Durée du CC : 1h30.

Les seuls documents autorisés sont la Fiche-Triche Python et la documentation de la bibliothèque Beautiful Soup.
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 1000 et 2000 qui sont divisibles par 3 ou par 7.
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 : on va donc créer une liste de taille `nb_place` contenant uniquement des `None`.
2. Définir une classe `Place_occupee` qui étend `Exception`. Cette classe pourra tout simplement hériter son constructeur d'`Exception`.
3. Doter cette classe d'une méthode `reserver(place,nom)` qui permet de réserver une place à un nom : concrètement, on veut ajouter la chaîne de caractère `nom` dans la liste, à la position `place`. Attention : si la place est déjà occupée, on veut à la place lever une exception `Place_occupee`.

Question 3 Écrire une fonction génératrice (en utilisant le mot-clef `yield`) `gen_puissances()`, qui renvoie au compte-gouttes les puissances de 2. En particulier, les quatre premières valeurs renvoyées seront respectivement `1`, `2`, `4` et `8`.

2 Beautiful Soup

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 `afficher_h1(soup)` qui renvoie le *h1* 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
```

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 `max_liens_par(soup)` qui renvoie le nombre de liens contenus dans le paragraphe (i.e. la balise *p*) qui en contient le plus.
Résultat attendu sur l'exemple :

```
7
```