Session d'exercices – Tuples (n-uplets) et Sets (Ensembles)

Aujourd'hui, vous apprendrez à utiliser les n-uplets (tuples) et les ensembles (sets) en Python. Vous explorerez leur syntaxe, leurs propriétés distinctives, et leurs applications pratiques. Ces compétences vous seront utiles pour des tâches plus complexes, notamment l'analyse de données.

Exercice 1: n-uplets

Un n-uplet (*tuple*) est une séquence finie d'éléments qui se distingue d'un ensemble par deux propriétés principales :

- Présence de doublons : Un tuple peut contenir plusieurs occurrences du même élément. Par exemple, $(1, 2, 2, 3) \neq (1, 2, 3)$, tandis que l'ensemble $\{1, 2, 2, 3\} = \{1, 2, 3\}$.
- Ordre des éléments : Les tuples respectent l'ordre des éléments, contrairement aux ensembles. Ainsi, $(1, 2, 3) \neq (3, 2, 1)$, mais $\{1, 2, 3\} = \{3, 2, 1\}$.

Utilisez le shell Python interactif ou un fichier dédié (`tuples.py`) pour compléter les tâches suivantes.

1. [Difficulté: Warm up]

   Testez et analysez les exemples suivants :

   ```
   # Un tuple peut contenir différents types.
   tuple_icc = ("icc", "theory", "exercises")
   print(tuple_icc[0], tuple_icc[2])

   # Décomposition des tuples.
   first, second, _ = tuple_icc # L'underscore '_' ignore les éléments inutiles.
   print(first, second)

   # Les tuples sont immuables. Pouvez-vous expliquer ce que cela signifie ?
   ```

   Consultez la documentation Python officielle pour approfondir.

2. [Difficulté: *] Déclarez un tuple contenant vos fruits préférés et utilisez une boucle `for-in` pour afficher chaque fruit.

3. [Difficulté: *] Vérifiez si un fruit particulier est présent dans votre tuple. Existe-t-il une méthode dédiée pour cela, ou devez-vous parcourir le tuple manuellement ?

4. [Difficulté: *] Essayez de modifier le deuxième fruit de votre tuple préféré en "strawberry". Que se passe-t-il ? Pourquoi ?

5. [Difficulté: **] Écrivez une fonction `slice_tuple()` qui divise un tuple en deux. La première moitié doit contenir un élément de plus si la longueur est impaire. Exemple :

   ```
   >>> slice_tuple(("icc", "theory", "exercises"))
   (('icc', 'theory'), ('exercises',))
   ```

Exercice 2: Ensembles

1. [Difficulté: Warm up]

   Testez les exemples suivants et effectuez les opérations d'ensembles classiques : union, intersection, différence, et différence symétrique. Quelle est la différence entre la différence simple et la différence symétrique ?

```
students_icc = set(["Jeremy", "Victoria", "Paul"])
students_in = set(["Victoria", "Alexander"])
students_all = set(["Jeremy", "Victoria", "Paul", "Alexander"])
```

   Que se passe-t-il lorsque vous essayez d'imprimer un ensemble vide après une opération ?

   Le résultat attendue est ci-dessous.

```
Union between students_icc and students_in:
{'Alexander', 'Jeremy', 'Victoria', 'Paul'}
Intersection between students_icc and students_in:
{'Victoria'}
Symmetric Difference between students_icc and students_in:
{'Alexander', 'Paul', 'Jeremy'}
Difference:
students_icc - students_in:
{'Paul', 'Jeremy'}
students_in - students_icc:
{'Alexander'}
students_icc - students_all:
set()
```

2. [Difficulté: **] Pensez aux propriétés des ensembles vues en cours et essayez de les vérifier en python: Créez plusieurs ensembles avec des éléments de types et ordres différents. Qu'observez-vous ?

3. [Difficulté: *] Supprimez efficacement les doublons dans la liste suivante :

```
['foo.py', 'bar.py', 'a', 'b', 'foo.py', 'b']
```

4. [Difficulté: **] Les ensembles ne sont pas immuables, mais vous pouvez créer un ensemble immuable avec `frozenset()` (documentation). Quand pourriez-vous en avoir besoin ? Proposez un exemple simple.