How Did  Get Into My Computer?

David Branner

CSTUY
St. Joseph's College, Brooklyn
2014.07.22

How does something like 鬻 get into the computer?

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

follow simple instructions, such as

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

follow simple instructions, such as:

simple math

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

- follow simple instructions, such as:
  - simple math
  - looking things up

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

- follow simple instructions, such as:
  - simple math
  - looking things up
  - moving things around

How does something like 鬱 get into the computer?

How does a computer deal with something like Chinese?

What can computers do?

- follow simple instructions, such as:
  - simple math
  - looking things up
  - moving things around
  - storing things (so they can look them up later)

How does something like 鬻 get into the computer?

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding. = System for encoding things.

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding. = System for encoding things. Each letter is assigned a number

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding. = System for encoding things. Each letter is assigned a number, and when you want the letter you work with the number. Computers can only work with numbers.

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding. = System for encoding things. Each letter is assigned a number, and when you want the letter you work with the number. Computers can only work with numbers.

Encodings are usually implemented as hash-tables

How does something like 鬱 get into the computer?

How does a letter of the alphabet get into the computer?

Encoding. = System for encoding things. Each letter is assigned a number, and when you want the letter you work with the number. Computers can only work with numbers.

Encodings are usually implemented as hash-tables — tables in which you look up a letter or other symbol using a number.

Let’s build an encoding for the alphabet.

Let’s build an encoding for the alphabet. (The Roman alphabet.)

Let's build an encoding for the alphabet. (The Roman alphabet.) How many numbers do we need for this?

Let’s build an encoding for the alphabet. (The Roman alphabet.) How many numbers do we need for this?

There are twenty-six letters. How many numbers do we need to represent them?

Suppose we use a table of twenty-six elements:

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

This is going to be our encoding system.

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

This is going to be our **encoding system**.

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

This is going to be our encoding system. We all actually use this system in daily life all the time.

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

This is going to be our encoding system. We all actually use this system in daily life all the time.

Wait — we have to zero-index.

Suppose we use a table of twenty-six elements:

1	a	7	g	13	m	19	s	25	y
2	b	8	h	14	n	20	t	26	z
3	c	9	i	15	o	21	u		
4	d	10	j	16	p	22	v		
5	e	11	k	17	q	23	w		
6	f	12	l	18	r	24	x		

This is going to be our encoding system. We all actually use this system in daily life all the time.

Wait — we have to **zero-index**.

Suppose we use a zero-indexed table of twenty-six elements:

Suppose we use a zero-indexed table of twenty-six elements:

0	a	6	g	12	m	18	s	24	y
1	b	7	h	13	n	19	t	25	z
2	c	8	i	14	o	20	u		
3	d	9	j	15	p	21	v		
4	e	10	k	16	q	22	w		
5	f	11	l	17	r	23	x		

Suppose we use a zero-indexed table of twenty-six elements:

0	a	6	g	12	m	18	s	24	y
1	b	7	h	13	n	19	t	25	z
2	c	8	i	14	o	20	u		
3	d	9	j	15	p	21	v		
4	e	10	k	16	q	22	w		
5	f	11	l	17	r	23	x		

Is that enough for the whole alphabet?

We actually need fifty-two

We actually need fifty-two: one for each lower-case letter and one for each upper-case letter.

We actually need fifty-two: one for each lower-case letter and one for each upper-case letter.

0	a	9	j	18	s	27	B	36	K	45	T
1	b	10	k	19	t	28	C	37	L	46	U
2	c	11	l	20	u	29	D	38	M	47	V
3	d	12	m	21	v	30	E	39	N	48	W
4	e	13	n	22	w	31	F	40	O	49	X
5	f	14	o	23	x	32	G	41	P	50	Y
6	g	15	p	24	y	33	H	42	Q	51	Z
7	h	16	q	25	z	34	I	43	R		
8	i	17	r	26	A	35	J	44	S		

We actually need fifty-two: one for each lower-case letter and one for each upper-case letter.

0	a	9	j	18	s	27	B	36	K	45	T
1	b	10	k	19	t	28	C	37	L	46	U
2	c	11	l	20	u	29	D	38	M	47	V
3	d	12	m	21	v	30	E	39	N	48	W
4	e	13	n	22	w	31	F	40	O	49	X
5	f	14	o	23	x	32	G	41	P	50	Y
6	g	15	p	24	y	33	H	42	Q	51	Z
7	h	16	q	25	z	34	I	43	R		
8	i	17	r	26	A	35	J	44	S		

We actually need fifty-two: one for each lower-case letter and one for each upper-case letter.

0	a	9	j	18	s	27	B	36	K	45	T
1	b	10	k	19	t	28	C	37	L	46	U
2	c	11	l	20	u	29	D	38	M	47	V
3	d	12	m	21	v	30	E	39	N	48	W
4	e	13	n	22	w	31	F	40	O	49	X
5	f	14	o	23	x	32	G	41	P	50	Y
6	g	15	p	24	y	33	H	42	Q	51	Z
7	h	16	q	25	z	34	I	43	R		
8	i	17	r	26	A	35	J	44	S		

Anything else?

We need another ten: the numerals 0 to 9. That makes sixty-two.

0	0	11	b	22	m	33	x	44	I	55	T
1	1	12	c	23	n	34	y	45	J	56	U
2	2	13	d	24	o	35	z	46	K	57	V
3	3	14	e	25	p	36	A	47	L	58	W
4	4	15	f	26	q	37	B	48	M	59	X
5	5	16	g	27	r	38	C	49	N	60	Y
6	6	17	h	28	s	39	D	50	O	61	Z
7	7	18	i	29	t	40	E	51	P		
8	8	19	j	30	u	41	F	52	Q		
9	9	20	k	31	v	42	G	53	R		
10	a	21	l	32	w	43	H	54	S		

We need another ten: the numerals 0 to 9. That makes sixty-two.

0	0	11	b	22	m	33	x	44	I	55	T
1	1	12	c	23	n	34	y	45	J	56	U
2	2	13	d	24	o	35	z	46	K	57	V
3	3	14	e	25	p	36	A	47	L	58	W
4	4	15	f	26	q	37	B	48	M	59	X
5	5	16	g	27	r	38	C	49	N	60	Y
6	6	17	h	28	s	39	D	50	O	61	Z
7	7	18	i	29	t	40	E	51	P		
8	8	19	j	30	u	41	F	52	Q		
9	9	20	k	31	v	42	G	53	R		
10	a	21	l	32	w	43	H	54	S		

One question is: how do we decide what order to put things in?

A different order:

A different order:

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k

11	l
12	m
13	n
14	o
15	p
16	q
17	r
18	s
19	t
20	u
21	v

22	w
23	x
24	y
25	z
26	A
27	B
28	C
29	D
30	E
31	F
32	G

33	H
34	I
35	J
36	K
37	L
38	M
39	N
40	O
41	P
42	Q
43	R

44	S
45	T
46	U
47	V
48	W
49	X
50	Y
51	Z
52	0
53	1
54	2

55	3
56	4
57	5
58	6
59	7
60	8
61	9

A different order:

0	a	11	l	22	w	33	H	44	S	55	3
1	b	12	m	23	x	34	I	45	T	56	4
2	c	13	n	24	y	35	J	46	U	57	5
3	d	14	o	25	z	36	K	47	V	58	6
4	e	15	p	26	A	37	L	48	W	59	7
5	f	16	q	27	B	38	M	49	X	60	8
6	g	17	r	28	C	39	N	50	Y	61	9
7	h	18	s	29	D	40	O	51	Z		
8	i	19	t	30	E	41	P	52	0		
9	j	20	u	31	F	42	Q	53	1		
10	k	21	v	32	G	43	R	54	2		

Another different order:

Another different order:

0	a	11	l	22	w	33	7	44	I	55	T
1	b	12	m	23	x	34	8	45	J	56	U
2	c	13	n	24	y	35	9	46	K	57	V
3	d	14	o	25	z	36	A	47	L	58	W
4	e	15	p	26	0	37	B	48	M	59	X
5	f	16	q	27	1	38	C	49	N	60	Y
6	g	17	r	28	2	39	D	50	O	61	Z
7	h	18	s	29	3	40	E	51	P		
8	i	19	t	30	4	41	F	52	Q		
9	j	20	u	31	5	42	G	53	R		
10	k	21	v	32	6	43	H	54	S		

Another different order:

0	a	11	l	22	w	33	7	44	I	55	T
1	b	12	m	23	x	34	8	45	J	56	U
2	c	13	n	24	y	35	9	46	K	57	V
3	d	14	o	25	z	36	A	47	L	58	W
4	e	15	p	26	0	37	B	48	M	59	X
5	f	16	q	27	1	38	C	49	N	60	Y
6	g	17	r	28	2	39	D	50	O	61	Z
7	h	18	s	29	3	40	E	51	P		
8	i	19	t	30	4	41	F	52	Q		
9	j	20	u	31	5	42	G	53	R		
10	k	21	v	32	6	43	H	54	S		

Import point: These three encodings contain all the same elements but they are not the same encoding.

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items?

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets:

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets: 3! = 6.

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets: 3! = 6. If we scramble the 62 items:

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets: 3! = 6. If we scramble the 62 items: 62!

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets: 3! = 6. If we scramble the 62 items: 62! = 31469973260387937525653122354950764088012280797258232192163168247821107200000000000000.

Import point: These three encodings contain all the same elements but they are not the same encoding.

26	q
26	A
26	0

And so on. How many possible different orders are there for these items? If we keep numerals, lower-case, and upper-case together as three whole sets: 3! = 6. If we scramble the 62 items: 62! = 31469973260387937525653122354950764088012280797258232192163168247821107200000000000000.

So when we as programmers choose an encoding, we have to stick with it.

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others.

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many **characters** do we need in all?

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

The encoding system you are most likely to encounter is ASCII ("American Standard Code for Information Interchange")

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

The encoding system you are most likely to encounter is **ASCII**

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

The encoding system you are most likely to encounter is **ASCII** ("American Standard Code for Information Interchange")

It contains 95 letters, numerals, and symbols in all, plus 33 "control characters."

Also needed are various symbols: (space), (comma), (open/close-parentheses), +, -, =, *, and a bunch of others. How many characters do we need in all?

To make a working encoding system we need to know how big a list we have to construct.

The encoding system you are most likely to encounter is ASCII ("American Standard Code for Information Interchange")

It contains 95 letters, numerals, and symbols in all, plus 33 "**control characters**."

ASCII encoding:

ASCII encoding:

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding:

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding:

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding:

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding:

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding. What is blue?

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding (blue is the 33 **control characters**):

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

ASCII encoding

Control characters are for functions:

ASCII encoding

Control characters are for functions:

8	\x08	backspace
9	\t	**T**ab
10	\n	li**N**e-feed
13	\r	carriage-**R**eturn
127	\x7f	delete

ASCII encoding (green is the symbols):

0	\x00	19	\x13	38	&	57	9	76	L	95	_	114	r
1	\x01	20	\x14	39	'	58	:	77	M	96	`	115	s
2	\x02	21	\x15	40	(	59	;	78	N	97	a	116	t
3	\x03	22	\x16	41	)	60	<	79	O	98	b	117	u
4	\x04	23	\x17	42	*	61	=	80	P	99	c	118	v
5	\x05	24	\x18	43	+	62	>	81	Q	100	d	119	w
6	\x06	25	\x19	44	,	63	?	82	R	101	e	120	x
7	\x07	26	\x1a	45	-	64	@	83	S	102	f	121	y
8	\x08	27	\x1b	46	.	65	A	84	T	103	g	122	z
9	\t	28	\x1c	47	/	66	B	85	U	104	h	123	{
10	\n	29	\x1d	48	0	67	C	86	V	105	i	124	|
11	\x0b	30	\x1e	49	1	68	D	87	W	106	j	125	}
12	\x0c	31	\x1f	50	2	69	E	88	X	107	k	126	~
13	\r	32		51	3	70	F	89	Y	108	l	127	\x7f
14	\x0e	33	!	52	4	71	G	90	Z	109	m		
15	\x0f	34	"	53	5	72	H	91	[	110	n		
16	\x10	35	#	54	6	73	I	92	\	111	o		
17	\x11	36	$	55	7	74	J	93	]	112	p		
18	\x12	37	%	56	8	75	K	94	^	113	q		

What does a **string** of text look like when it is actually ASCII-encoded?

What does a **string** of text look like when it is actually ASCII-encoded? (Ask this question using Python3.)

What does a **string** of text look like when it is actually ASCII-encoded? (Ask this question using Python3.)

```
>>> x = 'How Did x Get Into My Computer?'
```

What does a **string** of text look like when it is actually ASCII-encoded? (Ask this question using Python3.)

```
>>> x = 'How Did x Get Into My Computer?'
>>> [(i, chr(i)) for i in x.encode()]
```

What does a **string** of text look like when it is actually ASCII-encoded? (Ask this question using Python3.)

```
>>> x = 'How Did x Get Into My Computer?'
>>> [(i, chr(i)) for i in x.encode()]
```

72	H	100	d	116	t	32		109	m	63	?
111	o	32		32		77	M	112	p		
119	w	120	x	73	I	121	y	117	u		
32		32		110	n	32		116	t		
68	D	71	G	116	t	67	C	101	e		
105	i	101	e	111	o	111	o	114	r		

There is one more important question we have to ask before we ask how the Chinese character 鬱 got into my computer…

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
b'\xc3\xa9'
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
b'\xc3\xa9'

>>> [i for i in 'é'.encode()] # hex => decimal
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
b'\xc3\xa9'

>>> [i for i in 'é'.encode()] # hex => decimal
```

```
>>> B = 'How Did Beyoncé Get Into My Computer?'
>>> [(i, chr(i)) for i in B.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
66	B
101	e
121	y
111	o
110	n
99	c

195	Ã
169	©
32	
71	G
101	e
116	t
32	

73	I
110	n
116	t
111	o
32	
77	M
121	y

32	
67	C
111	o
109	m
112	p
117	u
116	t

101	e
114	r
63	?

```
>>> 'é'.encode() # char => bytestring (hex)
b'\xc3\xa9'

>>> [i for i in 'é'.encode()] # hex => decimal
[195, 169]
```

So much for Beyoncé.

So much for Beyoncé.

Now let’s ask our original question…

```
>>> h = 'How Did 釁 Get Into My Computer?'
```

```
>>> h = 'How Did 劘 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

```
>>> h = 'How Did 劈 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
233	é
135	\x87
129	\x81
32	
71	G
101	e

116	t
32	
73	I
110	n
116	t
111	o
32	

77	M
121	y
32	
67	C
111	o
109	m
112	p

117	u
116	t
101	e
114	r
63	?

```
>>> h = 'How Did 鬺 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
233	é
135	\x87
129	\x81
32	
71	G
101	e

116	t
32	
73	I
110	n
116	t
111	o
32	

77	M
121	y
32	
67	C
111	o
109	m
112	p

117	u
116	t
101	e
114	r
63	?

```
>>> '鬺'.encode() # char => bytestring (hex)
```

```
>>> h = 'How Did 鬺 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
233	é
135	\x87
129	\x81
32	
71	G
101	e

116	t
32	
73	I
110	n
116	t
111	o
32	

77	M
121	y
32	
67	C
111	o
109	m
112	p

117	u
116	t
101	e
114	r
63	?

```
>>> '鬺'.encode() # char => bytestring (hex)
b'\xe9\x87\x81'
```

```
>>> h = 'How Did 鬺 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
233	é
135	\x87
129	\x81
32	
71	G
101	e

116	t
32	
73	I
110	n
116	t
111	o
32	

77	M
121	y
32	
67	C
111	o
109	m
112	p

117	u
116	t
101	e
114	r
63	?

```
>>> '鬺'.encode() # char => bytestring (hex)
b'\xe9\x87\x81'

>>> [i for i in '鬺'.encode()] # hex => decimal
```

```
>>> h = 'How Did 鬺 Get Into My Computer?'
>>> [(i, chr(i)) for i in h.encode()]
```

72	H
111	o
119	w
32	
68	D
105	i
100	d

32	
233	é
135	\x87
129	\x81
32	
71	G
101	e

116	t
32	
73	I
110	n
116	t
111	o
32	

77	M
121	y
32	
67	C
111	o
109	m
112	p

117	u
116	t
101	e
114	r
63	?

```
>>> '鬺'.encode() # char => bytestring (hex)
b'\xe9\x87\x81'

>>> [i for i in '鬺'.encode()] # hex => decimal
[233, 135, 129]
```

Another question we can ask about a character:

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
```

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
```

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

This assumes there is one standard sequence of characters — all characters.

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

This assumes there is one standard sequence of characters — all characters.

That standard sequence of characters is called Unicode (late 1980s).

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

This assumes there is one standard sequence of characters — all characters.

That standard sequence of characters is called **Unicode** (late 1980s).

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

This assumes there is one standard sequence of characters — all characters.

That standard sequence of characters is called Unicode (late 1980s).

```
>>> ord('鬱')
```

Another question we can ask about a character:

```
>>> ord('é') # What is the ordinal value?
233
```

“Ordinal value”: the position of the character “in order” in the sequence of the encoding.

This assumes there is one standard sequence of characters — all characters.

That standard sequence of characters is called Unicode (late 1980s).

```
>>> ord('鬱')
37313
```

I said that to create an encoding, we need to know how many characters we have to be able to handle.

I said that to create an encoding, we need to know how many characters we have to be able to handle.

How many Chinese characters are there?

I said that to create an encoding, we need to know how many characters we have to be able to handle.

How many Chinese characters are there?

2600-2700 needed for good command of language.

I said that to create an encoding, we need to know how many characters we have to be able to handle.

How many Chinese characters are there?

2600-2700 needed for good command of language.

At least 60000-70000 are known historically, if we ignore differences in shape.

I said that to create an encoding, we need to know how many characters we have to be able to handle.

How many Chinese characters are there?

2600-2700 needed for good command of language.

At least 60000-70000 are known historically, if we ignore differences in shape.

More of Unicode is Chinese than any other writing system!

I said that to create an encoding, we need to know how many characters we have to be able to handle.

How many Chinese characters are there?

2600-2700 needed for good command of language.

At least 60000-70000 are known historically, if we ignore differences in shape.

More of Unicode is Chinese than any other writing system! That's why 鬱 can have an ordinal value like 37313.

Before Unicode, there was chaos! (many incompatible systems)

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]
```

What about Japanese, which also uses many Chinese characters?

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]
```

What about Japanese, which also uses many Chinese characters? (They call them *kanji*, which just means “Chinese characters.”)

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]
```

What about Japanese, which also uses many Chinese characters? (They call them ***kanji***, which just means "Chinese characters.")

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]

>>> [i for i in '鬱'.encode('euc_jp')] # & JA
[238, 215]
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]

>>> [i for i in '鬱'.encode('euc_jp')] # & JA
[238, 215]

>>> [i for i in '鬱'.encode('iso2022_jp')] # JA
```

Before Unicode, there was chaos! (many incompatible systems)

```
>>> [i for i in '鬱'.encode()] # Unicode
[233, 135, 129]

>>> [i for i in '鬱'.encode('big5')] # HK, TW
[198, 93]

>>> [i for i in '鬱'.encode('gb18030')] # PRC
[225, 133]

>>> [i for i in '鬱'.encode('euc_jp')] # & JA
[238, 215]

>>> [i for i in '鬱'.encode('iso2022_jp')] # JA
[27, 36, 66, 110, 87, 27, 40, 66]
```

What about Korean?

What about Korean? (a question you should always ask when talking about East Asia, since many people only remember China and Japan)

What about Korean? People have used Chinese characters in Korea for a couple of thousand years.

What about Korean? People have used Chinese characters in Korea for a couple of thousand years. They call them *hanja*, which just means "Chinese characters."

What about Korean? People have used Chinese characters in Korea for a couple of thousand years. They call them ***hanja***, which just means "Chinese characters."

What about Korean? People have used Chinese characters in Korea for a couple of thousand years. They call them *hanja*, which just means "Chinese characters."

```
>>> [i for i in '鬱'.encode('iso2022_kr')]
```

What about Korean? People have used Chinese characters in Korea for a couple of thousand years. They call them *hanja*, which just means "Chinese characters."

```
>>> [i for i in '鬱'.encode('iso2022_kr')]
UnicodeEncodeError: 'iso2022_kr' codec can't
encode character '\u91c1' in position 0:
illegal multibyte sequence
```

What about Korean? People have used Chinese characters in Korea for a couple of thousand years. They call them *hanja*, which just means "Chinese characters."

```
>>> [i for i in '鬱'.encode('iso2022_kr')]
UnicodeEncodeError: 'iso2022_kr' codec can't
encode character '\u91c1' in position 0:
illegal multibyte sequence
```

The character 鬱 is not in the *iso2022_kr* encoding — it's not possible to represent using that encoding.

One other useful word you may hear:

One other useful word you may hear:

codec

One other useful word you may hear:

codec

- machine (or program) that can encode and decode things
- short for “coder-decoder”

Things to remember:

- encoding (encoding system)
- Roman alphabet
- zero-indexing
- characters
- ASCII
- control characters
- string
- bytestring
- hex (hexadecimal)
- ordinal value
- Unicode
- *kanji*
- *hanja*
- codec
- input method

Things to remember:

- encoding (encoding system) — for representing characters
- Roman alphabet — the alphabet we use in English
- zero-indexing — to start numbering at 0 (remember displacement)
- characters — single unit of text (not necessarily a “letter”)
- ASCII — the standard American encoding system
- control characters — “characters” representing a function
- string — sequence of characters
- bytestring — type of string used internally, non-readable
- hex (hexadecimal) — base 16 (“numerals” go from 0 to f)
- ordinal value — position “in order” in some sequence
- Unicode — a universal character-encoding system
- *kanji* — Japanese name for Chinese characters
- *hanja* — Korean name for Chinese characters
- codec — machine or program for coding/decoding
- input method — a utility for the user to generate characters

劇終