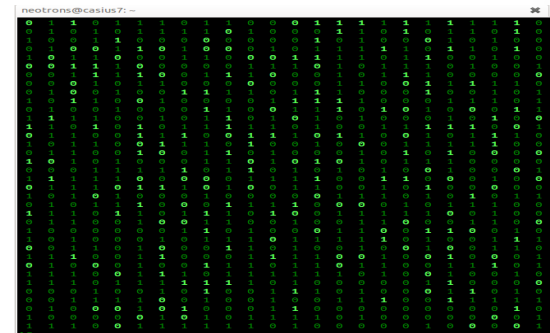
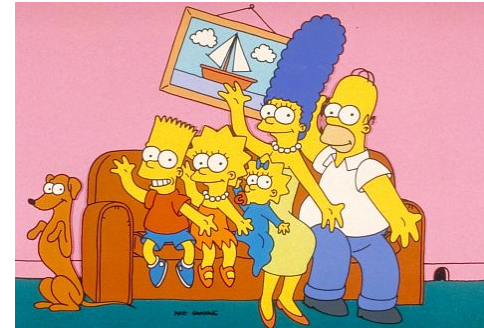


Encodings

Midterm Review

# External vs. Internal Representation

- **External representation:**
  - Convenient for programmer
  - Decimal (base 10)
- **Internal representation:**
  - Actual representation of data in the computer's memory:  
Always binary (1's and 0's)



# Binary representation (base 2)

- On a computer all data is stored in binary
- Only two symbols: 0 and 1
- Each position is called a *bit*
- *Bits take up space*
- 8 bits make a *byte*
- *Example of a 4-bit number*



- Actually the data is voltages
- We use the abstraction:
  - High voltage: 1 (true)
  - Low voltage: 0 (false)

# Positional encoding for non-negative numbers

- Each position represents some power of the base
- Decimal (Base 10), Digits (0-9)
- Binary (Base 2), Digits (0,1)
- Hex (Base 16), Digits (0-9, A-F)

# $101_5 = ?$ In decimal

A. 26

B. 51

C. 126

D. 130

# Converting between binary and decimal

Binary to decimal:  $1\ 0\ 1\ 1\ 0_2 = ?_{10}$

Decimal to binary:  $34_{10} = ?_2$

# Hex to binary

- Each hex digit corresponds directly to four binary digits
- Programmers love hex, why?

$25B_{16} = ?$  In binary

00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

# Hexadecimal to decimal

$$25B_{16} = ? \text{ Decimal}$$



# Hexadecimal to decimal

- Use polynomial expansion
- $25B_{16} = 2*256 + 5*16 + 11*1 = 512 + 80 + 11$   
 $= 603$
- Decimal to hex:  $36_{10}=?_{16}$

# Binary to hex: 1000111100

A. 8F0

B. 23C

C. None of the above

# BIG IDEA: Bits can represent anything!!

<b>Numbers</b>	<b>Binary Code</b>
----------------	--------------------

0	
---	--

1	
---	--

2	
---	--

3	
---	--

How many (minimum) bits are required to represent the numbers 0 to 3?

# What is the maximum positive value that can be stored in a byte?

- A. 127
- B. 128
- C. 255
- D. 256

# BIG IDEA: Bits can represent anything!!

**Colors**



**Binary code**

How many (minimum) bits are required to represent the three colors?

# BIG IDEA: Bits can represent anything!!

## Characters

'a'

'b'

'c'

'd'

'e'

N bits can represent at most  $2^N$  things

# BIG IDEA: Bits can represent anything!!

- Logical values?
  - 0  $\Rightarrow$  False, 1  $\Rightarrow$  True
- colors ?
- Characters?
  - 26 letters  $\Rightarrow$  5 bits ( $2^5 = 32$ )
  - upper/lower case + punctuation  $\Rightarrow$  7 bits (in 8) (“ASCII”)
  - standard code to cover all the world’s languages  $\Rightarrow$  8,16,32 bits (“Unicode”)  
[www.unicode.com](http://www.unicode.com)
- locations / addresses? commands?
- **MEMORIZE:** N bits  $\Leftrightarrow$  at most  $2^N$  things

*Red*

*Green*

*Blue*



# Encodings in Python

`ord('a')` -> returns the ascii value of 'a' (97)

`chr(97)` -> returns the character corresponding with ascii value 97

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	:	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: [www.LookupTables.com](http://www.LookupTables.com)



# Unicode

- Universal character encoding (extends ASCII to handle other languages)

```
>>> chinese = '\u4e16\u754c\u60a8\u597d!'
>>> print(chinese)
世界您好!
```

- Includes all ASCII characters using the same ascii encoding

```
>>> print('\u0048\u0049')
HI
>>>
```

---

# Midterm Review

- We'll go over a practice exam from Fall 2017:

<https://drive.google.com/file/d/1OcgXYZhtfrNQgtMdyZ1a8koHKbO9LI6c/view?usp=sharing>

Hand out:

[https://drive.google.com/file/d/1wOLtOH3vLANqUEG\\_tBmA6cl9ld2eQ0V3/view?usp=sharing](https://drive.google.com/file/d/1wOLtOH3vLANqUEG_tBmA6cl9ld2eQ0V3/view?usp=sharing)