Data Representation

15-110 – Friday 08/30

Announcements

- Check1 is due at noon Tuesday. If you forget to turn it in, you can still submit up until the revision deadline for 90% credit, but start early and ask for help on Piazza if you need it!
  - We'll try to get feedback released by the next lecture

- Note that Hw1 (due next Monday!) has a programming component. This will be completed in a separate Python file.
  - Tutorial: how to use and submit the programming starter file

- When you ask for help on Piazza, address the post to “Instructors” so that our TAs can get you the fastest possible answer!

Learning Objectives

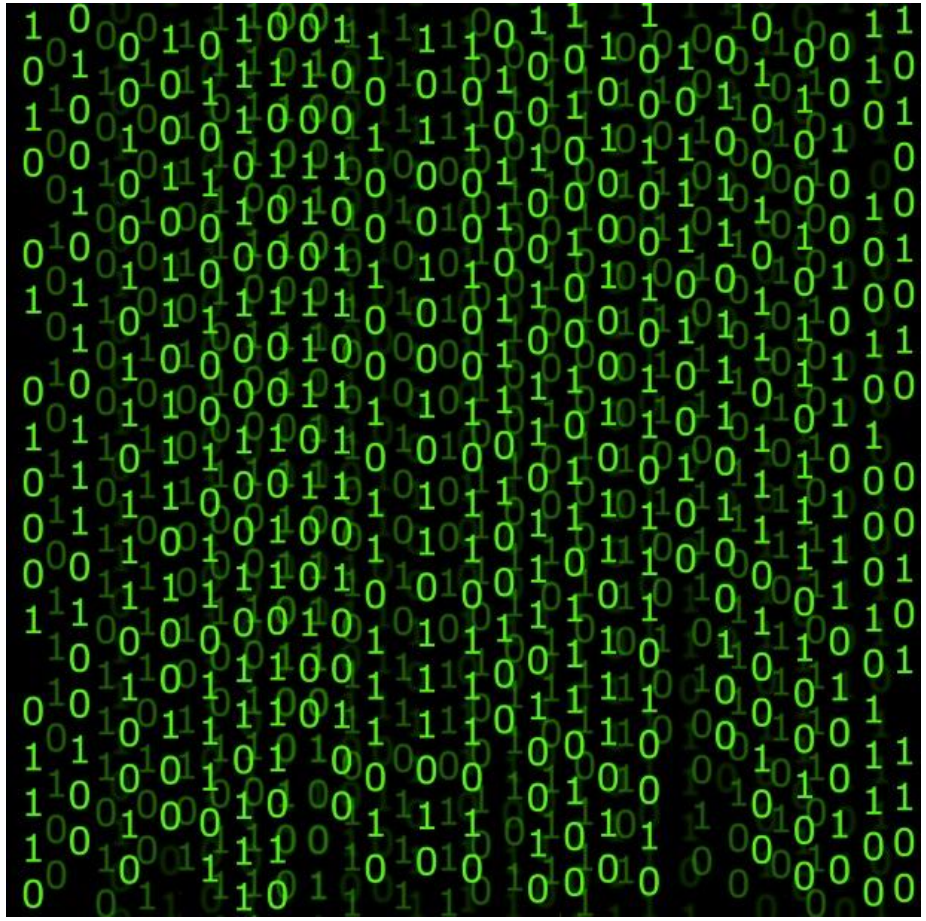
- Understand how different **number systems** can represent the same information
- Translate **binary numbers** to decimal, and vice versa
- Interpret binary numbers as abstracted types, including **colors** and **text**

Activity: Trace the Variable Values

You do: Trace through the following lines of code. What values do `a` and `b` hold at the end?

```
a = 4
b = 7
b = a - 2
a = a + 1
print(a+1)
print(b-1)
```

Computers represent everything by **0s and 1s**.



How can we use 0s and 1s to represent all types of data?

- Numbers
- Letters
- Emojis!
- Images
- Sounds
- Colors
- Videos

We will use **abstraction**! Abstraction is about **representation**: We will translate 0s and 1s to decimals and then translate them to other types.

Number Systems

A **number system** is a way of representing numbers using symbols.

Example: **Currency**

In US currency system, how much is each of the following symbols worth?

Penny
1 cent

Nickel
5 cents

Dime
10 cents

Quarter
25 cents

A **number system** is a way of representing numbers using symbols.

Example: **Dollars + Cents**

We can represent money in decimal form, with dollars and cents.

A medium coffee at La Prima Cafe is **$2.45.**

We can convert **between number systems** by translating a value from one system to the other.

Let's make a
new representation system
for coins!

To represent coins, we'll make a **number with four digits**.

$0.87

c 3 1 0 2

c.3.1.0.2

= **3***$0.25 + **1***$0.10 + **0***$0.05 + **2***$0.01

= $0.87

Conversion Example

What is `$0.59` in coin representation?

```
$0.59
= 2*$0.25 + 0*$0.10 + 1*$0.05 + 4*$0.01
= c.2.0.1.4
```

Activity: Coin Conversion

You do: Now try the following calculations:

What is **`c.1.1.1.2`** in dollars?

What is **`$0.61`** in coin representation?

When working with numbers, we usually use the **decimal number system**.

Decimal number systems uses digits: 0 1 2 3 4 5 6 7 8 9

Each digit represents a power of 10. For example 1980 in decimal:

10^3	10^2	10^1	10^0
1000	**100**	**10**	**1**
1	9	8	0

This is not the only abstract number system we can use!

We can represent numbers with the **binary number system**.

Binary number systems uses 0s and 1s.

Each digit represents a power of 2. For example 1101 in binary:

2^3	2^2	2^1	2^0
8	**4**	**2**	**1**
1	1	0	1

Counting in Binary

0 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	0	0	0

1 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	0	0	1

2 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	0	1	0

3 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	0	1	1

4 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	1	0	0

5 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	1	0	1

6 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	1	1	0

7 =

2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0	1	1	1

Converting Binary to Decimal

Algorithm: Add each power of 2 that is represented by a 1.

Example: 00011000 = **16** + **8** = 24

2^7	2^6	2^5	**2^4**	**2^3**	2^2	2^1	2^0
128	64	32	**16**	**8**	4	2	1
0	0	0	**1**	**1**	0	0	0

16 + 8 = 24

Converting Binary to Decimal

Algorithm: Add each power of 2 that is represented by a 1.

Example: 10010001 = **128** + **16** + **1** = 145

$\mathbf{2^7}$	2^6	2^5	$\mathbf{2^4}$	2^3	2^2	2^1	$\mathbf{2^0}$
128	64	32	**16**	8	4	2	**1**
1	0	0	**1**	0	0	0	**1**

128 **+ 16** **+ 1 = 145**

Converting Decimal to Binary

Algorithm: Look for the largest power of 2 that can fit in the number and subtract it from the number. Repeat with the next power of 2, etc., until you reach 0.

Example: 36 = **32** + **4** = 100100

		32			**+ 4**		**= 36**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	**32**	16	8	**4**	2	1
0	0	**1**	0	0	**1**	0	0

Converting Decimal to Binary

Algorithm: Look for the largest power of 2 that can fit in the number and subtract it from the number. Repeat with the next power of 2, etc., until you reach 0.

Example: 103 = **64** + 39 = **32** +7 = **4** + **2** + **1** = 1100111

	64	**+ 32**			**+ 4**	**+ 2**	**+ 1 = 103**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	16	8	**4**	**2**	**1**
0	**1**	**1**	0	0	**1**	**1**	**1**

Activity: Converting Binary

You do: Now try converting numbers on your own.

First: what is **01011011** in decimal?

Second: what is **75** in binary?

Abstracted Types

Using abstraction we can represent everything computers store using binary.

A single binary value is called a **bit**.

A set of 8 bits is called a **byte**.

We commonly use some number of bytes to represent data values.

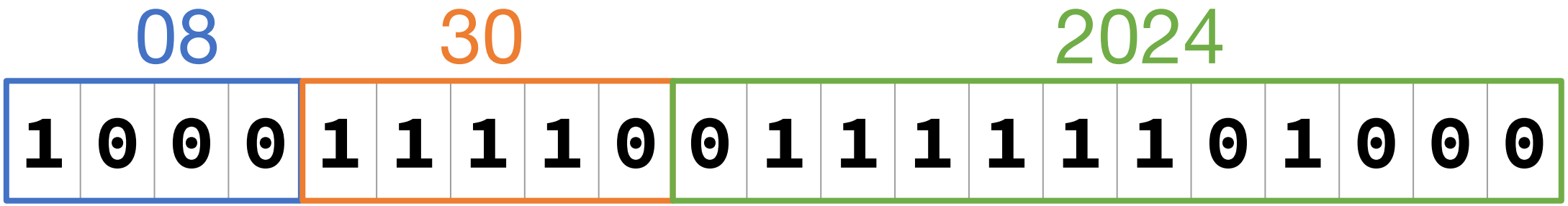
Discussion: Representing Dates

It can be helpful to think logically about how to represent a value before learning how it's done in practice. Let's do that now.

Discuss: We can convert binary directly into numbers, but how could we use binary and number to represent a date (i.e., 10/15/2023)?

Answer: Representing Dates

Simple Approach: reserve 4 bits to represent the month (1-12), 5 bits to represent the date (1-31) and 12 bits to represent the year (1-4095). Convert the month, day, year normally from decimal to binary.



Answer: Representing Dates

Actual Approach: Unix Timestamp – you count the seconds from a certain date (00:00:00 of 01/01/1970) and convert the number of seconds to binary. Any dates that occur before this time would be negative numbers, and any dates after would be positive numbers!

We use 32 bits to represent each date; the first bit is used to indicate if the number was positive (0) or negative (1), and the remaining 31 bits are used to represent the number of seconds elapsed. Thus, we restrict the number of bits to represent the date to 31 bits.

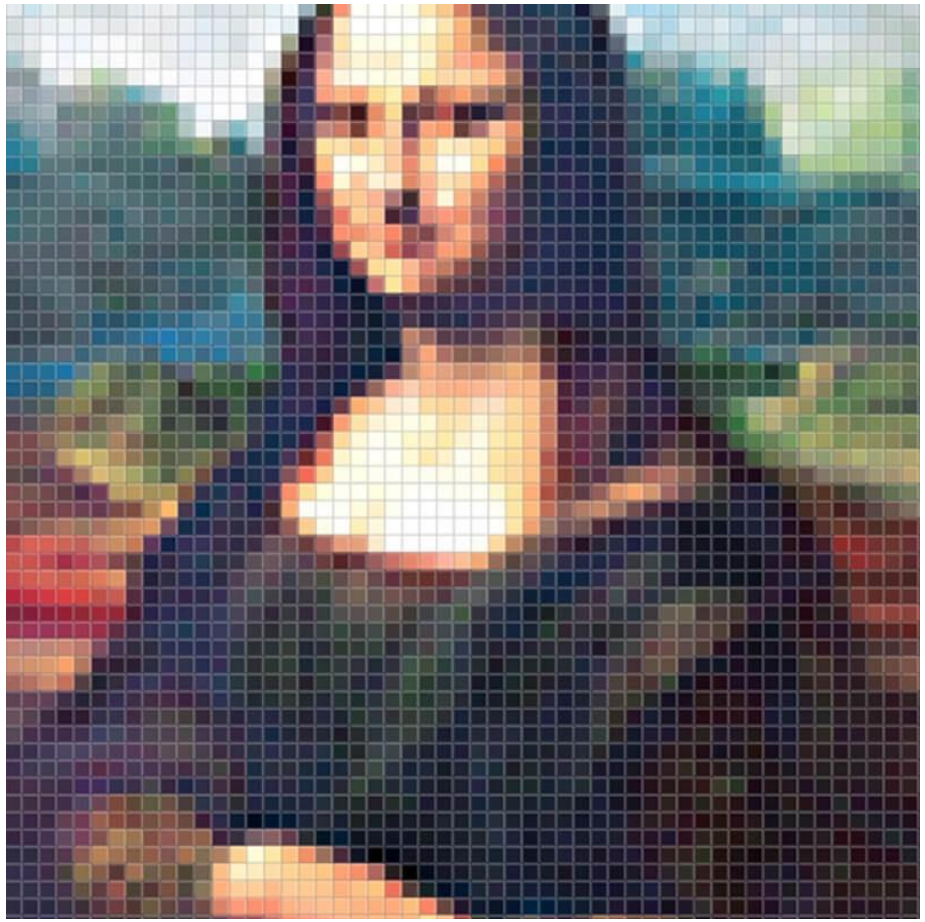
More on dates: https://busyintelligence.blog/2019/05/22/how-does-a-computer-store-data-dates/

We represent images as a **grid of colors**.

Break image into a grid of colors.

Each square of color has a distinct hue. Each square is called a **pixel**.

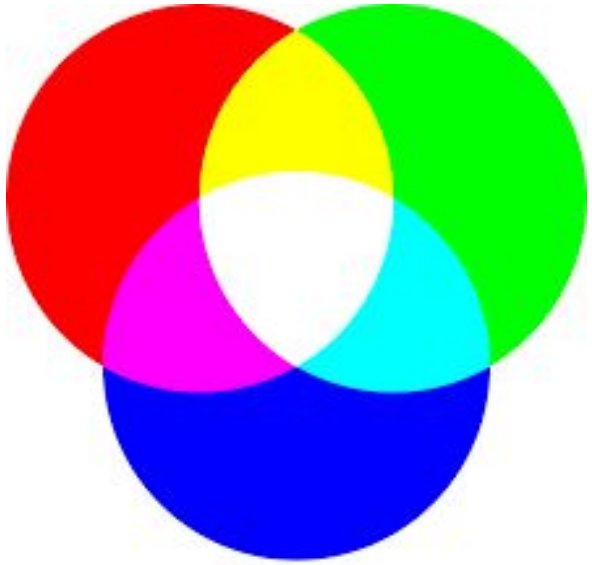
Convert colors -> numbers.



Every color can be represented as a combination of **R**ed, **G**reen, and **B**lue **(RGB)**.

Red, green, and blue intensity can be represented using one byte each, where 00000000 **(0)** is none and 11111111 **(255)** is very intense.

Each pixel will therefore require **3 bytes** to encode.



Try it out here:

https://www.w3schools.com/colors/colors_rgb.asp

Example: Representing Beige

To make the campus-building beige, we'd need:

Red = 249 = 11111001

Green = 228 = 11100100

Blue = 183 = 10110111

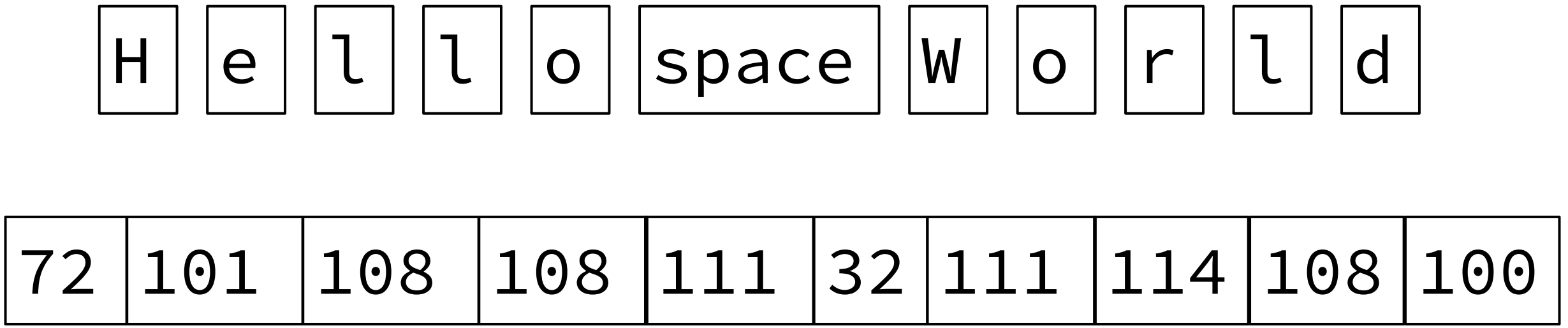
Which makes beige!

We represent text as **individual characters**.

Break text into characters.

```
Hello World
```

We us a **lookup table** to map each character to a number.



We use a **lookup table** to map each character to a number.

ASCII maps the numbers 0-255 to characters.

1 character represented by 1 byte.

Check it out here:

https://www.asciitable.com/

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

We use a **lookup table** to map each character to a number.

Y	A	Y

89	65	89

01011001	01000001	01011001

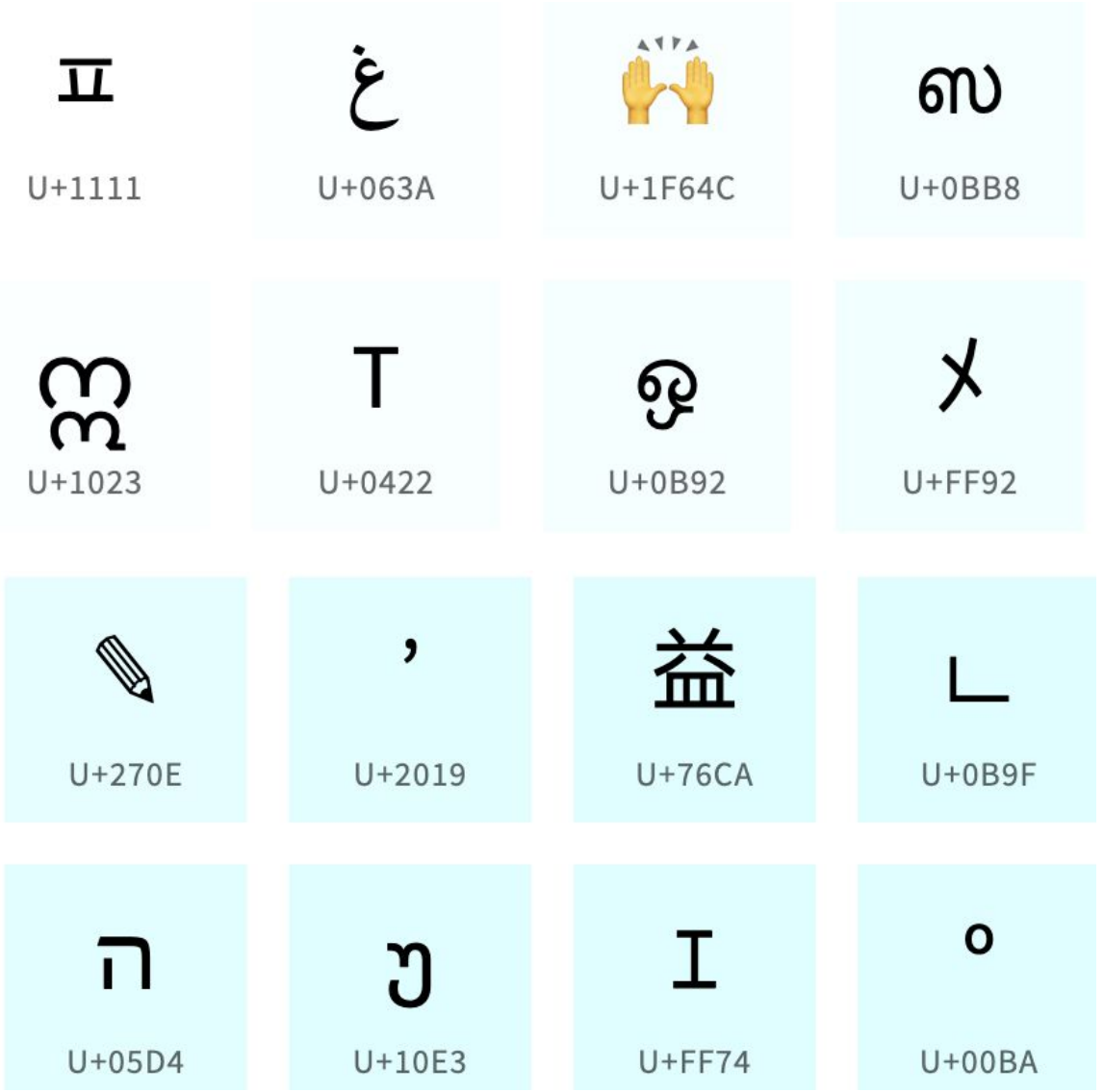
Dec	Char	Dec	Char	Dec	Char
32	SPACE	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

We use a **lookup table** to map each character to a number.

Unicode represents every character that can be typed into a computer.

It uses up to 5 bytes, which can represent up to 1 trillion characters!

It is the standard for all text on computers and is actively under development by the Unicode Consortium.



Learning Objectives

- Understand how different **number systems** can represent the same information
- Translate **binary numbers** to decimal, and vice versa
- Interpret binary numbers as abstracted types, including **colors** and **text**

Bonus Slides

In case you want to learn even more about data representation

Size of Integers

Your machine is either classified as 32-bit or 64-bit. This refers to the size of integers used by your computer's operating system.

The largest signed integer that can be represented with N bits is 2^N-1 (why?). This means that...

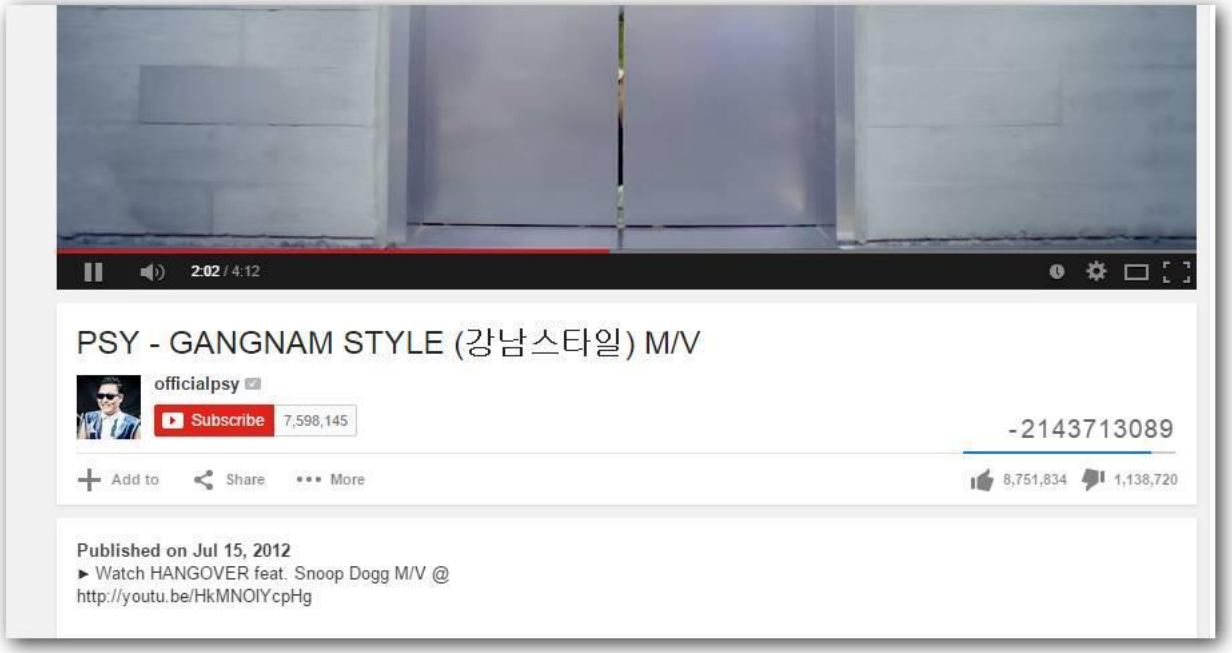
Largest int for 32 bits: 4,294,967,295 (or 2,147,483,647 with negative numbers)

Largest int for 64 bits: 18,446,744,073,709,551,615 (18.4 quintillion)

Integer Overflow

Why does this matter?

By late 2014, the music video Gangnam Style received more than 2 billion views. When it passed the largest positive number that could be represented with 32 bits, YouTube showed the number of views as negative instead!

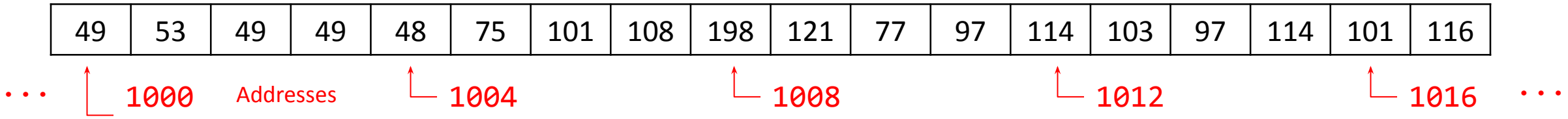


Now YouTube uses a 64-bit counter instead.

Computer Memory is Stored as Binary

Your computer keeps track of saved data and all the information it needs to run in its memory, which is represented as binary. You can think about your computer's memory as a really long list of bits, where each bit can be set to 0 or 1. But usually we think in terms of bytes, groups of 8 bits.

Every byte in your computer has an address, which the computer uses to look up its value.



Binary Values Depend on Interpretation

When you open a file on your computer, the application goes to the appropriate address, reads the associated binary, and interprets the binary values based on the file encoding it expects. That interpretation depends on the application you use when opening the file, and the filetype.

You can attempt to open any file using any program, if you convince your computer to let you try. Some programs may crash, and others will show nonsense because the binary isn't being interpreted correctly.

Example: try changing an (unimportant) .docx filetype to .txt, then open it in a plain text editor. .docx files have extra encoding, whereas .txt files use plain ASCII.

We Use Lots of Bytes!

In modern computing, we use a lot of bytes to represent information.

Smartphone Memory: 64 gigabytes = 64 billion bytes

Google databases: Over 100 million gigabytes = 100 quadrillion bytes!

CMU Wifi: 15 million bytes per second