M1102: Introduction à l'algorithmique et à la programmation

Feuille de TD n°6

Structures de données imbriquées

INSTRUCTIONS: à la fin du TD, vous devez déposer sous Celene une archive zip contenant les fichiers Python correspondant à ce TD (fournis avec la feuille). Vous pourrez déposer une version améliorée de votre travail jusqu'à la **VEILLE** du TD suivant.

L'évaluation de la période 2 sera basée sur ces rendus.

Objectif

Cette feuille de TD a pour objectif de manipuler des structures de données plus compliquées que celles utilisées jusqu'à présent. Par exemple, on va travailler avec des tuples qui contiennent des listes, des listes qui contiennent des tuples ou même des listes qui contiennent des listes. A quoi cela peut-il bien servir ? Par exemple, si on prend la liste des meilleurs scores d'un jeu, on aimerait bien que cette liste contienne des éléments qui donnent à la fois le score obtenu ET le nom du joueur qui a effectué ce score. Si notre classement est

1. Batman 320043
2. Robin 280481
3. Batman 245023
4. Joker 124566
5. Batman 113456

on peut le représenté par la liste de couple

```
best=[("Batman",320043),("Robin",280481),("Batman",245023),("Joker",124566),("Batman",113456)]
```

Ainsi `best[2]` est le couple `("Batman",245023)` et `best[2][0]` donne le nom du joueur qui a obtenu ce score et `best[2][1]` donne le nombre de points obtenus.
On peut aussi imbriquer des listes dans des tuples. Par exemple, si on veut représenter un joueur et la liste de tous les scores qu'il a obtenus, on pourrait utiliser un couple dont le premier composant est le nom du joueur et le second la liste de ses scores :

```
joueur=("Batman",[320043,245023,113456,102355])
```

On remarque que `joueur[0]` donne le nom du joueur et `joueur[1]` donne la liste de ses scores. Enfin `joueur[1][2]` donne le 3^e score du joueur (`113456` sur l'exemple).
Dans ce contexte, une liste de joueurs de ce type serait une liste dont les éléments sont des tuples qui eux-mêmes contiennent des listes.

```
lj=[("Batman",[320043,245023,113456,102355]),("Robin",[280481,102365])]
```

Les exercices de la feuille vont vous permettre de vous familiariser avec ce type de structures de données.

Exercice 1 *Accéder aux éléments d'une structure imbriquée*

Instructions: *Pour cet exercice vous devez compléter le fichier* `td6-Exercice1.py`

A. Listes de tuples

Pour un site de vente par internet, les articles mis en vente possèdent trois propriétés : une référence, un nom et un prix unitaire. Pour représenter ces articles en Python on va utiliser des triplets `(ref,nom,prix)`. Voici un exemple de liste d'articles mis en vente :

```
art=[(152,"Chaussures",37.5),(145,"Veste",87.2),(147,"T-shirt",25.3),(165,"Bonnet",11.0)]
```

1. Que valent `art[1]`, `art[0][1]`, `art[0][1][0]` et `art[3][2]` ?
2. Quelle expression permet d'obtenir le troisième article de la liste ?
3. Quelle expression permet d'obtenir le prix du deuxième article de la liste ?
4. Quelle expression permet d'obtenir le nom du dernier article de la liste ?
5. Donner l'instruction qui permet d'ajouter à la fin de la liste une chemise de référence 256 et de prix 51.1.

B. Tuples contenant des listes

Une commande va contenir un numéro, un nom de client, et la liste indiquant les produits commandés et leur quantité commandée. Voici un exemple de commande dont le numéro est 174 et le client Dupont : `commande=(174,"Dupont",[(145,1),(147,5),(152,1)])`
Ce client a donc commandé 1 exemplaire de l'article 145, 5 exemplaires de l'article 147 et un exemplaire de l'article 152.

1. Que représente respectivement `commande[0]`, `commande[1]` et `commande[2]` ?
2. Que valent `commande[2][0]`, `commande[2][0][0]` et `commande[2][0][1]` ?
3. Quelle expression donne la référence du deuxième produit commandé par Dupont ?
4. Quelle expression donne le nombre d'exemplaires du troisième produit commandé par Dupont ?

Exercice 2 *Collection de Python3 et autres serpents*

Instructions: *Pour cet exercice vous rendrez le fichier* `serpents.py` *complété.*

Un passionné de serpents décide de programmer une petite application lui permettant d'archiver un certain nombre d'informations sur ses animaux préférés. Il choisit donc le langage Python pour développer cette application. Les informations qu'il souhaite garder sont les suivantes :

— son nom,
— sa longueur moyenne en mètres,
— sa dangerosité (un entier de 0 à 5 : 0 étant inoffensif et 5 extrêmement dangereux).

Il décide donc de représenter les serpents par des triplets de la forme `(nom,longueur,danger)`. Le triplet `("Python3",0.3,0)` nous indique que Python3 est très court (30 cm) et inoffensif. Complétez le script `serpents.py` pour implémenter ces fonctions. Ce script contient un programme principal qui utilise les fonctions que vous devez développer. Vous pourrez essayer ce programme notamment en tapant la commande `python3 serpents.py` dans un terminal. Vous pouvez aussi lancer la baterrie de tests prévue pour cette feuille en tapant la commande `./test_serpents.py`.

A. Fonctions d'entrées-sorties

1. Une fonction qui transforme les informations d'un serpent sous la forme d'une chaine de caractères qui a la forme suivante :

```
--------------------
Nom: Python3
Taille: 0.3
Danger: 0
--------------------
```

2. Une fonction qui transforme une liste de serpents en une chaine de caractères donnant les informations de chaque serpent de la liste (en utilisant la précédente)
3. Une fonction qui permet de demander à l'utilisateur de saisir les informations d'un serpent et retourne un triplet contenant ces informations (avec le bon type).
4. Une fonction qui permet de rajouter des serpents saisis par l'utilisateur dans la liste (cette fonction utilise la fonction précédente). L'utilisateur doit pouvoir ajouter autant de serpents qu'il le souhaite.
5. Une fonction qui charge une liste de serpents à partir d'un fichier texte dont chaque ligne a la forme suivante `nom du serpent,taille,dangerosité`. Par exemple, pour notre serpent préféré la ligne sera `Python3,0.3,0`. Cette fonction doit prendre en paramètre le nom du fichier et doit retourner la liste des serpents qu'il contient.
   Indication : La méthode `split` permet de découper une chaîne de caractères suivant un séparateur. Cette méthode retourne une liste de chaînes de caractères. Il vous est fourni un fichier `serpents.txt` contenant 13 serpents.
6. Une fonction qui sauvegarde une liste de serpents dans un fichier texte de même format que la question précédente. Cette fonction prend en paramètres un nom de fichier et une liste de serpents et elle retourne le nombre de serpents sauvegardés.

B. Fonction de consultation et de calculs

1. Une fonction qui recherche la liste des noms des serpents les plus dangereux.
2. Une fonction qui calcule la taille moyenne des serpents de dangerosité 4.
3. Une fonction qui retourne une liste qui indique le nombre de serpents qui correspond à chaque niveau de dangerosité.