Exercice 2.1

Créer une classe **`Chien`**, avec au minimum les attributs « nom » et « age ».

Il ne doit pas être possible de changer le nom d'un chien.

Quand on demande l'âge d'un chien, on doit avoir deux valeurs, son âge réel et son âge « humain » (âge réel multiplié par 7).

Exercice 2.2

Toujours dans la classe **`Chien`**, il faut avoir la possibilité de comparer des chiens en utilisant les opérateurs de comparaison.

Un chien est supérieur (ou égal) ou inférieur (ou égal) à un autre en fonction de son âge.

Deux chiens sont égaux s'ils ont le même âge et le même nom.

Exercice 2.3

Créer une classe **`Animal`**, avec les attributs « nom » et « age ». Ajouter des méthodes **`dormir()`** et **`manger()`**.

Modifier la classe **`Chien`** pour qu'elle hérite de **`Animal`**. Ajouter des méthodes **`aboyer()`** et **`courir()`**.

Rendre la classe **`Animal`** abstraite.

Exercice 2.4

Faire en sorte que la classe **`Animal`** ne puisse plus être instanciée même si elle ne contient pas de méthode abstraite.

De même, faire en sorte que la méthode **`manger()`** de la classe **`Animal`** soit abstraite.

Exercice 2.5

Créer un décorateur qui permet de faire un contrôle de type sur des fonctions, mais en se basant sur les type hints cette fois-ci. Il faut contrôler à la fois les paramètres, mais aussi idéalement le type de retour.

Exercice 2.6

La fonction `range()` génère une série d'entiers continus.

Créer un itérable **`RandomRange()`** qui permet de générer des entiers aléatoires.

Les caractéristiques de ce **`RandomRange`** doivent être les suivantes :

- Il ne prend que des `int` en paramètre, et les trois paramètres sont obligatoires.
- Le premier paramètre est la borne initiale (inclusive), le deuxième paramètre est la borne finale (exclusive), le troisième paramètre est le nombre maximal de valeurs dans le `RandomRange`.
- Le `RandomRange` est vide initialement. Les valeurs sont ajoutées au fur et à mesure, soit à chaque `next()`, soit à chaque tour de boucle `for`.
- Lorsqu'on *print* un objet de type `RandomRange`, les valeurs actuelles doivent s'afficher entre chevrons (exemple : `<5, 3, 0, 9, 5, 1>`)

Pour simplifier, l'itérable sera son propre itérateur.

Exercice 2.7

La connexion à une base de données avec le module **`sqlite3`** se fait toujours sur le même principe :

```
import sqlite3

conn = sqlite3.connect("fichier.db")

cursor = conn.cursor()

cursor.execute("""Requête SQL ...""")

# Opérations sur l'objet cursor

conn.commit()

conn.close()
```

Étant donné que certaines de ces instructions sont redondantes, créer un *context manager* qui permet de simplifier ces appels, dans lequel on n'aura à écrire que les instructions sur l'objet **`cursor`**.

Utiliser les deux méthodes de création de *context manager* : avec une classe et avec une méthode. Faire quelques tests en exécutant des requêtes SQL d'insertion et de consultation.

Quelques exemples de requêtes SQL :

```
CREATE TABLE IF NOT EXISTS users(
     id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
     name TEXT,
     age INTERGER
)
```

```
INSERT INTO users(name, age) VALUES('John', 40)
```

```
SELECT name, age FROM users
```

Exercice 2.8

Reprendre la classe `Chien`.

Créer une classe d'exception personnalisée **`AgeIncoherentError`**. Une exception de ce type doit être levée lorsque l'on tente de donner un âge négatif à un chien ou bien supérieur à 20 ans.