La commande usuelle de la librairie Pandas, pour importer un fichier texte est **read\_table** (il existe une commande **read\_csv** qui est identique mais avec des paramètres par défaut différents).

Les options de **read\_table** permettent de préciser comment doit se faire la lecture du fichier externe.

Ci-dessous une liste d'options classiques, qui correspondent à des questions que l'on se pose habituellement dans cette situation, quel que l'environnement dans lequel se fait la lecture :



Questions classiques à se poser au moment de l'importation d'un fichier texte

- Les noms des variables sont-ils dans la première ligne ?
- Quel est le séparateur des champs (colonnes) dans les lignes du fichier texte ?
- Le séparateur des décimales dans le fichier texte est-il le « . » ?
- Existe-t-il dans le fichier texte une variable identifiant les individus (variable identificatrice, clé primaire) ?
- Comment sont codées les valeurs manquantes dans le fichier texte ?
- J'ai réalisé une première importation et j'ai des problèmes avec les accents...

- **Option header : les noms des variables sont-ils dans la première ligne ?**

header indique le numéro de la ligne (dans le fichier texte) où se trouvent les noms de variables.

**header=0** : les noms sont dans la première ligne (valeur par défaut)

**header=None** : il n'y a pas de ligne avec les noms de variables, dans ce cas Python affecte des noms de colonnes par défaut, mais on peut aussi préciser ces derniers en utilisant une option

**names=[liste des noms de variables]**

- **Option sep : Quel est le séparateur des champs (colonnes) dans les lignes du fichier texte ?**

**sep=' ; '** : le séparateur est le « ; » (fichier CSV dans nos environnements)

**sep=' , '** : le séparateur est la « , » (fichier CSV d'un environnement anglophone)

**sep=' \t '** : le séparateur est la tabulation (fichier TXT classique) – valeur par défaut

**sep=' \s+ '** : le séparateur est un nombre variable d'espaces (il s'agit ici d'une expression régulière, \s représente un espace et + signifie 1 ou plusieurs)

- **Option decimal : le séparateur des décimales est-il le « . » ?**

Dans la plupart des langages, et dans Python notamment, le séparateur des décimales est le « . ». Ce n'est pas le cas dans notre environnement Excel par exemple et ce ne sera pas le cas des CSV créés dans cet environnement.

**decimal=' . '** : le séparateur des décimales dans le fichier lu est le « . » (option inutile, car valeur par défaut)

**decimal=' , '** : le séparateur des décimales dans le fichier lu est la « , »

- **Option index\_col : existe-t-il dans le fichier texte une variable identifiant les individus (variable identificatrice, clé primaire) ?**

Dans les DataFrame, les lignes sont identifiées par un **index**.

Par défaut index\_col=None : il n'y a pas de colonne dans le fichier qui permette d'identifier les lignes et Python créera un index numérotant ces dernières, de 0 à n.

**index\_col= 0** : la première colonne du tableau lu est une variable identificatrice

**index\_col='nom'** : la colonne « nom » du tableau lu est une variable identificatrice

- **Options na\_values : comment sont codées les valeurs manquantes dans le fichier texte?**

Ce n'est pas un problème fréquent, mais il est très important pour importer correctement les données. Par défaut **Python** interprète comme valeur manquante les symboles suivants : '' (la chaîne vide), '#N/A', '#N/AN/A', '#NA', '-1.#IND', '-1.#QNAN', '-NaN', '-nan ', '1.#IND', '1.#QNAN', '<NA>', 'N/A', 'NA', 'NULL', 'NaN', 'n/a', 'nan', 'nul'.

Si dans le fichier les valeurs manquantes sont codées « NC », il faudra utiliser l'option

**na\_values= 'NC'**

- **Option encoding : j'ai réalisé une première importation et j'ai des problèmes avec les accents.**

Pour un fichier venant d'Excel, on pourra utiliser l'option encoding='ANSI' ou encoding='latin\_1' (<https://docs.python.org/3/library/codecs.html#standard-encodings>)

Bricolage : on peut aussi changer l'encodage d'un fichier texte en utilisant le Bloc-Notes (enregistrer sous et on modifie l'encodage)

Encodage : ANSI

Dernière remarque : prendre l'habitude de consulter la documentation de pandas :

[https://pandas.pydata.org/docs/user\\_guide](https://pandas.pydata.org/docs/user_guide)

Exemple pour **read\_table** ([https://pandas.pydata.org/docs/reference/api/pandas.read\\_table.html](https://pandas.pydata.org/docs/reference/api/pandas.read_table.html)) :

## pandas.read\_table

```
pandas.read_table(filepath_or_buffer, sep=NoDefault.no_default, delimiter=None,
header='infer', names=NoDefault.no_default, index_col=None, usecols=None, squeeze=None,
prefix=NoDefault.no_default, mangle_dupe_cols=True, dtype=None, engine=None,
converters=None, true_values=None, false_values=None, skipinitialspace=False,
skiprows=None, skipfooter=0, nrows=None, na_values=None, keep_default_na=True,
na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False,
infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False,
cache_dates=True, iterator=False, chunksize=None, compression='infer', thousands=None,
decimal='.', lineterminator=None, quotechar='\"', quoting=0, doublequote=True,
escapechar=None, comment=None, encoding=None, encoding_errors='strict', dialect=None,
error_bad_lines=None, warn_bad_lines=None, on_bad_lines=None, delim_whitespace=False,
Low_memory=True, memory_map=False, float_precision=None, storage_options=None) ; [source]
```

Ci-dessus : la syntaxe de read\_table avec la liste de ses options. En dessous de la syntaxe générale, les options sont détaillées avec leurs valeurs par défaut (inutile d'écrire une option si on utilise sa valeur par défaut)

**sep** : str, default '\t' (tab-stop)

Exemples pour sep :

Delimiter to use. If sep is None, the C engine cannot automatically detect the separator, but the Python parsing engine can, meaning the latter will be used and automatically detect the separator by Python's builtin sniffer tool, `csv.Sniffer`. In addition, separators longer than 1 character and different from '\s+' will be interpreted as regular expressions and will also force the use of the Python parsing engine. Note that regex delimiters are prone to ignoring quoted data. Regex example: '\r\t'.

# ALLONS-Y !

## 1. Importation du fichier ProfilEntrant.csv

```
import pandas as pd
profilentrant=pd.read_table('C:\\\\INFO\\\\Pandas\\\\ProfilEntrant.csv', sep=";")
```


↑  
Chemin d'accès à modifier

- Vérifier que l'importation s'est correctement déroulée : il ne doit pas y avoir de message d'erreur et le DataFrame **profilentrant** doit se trouver dans les variables de votre environnement (fenêtre *variable explorer*, normalement en haut à droite)

Il est possible de visualiser le contenu du DataFrame en cliquant sur son nom dans la fenêtre *variable explorer* :

Name	Type	Size	Value
profilentrant	DataFrame	(242, 5)	Column names: Classement recrutement (dans le groupe), Type De Formati ...

Help Variable explorer Plots Files



Index	Num	utement (	Type De Formation	erie Diplome Cod	Mention Diplome	Entre
0	140	nan	Terminale	Bac etranger	nan	2017
1	18	79	Terminale	S	Admis sans mention	2017
2	196	nan	Terminale	S	nan	2017
3	83	119	Terminale	S	Admis sans mention	2017
4	30	3	Terminale	STI2D	Admis mention TB	2016
5	40	61	Terminale	STI2D	Admis mention AB	2018

- Un index a été créé qui permet d'identifier les lignes : modifier les options de **read\_csv** pour que la variable *Num* soit considérée comme *index* (variable identificatrice)

## 2. Importer le fichier Poursuite.csv en utilisant comme index la colonne « code ».

Si vous n'avez pas changé les options dans le `read_table`, vous devez observer un petit problème dans le DataFrame. Corrigez-le ?

## 3. Importer le fichier Resultats.csv en utilisant comme index la colonne « num ».

Si vous n'avez pas changé les options dans le `read_table`, vous devez observer nouveau problème\* dans le DataFrame. Corrigez-le ?

*Indication : dans l'affichage ci-dessus, les variables numériques sont représentées en couleur.*

**Remarque : les variables d'un DataFrame sont des objets de type « Serie » et un DataFrame se comporte comme un dictionnaire de variables c'est-à-dire de d'objets « Serie »**

# MANIPULATIONS ÉLÉMENTAIRES DES DATAFRAME

## CONTENU DU DATAFRAME

### ✓ Dimensions (nombre d'individus et de variables)

<code>poursuite.shape</code>	→	<code>(102, 5)</code>	<b>Tuple qui se manipule comme une liste (de taille fixe)</b>
<code>poursuite.shape[0]</code>	→	<code>102</code>	<b>Nombre de lignes ou d'individus</b>
<code>poursuite.shape[1]</code>	→	<code>5</code>	<b>Nombre de colonnes ou de variables</b>

## VARIABLES

```
poursuite.columns →  
Index(['Situation', 'Alternance', 'Niveau formation', 'complement', 'promo'], dtype='object')  
poursuite.columns[2] → 'Niveau formation' Nom de la troisième colonne
```

### ✓ Renommer une variable

Il n'est pas possible de renommer individuellement une variable (objet non mutable) :

```
poursuite.columns[2]= 'Formation' ne marche pas
```

En revanche, on peut renommer l'ensemble des variables :

```
poursuite.columns=['Situation', 'Alternance', 'Formation', 'Complement', 'Promo']
```

Pour renommer une seule variable, on peut passer par l'intermédiaire d'une liste qui elle, est mutable :

```
l=list(poursuite.columns)  
l[2]='Formation'  
poursuite.columns=l
```

### ✓ Accès à une variable

<code>poursuite['Alternance']</code>	→	<pre>code 106  En contrat d'apprentissage 149  En formation initiale 102  NaN 181  En formation initiale 63   En contrat d'apprentissage</pre>
--------------------------------------	---	--

**Contenu de la variable Alternance**

### ✓ Changer le type d'une variable – Variables catégorielles (qualitatives)

Lorsque nous importons des données externes, les colonnes numériques sont importées dans des types numériques (**int64** pour les entiers, **float64** pour les flottants). Les autres colonnes reçoivent un type **objet** qui recouvre les autres types de variables. Il s'agit en général de variables qualitatives.

Pour des traitements simples, ce n'est pas très gênant de garder le format **objet**, mais si on veut manipuler plus aisément les modalités, il vaut mieux leur affecter le type **categorical\*** qui correspond au type **qualitatif des variables statistiques**.

Cette transformation peut aussi être faite sur des variables numériques quand elles représentent des variables qualitatives

*\*Cette opération qui consiste à préciser les variables qualitatives (ici categorical) d'un tableau de données, n'est pas fondamentale sous Python, mais est souvent indispensable dans la plupart des logiciels spécialisés dans le traitement des données.*

Exemple avec le DataFrame **poursuite** :

```
poursuite.dtypes →
```

Situation	object
Alternance	object
Formation	object
complement	object
promo	int64

```
poursuite['Situation']=poursuite['Situation'].astype('category')
poursuite.dtypes
```

Situation	category
Alternance	object
Formation	object
complement	object
promo	int64

```
poursuite[['Alternance','Formation','complement',
            'promo']] =poursuite[['Alternance','Formation','complement',
            'promo']].astype('category')
```

```
poursuite.dtypes
```

Situation	category
Alternance	category
Formation	category
complement	category
promo	category

## OBSERVATIONS ET INDIVIDUS

### ✓ **Sélections d'observations : *loc* (sélection par le nom-index), *iloc* (sélection par l'indice) :**

Les observations du DataFrame peuvent être identifiées par leurs indices (*iloc*) ou par leur nom (*loc*)

poursuite - DataFrame

code	Situation	Alternance	
106	Etudes longues	En contrat d'apprentissage	L3
149	Etudes longues	En formation initiale	L3
102	Année sabbatique	nan	na

```
poursuite.iloc[0,0] → 'Etudes longues'
```

**Observation correspondant à la première ligne et première colonne du tableau**

```
poursuite.loc[106,'Situation'] → 'Etudes longues'
```

**Observation correspondant à la ligne d'index 106 (la colonne *code* est l'index du DataFrame) et la colonne « Situation »**

**Ou encore : l'observation correspondant à l'individu 106 et la variable « Situation »**  
**Ce qui correspond aussi à la valeur de la première ligne et la première colonne du tableau**

### ✓ **Sélection de plusieurs lignes et/ou plusieurs colonnes**

```
poursuite.iloc[[1,2,3,4],[0,3]] ⇔ poursuite.iloc[1:5,[0,3]]
⇔ poursuite.loc[[149,102,181,63],['Situation','complement']]
```

→

	Situation	complement
code		
149	Etudes longues	Univ Publique
102	Année sabbatique	NaN
181	Etudes longues	Univ Publique
63	Etudes longues	Ecole Ingénieur Publique

### ✓ **Sélection d'individus selon leur rang ou leur index**

```
poursuite.iloc[1:5,:] ⇔ poursuite.loc[[149,102,181,63],:]
```

→ les lignes 1 à 4 et toutes les colonnes

Ou : les individus d'index 149,102,181,63 et toutes les variables

✓ **Sélection d'individus à partir de conditions logiques sur les variables**

```
poursuite.loc[poursuite['Situation']=='Etudes longues',:]
```

→

	Situation	Alternance	...	promo	nvlvar
code					
106	Etudes longues	En contrat d'apprentissage	...	2019	0
149	Etudes longues	En formation initiale	...	2019	0
181	Etudes longues	En formation initiale	...	2019	0
63	Etudes longues	En contrat d'apprentissage	...	2019	0
105	Etudes longues	En contrat de professionnalisation	...	2019	0

## CRÉATION ET TRANSFORMATION DE VARIABLES

✓ **Création d'une variable**

Il est possible de créer une nouvelle variable en lui affectant directement une valeur, une valeur par défaut par exemple :

```
poursuite['nvlvar1']=0
```

```
poursuite['nvlvar2']=numpy.nan →
```

	Situation	Alternance	...	nvlvar1	nvlvar2
code					
106	Etudes longues	En contrat d'apprentissage	...	0	NaN
149	Etudes longues	En formation initiale	...	0	NaN
102	Année sabbatique	NaN	...	0	NaN
181	Etudes longues	En formation initiale	...	0	NaN

✓ **Mise en classe d'une variable quantitative**

Pour mettre en classe une variable quantitative nous pouvons utiliser les fonction **qcut** et **cut**.

→ **qcut** pour des classes d'effectifs équilibrées

Il suffit de donner le nombre de classes (4 dans les exemples ci-dessous) et les bornes de classes sont calculées à partir des quantiles correspondants (ci-dessous les quartiles)

```
resultats["ClassMoy1"]=pd.qcut(resultats["Moyenne S2"],4)
```

```
resultats["ClassMoy2"]=pd.qcut(resultats["Moyenne S2"],4,
                                labels=["Ag1","Ag2","Ag3","Ag4"])
```

→

	UE 21	UE 22	Moyenne S2	...	annee S1	ClassMoy1	ClassMoy2
Num							
30	14.00	12.86	13.47	...	16-17	(12.34, 17.93]	Ag4
97	13.38	12.68	13.05	...	16-17	(12.34, 17.93]	Ag4
143	12.67	11.88	12.30	...	16-17	(11.13, 12.34]	Ag3
102	10.56	10.04	10.32	...	16-17	(9.51, 11.13]	Ag2
82	7.99	9.61	8.93	...	16-17	(1.709, 9.51]	Ag1

→ **cut** pour des classes personnalisées ou d'amplitudes égales

Pour des classes d'amplitudes égales, il suffit de donner leur nombre :

```
resultats["ClassMoy3"]=pd.cut(resultats["Moyenne S2"],3)
```

```
resultats["ClassMoy4"]=pd.cut(resultats["Moyenne S2"],3,right=False)
```

Pour des classes personnalisées, il suffit de donner les bornes des classes (option bins)

```
resultats["ClassMoy5"]=pd.cut(resultats["Moyenne S2"],bins=[0,8,10,12,14,20])
```

	UE 21	UE 22	Moyenne S2	...	ClassMoy3	ClassMoy4	ClassMoy5
Num							
30	14.00	12.86	13.47	...	(12.523, 17.93]	[12.523, 17.946)	(12, 14]
97	13.38	12.68	13.05	...	(12.523, 17.93]	[12.523, 17.946)	(12, 14]
143	12.67	11.88	12.30	...	(7.117, 12.523]	[7.117, 12.523)	(12, 14]
102	10.56	10.04	10.32	...	(7.117, 12.523]	[7.117, 12.523)	(10, 12]
82	7.99	9.61	8.93	...	(7.117, 12.523]	[7.117, 12.523)	(8, 10]



## ✓ Recodage d'une variable qualitative

### → Affichage des modalités

```
poursuite["Situation"].unique()  
['Etudes longues', 'Année sabbatique', 'Emploi / Recherche d'emploi', 'Licence Pro',  
'année sabbatique', 'réorientation']
```

ou

```
poursuite["Situation"].cat.categories  
Index(['Année sabbatique', 'Emploi / Recherche d'emploi', 'Etudes longues',  
       'Licence Pro', 'année sabbatique', 'réorientation'],  
       dtype='object')
```

## ✓ Regroupement de modalités

### → Utilisation de la fonction map

**#Création d'un dictionnaire *corresp* pour les correspondances  
#entre les anciennes et nouvelles modalités**

```
corresp = {'Année sabbatique': 'Autre',  
          'année sabbatique': 'Autre',  
          'réorientation': 'Autre',  
          'Etudes longues': 'Etudes longues',  
          'Emploi / Recherche d\'emploi': 'Emploi / Recherche d\'emploi',  
          'Licence Pro': 'Licence Pro'}
```

**#Attribution des nouvelles modalités dans la variable *sit2***

```
poursuite['sit2'] = poursuite["Situation"].map(corresp)
```

### → Ou en utilisant des sélections conditionnelles

```
poursuite['sit2'] = numpy.nan  
poursuite['sit2'][poursuite['Situation']=='Etudes longues']='EL'  
poursuite['sit2'][poursuite['Situation']=='Année sabbatique']='AUT'  
poursuite['sit2'][poursuite['Situation']=='année sabbatique']='AUT'  
poursuite['sit2'][poursuite['Situation']=='réorientation']='AUT'  
poursuite['sit2'][poursuite['Situation']=='Emploi / Recherche d\'emploi']='EMP'  
poursuite['sit2'][poursuite['Situation']=='Licence Pro']='LP'
```

```
poursuite['sit2'].unique()
```

```
array(['EL', 'AUT', 'EMP', 'LP'], dtype=object)
```

La variable sit2 a quatre modalités : 'EL', 'AUT', 'EMP', 'LP'

## SUPPRESSION DE VARIABLES ET D'INDIVIDUS

```
poursuite.drop(index=[106,149],inplace = True)  
poursuite.drop(columns=['Situation','complement'],inplace = True)
```

option inplace : pour remplacer le DataFrame

Voir aussi : <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html>



## CONCATÉNATION DE TABLES

### JOINTURE (« CONCATÉNATION HORIZONTALE »)

La fonction `merge` permet de faire un `SELECT... JOIN`, ce qui revient, d'un point de vue statistique, à ajouter des variables à un tableau individus x variables.

```
pd.merge(profilentrant,
         resultats,
         left_index=True,
         right_index=True,
         how="inner")
```

⇔

```
SELECT *
FROM profilentrant
INNER JOIN resultats
ON profilentrant.code=resultats.Num
```

Il est possible d'utiliser d'autres clés que les index des tables, on remplacera pour cela les options `left_on` les `o right_on`.

(Voir la doc : <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html>)

### CONCATÉNATION VERTICALE (FACULTATIF)

La fonction **`pd.concat`** permet de faire des concaténations horizontales (options `axis=1`) et verticales (options `axis=0`). D'un point de vue statistique, la concaténation verticale revient à ajouter des individus à un tableau individus x variables.

(Voir la doc : <https://pandas.pydata.org/docs/reference/api/pandas.concat.html>)

## ALLONS-Y !

### 1. Manipulations du DataFrame `profilentrant`.

- a. Afficher les noms de variables de ce DataFrame
- b. Renommer la variable « Serie Diplome Code » en « seriebac »
- c. Afficher les modalités de la variable « seriebac ».
- d. Créer une nouvelle variable nommée « bac », n'ayant plus de 3 modalités : « S », « STI2D », « Autre »
- e. Renommer la variable « Classement recrutement (dans le groupe) » en « classement »
- f. Créer une variable nommée « classcod1 » correspondant à la mise en classe de la variable de « classement » en 5 classes de mêmes effectifs
- g. Créer une variable nommée « classcod1 » correspondant à la mise en classe de la variable de « classement » en 5 classes de même amplitude
- h. Créer une variable nommée « classcod1 » correspondant à la mise en classe de la variable de « classement » en 3 classes dont vous choisirez les bornes vous-même

### 2. Jointure.

Concaténer les trois tables `profilentrant`, `poursuite` et `resultats`. On ne gardera que les étudiants communs aux trois tables

## Projet Python (1<sup>ère</sup> partie)

3. Importer les données de votre projet Excel dans un ou des DataFrame Python.
4. Faire une jointure des DataFrame obtenus, le cas échéant.
5. Effectuer sous Python les mêmes modifications et créations de variables que celles effectuées sous Excel.

# TABLE DES MATIÈRES

Présentation des données.....	1
Extrait des fichiers obtenus .....	1
La bibliothèque Pandas et le Dataframe .....	2
Importation d'un fichier texte dans un DataFrame.....	2
 Allons-y ! (Exercice).....	4
 Manipulations élémentaires des DataFrame .....	5
Contenu du DataFrame .....	5
✓ <i>Dimensions (nombre d'individus et de variables)</i>	
 Variables .....	5
✓ Renommer une variable	
✓ Accès à une variable	
✓ Changer le type d'une variable – Variables catégorielles (qualitatives)	
 Observations et individus .....	6
✓ Sélections d'observations : <i>loc (sélection par le nom-index), iloc (sélection par l'indice) :</i>	
✓ <i>Sélection de plusieurs lignes et/ou plusieurs colonnes</i>	
✓ <i>Sélection d'individus selon leur rang ou leur index</i>	
✓ <i>Sélection d'individus à partir de conditions logiques sur les variables</i>	
 Création et transformation de variables .....	7
✓ Création d'une variable	
✓ Mise en classe d'une variable quantitative	
✓ Recodage d'une variable qualitative	
✓ Regroupement de modalités	
 Suppression de variables et d'individus .....	8
 Concaténation de tables.....	8
Jointure (« concaténation horizontale ») .....	9
concaténation verticale (facultatif) .....	9
 Allons-y ! (Exercice et Mini-projet Partie1).....	9