A. Notion d'encodage / Code ASCII :

Ce qu'il faut savoir :

● Coder un caractère sur un ordinateur relève d'un principe simple : il faut le faire correspondre à un nombre entier codé en binaire. Cette action action se nomme ***l'encodage***, et elle doit respecter 2 contraintes :

1. ***Pouvoir coder suffisamment de caractères*** (ceux propres à chaque langue ansi que les caractères non imprimables tels que le saut de ligne, la tabulation …).
2. ***Être le plus économe possible*** en terme d'espace mémoire.

● Dans les années 50, il existait de nombreux encodages différents qui rendaient très compliquée la communication des différentes machines les unes avec les autres (ordinateurs, imprimantes etc...).
Ainsi, au début des années 60, l'***ANSI*** (***A***merican ***N***ational ***S***tandards ***I***nstitute), dans un souci d'uniformisation, met en place ***l'encodage ASCII*** (***A***merican ***S***tandard ***C***ode for ***I***nformation ***I***nterchange).

La correspondance entre 128 caractères et les 128 premiers entiers est donnée dans la ***table ASCII*** suivante :

0	*NUL*	16	*DLE*	32	SPC	48	0	64	@	80	P	96	`	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	'	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	DEL

Remarques :
Les 128 caractères ASCII sont codés ***en binaire sur 1 octet dont le bit de poids fort est 0***.
Mais un souci d'économie d'écriture et de confort de lecture, on peut utiliser ***la conversion hexadécimale*** plutôt que binaire d'un entier.

Exemple :
La lettre K correspond à l'entier 75 qui si code 01001011 en binaire, mais seulement 4B en hexadécimal.
(Revoir si nécessaire la partie D, exercice 4, TP : « Ecrire d'un entier naturel en base $b \geqslant 2$)

Exercice 1 : Voici ci-dessous un texte codé au format ASCII en binaire.

01000011001001110110010101110011011101000010000001100110011000010110001101101001011011000110010 1.

1. Décode ce texte à l'aide de la table ASCII.

2. Convertis ce code en hexadécimal.

Exercice 2 :

Inversement, traduis le texte : ***« j'aime l'informatique »*** au format ASCII en binaire puis en hexadécimal.

Exercice 3 :

1. Sur la console Edupython, saisis : « >>>bin(65), puis >>>bin(75) etc...
Explique l'utilité de la méthode ***bin*** sur Python.

2. Sur la console Edupython, saisis : « >>>hex(65), puis >>>hex(75) etc...
Explique l'utilité de la méthode ***hex*** sur Python.

3. Sur la console Edupython, saisis : « >>>int(''01001011'',2), puis >>>int(''4B'',16) etc...
Explique l'utilité de la méthode ***int*** sur Python.

4. Sur la console Edupython, saisis : « >>>ord('A'), puis >>>ord('K') etc... ».
Explique l'utilité de la méthode ***ord*** sur Python.

5. Sur la console Edupython, saisis : « >>>chr(65), puis >>>chr(0b01001011) etc... ».
Explique l'utilité de la méthode ***chr*** sur Python.

Ce qu'il faut savoir : Les méthodes bin / hex / int / ord / chr sur Python.

	utilité	exemples
bin		
hex		
int		
ord		
chr		

Exercice 4 :

1. Ecris une fonction ***printASCII(s)*** en langage Python dont le paramètre est une chaîne de caractères ***s*** et qui renvoit le code ASCII en binaire de cette chaîne de caractères.
2. Programme cette fonction sur Edupython et teste la dans la console.
3. Crée sur Edupython un programme qui demande à l'utilisateur de saisir une phrase ***en Anglais*** (***pas d'accents***) et d'afficher le code ASCII en binaire correspondant.
4. Sauvegarde en nommant « ASCII.NOM » le fichier Python dans lequel il y a ton travail et envoie le par mail au professeur : jerome.gauthier@lyceemermozdakar.org .

C. Normes ISO 8859 et ISO 10646 :

Ce qu'il faut savoir :

● Les 128 caractères imprimables de la table ASCII sont évidemment insuffisants pour des langues autres que l'Anglais (lettres accentuées, symboles de monnaies, caractères arabes, mandarins etc...).
Ainsi l'***ISO*** (***I***nternational ***O***rganization for ***S***tandardization) a créé ***la norme ISO 8859*** constitué de 16 tables de correspondance entre 256 caractères et les 256 premiers nombres entiers.
Dans chacune de ces tables, les entiers de 0 à 127 codent les caractères de la table ASCII (un nombre binaire d'***un octet dont le bit de poids fort est 0***), et les entiers de 128 à 255 codent les caractères spécifiques à chaque table (un nombre binaire d'***un octet dont le bit de poids fort est 1***).

Exemples : ISO 8859-1 ou ***Latin-1*** pour la zone Europe Occidentale.
ISO 8859-6 pour la zone Arabe.
ISO 8859-16 ou ***Latin-10*** pour la zone Europe du sud-est

● Les encodages de la norme ISO 8859 permettent de coder un très grand nombre de caractères, mais ils peuvent être insuffisants si on écrit un texte avec des caractères de différentes zones géographiques par exemple. Ainsi l'ISO a créé ***la norme ISO 10646*** qui rassemble un jeu de plus de 110 000 caractères de toutes les langues. Cette norme définit plusieurs encodages nommés ***UTF*** (***U***niversal ***T***ransformation ***F***ormat) où chaque caractère est codé sur 32 bits (4 octets) maximum.
L'encodage de cette norme le plus utilisé est ***UTF-8*** où l'on code chaque caractère sur 8 bits minimum et jusqu'à 32 bits. ***Les 256 premiers caractères de cet encodage sont ceux de l'encodage Latin-1***.
Chacun des caractères est associé à un entier appelé ***point de code***.

- Si le bit de poids fort d'un octet est 0, il s'agit d'un caractère codé sur un seul octet et seuls les 7 bits suivants sont des bits codants. Cela correspond aux caractère ASCII.
- Sinon, dans le premier octet d'un caractère, les premiers bits de poids fort sont des 1 qui déterminent le nombre d'octets utilisés pour coder le caractère. Cette suite de un à quatre bits 1 est toujours suivie d'un bit 0. Les bits suivants sont des bits codants. Les octets suivants commencent par les bits 1 0 et seuls les 6 bits suivants sont des bits codants.

Table des codes possibles en UTF-8 :

Point de code (décimal)	Point de code (hexadécimal)	Code UTF8 (binaire) (x=bit codant)
0 à 127	0000 à 007F	0xxxxxxx
128 à 2 047	0080 à 07FF	110xxxxx 10xxxxxx
2 048 à 65 535	0800 à FFFF	1110xxxx 10xxxxxx 10xxxxxx
65 536 à …	10000 à 10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Exercice 5 :

1. Pourquoi ne peux-tu pas coder le mot ***« Numérique »*** au format ASCII ?

2.Code le mot ***« Numérique »*** au format Latin-1 sachant que le caractère « é » correspond à l'entier 233.

Exercice 6 :

1. Sachant que le point de code en décimal du caractère « é » est 233, donne le codage de ce caractère au format UTF-8.

2. Donne le codage au format UTF-8 du mot « ***Numérique*** ».

Exercice 7 : Pour chacun des caractères ci-dessous, coder leur code au format UTF-8 en écrivant les octets en binaire, puis en hexadécimal.

1. A (point de code = 65). ***2.*** è (point de code = 232). ***3.*** * (point de code = 8902).

Exercice 8 : Pour chacune des séquences d'octets suivantes, qui sont écrites en binaire, dire si elle représente une séquence UTF-8 valide et si oui la décoder.

On pourra utiliser le lien : https://www.w3schools.com/charsets/ref_utf_dingbats.asp .

1. 01111110 01000000 01100100.

2. 11000110 10000001 10000001.

3. 11100010 10011100 10001000.

Exercice facultatif :

Ecris une fonction ***decalage(s)*** en langage Python dont le paramètre est une chaîne de caractères ***s*** constituée des 128 caractères (en minuscules uniquement) du code ASCII et qui renvoit une chaîne de caractères ou chaque lettre a été décalée de 3 rangs. Par exemple « a » devient « d », « f » devient « i » ou « y » devient « b ».

Attention : seules les lettres doivent être décalées. Pas les espaces ou la ponctuation.

Programme cette fonction sur Edupython et teste la dans la console.

Sauvegarde en nommant « facultatif.NOM » le fichier Python dans lequel il y a ton travail et envoie lepar mail au professeur : jerome.gauthier@lyceemermozdakar.org .

A. Notion d'encodage / Code ASCII :

Ce qu'il faut savoir :

● Coder un caractère sur un ordinateur relève d'un principe simple : il faut le faire correspondre à un nombre entier codé en binaire. Cette action action se nomme ***l'encodage***, et elle doit respecter 2 contraintes :

1. ***Pouvoir coder suffisamment de caractères*** (ceux propres à chaque langue ansi que les caractères non imprimables tels que le saut de ligne, la tabulation …).
2. ***Être le plus économe possible*** en terme d'espace mémoire.

● Dans les années 50, il existait de nombreux encodages différents qui rendaient très compliquée la communication des différentes machines les unes avec les autres (ordinateurs, imprimantes etc...).
Ainsi, au début des années 60, l'***ANSI*** (***A***merican ***N***ational ***S***tandards ***I***nstitute), dans un souci d'uniformisation, met en place ***l'encodage ASCII*** (***A***merican ***S***tandard ***C***ode for ***I***nformation ***I***nterchange).

La correspondance entre 128 caractères et les 128 premiers entiers est donnée dans la ***table ASCII*** suivante :

0	*NUL*	16	*DLE*	32	SPC	48	0	64	@	80	P	96	`	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	'	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	DEL

Remarques :
Les 128 caractères ASCII sont codés ***en binaire sur 1 octet dont le bit de poids fort est 0***.
Mais un souci d'économie d'écriture et de confort de lecture, on peut utiliser ***la conversion hexadécimale*** plutôt que binaire d'un entier.

Exemple :
La lettre K correspond à l'entier 75 qui si code 01001011 en binaire, mais seulement 4B en hexadécimal.
(Revoir si nécessaire la partie D, exercice 4, TP : « Ecrire d'un entier naturel en base $b \geqslant 2$)

Exercice 1 : Voici ci-dessous un texte codé au format ASCII en binaire.

01000011001001110110010101110011011101000010000001100110011000010110001101101001101100101.

1. Décode ce texte à l'aide de la table ASCII.

01000011	00100111	01100101	01110011	01110100	00100000	01100110	01100001	01100011	01101001	01101100	01100101
67	39	101	115	116	32	102	97	99	105	108	101
C	*'*	*e*	*s*	*t*		*f*	*a*	*c*	*i*	*l*	*e*

2. Convertis ce code en hexadécimal.

01000011	00100111	01100101	01110011	01110100	00100000	01100110	01100001	01100011	01101001	01101100	01100101
43	27	65	73	74	20	66	61	63	69	6C	65

Exercice 2 :

Inversement, traduis le texte : ***« j'aime l'informatique »*** au format ASCII en binaire puis en hexadécimal.

j	'	a	i	m	e		l	'
106	39	97	105	109	101	32	108	39
01101010	00100111	01100001	01101001	01101101	01100101	00100000	01101100	00100111
6A	27	61	69	6D	65	20	6C	27

i	n	f	o	r	m	a	t	i	q	u	e
105	110	102	111	114	109	97	116	105	113	117	101
01101001	01101110	01100011	01101111	01110010	01101101	01100001	01110100	01101001	01110001	01110101	01100101
69	6E	66	6F	72	6D	61	74	69	71	75	65

Exercice 3 :

1. Sur la console Edupython, saisis : « >>>bin(65), puis >>>bin(75) etc...
Explique l'utilité de la méthode ***bin*** sur Python.

bin convertit un entier en binaire (le code binaire débute par « 0b »).

2. Sur la console Edupython, saisis : « >>>hex(65), puis >>>hex(75) etc...
Explique l'utilité de la méthode ***hex*** sur Python.

hex convertit un entier en hexadécimal (le code binaire débute par « 0x »).

3. Sur la console Edupython, saisis : « >>>int(''01001011'',2), puis >>>int(''4B'',16) etc...
Explique l'utilité de la méthode ***int*** sur Python.

int calcule l'entier corresponpant à un code dans une base quelconque (2, 16, etc...).

4. Sur la console Edupython, saisis : « >>>ord('A'), puis >>>ord('K') etc... ».
Explique l'utilité de la méthode ***ord*** sur Python.

ord donne l'entier correspondant à un caractère.

5. Sur la console Edupython, saisis : « >>>chr(65), puis >>>chr(0b01001011) etc... ».
Explique l'utilité de la méthode ***chr*** sur Python.

ch donne le caractère correspondant à un entier écrit en décimal, en binaire ou en hexadécimal.

Ce qu'il faut savoir : **Les méthodes bin / hex / int / ord / chr sur Python.**

	utilité	exemples
bin	bin convertit un entier en binaire	bin(65)=0b1000001
hex	hex convertit un entier en hexadécimal	hex(75)=0x4B
int	int calcule l'entier correspondant à un code dans une base quelconque (2, 16, etc...)	int('4B',16)=75
ord	ord donne l'entier correspondant à un caractère.	ord('A')=65
chr	chr donne le caractère correspondant à un entier écrit en décimal, en binaire ou en hexadécimal.	chr(75)='K'

Exercice 4 :

1. Ecris une fonction ***printASCII(s)*** en langage Python dont le paramètre est une chaîne de caractères ***s*** et qui renvoit le code ASCII en binaire de cette chaîne de caractères.

```
def ASCII(s):
    code=''
    for i in s:
        code=code+bin(ord(i))
    return (code)
```

2. Programme cette fonction sur Edupython et teste la dans la console.

```
>>> ASCII("Je m'appelle Jerome")
'0b1001010000b1100101000b1000000b1101101000b1001110b1100001000b1110000000b1110000000b1100101000b1101100000b1101100000b1100101000b1000000000b1001010000b1100101000b1110010000b1101111000b1110100000b1100101000'
```

3. Crée sur Edupython un programme qui demande à l'utilisateur de saisir une phrase ***en Anglais*** (***pas d'accents***) et d'afficher le code ASCII en binaire correspondant.

```
def ASCII(s):
    code=''
    for i in s:
        code=code+bin(ord(i))
    return (code)


phrase=input("Saisis une phrase en Anglais")
print("Le code ASCII de",phrase,"est :",ASCII(phrase))
```

4. Sauvegarde en nommant « ASCII.NOM » le fichier Python dans lequel il y a ton travail et envoie le par mail au professeur : jerome.gauthier@lyceemermozdakar.org .

C. ***Normes ISO 8859 et ISO 10646 :***

Ce qu'il faut savoir :

● Les 128 caractères imprimables de la table ASCII sont évidemment insuffisants pour des langues autres que l'Anglais (lettres accentuées, symboles de monnaies, caractères arabes, mandarins etc...).
Ainsi l'***ISO*** (***I***nternational ***O***rganization for ***S***tandardization) a créé ***la norme ISO 8859*** constitué de 16 tables de correspondance entre 256 caractères et les 256 premiers nombres entiers.
Dans chacune de ces tables, les entiers de 0 à 127 codent les caractères de la table ASCII (un nombre binaire d'***un octet dont le bit de poids fort est 0***), et les entiers de 128 à 255 codent les caractères spécifiques à chaque table (un nombre binaire d'***un octet dont le bit de poids fort est 1***).

Exemples : ISO 8859-1 ou ***Latin-1*** pour la zone Europe Occidentale.
ISO 8859-6 pour la zone Arabe.
ISO 8859-16 ou ***Latin-10*** pour la zone Europe du sud-est

● Les encodages de la norme ISO 8859 permettent de coder un très grand nombre de caractères, mais ils peuvent être insuffisants si on écrit un texte avec des caractères de différentes zones géographiques par exemple. Ainsi l'ISO a créé ***la norme ISO 10646*** qui rassemble un jeu de plus de 110 000 caractères de toutes les langues. Cette norme définit plusieurs encodages nommés ***UTF*** (***U***niversal ***T***ransformation ***F***ormat) où chaque caractère est codé sur 32 bits (4 octets) maximum.
L'encodage de cette norme le plus utilisé est ***UTF-8*** où l'on code chaque caractère sur 8 bits minimum et jusqu'à 32 bits. ***Les 256 premiers caractères de cet encodage sont ceux de l'encodage Latin-1***.
Chacun des caractères est associé à un entier appelé ***point de code***.

- Si le bit de poids fort d'un octet est 0, il s'agit d'un caractère codé sur un seul octet et seuls les 7 bits suivants sont des bits codants. Cela correspond aux caractère ASCII.
- Sinon, dans le premier octet d'un caractère, les premiers bits de poids fort sont des 1 qui déterminent le nombre d'octets utilisés pour coder le caractère. Cette suite de un à quatre bits 1 est toujours suivie d'un bit 0. Les bits suivants sont des bits codants. Les octets suivants commencent par les bits 1 0 et seuls les 6 bits suivants sont des bits codants.

Table des codes possibles en UTF-8 :

Point de code (décimal)	Point de code (hexadécimal)	Code UTF8 (binaire) (x=bit codant)
0 à 127	0000 à 007F	0xxxxxxx
128 à 2 047	0080 à 07FF	110xxxxx 10xxxxxx
2 048 à 65 535	0800 à FFFF	1110xxxx 10xxxxxx 10xxxxxx
65 536 à …	10000 à 10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Exercice 5 :

1. Pourquoi ne peux-tu pas coder le mot ***« Numérique »*** au format ASCII ?
A cause du caractère accentué « é » qui n'est pas dans la table ASCII.

2.Code le mot ***« Numérique »*** au format Latin-1 sachant que le caractère « é » correspond à l'entier 233.

01001110	01110101	01101101	11101001	01110010	01101001	01110001	01110101	01100101
78	117	109	233	114	105	113	117	101
N	u	m	é	r	i	q	u	e

Exercice 6 :

1. Sachant que le point de code en décimal du caractère « é » est 233, donne le codage de ce caractère au format UTF-8.

« é » va être codé sur 2 octets car 1 octet commençant par 0 ne suffit pas pour coder 233 (127 au maximum est codable sur un octet commençant par 0). Donc son code sera au format 110xxxxx 10xxxxxx.

Les bits codants (xxxxx xxxxxx) servent à coder 233 en binaire, soit 11101001

Ainsi « é » se code en UTF-8 : 11000011 10101001

2. Donne le codage au format UTF-8 du mot « ***Numérique*** ».

N	u	m	é	r	i	q	u	e
78	117	109	233	114	105	113	117	101
01001110	01110101	01101101	11000011 10101001	01110010	01101001	01110001	01110101	01100101

Exercice 7 : Pour chacun des caractères ci-dessous, coder leur code au format UTF-8 en écrivant les octets en binaire, puis en hexadécimal.

1. A (point de code = 65). ***2.*** è (point de code = 232). ***3.*** * (point de code = 8902).

01000001 *11000011 10101000* *11100010 10100101 10100110*

Exercice 8 : Pour chacune des séquences d'octets suivantes, qui sont écrites en binaire, dire si elle représente une séquence UTF-8 valide et si oui la décoder.

On pourra utiliser le lien : https://www.w3schools.com/charsets/ref_utf_dingbats.asp .

1. 01111110 01000000 01100100.

~ @ d

2. 11000110 10000001 10000001.

Impossible car le premier octet signale un codage sur 2 octets, de fait le 3è octet devrait être le premier octet d'un nouveau carcatère et être du type 0xxxxxxxx ou 110xxxxx ou 1110xxxx ou 11110xxx.

3. 11100010 10011100 10001000.

Un caractère sur 3 octets. Les bits codants sont 0010011100001000 soit 9 992 en décimal qui correspond en UTF-8 au caractère ✈

Exercice facultatif :

Ecris une fonction ***decalage(s)*** en langage Python dont le paramètre est une chaîne de caractères ***s*** constituée des 128 caractères (en minuscules uniquement) du code ASCII et qui renvoit une chaîne de caractères ou chaque lettre a été décalée de 3 rangs. Par exemple « a » devient « d », « f » devient « i » ou « y » devient « b ».

Attention : seules les lettres doivent être décalées. Pas les espaces ou la ponctuation.

Programme cette fonction sur Edupython et teste la dans la console.

Sauvegarde en nommant « facultatif.NOM » le fichier Python dans lequel il y a ton travail et envoie lepar mail au professeur : jerome.gauthier@lyceemermozdakar.org .