Encodage des textes

Pour écrire du texte brut (c'est-à-dire sans aucun effet) on peut utiliser Notepad++ (windows), TextEdit (MacOs), Kwrite ou Geany (Linux).
Fonctions Pyhton utiles : ord(), bin(), chr().

1 Norme ASCII

Avant 1960, chaque système codait à sa manière les informations (dont les caractères) ce qui se traduisait par d'énormes problèmes de compatibilité et d'échange.
En 1960, l'organisation internationale de normalisation (ISO) décide de créer la norme **ASCII** (American Standard Code for Information Interchange).
À chaque caractère est associé un nombre binaire sur 8 bits (1 octet).
Seuls 7 bits sont utilisés pour coder un caractère.
Les caractères en Ascii sont ainsi des octets dont le bit de poids fort est 0.
Avec 7 bits il est possible de coder jusqu'à 2^7=128 caractères ce qui est largement suffisant pour un texte écrit en langue anglaise (pas d'accents et autres lettres particulières) :

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]

Ainsi la lettre A est associée à $65_{10} = 01000001_2$ et la lettre a est associée à $97_{10} = 01100001_2$.
On retrouve ces valeurs avec Python :

```
>>>ord('A')
65
>>>bin(65)
'0b1000001'
```

Exemple : La chaine de caractères "Yes!" se code 89 101 115 32 33 en décimal soit 01011001 01100101 01110011 00100000 00100001 en binaire.

2 Norme Latin

L'Ascii est idéal pour les pays anglo-saxons. Cependant, nous avons dans nos pays européens des caractères particuliers, comme ç ou les caractères accentués. Pour coder ces caractères, on a étendu l'Ascii à la norme ISO 8859-1 dit Latin-1. Le huitième bit de chaque octet a ainsi été utilisé. Plusieurs versions de cette norme ont été créées dont la dernière, Latin-9, comprenant le symbole de l'euro €.
Problème, cette norme restait encore insuffisante au niveau international.

3 Norme Unicode

Pour résoudre ce problème, en 1991 une nouvelle norme a vu le jour : Unicode. Unicode a pour ambition de rassembler tous les caractères existant afin qu'un texte codé dans cette norme puisse être lu dans n'importe quel pays.

Unicode est uniquement une table qui regroupe tous les caractères existant au monde, 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, notamment sur le Web, est UTF-8.

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.

Avec Python, UTF-8 est utilisé par défaut, et on a :
>>> ord('Ê')
202

On se retrouve donc avec un octet supérieur à 127 (le bit de poids fort est 1) : 11001010

En Python, la fonction *chr()* permet de traduire un code en caractère :
>>> chr(202)
Ê
Si on dispose du code binaire, on utilisera : *chr(0b00100001)* qui donne '!'

UTF-16 et UTF-32 utilisent la même table de caractères mais codent ceux-ci sur 16 et 32 bits (donc sont plus gourmands en ressource).