1 Intro

Comme pour les nombres (ou plus généralement toute information numérique), les caractères sont représentés en machine par une suite de 0 et de 1.

> L'**encodage** des caractères définit l'association entre un code binaire et un caractère donné.

Les systèmes d'encodage ont subi plusieurs évolutions majeures.
À l'origine du développement des ordinateurs, c'était le « grand bazar »! En changeant de type d'ordinateur ou de système d'exploitation, les textes ne pouvaient pas être lus correctement car les systèmes d'encodage étaient tous différents.

2 ASCII

> En 1960, la norme **ASCII** (American Standard Code for Information Interchange) est créée. Elle définit un encodage des caractères sur 7 bits, permettant ainsi de définir $2^7 = 128$ caractères différents.

Cela était bien suffisant pour les caractères de langue anglo-saxonne et quelques signes de ponctuation (plus quelques encodages particuliers comme le retour à la ligne par exemple).

Voici une retranscription de la table de caractères ASCII (codes exprimés dans différentes bases):

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

3 ASCII étendu : ISO-8859-1 (Latin-1)

Par la suite, pour augmenter le nombre de caractères à représenter, le 8ème bit de l'octet a été ajouté pour pouvoir représenter $2^8 = 256$ caractères (comme les caractères accentués par exemple). Les normes qui se sont développées sur ce modèle ont fait en sorte de conserver une compatibilité avec l'ASCII d'origine (par exemple le A majuscule reste codé 41 en hexadécimal dans ces nouveaux encodages) : cela définit des tables ASCII **étendues**.

> La norme **ISO-8859-1 (Latin-1)** est utilisée pour encoder tous les caractères de langue européenne (de l'Ouest) comme le Français.

Voici une retranscription de la table de caractères ISO-8859-1 (codage hexadécimal):

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

4 Unicode et UTF-8

> Pour unifier complètement l'ensemble des caractères existant dans toutes les langues du monde, le standard Unicode (lié à la norme ISO/CEI 10646) a été développé à partir de 1991 par le Consortium Unicode. Il vise au codage de texte écrit en donnant à tout caractère un nom et un identifiant numérique appelé **point de code**.

Le répertoire Unicode peut contenir plus d'un million de caractères, ce qui est bien trop grand pour être codé par un seul octet. La norme Unicode définit donc des méthodes standardisées pour coder cet index sous forme de séquence d'octets.
L'encodage UTF-8 est l'une d'entre elles et la plus majoritairement utilisée.

> En **UTF-8**, chaque point de code Unicode est encodé sous forme de séquences **de un à quatre octets**.

Les caractères ASCII restent codés sur un seul octet pour garder la comptabilité avec l'ASCII d'origine. Par exemple le A majuscule reste codé par l'unique octet 41 (en hexa).
Les caractères dont le point de code est supérieur à 127 (caractère non ASCII) se codent sur 2 à 4 octets. Le caractère € (euro) se code par exemple sur 3 octets : E2-82-AC.

5 Encodage et Python

Les fonction `ord('X')` et `chr(n)` renvoient respectivement le point de code du caractère `'X'` ou le caractère du point de code `n`.

Exemples :

```
>>> chr(65)
'A'
>>> ord('A')
65
>>> chr(8364)
'€'
>>> ord('€')
8364
```

Python encode les textes en utf8 par défaut.
Il est possible de convertir une chaîne de caractères (type `str`) en chaîne d'octets (type `bytes`). En fonction du type d'encodage, la taille de la chaîne d'octets obtenue sera variable.
Évidemment, il n'est pas possible d'encoder en mode "ascii" une chaîne de caractères contenant des accents ou tout autre caractère non défini dans la table ASCII.

Les fonctions d'encodage ou de décodage sont simplement :
`"texte".encode("nom-encodage")` et `b"chaine_octets".decode("nom-encodage")`.

```
>>>'€'.encode('utf8')
b'\xe2\x82\xac'
>>> 'déjà'.encode('utf8')
b'd\xc3\xa9j\xc3\xa0'
>>> len('déjà'.encode('utf8')),  len('déjà'.encode('utf16'))
(6, 10)
>>> 'déjà'.encode('ascii')
----------------------------------
UnicodeEncodeError: 'ascii' codec can't encode character '\xe9' in position 1:
ordinal not in range(128)
```

```
>>> b'd\xc3\xa9j\xc3\xa0'.decode('utf8')
'déjà'
>>> b'd\xc3\xa9j\xc3\xa0'.decode('utf16')

'썤檩ꃃ'
```

6 Déclarer un encodage en programmation

En HTML5, on commencera toujours par « prévenir » le navigateur de l'encodage utilisé. Il est recommandé d'encoder les pages html en utf8.

```
<meta charset="utf8">
```

En Python, la 1ère ligne du programme permet de déclarer l'encodage sous la forme :

```
# coding: utf-8
```

Les bons éditeurs de texte proposent toujours de choisir l'encodage lors de l'enregistrement d'un document.

7 Chaînes de caractères et Python

En Python, une chaîne de caractères se comporte comme un **tableau** de caractères **non mutable**.

On peut donc accéder à chaque caractère de la chaîne par un indice, et donc aussi faire des parcours de chaîne comme des parcours de tableau, mais il n'est pas possible de modifier un caractère au sein d'une chaîne.

Exemple :

```
>>> chaine = "bonjour à tous"
>>> chaine[0]
'b'
>>> chaine[-1]
's'
>>> len(chaine)
14
>>> for lettre in chaine:
>>>     print(lettre, end=' * ')
b * o * n * j * o * u * r *   * à *   * t * o * u * s *
>>> chaine[0] = 'B'
---------------------------------------------
TypeError: 'str' object does not support item assignment
```

On peut **concaténer** 2 chaînes avec l'opérateur +.
Exemple :

```
>>> ch1 = "bonjour"
>>> ch2 = " à tous"
>>> chaine = ch1 + ch2
>>> chaine
'bonjour à tous'
```

Remarque : beaucoup d'autres méthodes utiles existent pour la manipulation de chaînes.

8 Manipulation de fichier texte en Python

8.1 Lecture

Pour lire le contenu d'un fichier texte en Python, il faut **ouvrir le fichier en mode lecture** ("r" pour "read") selon la syntaxe suivante :

```
>>> fich = open("adresse du fichier", "r")
```

Ensuite, on peut appliquer la **méthode** `readline` à l'objet `fich` créé pour parcourir ligne par ligne le fichier.

```
>>> fich.readline()
'texte de la 1ère ligne'
>>> fich.readline()
'texte de la 2ème ligne'
```

On peut aussi **enregistrer directement toutes les lignes du fichier dans une liste-Python** avec **méthode** `readlines` (notez bien le 's' final).

```
>>> fich = open("adresse du fichier", "r")
>>> liste = fich.readlines()
>>> liste
['texte de la 1ère ligne\n', 'texte de la 2ème ligne\n']
```

Il faut aussi **bien penser à fermer un fichier** après son ouverture :

```
>>> fich.close()
```

Il est courant de devoir traiter l'ensemble des lignes d'un fichier texte, on écrira alors cette suite d'instructions :

```
>>> fich = open("adresse du fichier", "r")
>>> for ligne in fich:
>>>     < traitement de la ligne >
>>> fich.close()
```

Enfin, **il existe une instruction qui évite de devoir fermer de façon explicite le fichier**, et qui s'occupe même de le fermer si une erreur survient pendant le traitement du fichier. La syntaxe devient la suivante :

```
>>> with open("adresse du fichier", "r") as fich:
>>>     < traitement du fichier fich comme on le souhaite >
```

8.2 Écriture

Pour écrire dans un fichier texte, il faut **ouvrir le fichier en mode d'écriture** ("w" pour "write") [ce mode "w" écrase le contenu d'un fichier existant], **ou bien en mode d'ajout** ("a" pour "append") [dans ce mode, le texte est ajouté à la fin d'un fichier existant].
Ensuite, on peut appliquer la **méthode** `write` au fichier pour écrire une ligne dans le fichier, ou la **méthode** `writelines` pour écrire une liste de lignes.

Exemples :

```
>>> with open("adresse du fichier", "w") as fich:
>>>     fich.write('Bonjour,\n')
>>>     fich.write('Ceci est la deuxième ligne du fichier.\n')
>>>     fich.write('et voici la 3ème\n')
>>>     fich.writelines(['4ème\n', '5ème\n', 'et dernière']')
```

Remarque : le caractère de fin de ligne \n est indispensable pour passer à la ligne.