Représentation des textes

Sophie Chane-Lune

NSI - Première

Sommaire

Programme officiel

Contenus	Capacités attendues	Commentaires
Représentation d'un texte en machine. Exemples des encodages ASCII, ISO-8859-1, Unicode	Identifier l'intérêt des différents systèmes d'encodage. Convertir un fichier texte dans différents formats d'encodage.	Aucune connaissance précise des normes d'encodage n'est exigible.

1 Problématique

La représentation des caractères dans un ordinateur est l'élément clé pour stocker ou échanger des textes. En théorie, c'est très simple : il suffit d'associer un numéro unique à chaque caractère. En pratique, le choix de l'encodage doit respecter certaines contraintes. Tout d'abord, il faut que tous les ordinateurs utilisent le même encodage. Ensuite, il doit permettre de représenter le plus de caractères possible, en particulier des caractères dit *non imprimables* qui correspondent soit à des actions comme *passer à la ligne* mais également à des commandes de protocoles de communication comme *accuser réception, début de texte,* …. Enfin, il doit être le plus compacte possible afin d'économiser la mémoire ou le volume des échanges réseaux.

2 Codage ASCII

Dans les années 50, il existait un nombre important d'encodages de caractères dans les ordinateurs, les imprimantes ou les lecteurs de cartes. Tous ces encodages étaient incompatibles les uns avec les autres, ce qui rendait les échanges particulièrement difficiles car il fallait utiliser des programmes pour convertir les caractères d'un encodage dans un autre.

Pour tenter de mettre un peu d'ordre dans tout ça, l'ANSI (*American National Standards Insitute*) propose au début des années 60 une norme de codage des caractères appelée **ASCII** pour (*American Standard Code For*

Information Interchange). Cette norme définit un jeu de 128 caractères, chaque caractère étant représenté par un octet.

La correspondance entre les caractères et les octets qui les représentent est résumée dans une table, appelée **table ASCII**, donnée dans la figure ci-dessous. Chaque case de cette table contient un caractère. Pour trouver l'octet (représenté par un nombre hexadécimal à deux chiffres) associé à un caractère, il suffit de concaténer le chiffre de sa ligne avec celui de sa colonne. Par exemple, le caractère **A** correspond au nombre 41 et le caractère + au nombre $2B$.

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

La table ASCII contient plusieurs catégories de caractères :

- les lettres de l'alphabet latin en majuscule (entre 41 et $5A$) et en minuscule (entre 61 et $7A$)
- les chiffres de 0 à 9 (entre 30 et 39)
- des signes de ponctuations (comme la virgule, qui vaut $2C$), des parenthèses ou des crochets (comme le symbole (qui vaut 28 ou le crochet ouvrant qui vaut $2B$)

La table contient également des caractères spéciaux (entre 00 et 20). On trouve par exemple des caractères *blancs* (espaces, tabulation, …), des retours chariot, des suppressions, …. Le tableau ci-dessous résume quelques-uns de ces caractères.

caractère	numéro	signification
HT	09	Tabulation horizontale
LF	$0A$	Nouvelle ligne
VT	$0B$	Tabulation verticale
FF	$0C$	Nouvelle ligne
CR	$0D$	Retour chariot
SP	20	Espace
BS	08	Suppression
DEL	$7F$	Effacement

Un texte codé en ASCII est simplement une suite d'octets correspondant à cette séquence de caractères. Par exemple, la phrase (ou suite de caractères) suivante
Ceci est un texte !
correspond à la séquence d'octets ci-dessous :

C	e	c	i		e	s	t		u	n		t	e	x	t	e		!
43	65	63	69	20	65	73	74	20	75	$6E$	20	74	65	78	74	65	20	21

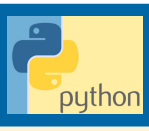
Exercice 1

Donner le codage ASCII des deux chaînes de caractères ci-dessous :

- `'Bonjour tout le monde !'`
- `'''programmer en Python'''`

Exercice 2

Décoder le texte correspondant à la liste des codes ASCII suivants, donnés en héxadécimal :
`47 65 6F 72 67 65 20 42 6F 6F 6C 65`



La fonction `ord` de Python renvoie le code ASCII correspondant à un caractère. L'entier renvoyé est en base 10 (que l'on peut convertir en hexadécimal avec la fonction `hex`)

```
>>> ord('a')
97
>>> hex(ord('a'))
'0x61'
```

Inversement, la fonction `chr` renvoie le caractère correspondant à un entier :

```
>>> chr(0x26)
'&'
```

Les caractères peuvent également être saisis directement par leur code ASCII en utilisant la notation `\xhh` où `hh` est le code hexadécimal du caractère.

```
(print('\x43e\x63i es\x74...')
Ceci est ...
```

Cette technique de saisie des caractères à l'aide du caractère {} est appelée *caractère échapée*

Exercice 3

Écrire une fonction `printASCII(s)` qui affiche à l'écran les codes ASCII au format hexadécimal d'une chaîne de caractères. Utiliser cette fonction pour vérifier les réponses de l'exercice 1.

3 Normes ISO 8859

Les caractères imprimables de la table ASCII se sont vite avérés insuffisants pour transmettre des textes dans des langues autres que l'anglais. En effet, rien qu'en considérant les langues reposant sur un alphabet latin, il manque dans la table ASCII de nombreux caractères comme les lettres accentuées, les symboles de monnaies, ….

Pour remédier à ce problème, l'ISO (Organisation Internationale de Normalisation) a proposé la norme ISO 8859, une extension de l'ASCII qui utilise les huit bits de chaque octet pour représenter les caractères. Au total, ce sont donc 256 caractères doublé, cela reste insuffisant pour représenter tous les caractères utilisés rien que dans les langues latines.

Pour représenter le plus de caractères possible, la norme ISO 8859 définit plusieurs tables de correspondances (on parle aussi de pages) notées ISO-8859-n, où n est le numéro de la table. Bien qu'indépendantes les unes des autres, ces tables ont été conçues pour qu'elles soient compatibles entre elles. Les premiers 128 caractères sont ceux de la norme ASCII. Les 128 suivants sont ceux spécifiques à la table n. De plus, les caractères identiques ont le même

code.
La norme 8859 compte seize table en tout et dix tables rien que pour les langues latines. Plutôt que de référencer ces dernières par leur nomenclature ISO, on les nomme parfois $latin-1$, $latin-2$, etc. Le tableau ci-dessous résume les spécificités de ces tables.

Code ISO	Zone
$8859-1$ ($latin-1$)	Europe occidentale
$8859-2$ ($latin-2$)	Europe centrale ou de l'est
$8859-3$ ($latin-3$)	Europe du sud
$8859-4$ ($latin-4$)	Europe du nord
$8859-5$	Cyrilique
$8859-6$	Arabe
$8859-7$	Grec
$8859-8$	Hébreu
$8859-9$ ($latin-5$)	Turc, Kurde
$8859-10$ ($latin-6$)	Nordique (réarrangement du $latin-4$)
$8859-11$	Thaï
$8859-12$	Devanagari (projet abandonné)
$8859-13$ ($latin-7$)	Balte
$8859-14$ ($latin-8$)	Celtique
$8859-15$ ($latin-9$)	Révision du $latin-1$ (avec €)
$8859-16$ ($latin-10$)	Europe du sud-est

4 Codage Unicode

Bien que les pages ISO-8859-n permettent l'encodage d'un très grand nombre de caractères, elles ne conviennent pas par exemple quand on souhaite écrire un texte avec un mélange de caractères dans différentes pages.

Pour remplacer l'utilisation des pages de code, l'ISO a défini un jeu universel de caractères (appelé UCS en anglais pour Universel Character Set) sous la norme ISO-10646. Il y a aujourd'hui plus de 110000 caractères recensés dans cette norme, qui est conçue pour contenir in fine les caractères de n'importe quelle langue. La capacité maximale de la norme a été fixée à 4294967295 caractères, c'est-à-dire le plus grand entier non signé représentable avec un mot de 32 bits. Par soucis de compatibilité, les 256 premiers entrées sont ceux de la norme ISO-8859-1 (latin-1).

4.1 Une table universelle

Afin de régler définitivement le problème d'encodage des caractères, une norme est apparue au début des années 1990 : **Unicode**. Cette table de caractères contient actuellement plus de 135000 symboles, l'objectif étant qu'elle couvre tous les besoins imaginables.

Unicode ne peut toutefois pas être utilisé directement. Chaque caractère étant actuellement codé sur 21 bits, utiliser le code Unicode à la place d'une table de la norme ISO 8859 multiplierait par 3 la taille des fichiers pour un même contenu.

4.2 Le codage UTF-8

Il existe plusieurs encodages parcimonieux dont UTF-8. Dans ce cas, les 128 caractères du codes ASCII sont codés sur 1 seul octet, et une séquence particulière permet d'accéder aux autres caractères de la table, qui sont codés sur 2 à 4 octets.

Unicode et l'encodage UTF-8 sont la solution actuelle aux problèmes d'encodage de texte. Depuis la version 3, Python représente ses chaînes de caractères en Unicode.

Exercice 4

En UTF-8, le codage des caractères coïncide avec l'ASCII pour les 128 premiers caractères (voir la table ASCII plus haut). Les autres caractères sont représentés par plusieurs octets.
La série d'octets suivants, donnés en hexadécimal, a été relevé dans un fichier codé en UTF-8 :
`43 6F 64 C3 A9 20 65 6E 20 55 54 46 2D 38`
Il contient uniquement des caractères de la table ASCII à l'exception d'un "é".

1. Quelle est la séquence d'octets qui représente le "é", et qu'est-ce qui est inscrit dans le fichier ?
2. Si le fichier avait été interprété en latin 1 (table ci-dessous), qu'est-ce qui se serait affiché ?

ISO-8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ