**Développement d'applications natives**

420-3N1-AA

Automne 2024

**Sommatif #03**

* L’examen se fait en équipe de 2.
* L’examen est sur 2.5 semaines
* L’usage de la théorie, exercice, capsule et internet est permis.
* L’examen compte pour **20%**.

Un **fichier SQLAR** (ou "SQLite Archive") est un format de fichier basé sur une base de données SQLite conçu pour archiver des ensembles de fichiers, de manière similaire aux formats d'archive classiques comme ZIP ou TAR. Ce format a été introduit par les développeurs de SQLite pour fournir un moyen simple et flexible de stocker un ensemble de fichiers au sein d'une base de données SQLite.

Il stocke les fichiers comme des enregistrements dans une base de données, où chaque fichier est représenté par un enregistrement comportant son contenu binaire, son chemin relatif, ainsi que d'autres métadonnées (comme la taille, la date de modification, les permissions, etc.).

Une base de données SQLAR contient minimalement une table (nommée « sqlar ») avec les champs suivants :

name (clé primaire), le chemin/nom du fichier

mode (entier), les permissions POSIX du fichier

mtime (horodatage), date de la dernière modification du fichier

sz (entier), taille originale du fichier

data (données binaires), données du fichier

Toute autre colonne jugée nécessaire.

**Réalisation**

Vous devez créer un programme python qui permet de manipuler une archive SQLAR en ligne de commande.

Le programme permet dans un terminal les opérations suivantes :

* Lister le contenu de l’archive
* Ajouter des fichiers à l’archive
* Retirer des fichiers de l’archive
* Extraire des fichiers de l’archive

La syntaxe du programme est la suivante :

sqlar archive verb [option1, option2, …, option\_n]

où *verb*  est une des actions entre « list », « extract », « add » ou « remove » pour lister, extraire, ajouter et retirer des fichiers respectivement.

*options1, options2, … options\_n*  sont les options associées à chaque verbe

Exemples :

*python sqlar monArchive.sqlar list*

*python sqlar monArchive.sqlar list -p10*

*python sqlar ar.sqlar add chemin\vers\fic1.txt chemin\vers\fic2.txt \chemin\vers\rep*

*python sqlar ar.sqlar extract chemin\vers\rep\destination*

**Contraintes**

Le programme doit vérifier les arguments en ligne de commande et afficher un message d’erreur s’ils sont invalides.

Le programme doit s’assurer que les fichiers/répertoires pour chacune des actions existent sinon afficher un message d’erreur.

Ajout : Si le fichier archive n’existe pas, il est créé, ensuite les fichiers sont ajoutés à l’archive, sinon ils sont simplement ajoutés à l’archive.

Retrait : L’archive doit exister. Extrait tous les fichiers.

Affichage : L’archive doit exister. Afficher le contenu en format tabulaire par pages. Si l’option -p est fourni, l’affichage se fait par page selon l’option -p (ex : -p10 se fait par 10 éléments par page, -p20 de 20 éléments par page, etc.). Par défaut, 10 éléments par page.

Extraction : L’archive doit exister. Extraire tous les fichiers de l’archive vers le répertoire renseigné en ligne de commande.

**Précisions**

* Les données des fichiers dans l’archive sont compressées avec « zlib ».
* Lorsqu’un fichier est ajouté à l’archive, le répertoire du fichier n’est pas conservé. Par exemple :

sqlar ar.sqlar add c:\rep1\fichier.txt c:\rep2\fichier2.txt

insérera les données « fichier.txt » et « fichier2.txt » dans la colonne « name ».

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

* S’il y a un conflit de nom lors de l’ajout d’un fichier, demander à l’utilisateur s’il veut écraser et remplacer le fichier présent dans l’archive ou ignorer l’ajout.
* Lorsqu’un répertoire et tout son contenu est ajouté à l’archive, ajouter le contenu du répertoire et de ses sous-répertoires avec le chemin de fichier relatif au répertoire ajouté. Par exemple, supposons l’arborescence suivante :

C:\temp

└───parent

├───rep1

│ fic1.txt

│ fic2.txt

│

└───rep2

fic1.txt

└─── fic1.txt

La commande :

sqlar ar.sqlar add c:\temp

ajoutera les fichiers dans l’archive avec les noms suivants :

fic1.txt

parent/rep1/fic1.txt

parent/rep1/fic2.txt

parent/rep2/fic1.txt

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

* Lors de l’extraction des fichiers, respectez les chemins relatifs inscrits dans l’archive et recréez ces chemins.
* Lors de l’affichage du contenu de l’archive, tronquez les noms trop longs :

Exemple :

L’ajout du fichier nommé : un\_nom\_de\_fichier\_tres\_tres\_long.txt

>python sqlar.py test.sqlar list  
-----------------------------------------------------------------------------  
| Name | Size | Last modified |  
-----------------------------------------------------------------------------  
| file2.txt | 26 | Mon Nov 4 14:24:01 2024 |  
| file1.txt | 40 | Mon Nov 4 14:24:01 2024 |  
| ...\_fichier\_tres\_tres\_long.txt| 42 | Mon Nov 4 14:24:01 2024 |  
| subdir/file1.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file10.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file11.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file12.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file13.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file14.txt | 4 | Mon Nov 4 14:24:01 2024 |  
| subdir/file15.txt | 4 | Mon Nov 4 14:24:01 2024 |  
-----------------------------------------------------------------------------

**Structure du programme**

* La connexion et la fermeture de la base de données est effectuée dans la fonction « main ».
* Chacune des actions (verbes) sont implantées dans des fonctions différentes.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* Créez une fonction pour initialiser la base de données avec le ou les tables requises.

**Aide :**

sys.argv # liste les arguments transmis depuis le terminal

os.path.isdir(filepath) # détermine si un chemin pointe vers un répertoire

os.path.join(…) # crée un chemin avec les / ou \ selon l’OS

os.path.basename(filepath) # obtient le nom du fichier en excluant les répertoires

os.path.dirname(filepath) # obtient le chemin de répertoire content le fichier

os.stat(file).st\_mode # obtient les permissions POSIX du fichier

os.chmod(file, mode) # applique les permissions POSIX sur le fichier

os.utime(file,(a\_time,m\_time)) # change l’horodate fichier dernier accès et modification

int(os.path.getmtime(file)) # obtient l’horodate du fichier

os.makedirs(path, exist\_ok=True) # créer les répertoires du chemin si inexistants

file.read() # lire les données binaire du fichier

zlib.compress(data) # compresser des données binaires

zlib.decompress(data) # décompresser des données binaire

Pour fournir des arguments en ligne de commande directement dans PyCharm (pour déboguer entre-autres):

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

**Critères de correction**

**Structure du programme et fonctions de base**

* **Organisation du code** : Le programme est structuré avec une fonction main qui gère la connexion et la fermeture de la base de données SQLite.
* **Séparation des actions** : Chaque verbe (list, add, remove, extract) est implémenté dans une fonction distincte.

**Initialisation et gestion de la base de données SQLAR**

* **Création de la table SQLAR** : La table SQLAR est correctement initialisée avec les champs nécessaires (name, mode, mtime, sz, data) et le bon type de données et toutes les colonnes jugées nécessaires sont présentes.
* **Validation de l’existence de l’archive** : Vérification correcte si l’archive existe avant d'exécuter les opérations nécessitant sa présence.

**Gestion des opérations de l'archive**

* **Liste des fichiers** : La commande « list » affiche le contenu de l’archive en format tabulaire avec pagination, avec 10 éléments par page par défaut, ou le nombre spécifié avec -p, tout en respectant les exemples d’exécutions fournies.
* **Ajout de fichiers** : Les fichiers ajoutés à l’archive sont correctement compressés avec zlib, et les chemins relatifs sont correctement gérés sans conserver les répertoires d’origine. Gestion des conflits de noms en demandant l’action souhaitée (écraser ou ignorer).
* **Retrait de fichiers** : La commande « remove » supprime correctement les fichiers spécifiés de l’archive, en s’assurant que l’archive existe avant la suppression.
* **Extraction des fichiers** : Les fichiers extraits respectent les chemins relatifs stockés dans l’archive et recréent l’arborescence correspondante. Les horodatages des fichiers sont restitués aux fichiers extraits ainsi que leur mode et leur données respectives.

**Gestion des erreurs et vérification des arguments**

* **Vérification des arguments** : Le programme vérifie la validité des arguments de la ligne de commande et affiche un message d’erreur en cas d’arguments invalides ou incomplets.
* **Existence des fichiers et répertoires** : Le programme s’assure que les fichiers et répertoires spécifiés existent avant d’exécuter les opérations, et affiche des messages d’erreur appropriés sinon.

**Aspects supplémentaires et fonctionnalité avancée**

* **Utilisation d’une nomenclature et d’une indentation systématique**: La nomenclature ainsi que l’indentation du code respecte le guide PEP8.
* **Rédaction de DocString et des variables d’entête de module**: Des docstrings et commentaires sont en nombre suffisant pour aider à la lecture et compréhension du code et les auteurs sont indiqués clairement pour chaque module.
* **Compression et décompression des données** : Les données des fichiers sont correctement compressées lors de l’ajout à l’archive et décompressées lors de l’extraction.
* **Usage adéquat des modules Python** : Utilisation des modules os et zlib pour manipuler les chemins de fichiers, les permissions et les horodatages est correcte et appropriée.
* **Codification adéquate des instructions**: Les instructions Python ne sont pas clairement redondantes et sont optimisées selon les standards du cours.