

**Durée du CC : 1h30.**

Les seuls documents autorisés sont la Fiche-Triche Python et la documentation de la bibliothèque BeautifulSoup.

Toutes les questions sont indépendantes ; on pourra les traiter dans l'ordre de son choix.

## 1 Questions générales

### Question 1

1. En utilisant une boucle *for*, stocker dans la variable `l1` la liste de tous les entiers (rangés par ordre croissant) compris entre 0 et 1000 qui sont divisibles par 4 mais pas par 5.
2. En utilisant une définition par compréhension, définir une liste `l2` contenant les mêmes éléments.

### Question 2

1. Écrire une classe `Wagon`, dont le constructeur attendra un entier `nb_place` (correspondant au nombre de places dans le wagon). Au départ, toutes les places sont libres. Cette classe devra être dotée
  - d'une méthode `est_complet()` qui répond `True` s'il ne reste plus de places libres.
  - d'une méthode `reserver()` qui réserve une place dans le wagon (autrement dit, qui décrémente le nombre de places libres) si le wagon n'est pas complet, et ne fait rien autrement.Un `Wagon` avec 34 places libres sur un total de 60 devra s'afficher sous la forme `"34 places libres / 60"`.
2. Définir une classe `Train`, dont le constructeur attendra une liste de `Wagon`. Doter cette classe d'une méthode `reserver()`, qui réservera une place dans le premier wagon non-complet (s'il en existe un). Pour implémenter cette méthode, on ne touchera évidemment pas aux attributs de `Wagon`.

**Question 3** Écrire une fonction génératrice (en utilisant le mot-clef `yield`) `gen_premiers()`, qui renvoie au compte-gouttes les nombres premiers. En particulier, les quatre premières valeurs renvoyées seront respectivement 2, 3, 5 et 7.

On ne cherchera pas à implémenter efficacement cette fonction.

## 2 BeautifulSoup

Téléchargez le fichier `King's_Landing.html` sur Eprel.  
Ajoutez le code suivant en début de fichier :

```
from bs4 import BeautifulSoup

kl = open("King's_Landing", 'r')
klsoup = BeautifulSoup(kl, "html.parser")
```

Ainsi, l'objet de type `soup` décrivant le fichier HTML sera stocké dans la variable `klsoup`.

On testera toutes les questions suivantes sur cette page HTML, en gardant bien en tête que le code proposé devra être générique et fonctionner sur n'importe quelle page.

**Question 4** Écrire une fonction `titre(soup)` qui renvoie le titre de la page représentée par `soup`. On fera bien attention à ne pas confondre *titre* et *h1*...

Résultat attendu sur l'exemple :

King's Landing | A Song of Ice and Fire

**Question 5** Écrire une fonction `afficher_h2(soup)` qui affiche le nom de tous les *h2* de la page représentée par `soup`, à raison de un par ligne.

Résultat attendu sur l'exemple :

Layout  
Population  
Military Forces  
History

**Question 6** Écrire une fonction `nb_par_avec_lien(soup)` qui renvoie le nombre de paragraphes (i.e. de balises *p*) contenant au moins un lien.

Résultat attendu sur l'exemple :

10