POO : Projet d'intégration

Acquis d'apprentissage évalués

Cette première partie évalue votre capacité à programmer une application en Python à l'aide de classes et de collections et à tester certains de ses composants. Vous disposez de deux heures à cette fin.

Description

Un problème de sécurité fréquent est celui des mots de passe faibles. Ces mots de passe, même quand ils sont hachés et cryptés, sont en effet vulnérables à des attaques par force brute. Votre projet consiste à développer une application qui sur base d'un fichier de mots et d'informations sur une base de données, liste les comptes utilisateurs qui sont liées à des mots de passe faibles.

Ce projet sera développé par phases. Dans cette première phase, votre programme utilise un terminal pour dialoguer avec l'utilisateur, une liste de mots de passe mémorisée en mémoire et une base de données également simulée en mémoire. **À terme,** l'application utilisera une interface graphique et attaquera une base de données réelle.

Concrètement

Préalable : Votre application doit prévoir deux classes : `WordSource` et `UserRepository`. Ces classes seront définies dans deux fichiers `word_source.py` et `user_repository.py` et placées dans un paquetage poo.

`WordSource` définit une séquence de mots et définit trois méthodes :

- `__init__(self, words)` qui initialise la séquence avec une séquence de mots ;
- `__getitem__(self, pos)` qui retourne le mot d'indice pos ou None en cas de pos incorrecte ;
- `count(self)` qui retourne le nombre mots.

Le comportement des objets de votre classe sera validé par des tests unitaires. Au minimum, ces deux tests doivent réussir.

```python
def test_provides_with_words_count(self):
    words = WordSource(["bonjour", "tout", "va", "bien"])

    found = words.count() # à modifier si propriété

    self.assertEqual(4, found)

def test_provides_with_word_given_pos(self):
    words = WordSource(["bonjour", "tout", "va", "bien"])

    found = words[2]

    self.assertEqual("va", found)
```

Degré de maitrise

- ✓ La classe encapsule la liste de mots (devez-vous exposer une propriété ?) ;

- ✓ La méthode __getitem__ valide le paramètre pos.
- ✓ La classe accepte des arguments variables
- ✓ `count` est manipulable comme une propriété.

`UserRepository` définit une collection d'utilisateurs et définit trois méthodes :

- `__init__(self)` qui initialise une liste de **dictionnaires ou d'objets User** correspondant au tableau ci-dessous ;
- `get_users_matching(self, clearPwd)` qui retourne la liste des utilisateurs dont le mot de passe correspond à `clearPwd`.

Le comportement des objets de votre classe sera validé par des tests unitaires. Au minimum, les tests suivants doivent réussir.

```
def test_get_user_for_pwd_bonjour(self):
    repository = UserRepository()

    found = repository.get_users_matching("password")

    self.assertListEqual(['P100077', 'P981234'], found)

def test_returns_empty_list_on_no_password_match(self):
    repository = UserRepository()

    found = repository.get_users_matching("rien à voir")

    self.assertEqual(0, len(found))
```

Degré de maitrise

- ✓ Les utilisateurs sont définis par des dictionnaires ou par une classe User
- ✓ La classe encapsule sa liste d'utilisateurs (devez-vous exposer une propriété users ?) ;
- ✓ La méthode `get_users_matching` valide le paramètre `clearPwd` (ce texte est défini et comporte au moins un caractère)

Vous initialiserez votre liste d'utilisateurs avec les données ci-dessous.

login	email	password
P070039	n.hendrikx@helmo.be	8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
P180039	d.lauwers@helmo.be	2cb4b1431b84ec15d35ed83bb927e27e8967d75f4bcd9cc4b25c8d879ae23e18
P100077	c.lorquet@helmo.be	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
P021234	l.swinnen@helmo.be	de77a8cd25c430bbbf84b7d7b1dbc3f62a17ecec5b9620a042e78c75d2700449
P051234	d.bayers@helmo.be	1e65dd734926e634e5e9b36bcbd1d6a6248b626fd2d2a6458f1e29b04fabaf3b
P981234	c.mathy@helmo.be	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

Notez que les mots de passe sont hachés avec l'algorithme sha-256.

Créez enfin un fichier program.py dans lequel vous créez un objet `WordSource` initialisé avec les mots suivants :

- 123456
- bonjour
- password
- mot de passe
- Die

Créez un objet `UserRepository`, puis bouclez sur les mots de `WordSource` et tentez de récupérer les utilisateurs correspondants à un mot de passe. En cas de correspondance, le programme affichera le login de l'utilisateur et son mot de passe en clair.

Pour ce faire, vous devrez hacher un mot avec l'algorithme sha-256. Utilisez à cette fin le module `hashlib`. Ce module propose des fonctions fabriquant des objets hachés. Vous pouvez alors obtenir le haché d'un string comme dans l'exemple ci-dessous.

```
import hashlib

pwd = "Je sens que je vais etre hache"

print(hashlib.sha256(pwd.encode('utf-8')).hexdigest())

>>>'c970792015dfc612986a75567c52bc02052b14b32b50754ef7a1679539bb374d'
```

Degré de maitrise

- ✓ Une structure contrôle vérifie que ce module est le module principal.

Sur quoi êtes-vous évalués ?

Pour être évaluable, votre projet doit comporter au minimum un paquetage poo et trois fichiers word_source.py, user_repository.py et program.py. Idéalement, le projet comporte également deux fichiers de tests unitaires.

Pour obtenir la note de 10/20, votre code doit :

- ✓ afficher le résultat attendu (indice : quatre utilisateurs ont un mot de passe foireux);
- ✓ utiliser à cette fin vos classes et le module hashlib;
- ✓ passer les tests unitaires donnés dans l'énoncé.

Les degrés de maitrise permettent alors d'obtenir une note supérieure à 10.

Exercice 1 : Utilisation de fichiers dans le projet d'intégration

Durée estimée : 120 minutes

Objectifs visés :

- Utiliser la bibliothèque csv
- Gérer des fichiers.

Description

Lors du projet d'intégration, vous avez mis en place un programme qui permettait de retrouver des personnes ayant un mot de passe faible même si celui-ci était hashé. Dans ce nouveau labo, il vous est demandé de modifier la manière dont sont chargées les informations depuis des dictionnaires vers des fichiers de type CSV.

Le programme

Dans votre package existant poo, créez un module `files`. Définissez-y une classe `FileUserRepository` et une autre `FileWordSource`. Ces deux classes doivent reprendre les mêmes méthodes que celles déjà existantes dans `UserRepository` et dans `WordSource`.

La réponse de l'exécution du programme, en plus de devoir être affichée à l'écran, sera ajoutée dans un fichier de rapport au format HTML sous ce format : `'{user} has a weak password = {password}'.`

WordSource

À la place de charger les différents mots de passe faibles comme dans l'extrait suivant, Il vous est demandé d'utiliser un fichier texte avec chaque mot de passe séparé par une virgule ",". Attention, il peut y avoir plusieurs mots par ligne. Cette classe sera testée par une suite de tests unitaires semblables à celle de la première évaluation.

```
words = WordSource(["123456", "bonjour", "password", "mot de passe",
"Nicolas"])
```

UserRepository

À la place de charger les différents utilisateurs comme dans l'extrait suivant, il vous est demandé d'utiliser un fichier de type csv avec chaque élément d'un utilisateur séparé par un point-virgule ";". Il y aura un utilisateur par ligne. Cette classe sera testée par une suite de tests unitaires semblables à celle de la première évaluation.

```
rows = [
        ('P070039', 'n.hendrikx@helmo.be',
'8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92'),
        ('P180039', 'd.lauwers@helmo.be',
'2cb4b1431b84ec15d35ed83bb927e27e8967d75f4bcd9cc4b25c8d879ae23e18'),
        ('P100077', 'c.lorquet@helmo.be',
'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8'),
        ('P048512', 'l.swinnen@helmo.be',
'de77a8cd25c430bbbf84b7d7b1dbc3f62a17ecec5b9620a042e78c75d2700449'),
        ('P023547', 'd.bayers@helmo.be',
'1e65dd734926e634e5e9b36bcbd1d6a6248b626fd2d2a6458f1e29b04fabaf3b'),
        ('P981234', 'c.mathy@helmo.be',
'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8')
 ]
```

Critères à valider

La plupart des critères de la première évaluation reste d'application. Nous serons également attentifs aux éléments suivants :

- Veuillez à toujours à bien fermer les fichiers ouverts une fois les informations parcourues.
- Veuillez à effectuer le moins de modifications de votre code possible pour permettre à cette nouvelle méthode de fonctionner.

Bonus

Pour ceux, qui, à cœur vaillant, rien n'est impossible, vous pouvez ajouter une fonctionnalité qui permettrait de gérer d'autres types de hash.

POO : Projet d'intégration

Acquis d'apprentissage évalués

Cette troisième itération évalue votre capacité à programmer une application en Python interagissant avec une BD. Vous disposez de deux heures à cette fin.

Description

Nous souhaitons appliquer le fichier de mots de passe à une base de données relationnelle (BD) de type SQLite. À cette fin, vous devez créer la BD, la remplir avec les instructions SQL fournies par le fichier annexe *database.sql*. La Figure 1 présente le schéma de la BD.

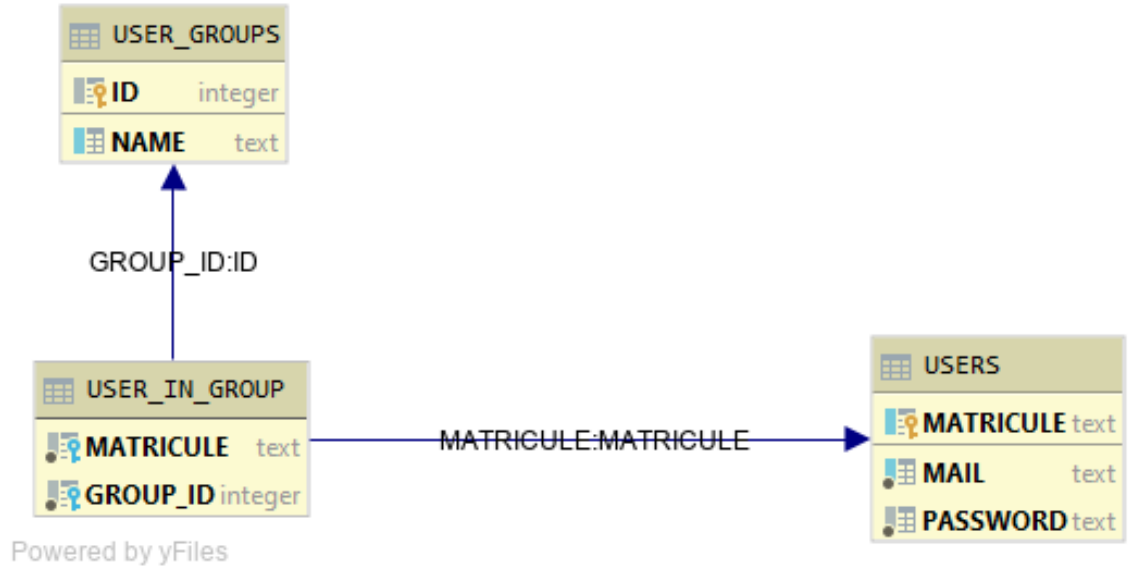

Figure 1 Schéma de la BD

Ensuite, comme lors de l'itération 2, vous produirez un fichier de rapport mentionnant les utilisateurs ayant un mot de passe faible et leurs rôles comme ci-dessous.

```
Résultat de l'inspection

| Critique | E-mail              | Mot de Passe | Roles         |
|----------|---------------------|--------------|---------------|
| x        | l.swinnen@helmo.be  | Bonjour      | ADMIN, LEADER |
|          | c.lorquet@helmo.be  | MotDePasse   | TEACHER       |
|          | d.lauwers@helmo.be  | 123456       | TEACHER       |
```

Une croix dans la colonne `Critique` signifie que l'utilisateur appartient au groupe des administrateurs.

Concrètement

Préalable : votre projet doit comporter les fichiers *poo\setup_db.py et poo\sql_user_repository.py* contenant une classe `SqlUserRepository`.

setup_db.py est un script lisant le fichier *database.sql* et générant une BD SQLite *poo.db* sur base des requêtes SQL qu'il contient. Attention, ce fichier contient des commentaires. Chaque commentaire occupe une ligne et commence par `--`.

Note : chaque requête du fichier est écrite sur une ligne.

Degré de maitrise

- ✓ Votre script affiche le message « schéma déjà généré » si la BD *poo.db* contient déjà les tables ou le message « données de la table xxx déjà présentes » si la BD *poo.db* contient déjà les données correspondantes.
- ✓ Vous appelez la méthode `commit()` de façon adéquate, c'est-à-dire à la fin d'une transaction logique (exemple : après les requêtes de création de tables).

`SqlUserRepository` définit une collection d'utilisateurs stockés dans une BD SQLite et définit les méthodes suivantes :

- `__init__(self, db_path, hash_algorithm)` initialise cet objet et crée une connexion à `db_path` et une référence sur l'algorithme de hachage utilisé ;
- `get_users_matching(self, clearPwd)` qui retourne la liste d'utilisateurs dont le mot de passe correspond à `clearPwd`.

Outre les données de l'itération précédente, chaque utilisateur retourné aura une entrée « roles » associée à un tuple énumérant ses rôles.

Degré de maitrise

- ✓ Les requêtes SQL sont paramétrées ;
- ✓ La connexion à la BD est fermée ;
- ✓ La connexion à la BD est fermée, même en cas d'erreur ;
- ✓ Les curseurs sont ouverts à l'aide de structures de contrôles `with`.

Adaptez le *program.py* pour qu'il travaille avec un objet `SqlUserRepository` et affiche le rapport dans un fichier *rapport.md* situé à la racine de votre projet.

Sur quoi êtes-vous évalués ?

Pour être évaluable, votre projet doit comporter au minimum un paquetage *poo* et les fichiers *poo\setup_db.py*, *poo\sql_user_repository.py* et *poo\program.py*.

Pour obtenir la note de 10/20, votre code doit :

- ✓ Créer la BD *poo.db* en exécutant le script *poo\setup_db.py*
- ✓ Créer un fichier *rapport.md* contenant le résultat attendu en exécutant le script *poo\program.py*
- ✓ Faire interagir *poo\program.py* avec votre BD *poo.db*
- ✓ Utiliser la classe `SqlUserRepository`

Les degrés de maitrise permettent alors d'obtenir une note supérieure à 10. Chaque degré de maitrise relatif à `SqlUserRepository` vaut 2 points.

Bon travail !