Encodings

Karën Fort

karen.fort@univ-lorraine.fr / https://members.loria.fr/KFort

Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

UNIVERSITÉ DE LORRAINE

The language of the computer

- Text does not exist in computers

The language of the computer

- Text does not exist in computers
- Which language do computers “speak” ? which are their “words” ?

The language of the computer

- Text does not exist in computers
- Which language do computers “speak” ? which are their “words” ?
- Basically, a computer only understands the binary language, i.e. a sequence of 0s and 1s
  Simply put : power/no power

The language of the computer

- Text does not exist in computers
- Which language do computers “speak” ? which are their “words” ?
- Basically, a computer only understands the binary language, i.e. a sequence of 0s and 1s
  Simply put : power/no power
- this elementary unit of information which can take 0 or 1 as value is called **bit** (BInary digiT)

Exercise

Exercise

How many different possibilities of combining 2 bits ?

A bit further on this

- In fact, processors handle groups of bits of fixed size
- The first processors were using 8 bits groups, i.e. a **byte** (*octet* in French)
- From eight bits (8086 processors), processors evolved to 32 (Pentium 4), to reach 64 as of today. These groups correspond to the transport (in buses) and processing (registry size) capacity of the computer

To be noted

8 bits = 1 byte = 1 octet (in French)

Say “A” !

- “A” is in fact an abstract entity which name is “uppercase a” and which **glyph** ressembles a triangle with a shortened and pulled up lower side [drawing]

Say “A” !

- “A” is in fact an abstract entity which name is “uppercase a” and which **glyph** ressembles a triangle with a shortened and pulled up lower side [drawing]
- How to say “A” to a computer ?

Say “A” !

- “A” is in fact an abstract entity which name is “uppercase a” and which **glyph** ressembles a triangle with a shortened and pulled up lower side [drawing]

- How to say “A” to a computer ?
- 01000001

Say “A” !

- “A” is in fact an abstract entity which name is “uppercase a” and which **glyph** ressembles a triangle with a shortened and pulled up lower side [drawing]

- How to say “A” to a computer ?
- 01000001

To be noted

1 character = 1 byte (beware, simplification !)

Byte and bits

Exercise

How many values can you code with one byte ?

A few examples

Character	Binary Code	Description
A	01000001	Character « A »
a	01100001	Character « a »
T	01010100	Character « T »
t	01110100	Character « t »
3	00110011	Character « 3 »
$	00100100	Character « $ »
BEL	00000111	Bip

To be noted

- to switch from uppercase to lowercase, you just need to change the third bit to 1
- numbers have their own code
- there is only one "A", which is displayed differently according to fonts or styles

Say “Bonjour”

Exercise

How many bits in “Bonjour” ?

So computers speak bytes, so what ?

A translation is needed for the text to be “coded” and “decoded” properly, in a **standardised** way

Conversion tables are needed, called encodings

In short

- computers only understand binary language, i. e. a sequence of 0s and 1s
- the object which takes 0 or 1 as value is called a bit
- if we simplify : 1 character = 1 byte = 8 bits

Da ASCII code

- In the early days of computing, 7-bit character encoding was considered sufficient, as it allows for the representation of 2^7=128 different characters. "A" is therefore encoded as 1000001.
- This led to the creation of the ASCII (American Standard Code for Information Interchange) table, published in 1968.
- For a long time, it was **the** standard reference table in computing -so much so that it became an official standard : ISO-646.

The ASCII table : isn't that enough ?

	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

No, that's not enough

Big Blue invents extended ASCII

- In 1981, IBM releases its first PC and adds a bit to ASCII.
- OEM (Original Equipment Manufacturer) extended ASCII thus allows for the encoding of 256 characters (2^8), and some are happy to be able to celebrate Noël.
- This extended ASCII code is not standardized and depends heavily on the platform used.

Extended ASCII table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
E	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	ø	∈	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	

You can imagine the look on the faces of those who need to write водка...

The ISO family

. . .hence the idea of creating different character sets to suit the needs of each language

- From 1987 onwards, the extended ASCII table was adapted into multiple variations (all encoded using a single byte).
- The first 128 characters of all ISO 8859 character sets correspond to the ASCII characters.
- The ISO 8859 standard currently comprises 16 tables, numbered 1 to 16 : 1 for so-called Western languages, 2 for Central/Eastern European languages, 5 for Cyrillic, 6 for Arabic, 7 for Greek, 8 for Hebrew, etc.

Example : ISO-8859-1

The ISO-8859-1 table defines what it calls Latin Alphabet No. 1, or Latin-1 : 191 characters of the Latin alphabet.

ISO/CEI 8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*inutilisés*															
1x																
2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*inutilisés*															
9x																
Ax		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
Bx	°	±	²	³	´	µ	¶	•	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Example : ISO-8859-5

Notice the first 128 characters. . .

ISO/IEC 8859-5																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*unused*															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*unused*															
9x																
Ax	*NBSP*	Ё	Ђ	Ґ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	*SHY*	Ў	Џ
Bx	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Cx	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Dx	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
Ex	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
Fx	№	ё	ђ	ґ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	§	ў	џ

Example : ISO-8859-15

Also known as Latin-9 (?), it is a direct extension of ISO 1 (though a later one), with the exception of 8 characters.

Differences between ISO 8859-1 and ISO 8859-15 :

Position	0xA4	0xA6	0xA8	0xB4	0xB8	0xBC	0xBD	0xBE
8859-1	¤	¦	¨	´	¸	¼	½	¾
8859-15	€	Š	š	Ž	ž	Œ	œ	Ÿ

You have enough ?

Naturally, among the most common encodings are "proprietary" encodings :

- Microsoft : Windows1252 et al.
- Apple : MacRoman

Limitations

- Incompleteness or display issues for some languages
- Impossible to write Russian and French (non-ASCII characters) in the same file
- Errors caused by the near-overlap of some encodings (e.g., $ turning into £)
- What about the billion Chinese ?

Parenthesis on Chinese

- There are two Chinese writing systems : Simplified, used in the People's Republic of China and Singapore ; and Traditional, more common in the diaspora, as well as in Taiwan, Hong Kong, and Macau.

Parenthesis on Chinese

- There are two Chinese writing systems : Simplified, used in the People's Republic of China and Singapore ; and Traditional, more common in the diaspora, as well as in Taiwan, Hong Kong, and Macau.
- Each writing system has its own encoding : specifically, GB 2312-80 (or Guobiao) for Simplified Chinese—with a mere 6,763 characters (! !)—and Big5 for Traditional Chinese, with 13,053 characters.

Parenthesis on Chinese

- There are two Chinese writing systems : Simplified, used in the People's Republic of China and Singapore ; and Traditional, more common in the diaspora, as well as in Taiwan, Hong Kong, and Macau.
- Each writing system has its own encoding : specifically, GB 2312-80 (or Guobiao) for Simplified Chinese—with a mere 6,763 characters (! !)—and Big5 for Traditional Chinese, with 13,053 characters.
- 13,053 characters ? ! But where do they fit them all ?

Parenthesis on Chinese

- There are two Chinese writing systems : Simplified, used in the People's Republic of China and Singapore ; and Traditional, more common in the diaspora, as well as in Taiwan, Hong Kong, and Macau.
- Each writing system has its own encoding : specifically, GB 2312-80 (or Guobiao) for Simplified Chinese—with a mere 6,763 characters (! !)—and Big5 for Traditional Chinese, with 13,053 characters.
- 13,053 characters ? ! But where do they fit them all ?
- The Chinese simply use more bits to encode their character sets : exactly 16 (or 2 bytes).

Asian elephants

Exercise

How many values can you represent with 16 bits ?

Asian elephants

Exercise

How many values can you represent with 16 bits ?
→ 2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2=2^{16}=65,536
→ It's enough for Chinese, even Traditional. Even 14 would have been enough...(2^{14}=16,384)
Then why 16 ?
→ Because it is incomparably more practical, given that the computer handles bytes.

Let's summarize all this

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !
- 1981 - extended ASCII (8 bits) : it's Noël without водка

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !
- 1981 - extended ASCII (8 bits) : it's Noël without водка
- 1987 - ISO (8 bits) : Noël with водка (but separate), but still no €

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !
- 1981 - extended ASCII (8 bits) : it's Noël without водка
- 1987 - ISO (8 bits) : Noël with водка (but separate), but still no €
- 1997 - ISO-8859-15 (8 bits) : upgrade of ISO-8859-1, adding € !

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !
- 1981 - extended ASCII (8 bits) : it's Noël without водка
- 1987 - ISO (8 bits) : Noël with водка (but separate), but still no €
- 1997 - ISO-8859-15 (8 bits) : upgrade of ISO-8859-1, adding € !
- In parallel, some "proprietary" encodings (8 bits) : same issues as ISO. Are not recognized as standards

Let's summarize all this

- 1968 - ASCII (7 bits) : it's not Noël !
- 1981 - extended ASCII (8 bits) : it's Noël without водка
- 1987 - ISO (8 bits) : Noël with водка (but separate), but still no €
- 1997 - ISO-8859-15 (8 bits) : upgrade of ISO-8859-1, adding € !
- In parallel, some "proprietary" encodings (8 bits) : same issues as ISO. Are not recognized as standards
- Still 8 bits = 256 possible characters, while Chinese (for example) needs much more !

The Unicode project

In 1988 (?), a somewhat ambitious project was conceived : to catalog every character from every written language- past or present - and to develop a universal reference table capable of encoding them all.
This colossal undertaking was immediately named the Unicode Unicode.

Unicode aims to be :

- universal : covering all languages (even the rarest ones)
- efficient : simple to parse
- uniform : a fixed number of bits
- unambiguous : one value = a single character (once encoded, it lasts forever !)

La couverture d'Unicode

- The first 255 characters of the Unicode table are those of the ISO-8859-1 table.

La couverture d'Unicode

- The first 255 characters of the Unicode table are those of the ISO-8859-1 table.
- First step : the 63,586 most frequently used characters are grouped in the Basic Multilingual Plane (BMP).

La couverture d'Unicode

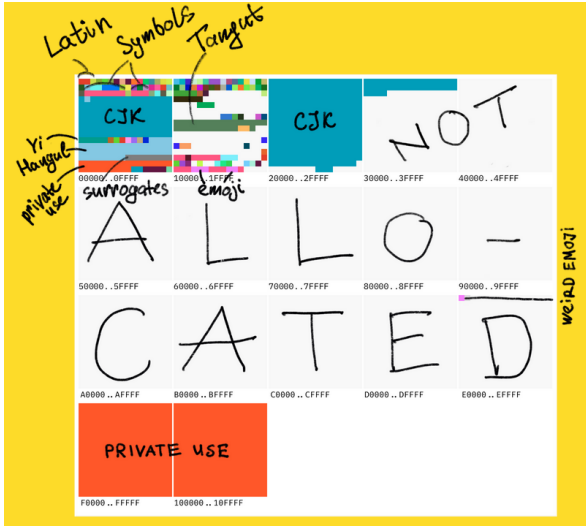
- The first 255 characters of the Unicode table are those of the ISO-8859-1 table.
- First step : the 63,586 most frequently used characters are grouped in the Basic Multilingual Plane (BMP).
- First publication of the Unicode standard in 1991 : 65,536 characters were listed and encoded.

La couverture d'Unicode

- The first 255 characters of the Unicode table are those of the ISO-8859-1 table.
- First step : the 63,586 most frequently used characters are grouped in the Basic Multilingual Plane (BMP).
- First publication of the Unicode standard in 1991 : 65,536 characters were listed and encoded.
- Today, Unicode version 17.0 (Sept. 2025) contains 159,801 characters.

What's in Unicode

https ://tonsky.me/blog/unicode/



Unicode as a character set (vs an encoding)

- In Unicode, a character corresponds to something called a code point, which is merely a theoretical concept
- Every character in every writing system was assigned a code point by the Unicode consortium
- This code point is of the form : U+XXXX. U+ meaning "Unicode", and the Xs being hexadecimal numbers. U+FEC9 is the Arabic character *Ain*. The character "A" corresponds to U+0041
- Example : "Bonjour"
  U+0042 U+006F U+006E U+006A U+006F U+0075 U+0072

(silly) Exercise

Which character is U+1F4A9 ?

Unicode encodingS

- The Unicode Consortium has defined three main "transformation" formats for encoding code points into binary. These transformation formats are known as UTF (Unicode Transformation Format) :
  UTF-32, UTF-16, UTF-8
- The fundamental principle behind these transformations is that any Unicode code point must be unambiguously recoverable from its transformed version.

The big simpleton : UTF-32

- In UTF-32 (or UCS-4), every Unicode code point is directly encoded as its binary value using a fixed number of bits (32), making the encoding process essentially a simple conversion to binary
- Conversely, this format is also the most resource-intensive, as every character - whether a common "e" or a South Indonesian ideogram -requires four bytes to encode.
- Consequently, UTF-32 is rarely used.

The undecisive : UTF-16

- UTF-16 breaks down Unicode characters into 16-bit units (2 bytes).
- These 2 bytes are more than sufficient to encode characters in the BMP (the part corresponding to UCS-2).
- Two 16-bit units are used to encode the others ; these are known as surrogate pairs.
- UTF-16 is therefore a variable-length encoding, using either 16 or 32 bits.

The efficient : UTF-8

- UTF-8 is - even more so than UTF-16 - a "variable-size" encoding method.

The efficient : UTF-8

- UTF-8 is - even more so than UTF-16 - a "variable-size" encoding method.
- In UTF-8, each code point from 0 to 127 (ASCII) is stored in a single byte.

The efficient : UTF-8

- UTF-8 is - even more so than UTF-16 - a "variable-size" encoding method.
- In UTF-8, each code point from 0 to 127 (ASCII) is stored in a single byte.
- Code points from 128 upwards are stored using 2, 3, or up to 4 bytes (theoretically 6) !

Conclusion : something-byte

- UTF-32 always uses 4 bytes
- UTF-16 is double-byte, as it uses 2 bytes (unless for characters needing "surrogate pairs")
- UTF-8 is multibyte, as it uses between 1 and 4 bytes to encode a character

UTF-8 multibyte conversion

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0*yyyzzzz*			
U+0080	U+07FF	110*xxxyy*	10*yyzzzz*		
U+0800	U+FFFF	1110*wwww*	10*xxxxyy*	10*yyzzzz*	
U+010000	U+10FFFF	11110*uvv*	10*vvwwww*	10*xxxxyy*	10*yyzzzz*

https://en.wikipedia.org/wiki/UTF-8

Let's summarize

- UTF-32 : ␣ ␣ ␣ ␣

Let's summarize

- UTF-32 : ␣ ␣ ␣ ␣
- UTF-16 : ␣ ␣ + ␣ ␣

Let's summarize

- UTF-32 : ⊔ ⊔ ⊔ ⊔
- UTF-16 : ⊔ ⊔ + ⊔ ⊔
- UTF-8 : ⊔ + ⊔ + ⊔ + ⊔

Which is the best one ?

- What are the pros and cons of each UTF ?
- UTF-32 is fixed-length, making it easy to process, but it takes up a lot of space.
- UTF-16 is sometimes trickier to handle, but it is more space-efficient and performs well, even with characters that are less common for us. It is the encoding used by the Java programming language.
- UTF-8 is optimal for Western use, but it requires a lot of space to encode certain characters. It is also tricky to handle.

Most modern tools are UTF-8-based by default

- Python : Use Python 3 and avoid Python 2 (if needed, in Python 2, use s = u"été" to declare an UTF-8 string)
- LaTeX :

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

- HTML-4 : <meta http-equiv="Content-Type" content="text/html ; charset=UTF-8">
- HTML-5 : The default character encoding is UTF-8. The usage of another encoding must be specified <meta charset="ISO-8859-1">
- XML : <?xml version="1.0" encoding="UTF-8" ?>

→ Make sure that your favourite text editor uses UTF-8 by default

Adapted from Bruno Guillaume, with his approval

Limitations of Unicode (1/2)

- The Unicode Consortium is the scene of numerous debates—which is actually a good sign !
- Example 1 : The Unicode Consortium and ISO consider Chinese, Korean, and Japanese characters to be the same, with only the glyphs differing... a view that is the subject of heated debate.
- Example 2 : Certain African writing systems used by large populations (such as Tifinagh) are not as well represented as much less widely used American scripts (such as Deseret).

Limitations of Unicode (2/2)

- Example 3 : Some Bretons complain about the absence of the barred K and of unique characters to represent CH and C'H.
- Example 4 : Some criticize Unicode for apparently favoring certain writing systems by encoding them in a simpler or more suitable way. Blackletter (Gothic script), for instance, is not encoded as such, and its correct display requires the addition of a higher-level protocol or control characters to its Latin transliteration.

WATCHING THE UNICODE PEOPLE TRY TO GOVERN THE INFINITE CHAOS OF HUMAN LANGUAGE WITH CONSISTENT TECHNICAL STANDARDS IS LIKE WATCHING HIGHWAY ENGINEERS TRY TO STEER A RIVER USING TRAFFIC SIGNS.

Naming languages is a delicate matter

The name used to designate a language is determined by the language of usage :

- E.g., allemand, Deutsch, German, and tedesco all refer to the **German** language (in French, English, Slovenian, Italian, etc.).

Choosing a single language is problematic (e.g., using English for everything).

Nor is it feasible to designate a language by its "native" name (e.g., English for English, français for French, etc.) $\Rightarrow$ difficult to write for oral languages and hard to pronounce for languages with writing systems unfamiliar to the reader.

How to name languages ?

Therefore, a system for naming a language must be found that is :

- simple to implement,
- easily portable (using only ASCII characters),
- understandable by everyone (via a standard).

$\rightarrow$ the ISO-639 standard

The ISO-639 standard

ISO 639 provides three sets of language codes :

- ISO 639-1 (alpha-2) uses 2-character codes and associates them with names in French, English, and the language itself ;
- ISO 639-2 (alpha-3) uses 3-character codes and offers two possible coding schemes : ISO 639-2/B (bibliographic) and ISO 639-2/T (terminological) ; the codes are associated with names in French and English.
- ISO 639-3 supplements ISO 639-2 by including a vast number of additional languages that were previously missing ; SIL [1] is the primary author and is responsible for registrations within this part of the standard ; only English names are associated with the added codes.

1. www.sil.org

The ISO-639 standard (more)

These three sets have different rationales :

- ISO 639-1 (alpha-2) was developed primarily for use in terminology, lexicography, and linguistics ; it covers widely spoken languages for which extensive terminological resources exist ;
- ISO 639-2 (alpha-3) was developed for use in bibliography and terminology ; it includes the languages from Part 1, as well as many others with extensive literature ;
- ISO 639-3 aims to provide the most comprehensive list of languages possible, including living, extinct, ancient, and artificially constructed languages [2] - whether major or minor, written or spoken.

2. such as Esperanto, Volapük, Klingon, etc.

Extract from the codes table

Id	Part2B	Part2T	Part1	Scope	Type	Ref_Name
aao				I	L	Algerian Saharan Arabic
abh				I	L	Tajiki Arabic
ara	ara	ara	ar	**M**	L	Arabic
czh				I	L	Huizhou Chinese
deu	ger	deu	de	I	L	German
epo	epo	epo	eo	I	**C**	Esperanto
fra	fre	fra	fr	I	L	French
frm	frm	frm		I	**H**	Middle French (ca. 1400-1600)
fro	fro	fro		I	H	Old French (842-ca. 1400)
mis	mis	mis		S	S	Uncoded languages
mjy				I	**E**	Mahican
mul	mul	mul		**S**	**S**	Multiple languages
nut				I	L	Nung (Viet Nam)
und	und	und		**S**	**S**	Undetermined
vie	vie	vie	vi	I	L	Vietnamese
zho	chi	zho	zh	M	L	Chinese
zxx	zxx	zxx		S	S	No linguistic content

Other issues related to multilingualism

- Right-to-left (Arabic, Hebrew) : or how do you insert a right-to-left writing system into a left-to-right writing system (bidirectional text) ? It's easy : control markers are provided to change the writing direction (in HTML, the DIR="rtl" or "ltr" attribute).
- Agglutinations (Arabic).

This is the end, my friend !

- Depending on the platform, the character representing the line break differs :
  - Windows : CR+LF (Carriage Return Line Feed, legacy from the typewriter !)
  - Unix world : LF
  - Mac : CR

Shorcuts everywhere !

- There is an UTF-7, used for emails, for example.

Shorcuts everywhere !

- There is an UTF-7, used for emails, for example.
- Actual numbers can be declared so that the computer can perform operations on them.

Shorcuts everywhere !

- There is an UTF-7, used for emails, for example.
- Actual numbers can be declared so that the computer can perform operations on them.
- There is a third Chinese character set, Nushu, used exclusively by women...

Tools that will help you

- A good text editor : Notepad++ (Windows), Atom (Windows, Mac, Linux), Gedit (Linux)

Really useful Unix/Linux commands

- `cat file` print the content of the file
- `file file` print info about the file (encoding, newline...) BUT not always reliable
- `diff file1 file2` print difference between files
- `iconv -f ISO-8859-1 -t UTF-8 file1 > file2` encoding conversion
- `iconv -l` list of known encodings
- `dos2unix file` change newlines in the file
- `hexdump -C file` show hexadecimal content and the text (ASCII) in the second columns

- bit, byte
- ASCII, ISO tables, Unicode
- ISO 639

Exercise

The Python 3 *ord* function returns the Unicode code of a character. *chr* is the inverse function.

The list [233, 112, 97, 116, 97, 110, 116, 32, 33] corresponds to the list of Unicode codes for a string.

Which one ?