

# Python 3 – Fichiers et Dossiers



# Plan de cours

- ↪ Les bases de python
- ↪ **Dossiers, Fichiers et Librairies avec python**
- ↪ **Bases de données SQL**
- ↪ Python pour l'administration
- ↪ Outils et Métiers de Développement

# Dossiers, Fichiers et Librairies

- ▮ Se déplacer dans des répertoires et parcourir leurs contenus
  - ▮ Le module « os »
- ▮ Créer, écrire, lire et supprimer (CRUD) de fichiers
- ▮ Présentation de Formats de fichiers « standard »
  - ▮ Csv
  - ▮ Xml
  - ▮ Json
  - ▮ Yaml
- ▮ Utilisation de librairie de manipulation de fichier
  - ▮ Logging, Json, csv, Lxml, pandas, openpyxl

# Création d'un module

- On crée un fichier exemple.py
- On saisit à l'intérieur les lignes suivantes

```
def welcome(message):  
    Print (message)
```
- On enregistre le fichier
- Il est possible de récupérer des modules avec la commande  
« pip install »

```
pip install logging
```
- Il est possible de stocker ses modules dans un sous-répertoire
- On parle alors de « paquet », leur nommage utilise un .

```
import toto.exemple
```

# Utilisation des modules

- ▮ Deux manières d'importer un module
- ▮ `import exemple`
  - ▮ On importe la totalité du module et l'on utilise son nom  
`exemple.welcome(« charlene »)`
- ▮ `from exemple import welcome`
  - ▮ On importe une seule partie du module  
`welcome(« charlene »)`
- ▮ Il est possible d'importer la totalité des fonctions des modules avec le  
\*  
`from exemple import *`

# La librairie « os »

- Contient tout un ensemble de méthode propre à la gestion de fichier et dossier

import os	# librairie native de python
print (dir(os))	# renvoie la liste de toutes les fonctions présente
de la librairie	
Print (help(os.listdir))	# renvoie l'aide de la fonction listdir

# Déplacement dans les dossiers

- ▮ `os.getcwd()` : retourne le dossier courant « Current Working Directory » (cwd)
- ▮ `os.sep()` : retourne le bon séparateur selon le système ( / , \ , : )
- ▮ `os.chdir('/tmp')` : change le cwd
- ▮ `os.listdir('/home')` : retourne une liste avec tous les répertoires d'un dossier (si pas de dossier transmis, il s'agit du cwd)
- ▮ `os.mkdir('/tmp/test')` : crée un répertoire dans le dossier tmp (non récursif)
- ▮ `os.makedirs ('/tmp/test/test/test')` : version récursive
- ▮ `os.rmdir('/tmp/test/test')` : suppression d'un répertoire (non récursif)
- ▮ `os.removedirs('/tmp/test')` : version récursive
- ▮ `os.rename('/tmp/test/', 'tmp/test2')` : renommer un nom de répertoire
- ▮ `os.stat('/tmp/test')` : renvoie les informations du répertoire (taille, date, ...)

# Autres fonctions de la librairie os

`os.environ.get('HOME')` : retourne la variable d'environnement HOME  
`os.path.join('/tmp/dossier', 'test.txt')` : retourne le chemin en ajoutant le / si absent  
`os.path.basename('/tmp/dossier/test.txt')` : retourne juste le nom du fichier  
`os.path.dirname('/tmp/dossier/test.txt')` : retourne juste le nom du dossier  
`os.path.split('/tmp/dossier/test.txt')` : retourne un tableau avec chemin + fichier  
`os.path.splitext('/tmp/dossier/test.txt')` : retourne l'extension du fichier séparément  
  
`os.path.exists('/tmp/dossier/test.txt')` : retourne True si l'élément existe  
`os.path.isdir('/tmp/dossier/')` : retourne True si c'est un dossier  
`os.path.isfile('/tmp/dossier/test.txt')` : retourne True si c'est un fichier

## BONUS

```

import pathlib # utilisation de la bibliothèque pathlib
myFolderpath= pathlib.Path(__file__).parent.resolve() # on récupère le chemin du
programme
os.chdir(myFolderpath) # on se positionne dans le dossier
  
```



# Création d'un fichier

```
monFichier = open('eleves.txt', 'w+') # Ouvre le fichier selon le mode défini
```

w+ => création du fichier en écrasant son contenu

a+ => ajoute à la fin du fichier existant

```
monFichier.write('eleve1\n') # \n : ajoute un saut de ligne
```

```
monFichier.write('eleve2\n')
```

```
monFichier.write('eleve3')
```

```
monFichier.close() # Ferme et enregistre le fichier
```

```
lstEleves = ['eleve1', 'eleve2', 'eleve3'] # Variante avec une liste
```

```
for eleve in lstEleves:
```

```
    monFichier.write(eleve + « \n »)
```

# Lecture d'un fichier

```

with open('eleves.txt', 'r+') as file:      # ouvre le fichier selon le mode
    r+      => lecture du fichier
    print (file.readlines())                # retourne une liste contenant les lignes
    file.close()                            # l'utilisation du with permet de se passer du close

# comment vérifier la présence du fichier???
import os
if os.path.exists('eleve.txt') :
    # on peut ouvrir le document existant
else :
    # le fichier n'existe pas
  
```

# Modification/Suppression d'une ligne d'un fichier

- ▢ On ouvre le fichier à modifier en lecture « FichierL »
  - ▢ Récupération du contenu du fichier
- ▢ On ouvre un autre fichier en écriture « FichierE »
- ▢ On parcourt la liste des lignes de FichierL
  - ▢ Quand on a une ligne qui n'est pas à modifier, on écrit la ligne dans FichierE
  - ▢ Quand on a une ligne à modifier, on modifie la ligne et on écrit la ligne modifiée dans FichierE
  - ▢ Quand on a une ligne à supprimer, on ne l'écrit pas dans FichierE
- ▢ On ferme les deux fichiers
- ▢ On supprime le FichierL
- ▢ On renomme le FichierE en FichierL

Complexité de réaliser un CRUD avec ce genre de fichier ...

# Structure des fichiers de données simples

- un fichier simple ne permet de stocker que des listes de lignes
- Une première évolution revient à créer des tableaux en deux dimensions.

Nom	Prenom	Age	Note
Dupont	Pierre	23	13
Martin	Paul	43	12
Durant	Jacques	30	15,5

`'nom','prenom','age','note'`      # on donne en première ligne le nom des colonnes

`'dupont','pierre',23,13`      # les colonnes sont séparées par un 'séparateur' : , ou ; ou tabulation

`'martin','paul',43,12`      # les valeurs numérique ne sont pas obligatoirement encadrées par des '

`'durant',jacques',30,15.5`      # attention aux « virgules » de décimale qui peuvent être des séparateurs

# Structure de fichiers avancées (non structuré)

- Besoin de stocker des données de manière non-structurées, arborescente
- Besoins de moyens rapides de lire et conserver les données dans un fichier

XML	JSON	YAML
<ul style="list-style-type: none"> <li>Même structure avec des balises et des attributs que le format HTML</li> </ul>	<ul style="list-style-type: none"> <li>Basée sur la structure des listes javascript, norme de fait</li> <li>Proche de la structure du dictionnaire python</li> </ul>	<ul style="list-style-type: none"> <li>Simple à lire et à écrire par des humains</li> <li>Il est possible de saisir des commentaires</li> </ul>
<pre>&lt;menu id="file" value="File"&gt;   &lt;popup&gt;     &lt;menuitem value="New" onclick="CreateNewDoc()"/&gt;     &lt;menuitem value="Open" onclick="OpenDoc()"/&gt;     &lt;menuitem value="Close" onclick="CloseDoc()"/&gt;   &lt;/popup&gt; &lt;/menu&gt;</pre>	<pre>{  "menu": {     "id": "file",     "value": "File",     "popup": {       "menuitem": [         { "value": "New", "onclick": "CreateNewDoc()" },         { "value": "Open", "onclick": "OpenDoc()" },         { "value": "Close", "onclick": "CloseDoc()" }       ]     }   } }</pre>	<pre>menu:   id: file   value: File   popup:     menuitem:       - value: New         onclick: CreateNewDoc()       - value: Open         onclick: OpenDoc()       - value: Close         onclick: CloseDoc()</pre>

# L'usage des librairies pour manipuler des fichiers

- XML

- <https://docs.python.org/3/library/xml.etree.elementtree.html>

- <https://python.doctor/page-xml-python-xpath>

- Json

- <https://docs.python.org/fr/3/library/json.html>

- Yaml

- <https://python.land/data-processing/python-yaml>

# CSV : manipulation de fichier csv

## ■ Ecriture de fichier csv

```
import csv
with open('eleve.csv', 'w', newline='') as csvfile:
    eleveWriter = csv.writer(csvfile,          # nom du fichier csv
                             delimiter=';',    # séparateur de champs
                             quotechar='"',    # champs d'encadrement des chaines
                             doublequote=True,
                             quoting=csv.QUOTE_NONNUMERIC # on n'encadre pas les valeurs numériques
    )

    eleveWriter.writerow(['nom', 'prenom', 'age', 'note'])
    eleveWriter.writerow(['martin', 'paul', '45', 15])
    eleveWriter.writerow(['dupont', 'pierre', '33', 12])
```

## ■ Lecture d'un fichier csv

```
import csv
with open('eleve.csv', 'r') as csvfile:
    eleveReader = csv.reader(csvfile)
    for row in eleveReader:
        print(' '.join(row))
```

# json : manipulation de fichier Json

```
import json      # on charge la bibliothèque
```

```
dicEleves = {
    'titi' : {'notes':{'tp1':10, 'tp2':13,'tp3':17}, 'appréciation': 'moyenne' },
    'toto' : {'notes' :{'tp1':19, 'tp2':11,'tp3':14}, 'appréciation': 'Très Bien' },
    'tata' : {'notes':{'tp1':15,'tp2':8,'tp3':13}, 'appréciation': 'Bonne' },
    'tutu' : {'notes':{'tp3':15,'tp4':13}, 'appréciation': 'Bonne' },
}
```

# pour enregistrer le dictionnaire

```
with open('listelevés.json', 'w+') as file: # on crée un fichier JSON
```

```
json.dump(dicEleves, file, indent=4)      # on 'dump' le dictionnaire dans le fichier
```

# pour recharger le dictionnaire

```
with open('listelevés.json', 'r') as file: # on ouvre le fichier JSON
```

```
dicEleves = json.load(file)      # on alimente les données dans le dictionnaire
```



# Gérer des fichiers log avec la bibliothèque logging

<https://docs.python.org/3/library/logging.html>

```
pip install logging      # récupère la bibliothèque sur notre pc
```

```
Import logging
```

```
# définition du format du fichier de log
```

```
logging.basicConfig(filename='filelog.log', encoding='utf-8', level=logging.WARNING,  
format='%(asctime)s %(message)s', datefmt='%d/%m/%Y')
```

```
# création du fichier de log reprenant le nom du fichier python
```

```
logging.debug('on passe par cette étape')
```

```
logging.info('on passe par cette fonction')
```

```
logging.warning('on passe par cette fonction')
```

```
logging.error('on passe par cette fonction')
```

```
logging.critical('il y a eu une erreur critique')
```

# Exemples d'utilisations de dossiers et fichiers

- ▢ Exo1 : Manipulations de dossiers
- ▢ Exo2 : Manipulation de fichiers
- ▢ Exo3 : Manipulation de fichier txt et CSV
  - ▢ On ouvre un fichier eleves.txt et on affiche son contenu
  - ▢ Sinon on crée le fichier et on ajoute des lignes

# Base de données SQL

- ▢ Présentation SGBDR
- ▢ DDL : Manipuler les tables
- ▢ SQL : Créer Lire, Modifier et supprimer (CRUD) des données dans une table
  - ▢ INSERT
  - ▢ SELECT
  - ▢ UPDATE
  - ▢ DELETE
- ▢ Conditions
- ▢ La joie des jointures
  
- ▢ Manipuler des données avec Python

# Historique

- ▮ 1969 : première définition du modèle de base de données relationnelle
- ▮ 1979 : première version exploitable (IBM et Oracle)
- ▮ 1986 : Début de la normalisation et définition des normes SQL
  - ▮ SQL-89 ou SQL-1 en 1989
  - ▮ SQL-92 ou SQL-2 en 1992
  - ▮ SQL-99 ou SQL-3 en ... 1999

En 50 ans le langage SQL c'est imposé comme une norme de stockage de données, même si d'autres Base de données non SQL (on parle de No-SQL) se développent (mongo-db, elasticsearch)

# Principe de fonctionnement

- ▮ Structurer les données
- ▮ Economiser la place (la mémoire coutait chère en 1970...)
- ▮ Gérer les droits d'accès aux données
- ▮ Assurer la cohérence des données (ACID : Atomicité, Cohérence, Isolation, Durabilité)
  - ▮ Transaction : enchainement cohérent de plusieurs actions
  - ▮ Cohérence : (mise en place de règle d'intégrité)
  - ▮ Isolation : les « requêtes » sont sérialisable, indépendante, et n'impacte pas (trop) d'autre transaction
  - ▮ Durabilité : tout changement est permanent et tracé (dans un changelog)

# Principe : Un langage simple à utiliser

- ▢ Un langage pour gérer le contenant : **Data Description Language (DDL)**
  - ▢ Créer des tables et leur liens (règles d'intégrité)
  - ▢ Modifier les tables (noms des champs, type, longueur, ... )
  - ▢ Supprimer les tables ou leur contenu (en totalité)
  - ▢ Droits d'accès (écriture, lecture) aux tables (et même leurs colonnes)
  - ▢ Ecriture de procédures stockées
- ▢ Un langage pour gérer le contenu : **Structured Query Language (SQL)**
  - ▢ Ajout de ligne dans une table : INSERT
  - ▢ Lecture des données d'une ou plusieurs tables : SELECT
  - ▢ Modification des données d'une table : UPDATE
  - ▢ Suppression des données d'une table : DELETE

# Mise en oeuvre

- ▢ Différentes méthodes d'installation
- ▢ Démarrage et arrêt du serveur
- ▢ Configuration du serveur
- ▢ Création d'une base

Serveur de Données (mysql / mariadb)

On se connecte au serveur

Base de données

- Droits d'accès
- Les tables
- Procédures Stockées

	Colonnes / Champs
Lignes / données	Tables

# Les différents types de champs

- ▢ Champs de type Texte (limité dans la longueur ou non)
- ▢ Champs numérique (entier ou autres)
- ▢ Champs dates et heures
- ▢ Booléen (petit entier 0:faux 1: Vrai)
- ▢ Existence d'une valeur « Null »
  - ▢ Null vs « »
  - ▢ Null vs 0
  - ▢ Null vs 01/01/1970
  - ▢ Null vs Faux

not True:

Date FR                      vs  
Date US

JJ/MM/AAAA MM/JJ/AAAA

02/06/2015

02/06/2015

25/09/2022

XXXXX



# Méthodes d'accès aux données

- ▢ Commande shell 'mysql'
- ▢ PhpMyAdmin
- ▢ Programmation
- ▢ ORM / Django

## Protections des données

- Accès aux données (utilisateur / fichier)
- Préservation de l'intégrité (sauvegarde/restauration)
- Synchronisation de serveur / base de données
- Journalisation / log

# Manipulation des tables DDL

- ▮ Création d'une table
  - ▮ `CREATE TABLE matable (champ1 varchar (30), champ2 integer)`
- ▮ Modification des champs d'une table
  - ▮ `ALTER TABLE matable ADD champ3 integer`
  - ▮ `ALTER TABLE matable MODIFY champ2 float`
- ▮ Suppression d'une table
  - ▮ `DROP TABLE matable`
- ▮ Copier une table dans une autre
  - ▮ `CREATE TABLE matable_clone LIKE matable`
  - ▮ Ne copie pas les données présente dans la table

# Requêtes sur les données SQL

- ▮ Ajout dans une table

```
INSERT INTO matable (champ1, champ2) VALUES ('valchamp1', 213)
```

- ▮ Sélection dans une table

```
SELECT champ1, champ2 FROM matable
```

- ▮ Modification

```
UPDATE matable SET champ1 = 'anotherValue'
```

- ▮ Suppression

```
DELETE FROM matable
```

# Filtrage (select, update et delete) et trie (select)

- ▢ Il est possible de sélectionner une ligne en filtrant les lignes avec une condition
  - ▢ `WHERE conditions`
  - ▢ `UPDATE matable SET champ1 = 'newvalue' WHERE champ1 = 'valchamp1'`
  - ▢ Le filtrage s'applique aux requêtes SELECT, UPDATE et DELETE
  
- ▢ Il est possible de trier les lignes sélectionnées
  - ▢ `ORDER BY champs_a_trier`
  - ▢ `SELECT * FROM matable ORDER BY champ1`

# SQL et le python

Récupérer la librairie dédiée à mysql pour python

`pip install mysql-connector-python`

Utiliser la librairie dans votre programme python

`Import mysql.connector`

Attention ne pas appeler votre programme de test « mysql.py »

Connection à la base de données :

```
conn = mysql.connector.connect(host=host, user=user, password=password,  
database=database)
```

host : nom ou adresse IP du serveur ou est installé mysql

database : nom de la base de données que l'on souhaite accéder

user + password : identifiant pour se connecter à la base de données

`conn.is_connected()` permet de savoir si l'on est connecté ou non à la base de données

# SQL et python

```
cursor = conn.cursor()  
données
```

# définit le curseur qui conservera les

```
query = "SELECT * FROM matable;"
```

```
cursor.execute(query)  
résultat
```

# on exécute la requête et on récupère le

```
for row in cursor:  
résultats
```

# on boucle sur les lignes de

```
    print(row)
```

```
cursor.close()
```

# on ferme le curseur (libère l'espace)

```
conn.close()
```

# on ferme la connexion à la base

# Exemple de programme python

- ▮ Définition des paramètres d'accès à la base données
- ▮ Connection à la base de données
- ▮ Récupération de données dans une table (SELECT)
- ▮ Affichage des lignes de données
- ▮ Fermeture de la connexion

# Accès par mysql

- ▮ Depuis un terminal dans le dossier des applications de mysql
- ▮ `> mysql -u root` // permet de se connecter au serveur de données
- ▮ `Show databases;` // la liste des bases présentes dans le serveur
- ▮ `Use mabase;` // se positionne sur la base de données nomBase
- ▮ `Show tables;` // la liste des tables présentes dans la DB
- ▮ `Desc nomTable` // décrit la structure de la table nomTable
- ▮ `select * from nomTable` // accède aux données de la table nomTable