Homework

- Lab open M–F 8:00am–4:30pm
- On github.com
  - Join team Everyone
  - Class materials at http://github.com/Ling583/notes
  - Read *Bad Data Handbook* Chapter 4: "Bad Data Lurking in Plain Text"
  - Updated python in baddata4.ipynb
- On datacamp.com
  - Finish *Intro to Python for Data Science*
  - Do *Introduction to Shell for Data Science*

#4,935,860

CERTIFICATE NUMBER

STATEMENT OF ACCOMPLISHMENT

HAS BEEN AWARDED TO

rmalouf

FOR SUCCESSFULLY COMPLETING

Intro to Python for Data Science Course

DataCamp

Why Unix?

- Big applications vs. toolkit approach
- Easier to document reproducible workflows
- Batch processing
- Open source software
- Widely used across platforms (Linux, cygwin, git-bash, MacOS X)
- File sharing, remote access, security, multiprocessing, etc all pluses

Why not Unix?

- Cryptic, hard to remember command names
- Vintage 1960’s user interface
- Documentation is sparse, confusing, filled with inside jokes, and sometimes wrong
- Open source software
- Multiple incompatible dialects (Linux, cygwin, git-bash, MacOS X)

Text

- Most basic format is “plain” or “raw” text
- No such thing!
- Computers can only deal with *bits* (ones and zeros), conventionally chunked into *bytes* (8 bits)
- Represent text by giving each letter a numeric *code* between 0 and 255

base 2	`01010100`	`01101000`	`01101111`
base 16	`54`	`68`	`6F`
base 10	`84`	`104`	`111`
	`T`	`h`	`o`

ASCII

- American Standard Code for Information Interchange
- developed out of 5 bit teletype codes in 1950’s and 60’s
- unambiguous 7 bit code that used all 2^7=128 positions
- included upper and (eventually) lower case letters, numbers, punctuation, control characters (XON/XOFF, CR/LF/FF)
- character codes arranged in collating sequence
- ASCII is (allegedly) suitable only for Latin, Swahili, Hawaiian, and American English, so national variants (ISO-646) were introduced

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

Table B-1. ISO Substitution Characters

<table>
<tr><th></th><th></th><th></th><th colspan="12">Decimal Character Equivalents</th></tr>
<tr><th>ISO</th><th>Name</th><th>ID</th><th>35</th><th>36</th><th>64</th><th>91</th><th>92</th><th>93</th><th>94</th><th>96</th><th>123</th><th>124</th><th>125</th><th>126</th></tr>
<tr><td>6</td><td>ASCII</td><td>0U</td><td>#</td><td>$</td><td>@</td><td>[</td><td>\</td><td>]</td><td>^</td><td>`</td><td>{</td><td>|</td><td>}</td><td>~</td></tr>
<tr><td>4</td><td>United Kingdom</td><td>1E</td><td>£</td><td>$</td><td>@</td><td>[</td><td>\</td><td>]</td><td>^</td><td>`</td><td>{</td><td>|</td><td>}</td><td>‾</td></tr>
<tr><td>69</td><td>French</td><td>1F</td><td>£</td><td>$</td><td>à</td><td>°</td><td>ç</td><td>§</td><td>^</td><td>µ</td><td>é</td><td>ù</td><td>è</td><td>‾</td></tr>
<tr><td>21</td><td>German</td><td>1G</td><td>#</td><td>$</td><td>§</td><td>Ä</td><td>Ö</td><td>Ü</td><td>^</td><td>`</td><td>ä</td><td>ö</td><td>ü</td><td>ß</td></tr>
<tr><td>15</td><td>Italian</td><td>0I</td><td>£</td><td>$</td><td>§</td><td>°</td><td>ç</td><td>é</td><td>^</td><td>ù</td><td>à</td><td>ò</td><td>è</td><td>ì</td></tr>
<tr><td>11</td><td>Swedish for Names</td><td>0S</td><td>#</td><td>¤</td><td>É</td><td>Ä</td><td>Ö</td><td>Å</td><td>Ü</td><td>é</td><td>ä</td><td>ö</td><td>å</td><td>ü</td></tr>
<tr><td>17</td><td>Spanish</td><td>2S</td><td>£</td><td>$</td><td>§</td><td>¡</td><td>Ñ</td><td>¿</td><td>^</td><td>`</td><td>°</td><td>ñ</td><td>ç</td><td>‾</td></tr>
<tr><td>60</td><td>Norwegian version 1</td><td>0D</td><td>#</td><td>$</td><td>@</td><td>Æ</td><td>Ø</td><td>Å</td><td>^</td><td>`</td><td>æ</td><td>ø</td><td>å</td><td>‾</td></tr>
<tr><td>2</td><td>Int'l. Ref. Version*</td><td>2U</td><td>#</td><td>¤</td><td>@</td><td>[</td><td>\</td><td>]</td><td>^</td><td>`</td><td>{</td><td>|</td><td>}</td><td>‾</td></tr>
<tr><td>25</td><td>French*</td><td>0F</td><td>£</td><td>$</td><td>à</td><td>°</td><td>ç</td><td>§</td><td>^</td><td>`</td><td>é</td><td>ù</td><td>è</td><td>‾</td></tr>
<tr><td></td><td>HP German*</td><td>0G</td><td>£</td><td>$</td><td>§</td><td>Ä</td><td>Ö</td><td>Ü</td><td>^</td><td>`</td><td>ä</td><td>ö</td><td>ü</td><td>ß</td></tr>
<tr><td>14</td><td>JIS ASCII*</td><td>0K</td><td>#</td><td>$</td><td>@</td><td>[</td><td>¥</td><td>]</td><td>^</td><td>`</td><td>{</td><td>|</td><td>}</td><td>‾</td></tr>
<tr><td>57</td><td>Chinese*</td><td>2K</td><td>#</td><td>¥</td><td>@</td><td>[</td><td>\</td><td>]</td><td>^</td><td>`</td><td>{</td><td>|</td><td>}</td><td>‾</td></tr>
<tr><td>10</td><td>Swedish*</td><td>3S</td><td>#</td><td>¤</td><td>@</td><td>Ä</td><td>Ö</td><td>Å</td><td>^</td><td>`</td><td>ä</td><td>ö</td><td>å</td><td>‾</td></tr>
<tr><td></td><td>HP Spanish*</td><td>1S</td><td>#</td><td>$</td><td>@</td><td>¡</td><td>Ñ</td><td>¿</td><td>°</td><td>`</td><td>{</td><td>ñ</td><td>}</td><td>‾</td></tr>
<tr><td>85</td><td>Spanish*</td><td>6S</td><td>#</td><td>$</td><td>·</td><td>¡</td><td>Ñ</td><td>Ç</td><td>¿</td><td>`</td><td>´</td><td>ñ</td><td>ç</td><td>‾</td></tr>
<tr><td>16</td><td>Portuguese*</td><td>4S</td><td>#</td><td>$</td><td>§</td><td>Ã</td><td>Ç</td><td>Õ</td><td>^</td><td>`</td><td>ã</td><td>ç</td><td>õ</td><td>°</td></tr>
<tr><td>84</td><td>Portuguese*</td><td>5S</td><td>#</td><td>$</td><td>´</td><td>Ã</td><td>Ç</td><td>Õ</td><td>^</td><td>`</td><td>ã</td><td>ç</td><td>õ</td><td>‾</td></tr>
<tr><td>61</td><td>Norwegian version 2*</td><td>1D</td><td>§</td><td>$</td><td>@</td><td>Æ</td><td>Ø</td><td>Å</td><td>^</td><td>`</td><td>æ</td><td>ø</td><td>å</td><td>|</td></tr>
</table>

ISO-646

- Not possible to mix languages in a single file with ISO-646
- Loss of punctuation characters a serious drawback:

  ```
  print(word, fdist[word], '\n')
  ```

  becomes in German:

  ```
  print(word, fdistÄwordÜ, 'Ön')
  ```

- ISO 646 is a seven bit code, but virtually all computers since the 1960’s work with multiples of eight bits

ISO-8859-1

- ISO-8859-1 is an 8 bit code that extends ASCII to use all 2^8=256 positions
- First 7 bits (codes 0–127) are the same as US ASCII
- adds accented and non-Latin characters in codes 128–255
- By convention, lines in text files are separated by LF (10), CR (13) or both
- SPACE (32) and TAB (9) characters used for spacing, indentation, etc

ISO-8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
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

ISO-8859

- ISO-8859-1 (aka “Latin1”) is backwards compatible with US ASCII and covers most Western European languages
- additional code tables added for other languages:

ISO-8859-1	Latin1 (W. Europe)	ISO-8859-8	Hebrew
ISO-8859-2	Latin2 (E. Europe)	ISO-8859-9	Latin5 (Turkish)
ISO-8859-3	Latin3 (Esperanto, Maltese)	ISO-8859-10	Latin6 (Nordic)
ISO-8859-4	Latin4 (N. Europe)	ISO-8859-11	Thai
ISO-8859-5	Cyrillic	ISO-8859-13	Latin7 (Baltic)
ISO-8859-6	Arabic	ISO-8859-14	Latin8 (Celtic)
ISO-8859-7	Greek	ISO-8859-15	Latin9 (W. Europe)

- Accented characters do not fit into blocks or sequence
- Some languages use more than 128 non-ASCII characters

CJKV scripts

- Character set vs. coded character set vs. encoding
- For all encodings, backwards compatibility with US ASCII is vital
- Some contexts assume eight bit characters, some seven
- Encodings:
  - ISO-2022 represents non-ASCII characters using two seven/eight bit codes
  - Extended Unix Codes (EUC) uses same strategy as ISO-8859

Problems

- Many, many other encodings, often specific to particular software
- Only languages which share an encoding can be included in the same document
- Not all encodings are easily distinguishable
- Searching and indexing collections of documents with multiple character encodings is impossible
- Not all alphabets use the same collating sequence or text direction
- Stateful encodings
- Mapping between upper and lower case is difficult

Unicode (ISO-10646)

- Extends ISO-8859-1 to 31 bits (2^{31}= more than two billion, space for every letter from every writing system ever)
- First 256 positions are the same as ISO-8859-1
- Only 16 bits (2^{16}=65,534 positions) used at first, with codes for most current world scripts plus numbers, punctuation, diacritics, mathematical symbols, IPA, dingbats, arrows, Braille patterns, Kangxi radicals, etc.
- Now uses 21 bits (more than two million positions), with additional positions used for specialist scripts like hieroglyphics and cuneiform
- Defines a standard name and code for every character

Unicode (ISO-10646)

- Codes are organized into **blocks** and **scripts**
- Includes composing characters that allow many more characters to be constructed but introduce ambiguities (e.g., Ä could be encoded as the single character U+00C4 or the sequence U+0041 U+0308)
- Extensive character database
    - Case mappings
    - Collation orders
    - Normalization tables
    - Properties: Letter (uppercase, lowercase, titlecase, other), Marks, Number, Punctuation, Symbol, Separator, Other

Unicode (ISO-10646)

- Offers round trip compatibility with existing national character sets
- Defines standard methods for collation and case folding
- Includes support for multiple text directions (left to right and right to left, but not top to bottom, bottom to top, or bustrophedonic)
- Space reserved for private use characters (Vai, Ethiopic, Klingon)
- Assigns a code position to each character, but doesn’t impose an encoding: two simple representations are UCS-2 (aka UTF-16) and UCS-4 (aka UTF-32), just the first 16 or 32 bits of the Unicode character code

UTF-8

- Unicode solves some of the problems of ISO-8859, but:
  - It doubles the size of characters: *q* is 71 in ISO-8859, U+0071 in UTF-16 and U+00000071 in UTF-32
  - Programs written for 8 bit characters may misinterpret parts of 16 bit characters: e.g., 00 may mean end of string

UTF-8

- One solution is UTF-8:
  - UTF-8 is backwards compatible with ISO-8859-1, can represent all 32 bits of UCS-4, and preserves the UCS-4 collating sequence.

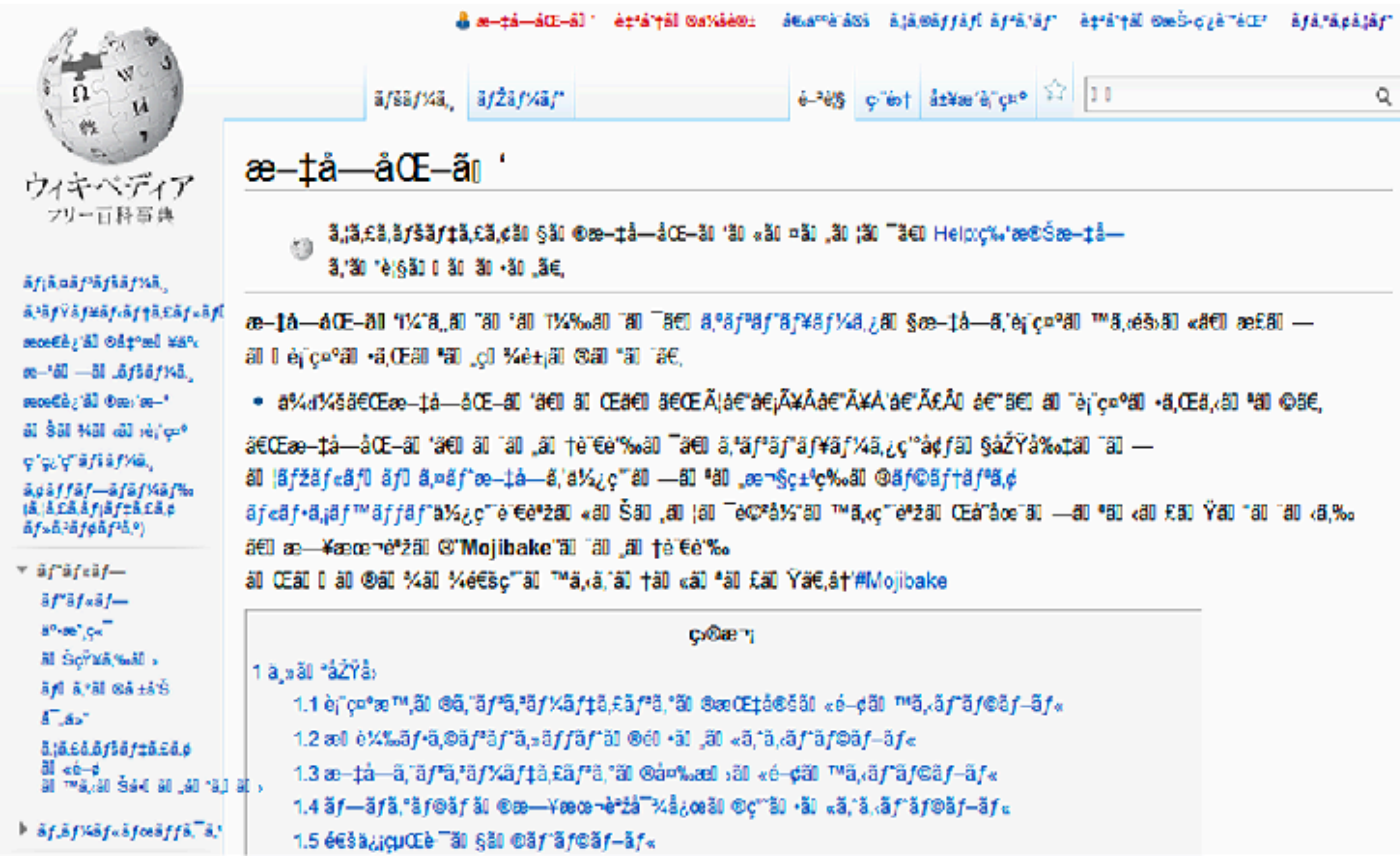
Bits of code point	First code point	Last code point	Bytes in sequence	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+0000	U+007F	1	0xxxxxxx					
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx				
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx			
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+200000	U+3FFFFFF	5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+4000000	U+7FFFFFFF	6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8

- UTF-8 is the default encoding for most systems now
- Plain text files usually use LF (`\n`) to mark end of line and SPACE and TAB (`\t`) for spacing
- Python3 (but not Python2!) uses Unicode internally with UTF-8 as the default input and output encoding
- If you are creating the file or if it's well documented, all is well
- Different encodings can be specified on input and output
- Convert Latin1 to EBCDIC

```
for line in open('infile.txt', encoding='latin1'):
    print(line, encoding='cp500')
```

Mojibake



Mojibake

- Even modern files in Latin scripts run into problems
- Smart quotes and dashes

—	→	â€”
“	→	â€œ
”	→	â€\x9d

- Accents

schön	→	schÃ¶n

- Ligatures

ﬂubberiﬁc	≠	flubberific

Mojibake

- Some problems are obvious, though may require knowledge of the (human) language
- In Python3
  - chardet module guesses what encoding a file is in

    https://chardet.readthedocs.io/
  - ftfy uses heuristics to fix common problems

    http://ftfy.readthedocs.io/
  - general Unicode support

    https://docs.python.org/3/howto/unicode.html

Text

- So, “plain” text is text in some encoding (usually UTF-8) with some line break code (usually \n)
  - Create or edit using TextEdit, Notepad, etc.
  - Read and write with Python’s built in functions (read, print)
  - Lingua franca format – convert from whatever to plain text for further processing

Text

- Other common “plain” formats for tabular data
- CSV (Comma-separated values)

```
Title,Author,ISBN13,Pages
1984,George Orwell,978-0451524935,268
Animal Farm,George Orwell,978-0451526342,144
Brave New World,Aldous Huxley,978-0060929879,288
Fahrenheit 451,Ray Bradbury,978-0345342966,208
Jane Eyre,Charlotte Brontë,978-0142437209,532
Wuthering Heights,Emily Brontë,978-0141439556,416
Agnes Grey,Anne Brontë,978-1593083236,256
Walden,Henry David Thoreau,978-1420922615,156
Walden Two,B. F. Skinner,978-0872207783,301
"Eats, Shoots & Leaves",Lynne Truss,978-1592400874,209
```

- Tab separated files use TAB character (\t) instead of comma
- Create and editing using Excel, etc
- In Python, use csv module or pandas

Text

- Rich Text Format (RTF)
- Microsoft proprietary formats
    - Word (.doc, .docx)
    - Excel (.xls, .xlsx)
    - Powerpoint (.ppt, .pptx)
- Convert to plain text using “Export” functions
- Convert Word to Text using catdoc
- Read Excel files using pandas

Text

- Searchable PDFs can be converted to text using pdftotext
- Non-searchable PDFs need to be made searchable first (Adobe Acrobat Pro)
- Ebooks (mobi, epub djvu) via Calibre
- More esoteric task-specific formats
    - Email: rfc822, outlook
    - Geodata: ARC shapefiles
- Web: HTML
- Serialization
    - XML, JSON