

2.4

Représentation des caractères

NSI 1ÈRE - JB DUTHOIT

On parle ici de codage des caractères, c'est-à-dire qu'il faut disposer d'un système de code permettant de traduire des caractères (lettres) en nombres.

2.4.1 Code ASCII

American Standard Code For Information Interchange → sur 7 bits, donc 2^7 représentations possibles

Tableau de conversion ASCII

				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
				b4	0	1	0	1	0	1	0	1
b3	b2	b1	b0	Hex	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(8	H	X	h	x
1	0	0	1	9	HT	EM)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	
1	1	0	1	D	CR	GS	-	=	M]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	S1	US	/	?	O	-	o	DEL

Savoir-Faire 2.14

Décoder le message suivant :

1000010 1110010 1100001 1010110 1101111 0100001

Savoir-Faire 2.15

Coder le message "Hello JB"

1. en binaire
2. En hexadécimal

2.4.2 Python et les codes ASCII

Python intègre nativement des méthodes et fonctions permettant de convertir un caractère en représentation ASCII décimale (entre 0 et 127) et vice-versa :

La fonction `ord()` donne la valeur décimale ASCII d'un caractère :

```
ord('B') affiche 66
```

`chr()` renvoie le caractère de valeur décimale ASCII donnée :

```
chr(65) affiche 'A'
```

Exercice 2.6

Écrire un script Python qui affiche les 256 caractères de la table ASCII étendue. De quel type sont ces caractères ?

2.4.3 code ASCII étendu

Le codage ASCII est souvent complété par des correspondances supplémentaires afin de permettre l'encodage informatique d'autres caractères, comme les caractères accentués par exemple. Cette norme s'appelle ISO-8859 et se décline par exemple en ISO-8859-1 lorsqu'elle étend l'ASCII avec les caractères accentués d'Europe occidentale -> sur 8 bits donc 2^8 représentations possibles.

2.4.4 La norme Unicode

L'Unicode désigne un système de codage utilisé pour l'échange de contenus à l'échelle internationale, dans différentes langues, en faisant fi de la langue, de la plateforme et du logiciel utilisé pour cet échange.

l'Unicode permet de coder l'ensemble des caractères couramment utilisés dans les différentes langues de la planète en spécifiant un nombre unique pour chacun de ces caractères.

Quelques langages ont un très grand nombre de caractères comme le chinois, le japonais ou le coréen. Il a donc fallu utiliser au moins deux octets au lieu d'un pour les représenter. Avec deux octets, on dispose de $256 \times 256 = 65536$ codes.

Le consortium Unicode a pour devise "**un seul nombre pour chaque caractère, quelque soit la plate-forme, le programme ou le langage**".

Chaque symbole d'écriture est représenté par une valeur hexadécimale préfixée par « U+ ».

Consulter la norme unicode

Contrôles C1 et supplément latin-1 [modifier | modifier le code]

PDF : en [archive] v · d · m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+0080	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
U+0090	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
U+00A0	NBSP	í	¢	£	¤	¥	¡	§	º	©	ª	«	¬	- SHY	®	—
U+00B0	º	±	²	³	‘	µ	¶	·	,	¹	º	»	¼	½	¾	¿
U+00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
U+00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
U+00E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
U+00F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Exemple de table unicode

Exercice 2.7

Soit la liste suivante : liste = [233, 112, 97, 116, 97, 110, 116, 32, 33], qui correspond à la liste des codes Unicode d'une chaîne. Laquelle ?

1. Méthode 1 Compléter le programme suivant afin qu'il affiche le résultat demandé :

```
for i in liste:
    print(.....)
```

2. Méthode 2 : Adapter le programme précédent afin qu'il affiche le résultat demandé :

```
for i in range(len(liste)):
    print(.....)
```

3. Méthode 3 : Vous écrirez une fonction python decodage(lst) :

```
def decodage(lst):
    '''lst est une liste d'entier contenant les codes
    La fonction renvoie une chaîne de caractères
    qui correspond au décodage.
    '''
    pass
```

2.4.5 L'UTF-8

Pour résoudre le problème du doublement (ou triplement) de la taille des fichiers, Ken Thompson, un des pères d'UNIX, a littéralement inventé le 2 septembre 1992 le codage UTF-8, une autre manière de coder les caractères Unicode. UTF-8 code chaque caractère sur 8, 16, 24

ou 32 bits L'idée a été de coder les caractères les plus utilisés de 0 à 127 sur un octet, de coder les caractères de 128 à 1023 sur 2 octets et au-dessus de 1023 sur 3 octets et ce jusqu'à 4 octets. il est devenu le standard de l'Internet, donc de l'informatique. Par exemple, l'UTF-8 est devenu le codage par défaut des documents en XML donc en XHTML.

⚠ Il ne faut pas confondre Unicode qui identifie des caractères par des nombres et l'encodage utf8 qui va associer à (la plupart) des caractères Unicode une suite d'octets. D'autres encodages des caractères Unicode comme utf-8 par exemple donneront en général une autre suite d'octets.

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet codant 1 à 7 bits
110xxxxx 10xxxxxx	2 octets codant 8 à 11 bits
1110xxxx 10xxxxxx 10xxxxxx	3 octets codant 12 à 16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4 octets codant 17 à 21 bits

Principe de l'UTF-8

Techniquement, il s'agit de coder les caractères Unicode sous forme de séquences de un à quatre octets d'un octet chacun. La norme Unicode définit entre autres un ensemble (ou répertoire) de caractères. Chaque caractère est repéré dans cet ensemble par un index entier aussi appelé « point de code ». Par exemple le caractère « € » (euro) est le 8365e caractère du répertoire Unicode, son index, ou point de code, est donc 8364 (0x20AC) (on commence à compter à partir de 0).

Le codage binaire du symbole € est donc **0010000010101100** (16 bits). On utilise donc la trame 1110xxxx 10xxxxxx 10xxxxxx

Ainsi, le symbole € est codé en UTF-8 par **111000101000001010101100**

Caractère	Unicode (hexadécimal)	Valeur scalaire		Codage UTF-8	
		décimal	binnaire	binnaire	hexadécimal
[NUL]	U+0000	0	0000000	00000000	00
[US]	U+001F	31	0011111	00011111	1F
[SP]	U+0020	32	0100000	00100000	20
A	U+0041	65	1000001	01000001	41
é	U+00E9	233	00011 101001	11000011 10101001	C3 A9
€	U+20AC	8 364	0010 000010 101100	11100010 10000010 10101100	E2 82 AC

Exemples de codage UTF-8

Remarque

En html, on utilise le codage utf-8 pour permettre au navigateur de lire correctement le document. On insère dans le "head" :

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

⚠ Bien distinguer unicode et encoding :

- unicode : chaque caractère est codé par un codepoint représenté par un nombre hexadécimal, par exemple U+00E8
- encoding : convertit l'unicode logique en stockage physique sur des octets, avec différentes solutions. Un de ces codages est UTF-8 (le plus standard et utilisé)

Exercice 2.8

Un caractère a pour code utf-8 hexadécimale EFBFBD.

1. Convertir EFBFBD en binaire
2. Déterminer le code binaire unicode associé à ce caractère
3. Convertir ce code binaire unicode en base 10
4. Afficher ce caractère avec python

2.4.6 Python et l'encodage utf-8

► En python3, les strings (type str) sont en unicode.

La méthode **encode()** permet d'encoder un caractère en utf-8 :

```
>>> mot_code = 'hello'.encode()
>>> mot_code
>>> b'hello'
>>> mot_code[0]
104
```

...et 104 correspond bien au caractère h en unicode !

A l'inverse, il existe la méthode **decode()**.

```
>>> mot_code = 'é'.encode()
>>> mot_code
b'\xc3\xa9'
```

Ici, on n'est plus avec un caractère ASCII donc il faut retrouver le caractère qui a pour encodage utf-8 c3a9 en binaire.

⚠ En Python 3, le type bytes est un tableau d'entiers

Exercice 2.9

On considère :

⚠ Le début de l'exercice se fait sans l'utilisation de la console Python

```
>>> mot = b'\xf0\x9f\x98\xb1'
```

- Déterminer son encodage utf8 binaire
- En déduire sa valeur unicode binaire
- En déduire sa valeur unicode décimale
- Faire une recherche sur le Web pour trouver ce caractère
- Vérifier avec python