REPRESENTATION DU TEXTE

(https://www.youtube.com/watch?v=MijmeoH9LT4) Représenter du texte pour un ordinateur, c'est pouvoir identifier plus de cent mille caractères dans toutes les langues, rendre compatible tous les systèmes entre eux, tout cela dans un espace mémoire optimal !

Cette vidéo retrace l'histoire d'un standard aujourd'hui majoritairement utilisé sur le web et en programmation : la norme Unicode.

Elle est en anglais, on peut activer les sous-titres français dans les paramètres de la vidéo.

1. ASCII

Aux Etats-Unis, dans les années 1970, à l'origine du développement des technologies informatiques et alors que les capacités de mémorisation des ordinateurs étaient encore assez limitées, on n'imaginait pas que ceux-ci seraient utilisés un jour pour traiter d'autres textes que des communications techniques,essentiellement en anglais. Voici ci-contre les caractères affichables dont on avait besoin.

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

Exercice 1:

1. Combien de caractères trouve-t-on dans l'image ci-dessus ?
2. Combien de bits sont nécessaires pour pouvoir représenter ces caractères ?

Réponses :

1.
2.

A retenir :

L'ASCII(American Standard Code for Information Interchange) définit 128 caractères associés aux entiers de 0 à 127 et codés en binaire de `00000000` à `11111111`. En plus des caractères affichables, ce standard attribue des valeurs à des caractères spéciaux invisibles (espace, effacer, retour à la ligne). Le codage s'effectue sur 8 bits (la plupart des ordinateurs utilisant des multiples de 8), le bit restant à gauche étant `0`. Ci-dessous, la table complète des caractères ASCII :

ASCII TABLE

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 2:

1. Par quelle valeur décimale est codée 'A' ? Quelle est sa valeur en binaire ?
2. Par quelle valeur décimale est codée 'a' ? Quelle est sa valeur en binaire ?
3. Quel caractère est représenté par la valeur décimale 64 ? Quelle est sa valeur en binaire ?
4. Quel caractère est représenté par la valeur décimale 32 ? Quelle est sa valeur en binaire ?
5. Quelle lettre est représentée par `01000100` ?

Réponses :

1.
2.
3.
4.
5.

Le saviez-vous ?

Il est possible d'afficher à l'écran un caractère grâce à son code : Sous Windows, pour générer le caractère 'y', on peut utiliser la combinaison de touches `Alt + 121` , 121 étant le code ASCII de cette lettre.

Exercice 3 :

Ecrire vos initiales avec des combinaisons de touches.

Réponses :

-
-

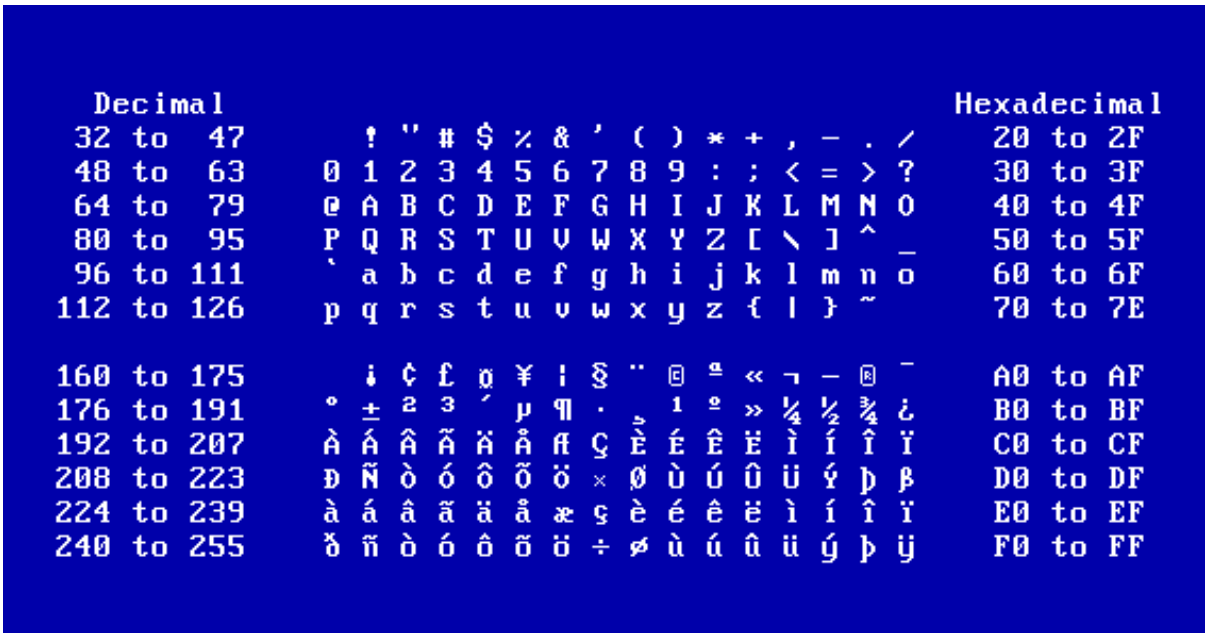
2. Latin-1: ISO-8859-1

La norme ASCII convient bien à la langue anglaise, mais pose des problèmes dans d'autres langues, par exemple le français(accents, tréma, cédille...).

C'est pour répondre à ce problème qu'est née la norme ISO-8859-1. Cette norme étend le standard l'ASCII(les codes précédents restent valables) en utilisant le 8ème bit qui était `0` , ce qui permet donc d'encoder 128 caractères supplémentaire. Il y a donc 256 caractères encodés.

Cette norme va être principalement utilisée dans les pays européens puisqu'elle permet d'encoder les caractères utilisés dans les principales langues européennes (la norme ISO-8859-1 est aussi appelée "Latin-1" car elle permet d'encoder les caractères de l'alphabet dit "latin").

Ci-dessous les valeurs utilisées par cet encodage :



Exercice 4 :

1. Quelle est la valeur du caractère è ?
2. Quel caractère est associé à la valeur 249 ?

Réponses :

1.
2.

Et les autres langues ?

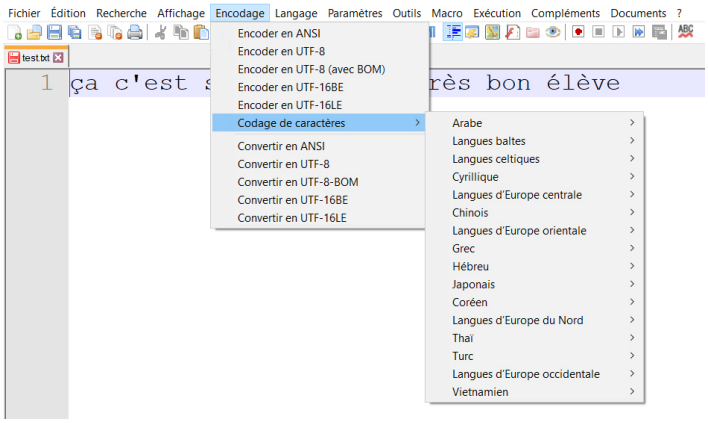
Beaucoup d'autres langues dans le monde n'utilisent pas l'alphabet dit "latin", par exemple le chinois , le japonnais le russe ou encore le grec... D'autres normes ont donc dû voir le jour, par exemple la norme "GB2312" pour le chinois simplifié,la norme "JISX0208" pour le japonais.

La multiplicité des normes pose alors problème puisqu'une même valeur peut potentiellement encoder plusieurs caractères suivant la norme choisie.

De plus, il est impossible avec ces normes d'écrire un texte comportant des mots de français et de japonais par exemple.

Exercice 5 :

1. Ouvrir le notepad++ :
    - Taper ce texte : 'ça c'est sûr,un très bon élève'.
    - Dans l'onglet 'encodage' puis 'codage des caractères', retrouver la norme 8859-1 et cliquer pour utiliser ce codage.
    - Que se passe-t-il ?



1. Tester d'autres codages dans le même menu. Quels sont les caractères qui posent toujours problème ?

1. Quels sont ceux qui s'affichent correctement dans la plupart des cas ?

1. Retrouver l'affichage correct du texte : quelle norme est alors utilisée ?

Réponses :

1.
2.
3.
4.

3. Unicode, UTF-8

Pour éviter ce genre de problème, en 1991 une nouvelle norme a vu le jour : Unicode

Cette norme réussit à rassembler tous les caractères existants 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 par exemple.

Le standard Unicode est uniquement une table qui regroupe tous les caractères existants au monde auxquels on a attribué un numéro unique, il ne s'occupe pas de la façon dont les caractères sont codés dans la machine. Il existe plusieurs systèmes de codage : UTF-8, UTF-16, UTF-32. Le plus utilisé, notamment sur le Web, est UTF-8.

- Exemple : le caractère '€' est identifié par le nombre entier 8364.

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(8 bits), 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.

Si ce standard a vocation a devenir universel, il reste encore quelques incompatibilités car tout le monde ne l'utilise pas encore.

Remarques : Python reconnait les caractères indéxés selon la norme Unicode. Chaque caractère est identifié par un nombre entier.

- La fonction `chr(N)` prend en paramètre l'entier naturel `N` et renvoie le caractère correspondant.
- la fonction `ord(car)` prend en paramètre le caractère `car` et renvoie l'entier correspondant.

In [5]:

```
#Exécuter les lignes ci-dessous pour tester ces deux fonctions
chr(65)
```

Out[5]:

```
'A'
```

In [6]:

```
ord('€')
```

Out[6]:

```
8364
```

4. Exercices

Exercice :

1. Quelle opération mathématique permet de calculer le nombre associé à une lettre majuscule à partir du nombre associé à cette même lettre minuscule.
2. Ecrire la fonction `majus(lettre)` qui prend en paramètre une lettre minuscule et qui renvoie la même lettre en majuscule.
3. Ecrire la fonction `minus(lettre)` qui prend en paramètre une lettre majuscule et qui renvoie la même lettre en minuscule.

Réponses :

1. Pour passer d'une lettre minuscule à une lettre majuscule, on soustrait 32.

In [2]:

```
#2.
```

In [1]:

```
#3.
```

Exercice :

1. Ecrire la fonction `majuscule(mot)` qui prend en paramètre une chaîne de caractères écrites en minuscules et qui renvoie la même chaîne en majuscule.
2. Ecrire la fonction `minuscule(MOT)` qui prend en paramètre une chaîne de caractères écrites en minuscules et qui renvoie la même chaîne en majuscule.

Aide : on pourra utiliser les fonctions de l'exercice précédent.

Réponses :

In [3]:

```
#1.
```

In [5]:

```
#2.
```

Exercice :

Les caractères minuscules grecs sont identifiés par les entiers successifs à partir de 945.Ecrire la fonction `grec()` qui renvoie une de chaîne de caractères contenant cet alphabet.

Le résultat affiché sera 'αβγδεζηθικλμνξοπρςστυφχψω'

In [4]:

```
#Réponse :
```

Exercice : convertir un fichier dans différentes normes

4. ASCII-Art

Une épéé : o()xxxx[{::::::::::::::::::::::::::::::::::::::>

Un poisson : ><((((('>

Un koala : @(*O*)@

Votre ASCII :

FIN