Charsets & encodings

Jacquelin Charbonnel

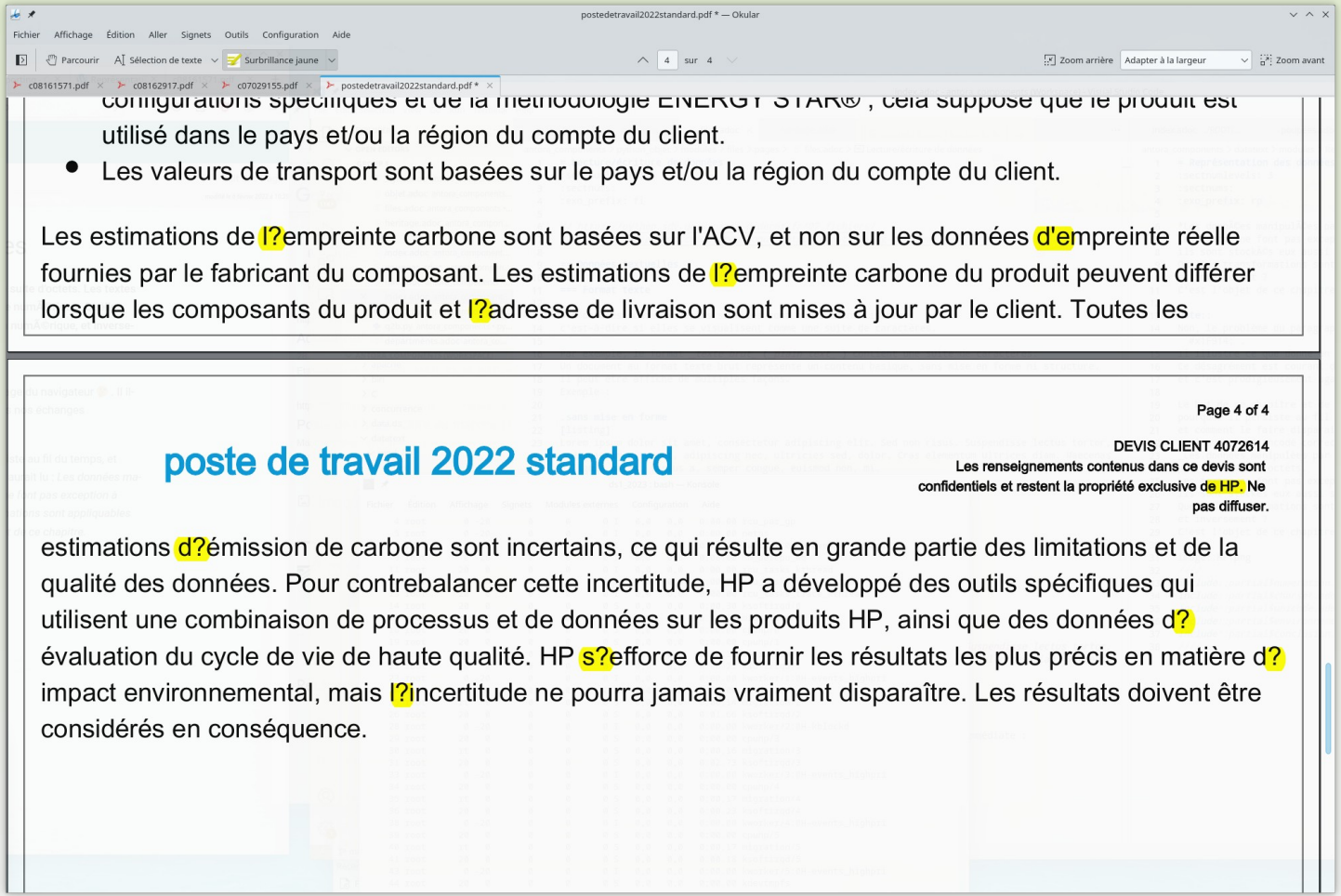
Journées Mathrice de Grenoble, mars 2024

Micmac & embrouillamini

Les donnÃ©es manipulÃ©es par un ordinateur sont reprÃ©sentÃ©es par une suite d’octets. Les textes ne font pas exception Ã cette rÃšgle : ils sont stockÃ©s eux aussi sous forme numÃ©rique. Quelles transformations sont nÃ©cessaires pour passer du texte Ã sa reprÃ©sentaton numÃ©rique, et inversement ?

Les données manipulées par un ordinateur sont représentées par une suite d’octets. Les textes ne font pas exception à cette règle : ils sont stockés eux aussi sous forme numérique. Quelles transformations sont nécessaires pour passer du texte à sa représentaton numérique, et inversement ?

Micmac & embrouillamini



Données textuelles

- Définitions
  - texte composé de signes typographiques (les *caractères*) issus d'un alphabet (le *charset*)
  - sur support numérique, chaque caractère d'un charset est représenté par un code (un entier)
  - la correspondances *caractère* ↔ *code* pour un charset donné est un codage (encodage, *encoding*)

- Historique
  - Les années 60 aux US
    - l'alphabet de la langue anglaise comporte peu de lettres (26)
    - le nombre de caractères pour rédiger un texte (min, MAJ, chiffres, signes) ne dépasse pas 90
    - 7 bits sont suffisants pour les coder
    - reste même de la place pour les caractères de contrôle
    - l'ASCII : un charset de 128 caractères

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Codage ASCII

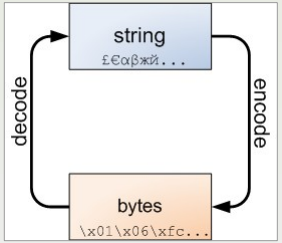
	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

- chaque caractère est codé par son indice dans la table

```
$ cat <<EOT | xxd -g1
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
EOT
00000000: 4c 6f 72 65 6d 20 69 70 73 75 6d 20 64 6f 6c 6f  Lorem ipsum dolo
00000010: 72 20 73 69 74 20 61 6d 65 74 2c 20 63 6f 6e 73  r sit amet, cons
00000020: 65 63 74 65 74 75 72 20 61 64 69 70 69 73 63 69  ectetur adipisci
00000030: 6e 67 20 65 6c 69 74 2c 0a 73 65 64 20 64 6f 20  ng elit,.sed do
00000040: 65 69 75 73 6d 6f 64 20 74 65 6d 70 6f 72 20 69  eiusmod tempor i
00000050: 6e 63 69 64 69 64 75 6e 74 20 75 74 20 6c 61 62  ncididunt ut lab
00000060: 6f 72 65 20 65 74 20 64 6f 6c 6f 72 65 20 6d 61  ore et dolore ma
00000070: 67 6e 61 20 61 6c 69 71 75 61 2e 0a              gna aliqua..
```

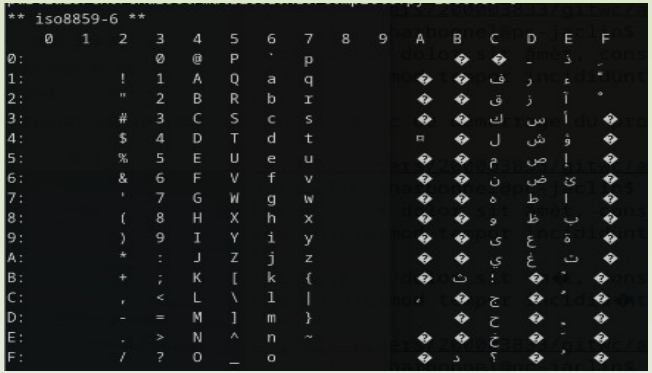
code compris entre 00 et 7F

- opérations de codage/décodage à chaque lecture/écriture



Années 80 revendicatives

- et les caractères nationaux ?
- idée simple : utiliser toute la tables (en utilisant le 8ème bit)




```
$ cat <<EOT | iconv -f utf8 -t latin1 | xxd -g1
Lorem ipsum dolor sit âmèt, consectetur adipisçing élit,
sêd do eiusmod tempor incididünt ut labore et dolore magna aliqua.
EOT
00000000: 4c 6f 72 65 6d 20 69 70 73 75 6d 20 64 6f 6c 6f  Lorem ipsum dolo
00000010: 72 20 73 69 74 20 e2 6d e8 74 2c 20 63 6f 6e 73  r sit .m.t, cons
00000020: 65 63 74 65 74 75 72 20 61 64 69 70 69 73 e7 69  ectetur adipis.i
00000030: 6e 67 20 e9 6c 69 74 2c 0a 73 ea 64 20 64 6f 20  ng .lit,.s.d do
00000040: 65 69 75 73 6d 6f 64 20 74 65 6d 70 6f 72 20 69  eiusmod tempor i
00000050: 6e 63 69 64 69 64 fc 6e 74 20 75 74 20 6c 61 62  ncidid.nt ut lab
00000060: 6f 72 65 20 65 74 20 64 6f 6c 6f 72 65 20 6d 61  ore et dolore ma
00000070: 67 6e 61 20 61 6c 69 71 75 61 2e 0a              gna aliqua..
```

code compris entre 00 et FF

Charsets nationaux

iso8859-1 (latin1)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p				°	À	Ð	à	ð
1:			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2:			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3:			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4:			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5:			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6:			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7:			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8:			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
9:			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A:			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B:			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C:			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
D:			-	=	M	]	m	}				½	Í	Ý	í	ý
E:			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
F:			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

ISO8859-5 (cyrillic)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p				А	Р	а	р	№
1:			!	1	A	Q	a	q			Ё	Б	С	б	с	ё
2:			"	2	B	R	b	r			Ђ	В	Т	в	т	ђ
3:			#	3	C	S	c	s			Ѓ	Г	У	г	у	ѓ
4:			$	4	D	T	d	t			Є	Д	Ф	д	ф	є
5:			%	5	E	U	e	u			S	Е	Х	е	х	s
6:			&	6	F	V	f	v			І	Ж	Ц	ж	ц	і
7:			'	7	G	W	g	w			Ї	З	Ч	з	ч	ї
8:			(	8	H	X	h	x			J	И	Ш	и	ш	j
9:			)	9	I	Y	i	y			Љ	Й	Щ	й	щ	љ
A:			*	:	J	Z	j	z			Њ	К	Ъ	к	ъ	њ
B:			+	;	K	[	k	{			Ћ	Л	Ы	л	ы	ћ
C:			,	<	L	\	l	|		Ќ	М	Ь	м	ь	ќ
D:			-	=	M	]	m	}				Н	Э	н	э	§
E:			.	>	N	^	n	~			Ў	О	Ю	о	ю	ў
F:			/	?	O	_	o				Џ	П	Я	п	я	џ

Iso8859-7 (grec)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p				°	ΐ	Π	ΰ	π
1:			!	1	A	Q	a	q			‘	±	A	P	α	ρ
2:			"	2	B	R	b	r			’	²	B	**?**	β	ς
3:			#	3	C	S	c	s			£	³	Γ	Σ	γ	σ
4:			$	4	D	T	d	t			€	΄	Δ	Τ	δ	τ
5:			%	5	E	U	e	u			₯	΅	E	Υ	ε	υ
6:			&	6	F	V	f	v			¦	Ά	Z	Φ	ζ	φ
7:			'	7	G	W	g	w			§	·	H	Χ	η	χ
8:			(	8	H	X	h	x			¨	Έ	Θ	Ψ	θ	ψ
9:			)	9	I	Y	i	y			©	Ή	I	Ω	ι	ω
A:			*	:	J	Z	j	z			ͺ	Ί	K	Ϊ	κ	ϊ
B:			+	;	K	[	k	{			«	»	Λ	Ϋ	λ	ϋ
C:			,	<	L	\	l	|		¬	Ό	M	ά	μ	ό
D:			-	=	M	]	m	}				½	N	έ	ν	ύ
E:			.	>	N	^	n	~			**?**	Ύ	Ξ	ή	ξ	ώ
F:			/	?	O	_	o				―	Ώ	O	ί	o	**?**

Charsets nationaux

iso8859-1 (latin1)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p				°	À	Ð	à	ð
1:			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2:			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3:			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4:			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5:			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6:			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
7:			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8:			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
9:			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A:			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B:			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C:			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
D:			-	=	M	]	m	}				½	Í	Ý	í	ý
E:			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
F:			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

iso8859-15

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p				°	À	Ð	à	ð
1:			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2:			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
3:			#	3	C	S	c	s			**£**	³	Ã	Ó	ã	ó
4:			$	4	D	T	d	t			**€**	Ž	Ä	Ô	ä	ô
5:			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
6:			&	6	F	V	f	v			Š	¶	Æ	Ö	æ	ö
7:			'	7	G	W	g	w			§	·	Ç	×	ç	÷
8:			(	8	H	X	h	x			š	ž	È	Ø	è	ø
9:			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A:			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B:			+	;	K	[	k	{			«	»	Ë	Û	ë	û
C:			,	<	L	\	l	|		¬	**Œ**	Ì	Ü	ì	ü
D:			-	=	M	]	m	}				**œ**	Í	Ý	í	ý
E:			.	>	N	^	n	~			®	**Ÿ**	Î	Þ	î	þ
F:			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

Années 80 : les marques aussi se distinguent

Apple - MacRoman

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p	Ä	ê	†	∞	¿	—	‡	
1:			!	1	A	Q	a	q	Å	ë	°	±	¡	—	·	Ò
2:			"	2	B	R	b	r	Ç	í	¢	≤	¬	“	‚	Ú
3:			#	3	C	S	c	s	É	ì	£	≥	√	”	„	Û
4:			$	4	D	T	d	t	Ñ	î	§	¥	ƒ	‘	‰	Ù
5:			%	5	E	U	e	u	Ö	ï	•	µ	≈	’	Â	ı
6:			&	6	F	V	f	v	Ü	ñ	¶	∂	Δ	÷	Ê	ˆ
7:			'	7	G	W	g	w	á	ó	ß	∑	«	◊	Á	˜
8:			(	8	H	X	h	x	à	ò	®	∏	»	ÿ	Ë	¯
9:			)	9	I	Y	i	y	â	ô	©	π	…	Ÿ	È	˘
A:			*	:	J	Z	j	z	ä	ö	™	∫		/	Í	˙
B:			+	;	K	[	k	{	ã	õ	´	ª	À	€	Î	°
C:			,	<	L	\	l	|	å	ú	¨	º	Ã	‹	Ï	
D:			-	=	M	]	m	}	ç	ù	≠	Ω	Õ	›	Ì	˝
E:			.	>	N	^	n	~	é	û	Æ	æ	Œ	fi	Ó	˛
F:			/	?	O	_	o		è	ü	Ø	ø	œ	fl	Ô	ˇ

Microsoft - windows_1250

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0:				0	@	P	`	p	€	￿		°	Ŕ	Đ	ŕ	đ
1:			!	1	A	Q	a	q	￿	‘	ˇ	±	Á	Ń	á	ń
2:			"	2	B	R	b	r	‚	’	˘	˛	Â	Ň	â	ň
3:			#	3	C	S	c	s	￿	“	Ł	˛	Ă	Ó	ă	ó
4:			$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
5:			%	5	E	U	e	u	…	•	Ą	µ	Ĺ	Ő	Í	ő
6:			&	6	F	V	f	v	†	–	¦	¶	Ć	Ö	ć	ö
7:			'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
8:			(	8	H	X	h	x	￿	￿	¨	¸	Č	Ř	č	ř
9:			)	9	I	Y	i	y	‰	™	©	ą	É	Ů	é	ů
A:			*	:	J	Z	j	z	Š	š	Ş	ş	Ę	Ú	ę	ú
B:			+	;	K	[	k	{	‹	›	«	»	Ë	Ű	ë	ű
C:			,	<	L	\	l	|	Ś	ś	¬	Ľ	Ě	Ü	ě	ü
D:			-	=	M	]	m	}	Ť	ť		˝	Í	Ý	í	ý
E:			.	>	N	^	n	~	Ž	ž	®	ľ	Î	Ţ	î	ţ
F:			/	?	O	_	o		Ź	ź	Ż	ż	Ď	ß	ď	˙

- tous ces charsets ont 256 caractères
- les codes tiennent sur 1 octet
- tous ces charsets englobent l’ASCII
- mais ils sont mutuellement exclusifs
  - par exemple, impossible d’écrire :

Le dicton russe « *Азбука -- к мудрости ступенька.* » signifie « *Nous devons d'abord apprendre à marcher avant d'apprendre à courir.* »

Fin des années 80 : ça part en vrille

- arrivée de charsets de plus de 256 caractères :
  - EUC-JP, pour le japonais (caractères codés sur 2 octets)
  - Big5, pour le chinois traditionnel à Taïwan (codes sur 1 ou 2 octets suivant les caractères)
- mais changer la taille des codes a un impact colossal
  - sur le matériel et le logiciel :
    - sur les langages de programmation, les compilateurs, les bibliothèques, les systèmes d'exploitation
  - **rupture technologique**
  - c'est pourquoi, les charsets codés sur 1 octet ont été exploités aussi longtemps que possible
- quitte à changer, autant repartir à zéro et voir loin, pour :
  - uniformiser le codage des signes de toutes les cultures (actuelles et passées), des symboles, smileys, etc.,
  - pouvoir utiliser tous ces signes au sein d'un même texte
  - obtenir des documents utilisables à l'identique sur tout système informatique

Années 90 : Unicode remet à plat

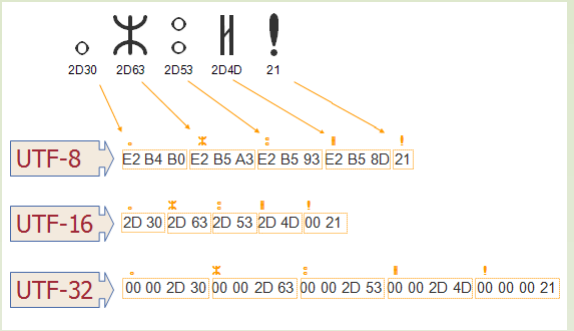
- un signe typographique est défini comme la plus petite composante d'un alphabet ayant une valeur sémantique :
  - l'accent circonflexe est un signe à part entière
  - la cédille en est un autre
  - Unicode référence plus d'un million de signes
- plus que des alphabets, Unicode gère des écritures
  - chaque caractère Unicode possède des attributs :
    - un nom : `solidus − Greek capital letter delta − left curly bracket`
    - un ***code point***
      - identifiant entier (sur 4 octets) canonique et unique
      - sans vocation à coder numériquement le caractère
      - noté `U+xxxx`, `U+xxxxx` ou `U+xxxxxx`
        - exemple : `U+0000`, `U+FFFF`, `U+10FFFF`
    - et des méta-données :
      - est-ce une majuscule ? une minuscule ? une ponctuation ? un chiffre (et si oui, sa valeur numérique) ?
      - se combine-t-il à d'autres caractères (accent, cédille, tilde, tréma, etc.) ?
      - son orientation (droite→gauche, gauche→droite, haut→bas, etc.)

Unicode attributs

Par exemple via Python

```
>>> import unicodedata

>>> unicodedata.name('/')
'SOLIDUS'

>>> unicodedata.name('{')
'LEFT CURLY BRACKET'

>>> unicodedata.lookup('solidus')
'/'

>>> unicodedata.category('A')
'Lu'                                    # Letter uppercase

>>> unicodedata.category('a')
'Ll'                                    # Letter lowercase

>>> unicodedata.category('€')
'Sc'                                    # Symbol currency

>>> c = "\u0664"
>>> c
'٤'

>>> unicodedata.name(c)
'ARABIC-INDIC DIGIT FOUR'

>>> unicodedata.category(c)
'Nd'                                    # Number decimal

>>> unicodedata.numeric(c)
4.0
```

Les codages d'Unicode



Unicode propose plusieurs encodage

- UTF-32
  - chaque caractère est codé directement avec son code point (sur 4 octets)
  - cette simplicité est son principal avantage : le codage/décodage est simple
  - par contre, il est particulièrement gourmand en taille
  - évidemment incompatible avec l'ASCII
  - peu utilisé
- UTF-16
  - la plupart des caractères (notamment ceux des langues modernes) est représentée sur 2 octets, les autres le sont sur 4 octets.
  - utilisé dans l'environnement Windows
  - évidemment incompatible avec l'ASCII
- UTF-8
  - encodage le plus utilisé en occident
  - son codage de taille variable (entre 1 et 4 octets) lui permet d'être parcimonieux en taille pour les alphabets latin, au détriment de la vitesse de codage/décodage.
  - compatible avec l'ASCII
  - inconvénient : la longueur variable de son codage complexifie le codage/décodage

UTF-8

- principe du codage :
  - si le code point est inférieur à 128, le code est égal au code point,
  - sinon, le code est une séquence de 2, 3 ou 4 octets, où chacun est compris entre 128 et 255.
- détail du codage :
  - les caractères ASCII sont codés sur un octet dont le bit de poids fort est 0 : 0xxxxxxx
  - les caractères non-ASCII utilisent au moins deux octets :
    - le premier octet commence par autant de 1 que d'octets dans la séquence, suivis par un 0.
    - les autres octets commencent par 10.
- ce qui donne :
  - code point ayant entre 8 à 11 bits : 110xxxxx 10xxxxxx
  - code point ayant entre 12 à 16 bits : 1110xxxx 10xxxxxx 10xxxxxx
  - code point ayant entre 17 à 21 bits : 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

```
$ cat <<EOT | xxd -g1
Lorem ipsum dolor sit âmèt, consectetur adipisçing élit,
sêd do eiusmod tempor incididünt ut labore et dolore magna aliqua.
EOT
00000000: 4c 6f 72 65 6d 20 69 70 73 75 6d 20 64 6f 6c 6f  Lorem ipsum dolo
00000010: 72 20 73 69 74 20 c3 a2 6d c3 a8 74 2c 20 63 6f  r sit ..m..t, co
00000020: 6e 73 65 63 74 65 74 75 72 20 61 64 69 70 69 73  nsectetur adipis
00000030: c3 a7 69 6e 67 20 c3 a9 6c 69 74 2c 0a 73 c3 aa  ..ing ..lit,.s..
00000040: 64 20 64 6f 20 65 69 75 73 6d 6f 64 20 74 65 6d  d do eiusmod tem
00000050: 70 6f 72 20 69 6e 63 69 64 69 64 c3 bc 6e 74 20  por incidid..nt
00000060: 75 74 20 6c 61 62 6f 72 65 20 65 74 20 64 6f 6c  ut labore et dol
00000070: 6f 72 65 20 6d 61 67 6e 61 20 61 6c 69 71 75 61  ore magna aliqua
00000080: 2e 0a
```

4c est inférieur à 128
donc le code est sur 1 octet

c3==11000011
donc le code est sur 2 octets

Écueils & curiosités

Faux-amis

```
>>> import unicodedata

>>> a, b = "\u0061", "\u0430"
>>> a, b, a==b
('a', 'а', False)

>>> [ unicodedata.name(x) for x in [a,b] ]
['LATIN SMALL LETTER A', 'CYRILLIC SMALL LETTER A']
```

Python

```
>>> import unicodedata

>>> a, b = "\u0050", "\u03a1"
>>> a, b, a==b
('P', 'Ρ', False)

>>> [ unicodedata.name(x) for x in [a,b] ]
['LATIN CAPITAL LETTER P', 'GREEK CAPITAL LETTER RHO']
```

Python

```
>>> import unicodedata

>>> s = "A\u216c\u2160B\u0391\u0392\u0391.COM"
>>> s
'AⅬIBABA.COM'

>>> [ unicodedata.name(c) for c in s ]
['LATIN CAPITAL LETTER A',
 'ROMAN NUMERAL FIFTY',
 'ROMAN NUMERAL ONE',
 'LATIN CAPITAL LETTER B',
 'LATIN CAPITAL LETTER A',
 'GREEK CAPITAL LETTER BETA',
 'GREEK CAPITAL LETTER ALPHA', … ]
```

Python

Unicode modélise des écritures

```
>>> import unicodedata

>>> a, b = "C", "Ç"
>>> c, d = "\u00B8", "\u0327"
>>> a, b, c, d
('C', 'Ç', '¸', '̧')

>>> [ unicodedata.name(x) for x in [a,b,c,d] ]
['LATIN CAPITAL LETTER C'
 'LATIN CAPITAL LETTER C WITH CEDILLA',
 'CEDILLA']
 'COMBINING CEDILLA']

>>> b, a+c, a+d
('Ç', 'Ç', 'Ç')

>>> b == a+d
False

>>> len(b), len(a+d)
(1, 2)

>>> unicodedata.name(a+d)
TypeError: name() argument 1 must be a unicode
character, not str
```

Python

```
$ echo C - $'\u0327' - C$'\u0327' - C$'\u0327''\u0327'
C - ̧ - Ç - Ç

$ echo AS$'\u0327'A - Live in Paris
AŞA - Live in Paris
```

bash