

TP 1: Package, module, sequences, generateurs, etc.

1 Package, module

Reprenons et simplifions l'exemple d'organisation de packages (`sound`, `formats` et `effects`) et modules (`wavread`, `wavwrite` et `echo`) proposés dans le cours :

```
sound
    formats
        wavread
        wavwrite
    effects
        echo
```

Chaque module possède une seule fonction de même nom qui l'affiche simplement. Par exemple le module `wavread` possède la fonction :

```
def wavread():
    print("wavread")
```

1. Mettez en place cette architecture ;
2. Lancez ipython depuis le répertoire où se trouve le package `sound`, puis :
 - importez le module `wavread` et exécutez la fonction `wavread` ;
 - importez uniquement la fonction `wavwrite` et exécutez la ;
 - importez uniquement la fonction `echo` en la renommant `ec` et exécutez la.
3. Recommencez ces actions mais depuis votre home (vous pouvez utiliser la commande UNIX `env` pour modifier temporairement des variables de votre environnement).

2 Module listtools

Dans l'archive `TP01.zip` disponible sur moodle il y a dans le répertoire `TP01` un module `listtools` qui propose les fonctions suivantes :

- `nb_occurrences` qui permet d'obtenir le nombre d'occurrences de `element_a_compter` dans `sequence` ;
- `mettre_en_majuscule` qui calcule une liste avec les chaînes de caractères en majuscule de `sequence` en majuscule ;
- `retirer` qui calcule une nouvelle liste qui contient les éléments de `sequence` excepté `element_a_retirer` ;
- `appliquer_l'équivalent_de_map` ;
- `decouper_en_listes_meme_longueur` qui découpe `sequence` en listes de même longueur (sauf peut être pour la dernière). Cette fonction retourne la liste de ces listes.

Lorsque l'on utilise ce module en tant que script, des tests unitaires permettent de valider (tests non exhaustifs) les algorithmes de ces fonctions :

```
% python listtools.py
nb_occurences ko
mettre_en_majuscule ko
retirer ko
appliquer sans paramatre ko
appliquer avec paramatre ko
decouper_en_listes_meme_longueur 1 ko
decouper_en_listes_meme_longueur 2 ko
```

Codez les algorithmes de ces fonctions en utilisant les compréhensions de liste lorsque c'est nécessaire.

3 Générateur

Dans l'archive TP01.tar.gz il y a aussi le module `fibonacci.py`.

Complétez le corps de la fonction `suite_fibonacci` tel que :

- sans argument cette fonction permet de calculer tous les nombres de la suite de fibonacci ;
- avec un argument de type naturel n , elle permet d'obtenir les n premiers éléments de la suite de fibonacci.

Encore une fois, l'exécution de ce script vous permettra de vérifier le bon fonctionnement de votre code.

4 Question subsidiaire

Reprenez l'algorithme de `decouper_en_listes_meme_longueur` pour que la fonction retourne un générateur plutôt qu'une liste.