Encodage des textes

I – ASCII

ASCII associe 128 valeurs encodées sur 7 bits à 128 caractères bien précis.

Il s'agit d'un standard respecté par toutes les autres tables.

128 caractères sur 7 bits car _____

Caractère	Binaire	Décimal	Hexédécimal
Saut ligne **\n**	000 1010	**10**	0A
Espace	010 0000	32	20
A	100 0001	65	41
B	100 0010	66	42

II – Encodage avec Python

2.1 Encodage

Encoder un texte veut dire le transformer en suite d'octets.

En Python, on applique la méthode **encode** aux strings qu'on veut transformer en octets en transmettant le système d'encodage qu'on veut utiliser.

```
>>> octets = "AB\nCDE".encode('ascii')
>>> type(octets)
 <class 'bytes'>

>>> print(octets)
b'AB\nCDE'
```

Le **b** signale qu'il s'agit d'octets. Si la valeur correspond à un caractère ASCII, Python l'affichage plutôt que sa valeur.

Le type **bytes** consiste donc en une simple suite d'octets sans signification propre. Bytes peut être traduit par octets.

2.2 Nombre d'octets

```
>>> len(octets)
6
```

On obtient 6 qui on encode bien 6 caractères : A, B, passage à la ligne, C, D et E.

2.3 Lecture des octets

```
for oct in octets: print(oct, end=" ")
 65 66 10 67 68 69

for i in range(len(octets)) : print(octets[i])
 même résultat

>>> t = list(octets)
>>> t
[65, 66, 10, 67, 68, 69]
```

III – Les tables «ASCII» étendues à 1 octet

Ces tables permettent d'utiliser le 8e bit comme bit de caractères et non plus comme bit de parité pour la transmission.

On dispose de 128 nouveaux caractères car
Il existe de mulitples tables, compatibles entre elles sur les 127 premières valeurs et associant des caractères différents de 128 à 255.

On a au total 256 caractères car _____

La table **latin-1** ou **iso8859_1** est la première table adaptée pour l'Europe de l'Ouest.

La table **latin9** ou **iso8859_15** est sa mise à jour suite à l'apparition de l'euro.

IV – Décodage

Lorsqu'on récupère une suite d'octets, il faut le décoder. Encode faut-il savoir comment.

Est-ce une image ? JPG ou PNG ?
Un texte ? Si c'est un texte, avec quel système d'encodage faut-il décoder ?

Avec Python et un flux d'octets correspondant à un texte, il faut utiliser la méthode **decode**.

```
texte = octets.decode('ascii')
texte = octets.decode('latin-1')
texte = octets.decode('latin9')
```

V – Unicode

UNICODE n'est pas un système d'encodage.
Il s'agit d'un système associant une glyphe ou un caractère à un numéro.

Nombre d'octets nécessaires théoriques

UNICODE 13 comporte 143 859 glyphes dont les numéros sont sur une plage 0 à 16 842 751.

Puisque 2^{24} ne vaut que 16 777 216, il faudrait 4 octets pour encoder cette plage de façon naïve.

Python et UNICODE : chr

La fonction native **chr** renvoie le caractère associé à un numéro UNICODE.

```
>>> chr(65)
'A'

>>> chr(10000)
'✐'

>>> chr(20000)
'北'

>>> chr(0x41)
'A'
```

Les numéros sont compatibles avec **ASCII sur 0-127.**
Les numéros sont compatibles avec **latin-1 sur 0-255**.
Plutôt que numéros, on parle de **points de code**.

Python et UNICODE : ord

La fonction native **ord** renvoie le numéro UNICODE associé à un caractère.

```
>>> ord('A')
65

>>> ord('✐')
10000

>>> ord('北')
20000
```

Python et HTML

On peut afficher n'importe quel caractère dans une page HTML si on connaît son numéro UNICODE.

Si on connaît la valeur décimale :
&#129497**;**

Si on connaît la valeur hexadécimale :
&#x1F9D9**;**

VI – UTF-8

Il s'agit de l'encodage le plus courant pour implémenter UNICODE actuellement.
Il s'agit presque d'un standard tellement il est utilisé.

Principe :

Caractères ASCII : sur 1 octet.

Caractères peu communs : 2 octets.

Caractères rares : 3 octets.

Caractères très rares : 4 octets.

UTF-8 veut donc dire que les caractères sont encodés sur 1 octet ou plus.

Et les deux autres ?

UTF-16 veut dire que les caractères sont encodés sur 2 octets ou plus.

UTF-32 veut dire que les caractères sont encodés sur 4 octets.

Comparaison

Si nb est le nombre de caractères du texte encodé, la taille en octets d'un fichier encodé en UTF-8 est donc comprise **entre nb et 4 nb.**

Annexes : méthodes des strings

Méthodes upper et lower

Ces méthodes renvoient une version en majuscules ou en minuscules.

```
>>> "AbCd".upper()
'ABCD'
>>> "AbCd".lower()
'abcd'
```

Méthode split

Cette méthode des strings renvoie un tableau en divisant le string.

```
>>> "A B CD".split(" ")
['A', 'B', 'CD']
```

Méthode rjust

Cette méthode renvoie une version du string de X caractères en complétant à gauche si besoin avec le caractère fourni.

```
>>> "0011".rjust(8, '-')
'----0011'
```

Méthode replace

Cette méthode renvoie un nouveau string en remplaçant les caractères désignés par les seconds. Elle sert notamment à supprimer certains caractères.

```
>>> "Bonjour à tous\n".replace("\n", "")
'Bonjour à tous'
```