Exercice 1 : Voici un message codé en ASCII :

56 49 56 45 20 4C 45 53 20 56 41 43 41 4E 43 45 53 0A

1. Les caractères sont-ils notés en décimal ou hexadécimal ?
2. Convertir le caractère 20_{hex} en binaire.
3. Le convertir en décimal.
4. En s'aidant d'une table ASCII, décoder le message.
5. Pour quel raison ne peut-on pas encoder le mot **éléphant** en ASCII ?

Exercice 2 :

1. Le caractère @ a le point de code **U+0040**. Convertir le point de code en binaire.
2. Combien d'octets sont nécessaires pour encoder ce caractère en UTF8 ?
3. Mêmes questions pour le caractère Ê.

Exercice 3 :

1. Rappeler et appliquer le principe de conversion d'un entier décimal en binaire.
2. Convertir 42 en binaire.
3. Écrire la fonction `deci_bin(entier: int) → str` qui convertit l'`entier` décimal en binaire. La représentation binaire sera renvoyée sous forme d'une chaîne de caractère. Par exemple :

```
>>> deci_bin(11)
'1011'
```

4. Donner le point de code en décimal des lettres du mot **SALUT**. Pour rappel, la norme UTF-8 est compatible avec le code ASCII.
5. La fonction native Python `chr(n: int) → str` renvoie le caractère correspondant au point de code `n`. Le point de code est ici représenté en écriture décimal. Par exemple, la lettre **é** est représenté par le point de code hexadécimal `U+00E9` équivalent à la valeur `233` décimale. Écrire la fonction `decoder(code_car: list) → str` qui renvoie le mot correspondant aux points de code décimaux stockés dans le tableau `code_car`.

Exercice 4 :

1. Trouver le rôle des fonctions natives Python **ord** et **hex**.
2. Écrire la fonction `utf8(car: str) → str` qui renvoie le point de code hexadécimal du caractère `car`.
3. Écrire la fonction `encoder_hexa(phrase: str) → list` qui renvoie le tableau des points de code hexadécimaux de chaque lettre de `phrase`.

Christophe Viroulaud