1 Problématique

En première approche il semble simple de représenter une chaîne de caractère en mémoire : il suffit d'associer un numéro à chaque lettre. En pratique plusieurs contraintes apparaissent. Il faut par exemple que chaque système respecte le même encodage. De plus tous les caractères doivent être représentés.

Comment encoder les caractères en mémoire ?

2 Première tentative de normalisation : ASCII

Dans les années 50, lors de l'apparition des premiers ordinateurs, chaque matériel utilisait son propre codage.
L'American National Standards Institute (ANSI) propose, au début des années 60, une première tentative de normalisation : l'**ASCII**.

Activité 1 : Répondre aux questions suivantes en effectuant des recherches sur le web.

1. Que signifie le sigle ASCII ?
2. Combien de bits sont utilisés dans le codage ASCII ?
3. À l'aide d'une table ASCII, encoder la phrase suivante en hexadécimal :

   La NSI est 1 super discipline !

4. Quelles sont les limitations de l'encodage ASCII ?

3 Prise en compte des différentes langues : ISO 8859

La norme ASCII est suffisante pour écrire l'anglais. Cependant de nombreuses langues utilisent des caractères additionnels non présents dans cette norme.
L'International Organization for Standardization (ISO) a alors proposé une extension de l'encodage ASCII : **ISO 8859**. Il utilise 8 bits et assure une compatibilité avec l'ASCII : les 128 premiers caractères de la norme ISO 8859 sont ceux de la norme ASCII.

Activité 2 : Répondre aux questions suivantes en effectuant des recherches sur le web.

1. Combien de caractères peut-on encoder dans une table ISO 8859 ?
2. Combien de tables ISO 8859 existe-t-il ?
3. Quelle est la table utilisée pour le français ?
4. Encoder la phrase française suivante en utilisant la norme ISO 8859 :

   À quelle âge apprendre la NSI ?

4 Encodage universel

4.1 Nouvelle norme

Bien que l'ISO 8859 permette de représenter un très grand nombre de caractère, il n'est par exemple pas possible d'encoder un texte qui contient des caractères de plusieurs tables. L'ISO a donc défini un

Christophe Viroulaud

jeu universel de caractère (norme ISO-10646). Chaque caractère, signe, idéogramme est associé à un numéro unique appelé **point de code**. La capacité maximale a été fixée à 32 bits.

Activité 3 :

1. Combien de caractères peut-on représenter avec la norme ISO-10646 ?
2. Combien d'octets représentent 32 bits ?
3. Trouver le point de code en hexadécimal de la lettre é. La convertir en binaire.

4.2 Représentation en mémoire

Une représentation naïve utiliserait quatre octets pour chaque caractère, ce qui peut s'avérer très coûteux. La norme **Unicode** définit plusieurs techniques pour économiser de l'espace. Nous étudierons particulièrement l'**UTF-8** (Universal Transformation Format), très utilisé sous Linux, les sites web... Le principe est le suivant :

— Si le *bit de poids fort* (le plus à gauche) est 0, il s'agit d'un caractère ASCII codé sur les 7 bits suivants.

— Sinon les premiers bits de poids fort de l'octet indiquent le nombre d'octets utilisés à l'aide de 1 et se terminant par 0.

Le tableau ci-après montre le nombre d'octets utilisés pour encoder les caractères en UTF-8.

Suite d'octets (en binaire)	Bits codant
0xxxxxxx	7 bits
110xxxxx 10xxxxxx	11 bits
1110xxxx 10xxxxxx 10xxxxxx	16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	21 bits

Activité 4 :

1. Combien d'octets sont nécessaires pour encoder la lettre é ?
2. Donner la représentation binaire en UTF-8 de la lettre é.
3. Trouver le rôle des fonctions natives Python `ord` et `hex`.
4. Écrire la fonction **utf8(car : str) → str** qui renvoie le point de code hexadécimal du caractère *car*.

Christophe Viroulaud