NSI : codage des caractères, ASCII et Unicode

1. Un premier codage : le code ASCII

Un ordinateur ne pouvant coder l'information que par des bits ayant pour valeur 0 ou 1, il a été nécessaire de mettre en place un codage approprié des caractères.

Dans les années 1960, apparait le code ASCII (American Standard Code for Information Interchange) qui propose un codage des caractères sur 7 bits soit 128 caractères possibles.
Les caractères dans cette table de code ASCII sont classés par code croissant du haut vers le bas, de la gauche vers la droite.
SI est ainsi un caractère de contrôle qui correspond aux codes ASCII : 0001111 en binaire, 0F en héxadécimal et 15 en décimal.

LSB \ MSB		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

Les 32 premiers codes sont réservés à des caractères de contrôle qui permettent des actions spécifiques.

Par exemple, CR signifie un retour à la ligne.

Exercice 1

a) Donnerl le codage ASCII du caractère de contrôle BEL.

En binaire :

En hexadécimal :

En décimal :

b) Donnerl le codage ASCII du caractère A.

En binaire :

En hexadécimal :

En décimal :

c) Donnerl le codage ASCII du caractère a.

En binaire :

En hexadécimal :

En décimal :

d) On peut passer facilement du codage d'une lettre majuscule à son codage en lettre minuscule.

- en ajoutant à son code décimal la valeur :
- en modifiant un seul bit de son codage ASCII. Lequel ?

2. Un première méthode de contrôle de la bonne transmission de l'information : le bit de parité.

Le codage ASCII n'utilisant que 7 bits et cette information étant transmise au minima sur un octet, on peut utiliser le bit restant pour s'assurer du bon transfert de l'information.

En effet, une information numérique peut toujours rencontrer lors de son transfert des erreurs (bruit de fond , surtension, ...).

Une première méthode de contrôle du bon transfert de l'information est d'utiliser la méthode du bit de parité ou VRC (Vertical Redundancy Checking).

Méthode avec une parité paire :

On veut transmettre le codage ASCII des 3 caractères NSI.

On code sur 7 bits chacun des caractères :

	N	S	I
Codage binaire sur 7 bits	_ _ _ _ _ _ _	_ _ _ _ _ _ _	_ _ _ _ _ _ _
On complète le bit de poids fort de l'octet pour que le nombre de bits à 1 soit pair.	_ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _
On complète le bit de poids faible de l'octet pour que le nombre de bits à 1 soit pair.	_ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _

Intérêt et désavantage de cette méthode de contrôle :

- Elle permet de déterminer un nombre impair d'erreurs sur les bits lors de la transmission de l'information,
- elle ne permet pas de détecter un nombre pair d'erreurs,
- elle ne permet pas de situer l'erreur.

2. Le code ASCII étendu

Le code ASCII a rapidement était étendu sur 8 bits pour donner accès à davantage de caractères.

LSB \ MSB		8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0	0000	Ç	É	á	▒	└	ð	Ó	-
1	0001	ü	æ	í	▒	┴	Ð		±
2	0010	é	Æ	Ó	▓	┬	Ê	ô	‗
3	0011	â	ô	ú	│	├	Ë	ò	¾
4	0100	ä	ö	ñ	┤	─	È	ő	¶
5	0101	à	ò	Ñ	Á	┼	ı	Ő	§
6	0110	å	û	ª	Â	ã	í	µ	÷
7	0111	ç	ù	º	À	Ã	î	þ	¸
8	1000	ê	ÿ	¿	©	╚	ï	Þ	°
9	1001	ë	Ö	®	╣	╔	┘	Ú	¨
A	1010	è	Ü	¬	║	╩	┌	Û	·
B	1011	ï	ø	½	╗	╦	█	Ù	¹
C	1100	î	£	¼	╝	╠	▄	ý	³
D	1101	ì	Ø	¡	¢	═	¦	Ý	²
E	1110	Ä	×	«	¥	╬	ì	¯	■
F	1111	Å	ƒ	»	┐	¤	▀	´	

Malheureusement, la multiplicité des caractères selon les langues et les besoins va multiplier les normes et les extensions du codage initial (certains "codes ASCII" étendus ne reprenant même pas les codes de la table ASCII initiale).

> Alors que les 96 caractères ASCII imprimables suffisent à l'échange d'informations en anglais courant, la plupart des autres langues qui utilisent l'alphabet latin ont besoin de symboles additionnels non couverts par l'ASCII, tels que ß (allemand), å (suédois et d'autres langues scandinaves). ISO 8859 a cherché à remédier à ce problème en utilisant le huitième bit de l'octet, pour donner de la place à 128 caractères supplémentaires (ce bit était jadis utilisé pour le contrôle de l'intégrité des données (bit de parité), ou était inutilisé). Cependant, il fallait plus de caractères qu'on n'en pouvait mettre dans un jeu de caractères 8 bits, aussi plusieurs tables de correspondances ont été développées, en incluant au moins 10 tables pour couvrir uniquement l'écriture latine.
>
> https://fr.wikipedia.org/wiki/ISO/CEI_8859

3. Le standart Unicode et l'UTF-8

> Unicode, dont la première publication remonte à octobre 1991, a été développé dans le but de remplacer l'utilisation de pages de code nationales.
>
> Ces pages de code avaient dans le passé quelques problèmes. Par exemple, sur les terminaux 3270 fonctionnant en EBCDIC : lorsqu'une note de service électronique comportait un caractère « signe monétaire », le même texte plafonnant une dépense en dollars pour le lecteur américain faisait afficher sur un écran britannique le même montant en livres sterling, puisque le signe monétaire était différent dans chacun des deux pays…
>
> Cependant, les écritures les plus utilisées dans le monde sont représentées, ainsi que des règles sur la sémantique des caractères, leurs compositions et la manière de combiner ces différents systèmes. — Par exemple, comment insérer un système d'écriture de droite à gauche dans un système d'écriture de gauche à droite
>
> https://fr.wikipedia.org/wiki/Unicode

Selon le standart Unicode, à chaque caractère est associé un nom décrivant ce caractère.

Par exemple, en python :

```
>>> "\N{GREEK CAPITAL LETTER DELTA}"
'Δ'
```

A ce nom est associé un point de code en héxadécimal qui permet de situer le caractère dans la table de caractères qui lui est associée et son numéro de position dans cette table.

```
>>>"\u0394"  # avec un point de code sur 16 bits
'Δ'
>>> "\U00000394" # avec un point de code sur 32 bits
'Δ'
```

> Le point de code est noté U+xxxx où xxxx est en hexadécimal, et comporte 4 à 6 chiffres :
>
> - 4 chiffres pour le premier plan, appelé *plan multilingue de base* (donc entre U+0000 et U+FFFF) ;
> - 5 chiffres pour les 15 plans suivants (entre U+10000 et U+FFFFF) ;
> - 6 chiffres pour le dernier plan (entre U+100000 et U+10FFFF).

On peut enfin repérer un caractère par son numéro d'ordre dans la table des caractères Unicode :

```
>>> ord("A") # donne le numéro d'ordre du caractère dans la table unicode
65
>>> ord('Δ')
916
>>> chr(65) # donne le caractère correspondant au caractère de numéro 65
'A'
>>> chr(916)
'Δ'
>>> pi="\u03c0" # codage d'un caractère par son point de code
```

```
>>> print(pi)
π
>>> ord(pi)  # fonction renvoyant le numéro d'ordre d'un caractère
960
>>> chr(960) # fonction affichant le caractère correspondant à un numéro d'ordre
'π'
```

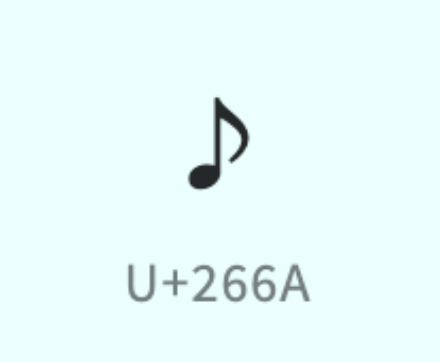
Certains points de code ou numéros sont réservés à des caractères spéciaux ou tout simplement réservés :

```
>>> chr(128) # le \x désigne un caractère spécial
'\x80'
>>> chr(0xda01) # le \ désigne un caractère réservé
'\uda01'
```

Actuellement, Unicode comporte plus de 130 000 caractères différents :
https://home.unicode.org/

Exercice

On veut écrire, en python, le caractère correspondant au glyphe suivant :



1. Quel est son point de code ?
2. Comment obtenir ce caractère en python ? (compléter la ligne de commande)

```
>>>
'♪'
```

3. Comment récupérer son numéro d'ordre et l'afficher avec la fonction chr ? (Comment compléter les 2 lignes de commandes)

```
>>>
9834
>>>
'♪'
```