Partie 1

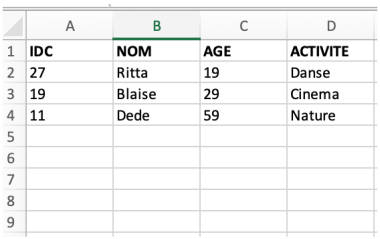
Données en table

1.1 Tables - généralités

1.1.1 Une table

Une table est une liste de lignes, dont les éléments partagent les mêmes colonnes (descripteurs).

Le contenu d'une table est formé de ses lignes, qui contiennent les *valeurs*.

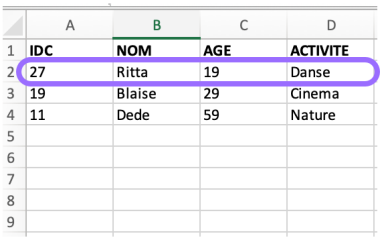


	A	B	C	D
1	**IDC**	**NOM**	**AGE**	**ACTIVITE**
2	27	Ritta	19	Danse
3	19	Blaise	29	Cinema
4	11	Dede	59	Nature
5				
6				
7				
8				
9				

FIGURE 1 – exemple de table

1.1.2 Une ligne

Chaque ligne est un p-uplet de valeurs (duet, triplet, ...)



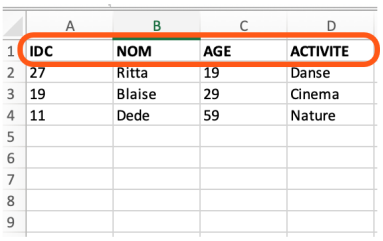
	A	B	C	D
1	**IDC**	**NOM**	**AGE**	**ACTIVITE**
2	27	Ritta	19	Danse
3	19	Blaise	29	Cinema
4	11	Dede	59	Nature
5				
6				
7				
8				
9				

FIGURE 2 – ligne, p-uplet

1.1.3 Une colonne

Dans chaque colonne, on trouve des valeurs de même type : le **domaine de valeurs**.

Chaque colonne porte un nom : le **descripteur** de la colonne. Celui-ci est important pour la représentation en *dictionnaire* de la table.



	A	B	C	D
1	IDC	NOM	AGE	ACTIVITE
2	27	Ritta	19	Danse
3	19	Blaise	29	Cinema
4	11	Dede	59	Nature
5				
6				
7				
8				
9				

FIGURE 3 – colonne, descripteur

1.2 Représenter une table

On peut représenter une table sous forme de texte structuré : en *csv*, *json*, ou *xml*.

Ces formats sont interopérables. On peut les manipuler avec de nombreuses applications. (tableurs, langages informatique).

Partie 2

Table dans un script python

2.1 Charger une table depuis un fichier csv

Pour ouvrir et lire un fichier *csv* en langage python, on peut utiliser le script suivant :

```
1 import csv
2 with open('classe.csv', mode='r', encoding='utf-8') as f:
3     table = list(csv.reader(f,delimiter=";"))
```

On suppose qu'il n'y a qu'une table par fichier *csv*.

La représentation d'une table est alors une LISTE de **lignes** en python. Mais il peut y avoir d'autres options : La représentation d'une ligne de la table peut être, au choix :

- une liste : `['eleve1', '12.5', 10.0, 8.9, ...]`
- un tuple (idem liste mais avec des parenthèses rondes `('eleve1', '12.5', 10.0, 8.9, ...)`
- un dictionnaire (nouveau) : `{'nom':'eleve1', 'moyenne':12.5, 'note1':10.0, 'note2':8.9, ...}`

Quelles sont les différences entre ces 3 représentations ? (séparateurs, position des données)

2.2 Liste ou tuple python

Si la table T est une liste de lignes, elles-mêmes stockées dans une liste :

Les données de la table T sont accessibles depuis le n° de ligne i et celui de colonne j : `T[i][j]`

Inconvenient : Se souvenir de la correspondance numero de colonne <-> information. Par exemple, pour le TP partie 3, la note 1 de l'élève 1 est accessible avec : `classe[1][2]`

Pour des lignes en tuple, c'est pareil, mais attention, le tuple est *non mutable*.

–> Faire la partie 4 de la fiche de TP (fichier des effectifs du lycée) en exercice.

2.3 Table mise dans un dictionnaire python

Un dictionnaire est un tableau associatif. Les éléments sont rangés en couples *clé :valeurs*, séparés par une virgule, et mis entre accollades {}.

Pour une représentation en *dictionnaire* :

```
ligne1 = {'nom':'eleve1', 'moyenne':12.5, 'note1':10.0, 'note2':8.9, ...}
```

Les *descripteurs* sont les *clés* du dictionnaire. A chaque clé est associée une valeur.

On accède à l'**une des valeurs** en plaçant la clé entre les `[]` :

```
>>> print(ligne1['nom'])
eleve1
```

La **liste des clés** est obtenue par la méthode de classe `keys()` du dictionnaire :

```
>>> for c in ligne1.keys():
    ....print(c)
eleve1
moyenne
note1
note2
```

Voici alors un extrait de la table classe mise dans une liste de dictionnaires (3 premiere lignes) :

```
table = [{'nom':'eleve1', 'moyenne':12.5, 'note1':10.0, 'note2':8.9, ...},
          {'nom':'          ', 'moyenne':None, 'note1':     , 'note2':    , ...},
          {'nom':          ', 'moyenne':None, 'note1':     , 'note2':    , ...},
           ...
        ]
```

2.4 Compléments sur les dictionnaires : introduction à la fiche 4

En programmation, on a souvent besoin de faire correspondre des éléments entre eux, comme un nom avec un numéro de téléphone, un produit avec son prix, ou un pays avec sa capitale. En Python, cette association est rendue possible grâce à une structure appelée **dictionnaire**.

Un dictionnaire est une **collection non ordonnée** (avant Python 3.7), **mutable** (modifiable), et **indexée par des clés**. Chaque élément est une **paire clé/valeur**.

Le traitement de la table change un peu

```
def moy(eleve):
    s = 0
    n = 0
    for cle in eleve.keys():
        if cle !='nom' and cle!='moyenne':
            s += eleve[cle]
            n+=1
    return s/n
```

Et pour placer la valeur de la moyenne :

```
eleve['moyenne'] = moy(eleve)
```