Encodage de Texte

1) Introduction

Pour représenter des caractères dans un ordinateur on va **associer un numéro unique à chaque caractère**.

L'encodage doit respecter les contraintes suivantes :

- pour échanger du texte entre ordinateurs, il faut qu'ils utilisent le **même encodage** ;
- il doit également permettre de représenter **le plus de caractères possibles** ;
- enfin, il doit être le plus **compact** possible pour économiser la mémoire ou le volume des échanges réseaux.

2) Historique

- Dans les **années 50**, il existait beaucoup d'encodages différents, incompatibles les uns avec les autres, ce qui rendait les **échanges difficiles**, car il fallait utiliser des programmes de conversion d'un encodage à l'autre.

- Au début des **années 60**, a été créé le **code ASCII** (American Standard Code for Information Interchange).

Les caractères sont codés sur **1 octet**. Mais seuls 7 bits sont utilisés pour le codage du caractère, le 8ième bit est utilisé pour le **contrôle de parité**, une sécurité pour éviter les erreurs, qui étaient très fréquentes dans les premières mémoires électroniques.

Avec 7 bits il est possible de coder jusqu'à **128 caractères** ce qui est largement suffisant pour un texte écrit en langue anglaise (pas d'accents et autres lettres particulières).

Le code ASCII contient les lettres majuscules et minuscules, les chiffres, plusieurs caractères de ponctuation, et des caractères invisibles comme l'espace, la tabulation ou le retour à la ligne.

La **correspondance entre les caractères et les octets** qui les représentent est résumée dans la table ASCII ci-dessous :

ASCII Code Chart

	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		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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

- Chaque case de cette table contient un caractère. Pour trouver l'octet correspondant on lit le numéro de la ligne, puis de la colonne dans laquelle il se trouve. On obtient un nombre hexadécimal à deux chiffres.

Par exemple : Le caractère **'A'** est codé par **41** en hexadécimal, soit **0100 0001** en binaire sur un octet.

- On peut voir sur les deux premières lignes des **caractères spéciaux**.

Par exemple : **HT** pour la tabulation horizontale, **LF** pour une nouvelle ligne...

Exercice 1 :

1. A l'aide de la table, codez la phrase suivante en ASCII, sous forme hexadécimale.

   Reveillez-vous !

2. Que signifie cette phrase codée en binaire :

   01000010 01110010 01100001 01110110 01101111 00100001.

Remarque : L'**encodage ASCII est insuffisant**, notamment il ne permet pas de coder les lettres accentuées.

- Ainsi, l'**ISO** (Organisation Internationale de Normalisation) a proposée la **norme ISO 8859**, une extension de l'ASCII, en utilisant le 8ième bit pour coder jusqu'à **256 caractères**. Mais cela n'était toujours pas suffisant !

Du coup, pour représenter le plus de caractères possibles, la norme a définie **plusieurs tables de correspondances** notées **ISO 8859-n** où n est le numéro de la table, conçues pour être **compatibles entre elles**.

Cette norme 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.

Les 128 premiers caractères sont ceux de la norme ASCII. Les 128 suivants sont ceux spécifiques à la table n. Il y a 16 tables, que l'on nomme également latin-1, latin-2...

Voici la table ISO 8859-1 ou **latin-1**, utilisée principalement en **Europe Occidentale** :

Code	...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	{	l	}	~	DEL
8...	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9...	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
A...	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B...	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C...	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D...	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E...	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F...	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Remarque :

Cela n'est toujours pas satisfaisant, notamment lorsque l'on veut écrire un texte mélangeant des caractères de différentes pages. On se retrouve parfois avec des caractères qui ne veulent rien dire...

	A	B	C	D
1	EntitÃ©			
2	gÃ©omÃ©trie			
3	Ã©tat			
4	RÃ©pertoriÃ©			
5	erronÃ©			
6	crÃ©Ã©			
7	CoordonnÃ©es			

- Au début des **années 90** une nouvelle norme a vu le jour, **Unicode**, pour remplacer l'utilisation de ces pages et **rassembler tous les caractères existant**.

Cette forme associe à chaque caractère **un nom unique**, ainsi qu'**un numéro**, un entier positif en base 10 **appelé point de code**.

Aujourd'hui, **plus de 110 000 caractères** sont recensés dans cette norme, qui est conçue pour contenir les caractère de **n'importe qu'elle langue**.

Exemple :

Caractère : α (alpha)

Nom (en anglais) : GREEK SMALL LETTER ALPHA

Code Point en décimal : 945

Code Point en hexadécimal : 3B1

Notation standard : U+3B1

Remarque :

Depuis sa version 3, **Python utilise Unicode** pour le codage des caractères.

Une fois le code Unicode déterminé, il convient ensuite de le transformer en code binaire compréhensible par les logiciels.

Unicode est uniquement une table qui regroupe tous les caractères existant au monde (Charset), 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 est **UTF-8**, notamment sur le web.

En effet, maintenant, par défaut, les navigateurs Internet utilisent ce codage et les concepteurs de sites pensent de plus en plus à créer leurs pages web en prenant en compte cette même norme ; c'est pourquoi il y a de moins en moins de problèmes de compatibilité.

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 c'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.

Table d'encodage UTF-8 :

Définition du nombre d'octets utilisés dans le codage (uniquement les séquences valides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0xxxxxxx	00 à 7F	1 octet, codant 7 bits
U+0080 à U+07FF	110xxxxx 10xxxxxx	C2 à DF	2 octets, codant 11 bits
U+0800 à U+0FFF	11100000 101xxxxx 10xxxxxx	E0 (le 2e octet est restreint de A0 à BF)	3 octets, codant 16 bits
U+1000 à U+1FFF	11100001 10xxxxxx 10xxxxxx	E1	
U+2000 à U+3FFF	11100010 10xxxxxx 10xxxxxx	E2 à E3	
U+4000 à U+7FFF	11100111 10xxxxxx 10xxxxxx	E4 à E7	
U+8000 à U+BFFF	11101000 10xxxxxx 10xxxxxx	E8 à EB	
U+C000 à U+CFFF	11101100 10xxxxxx 10xxxxxx	EC	
U+D000 à U+D7FF	11101101 100xxxxx 10xxxxxx	ED (le 2e octet est restreint de 80 à 9F)	
U+E000 à U+FFFF	11101111 10xxxxxx 10xxxxxx	EE à EF	
U+10000 à U+1FFFF	11110000 1001xxxx 10xxxxxx 10xxxxxx	F0 (le 2e octet est restreint de 90 à BF)	4 octets, codant 21 bits
U+20000 à U+3FFFF	11110000 101xxxxx 10xxxxxx 10xxxxxx		
U+40000 à U+7FFFF	11110001 10xxxxxx 10xxxxxx 10xxxxxx	F1	
U+80000 à U+FFFFF	11110011 10xxxxxx 10xxxxxx 10xxxxxx	F2 à F3	
U+100000 à U+10FFFF	11110100 1000xxxx 10xxxxxx 10xxxxxx	F4 (le 2e octet est restreint de 80 à 8F)	

Pour trouver la représentation binaire UTF-8 d'un point de code Unicode :

- tout d'abord, on convertit en binaire le point de code,

- ensuite, on détermine, à partir du tableau ci-dessus, le nombre d'octets nécessaires pour l'encodage UTF-8,

- enfin, on remplace les x par les bits du point de code, complétant à 0 les éventuels bits manquant en début de code.

Par exemple : La lettre « é » correspond au code U+00E9.

En binaire cela correspond au nombre 1110 1001.

8 bits sont nécessaires pour coder ce caractère.

Il faut donc utiliser 2 octets pour l'encodage UTF-8 :

11000011 **10**101001 , soit C3 A9 en hexadécimal.

Exercice 2 :

1. Ouvrez le navigateur firefox et affichez le code source de la page (clique droit).
   Où est indiqué le jeu de caractère (charset) utilisé ?
2. Quel est le point de code Unicode du caractère "b", codé avec UTF-8 ?
3. Quelle est la représentation binaire UTF-8 du caractère « ç » dont le point de code est U+00E7.
4. Vérifiez les résultats précédents en utilisant le site ci-dessous :
   https ://unicode-table.com/fr/

Remarques :

- La fonction **ord(c)** renvoie le nombre entier en base 10 représentant le code Unicode du caractère représenté par la chaîne donnée c.

Par exemple : **ord('a')** donne **97**, le résultat sous forme décimale et **hex(ord('a'))** donne **'0x61'** c'est à dire 61 en hexadécimal (le **0x** indique que le résultat est sous forme hexadécimal).

- La fonction **chr(i)** renvoie la chaîne représentant un caractère dont le code de caractère Unicode est le nombre entier i.

Par exemple : **chr(61)** renvoie '=' (61 est un entier sous forme décimale) et **chr(0x26)** renvoie **'&'** (0x26 est entier sous orme hexadécimale).

- Pour utiliser le code point Unicode, on utilise la syntaxe : u"\$uxxxx$ où les x sont remplacés par des chiffres.
Par exemple : en tapant u"\$u00E9$", on obtient le caractère **"é"**.