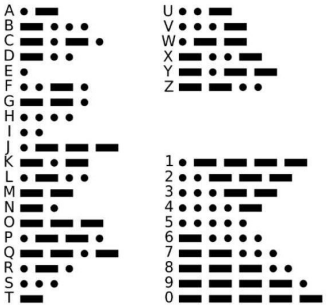
RD05 REPRÉSENTATION DU TEXTE

Quelle que soit la nature de l'information traitée par un ordinateur (image, son, texte, vidéo), elle est toujours représentée sous la forme d'un ensemble de nombres binaires.

I. Le codage Morse



Un des premiers systèmes de codage est le code Morse :

- Il a été développé par Samuel Morse et ses collaborateurs (1837)
- Il a servi pour la transmission de signaux par télégraphe
- Le système de codage est fait avec des séquences signal court/long

Le code Morse est au départ principalement utilisé par les militaires pour transmettre des informations, souvent chiffrées.

Le signal peut être transporté via un signal radio, des signaux électriques à travers un câble télégraphique ou des signaux visuels (lumière).

L'idée est de coder les caractères fréquents avec des séquences simples et les caractères rares par des séquences compliquées.

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]

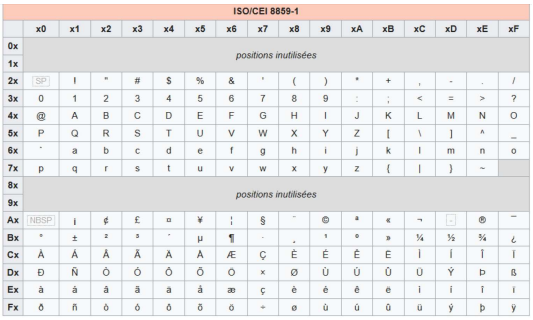
Exercice 1 : Coder en ASCII le mot suivant : Bonjour

II. Codage ISO 8859-1

La norme ASCII convient bien à la langue anglaise, mais pose des problèmes dans d'autres langues, par exemple le français. En effet l'ASCII ne prévoit pas d'encoder les lettres accentuées.

La norme ISO-8859-1 reprend les mêmes principes que l'ASCII, mais les nombres binaires associés à chaque caractère sont codés sur 8 bits, ce qui permet d'encoder jusqu'à 256 caractères.

D'autres normes ont donc dû voir le jour, par exemple la norme "GB2312" pour le chinois simplifié ou encore la norme "JIS X 0208" pour le japonais.



Exercice 2 : Décoder le texte suivant : 4C 61 20 43 69 67 61 6C 65 20 65 74 20 62 62 20 46 6F 75 72 6D 69 2E 20

III. Codage Unicode

Unicode a pour ambition de rassembler tous les caractères existant afin qu'une personne utilisant Unicode puisse, sans changer la configuration de son traitement de texte, à la fois lire des textes en français ou en japonais.

Unicode est uniquement une table qui regroupe tous les caractères existant au monde, il ne s'occupe pas de la façon dont les caractères sont codés dans la machine. Unicode accepte plusieurs systèmes de codage : UTF-8, UTF-16, UTF-32. Le plus utilisé, notamment sur le Web, est UTF-8.

Pour encoder les caractères Unicode, UTF-8 utilise un nombre variable d'octets : les caractères "classiques" (les plus couramment utilisés) sont codés sur un octet, alors que des caractères "moins classiques" sont codés sur un nombre d'octets plus important (jusqu'à 4 octets).

Un des avantages d'UTF-8 est qu'il est totalement compatible avec la norme ASCII : Les caractères Unicode codés avec UTF-8 ont exactement le même code que les mêmes caractères en ASCII.

On retrouve ici le codage 8 bits :

`http://sebastienguillon.com/test/jeux-de-caracteres/windows-ascii-fr.html`

Propriété 1 : Codage UTF-8

- On transforme le code point en binaire
- On code entre 1 et 4 octets selon la table suivante :

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF[12]	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Les "xxx" sont les chiffres de la représentation du code point en base 2.

Exercice 3 : Coder en UTF-8 la lettre é (U+00E9), le symbole ∅, → et ↑

Propriété 2 : Unicode et Python

En python il est possible de comparer des chaines de caractère directement

```
>>> 'abc' < 'abd'
True
>>> 'ab' < 'abc'
True
```

En python il existe des fonctions capables de transformer les caractère en nombre et les nombres en caractère.

```
>>> chr(65)      # valeur décimale -> caractère
'A'
>>> ord('A')     # caractère -> valeur décimale
65
```