Chapitre 5

Représentation de caractères

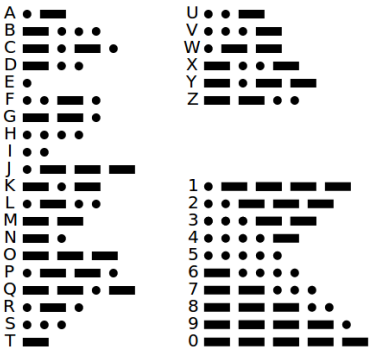
5.1 Une histoire d'encodage

Pour représenter un texte en langage machine - c'est à dire à l'aide de 0 et de 1 - on utilise un **encodage**.
Encoder une donnée, c'est la représenter sous forme d'un code simple, si possible composé de 0 et de 1.

Inventé en 1832 pour la télégraphie, le **code Morse** est un exemple d'encodage dans lequel chaque caractère est représenté par une série d'impulsions courtes et longues, produites par des signes, une lumière, un son ou un geste.

On peut représenter 36 caractères : les lettres de A à Z et les chiffres de 0 à 9.

On espace les différents codes par des « silences » marquant le passage d'un caractère au suivant.



En remplaçant une impulsion courte par un 0 et une longue par un 1, on obtient ainsi un codage binaire des caractères.

Mais le code Morse est incomplet : il manque les minuscules, les signes de ponctuations, les caractères spéciaux comme l'arobase @, « espace » ou « retour à la ligne »...

5.1.1 ASCII

Au début des années 1960 apparaît l'**American Standard Code for Information Interchange** (Code américain normalisé pour l'échange d'information), plus connu sous l'acronyme **ASCII** (prononcé « aski »).
Il définit 128 codes de 7 bits, correspondant à autant de caractères, dont 95 sont imprimables : les chiffres arabes de 0 à 9, les lettres minuscules et majuscules de A à Z, et des symboles mathématiques et de ponctuation.

▷ ***95 caractères imprimables... Et les autres ? À quoi peut bien servir un caractère non imprimable ?***

On appelle ces caractères des **caractères de contrôle**. Un exemple de caractère non imprimable est le **saut de ligne**.
Pour dire à une imprimante "maintenant, saute une ligne", on utilise le caractère adéquat, codé en binaire sur 7 bits par `000 1010`, soit 10 en décimal ou encore `0A` en héxadécimal.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Question 01 Le message héxadécimal suivant est codé avec l'encodage ASCII. Décoder ce message :

44 69 66 66 69 63 69 6C 65 20 3F

▷ ***Pourquoi coder un caractère sur 7 bits et non sur 8 bits, soit 1 octet ?***

Dans les années 1960, les ordinateurs ne travaillaient pas encore avec des octets ! Mais dans les années 1970, les octets apparaissent, et les caractères ASCII se codent alors sur 8 bits, en prenant simplement 0 pour le bit de poids fort.

Question 02 Le message binaire suivant est codé avec l'encodage ASCII (8 bits). Décoder ce message :

01010100 01110010 01101111 01110000 00100000 01101100 01101111 01101110 01100111

L'encodage ASCII suffit pour représenter un texte en anglais, mais il est trop limité pour les autres langues, dont le français et ses lettres accentuées. Les limitations du jeu de caractères ASCII sont encore sensibles au XXIe siècle, par exemple dans le choix restreint de caractères généralement offerts pour composer une adresse email.

Grâce au 8ème bit, on peut doubler le nombre de caractères encodables, comme les accents ou des lettres non latines. Mais ce n'est toujours pas suffisant... On utilise alors d'autres encodages.

5.1.2 ISO-8859-1

La norme **ISO 8859-1**, dont le nom complet est ISO/CEI 8859-1, et qui est souvent appelée **Latin-1** ou **Europe occidentale**, définit ce qu'elle appelle l'alphabet latin numéro 1, qui consiste en 191 caractères de l'alphabet latin, chacun d'entre eux étant codé sur un octet (soit 8 bits). ISO 8859-1 reprend le codage des caractères imprimables d'ASCII.

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

Question 03 Quel caractère correspond au code héxadécimal *B5*?

Question 04 Quel est le code héxadécimal de l'esperluette?

Question 05 43 6F 6D 62 69 65 6E 20 66 6F 6E 74 20 31 20 2B 20 31 20 3F

▷ ***Je ne vois pas de Ÿ... Est-ce normal?***

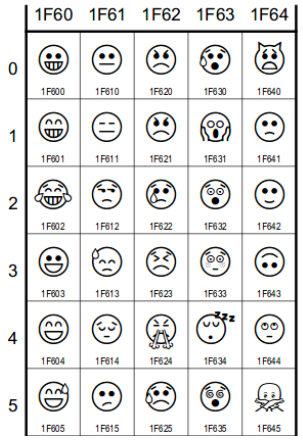
Bien vu! Le ÿ est bien encodé, mais pas la majuscule correspondante. Étant donnée sa rareté dans les langues latines, il a été en quelque sorte « sacrifié » au profit d'autres caractères. Pourtant, si on peut écrire Ÿ dans ce document, c'est qu'il peut se coder à l'aide d'un autre encodage.

5.1.3 Unicode et UTF-8

Le **Standard Unicode** est produit par une organisation à but non lucratif (le Consortium Unicode) ayant pour objectif d'attribuer un numéro à tout caractère utilisé dans une langue humaine. Cette entreprise vise à une certaine universalité. Les alphabets de beaucoup de langues sont déjà inscrits dans ce standard. On y trouve évidemment les alphabets latins, grecs et cyrilliques, mais aussi des dizaines de miliers de caractères chinois, japonais ou coréens. Et même encore plus :

- l'alphabet Braille
- des symboles musicaux
- les chiffres mayas
- des opérateurs mathématiques
- les symboles de monnaies
- des émoticônes...

Tout se trouve ici : `http://www.unicode.org/charts/`



Quelques émoticônes Unicode

Un **caractère Unicode** est un caractère défini dans le Standard Unicode. On y fait souvent référence par son numéro écrit en hexadécimal précédé de «U+». Par exemple, la lettre latine « a » correspond à U+0061 (« a » est bien associé au code héxadécimal *61*, comme vu dans le tableau page précédente), et l'émoticône « visage souriant » correspond à U+1F600, soit un code héxa de *1F600*.

Question 06 D'après le tableau ci-dessus, quel est le code héxadécimal de l'émoticône « visage triste » ?

Le procédé d'encodage de caractères Unicode en une suite d'octets s'appelle l'**UTF-8** : UTF pour *Unicode Transformation Format*, et 8 pour 8 bits. Il associe à tout numéro Unicode une suite d'un ou plusieurs octets (**jusqu'à quatre octets pour un seul caractère**). Une des propriétés importantes de cet encodage est que les caractères dont les numéros sont compris entre 32 et 126 possèdent la même représentation en UTF-8 et dans l'encodage ASCII. L'encodage UTF-8 est donc compatible avec ASCII, dans la mesure où il n'y a rien à faire pour convertir un fichier encodé en ASCII vers UTF-8.

Par sa nature, UTF-8 est d'un usage de plus en plus courant dans tout système devant échanger de l'information.
En octobre 2020, il est utilisé sur plus de 95% des sites Web !

☞ *Une dernière remarque...*

Ne pas confondre *encodage* et *police de caractères* : l'encodage définit la manière de représenter un caractère en langage machine, et la police de caractères définit la manière de représenter un caractère à l'écran.

5.2 Les fonctions `ord` et `chr` de Python

Python dispose des fonctions `ord` et `chr` permettant de convertir un caractère en entier et inversement :

- `ord(c)` retourne l'entier associé au caractère Unicode `c`

```
>>> ord('A')
65
>>> ord('ç')
231
```

Ces valeurs sont décimales. Pour obtenir les valeurs héxadécimales correspondantes, on peut utiliser la fonction `hex` de Python comme dans l'exemple :

```
>>> ord('Z')
90
>>> hex(ord('Z'))
'0x5a'
```

Question 07 Écrire une fonction `lettresMinuscules(chaine)` qui retourne `True` si `chaine` n'est composée que de lettres minuscules, sans accents, (de *a* à *z*), et `False` sinon.

```
>>> lettresMinuscules('Cet exemple est Faux !')
False
>>> lettresMinuscules('celuilaestvrai')
True
```

- `chr(n)` retourne le caractère Unicode associé à l'entier `n`

```
>>> chr(65)
'A'
>>> chr(100)
'd'
>>> chr(376)
'Ÿ'
>>> chr(0x29)
')'
```

`n` varie de 0 à 1 114 111 (`10FFFF` en héxadécimal).