COMP 150:
Basics of Computer Organization
Week 03, Lecture 05

Agenda

1. Review
2. Discussion
3. Web Browsers
4. Data Introduction
5. Data Operations

In Review: That was a lot.

1. Ancient computers were calculators
2. Industrial revolution pushed the technology (binary -- looms)
3. WWII forced the mathematicians and engineers into the same room
4. Tech races in the early cold war era meant more and better machines
5. Corporate needs ruled the 80s and 90s
6. From 1940 to 2000, went from:
    a. Massive machines
    b. Mainframes and terminals, supercomputers
    c. PCs and workstations, supercomputers
    d. PCs, supercomputers, and browser-based computing
7. Internet:
    a. Started as data transfer for scientists
    b. Expanded to commerce (ISPs, DNS)
    c. Expanded with the WWW (servers, clients, websites like pages of a book)
    d. Now, dynamic and interactive!

In Review: That was a lot. Discuss!

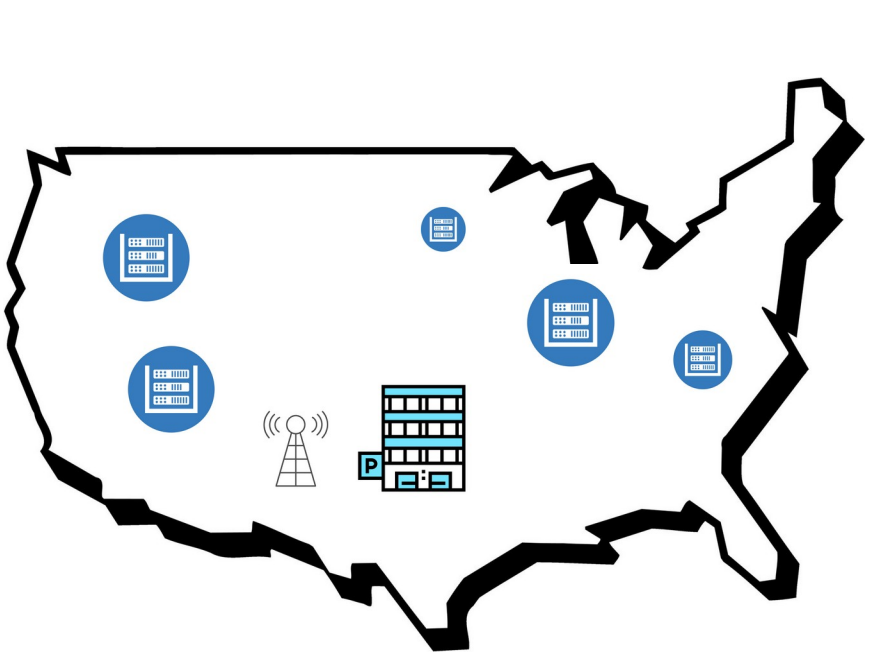
1. What were ancient computers?
    a. Calculators - business devices
2. What technological advancement made it possible for computers to get both smaller and faster?
    a. Transistors! Went from vacuum tubes (size of pinkie finger) to transistors, then smaller and smaller until 1,000 of them can fit across a red blood cell
3. Examples of computers in the last 100 years?
    a. Purpose-built rooms, mainframes, supercomputers, PCs
4. What was the original purpose of the internet?
    a. Data transfer across continents
5. What changed between ARPANET/NSFNET and the WWW?
    a. Went from simple data transfer to display of pages
6. How has the web changed in terms of interactivity?
    a. Used to simply get and give data to/from server - now, the server sends code to the browser to run interactively

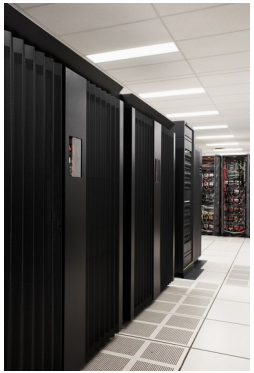
Attendance

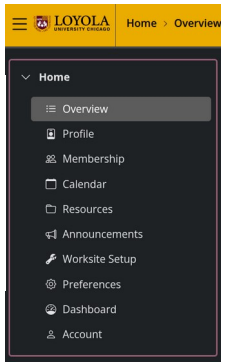
Recall: The Internet

- Communication at a distance
- Routing like mail addresses
- Send information as “packets”
- DNS acts like navigators on a map
- ISPs are like toll entries to the roads
- Changing interaction
  - Before: not-interactive, simply information transport
  - Now: server sends parts of its code for the client to run and interact with in the browser

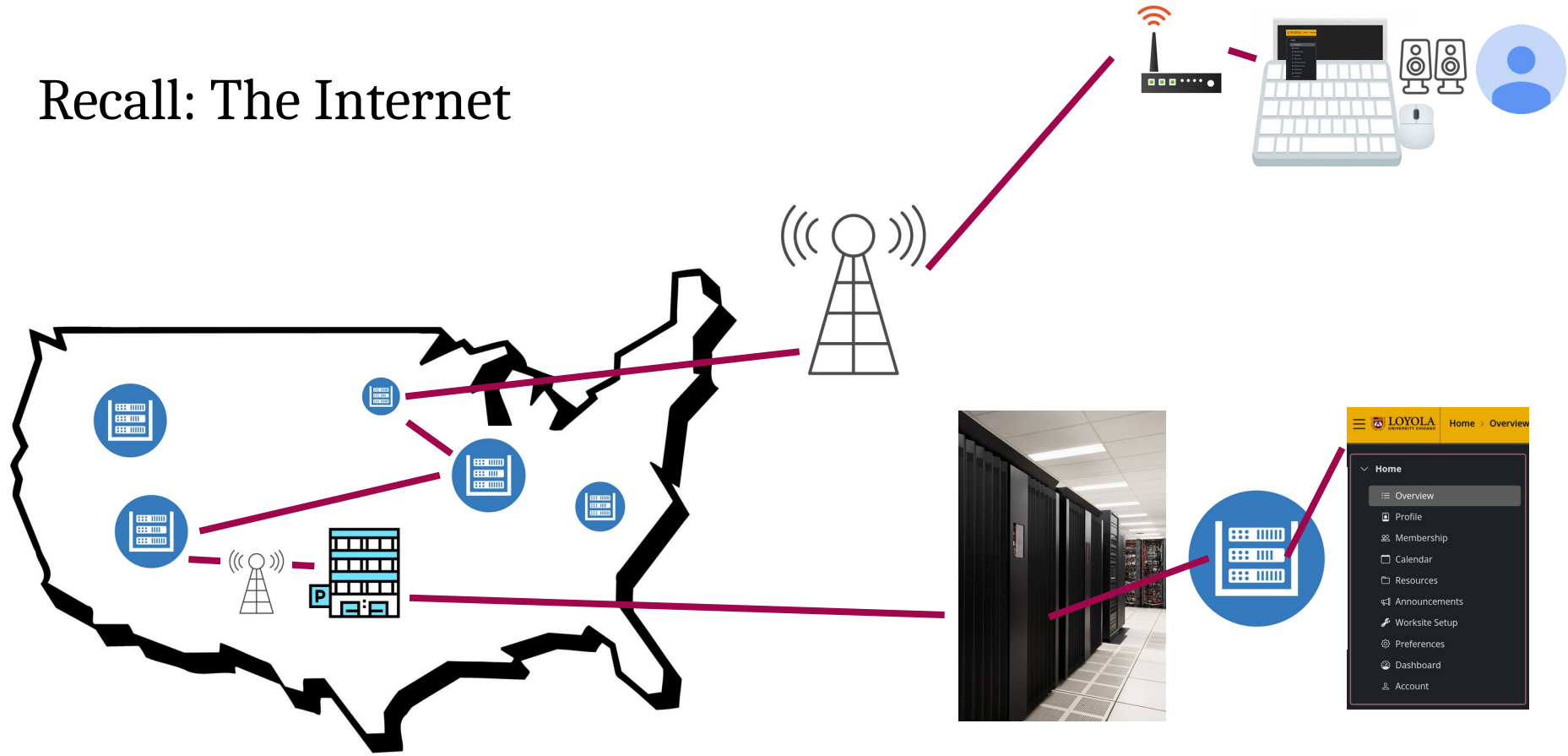
Recall: The Internet







Recall: The Internet



Web Browsers

Why do we care?

- In general: Everything is web-based
- In specific: We will be using Python in a web browser!

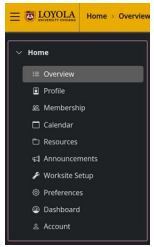
Web Browsers

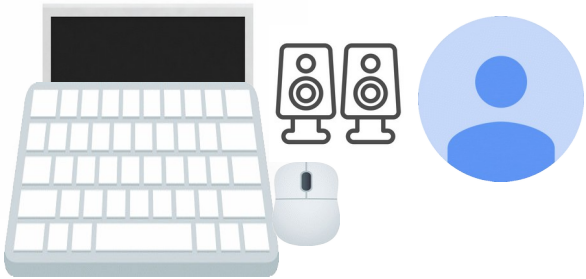
What is a browser?

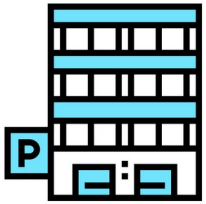
- Lets you get around (navigation)
- Sends commands to grab data from servers
- Security
- Identity
- Cookies (memory)

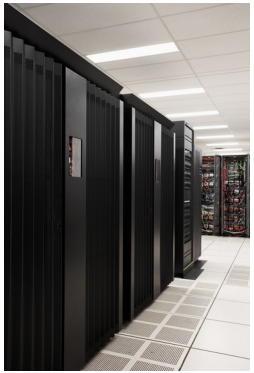
Web Browsers

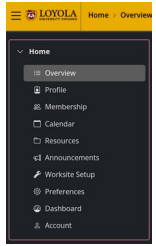
Where does the code that makes a website inside a browser “live”?











Changing Gears -- Data!

Types of Data

What might we want to store?

- Whole numbers
- Fractional numbers
- True/false
- Letters
- Groups of things
  - Ordered
  - Unordered
  - Mapped

Basic Types

- Integers (int)
  - Whole numbers
  - Binary!
- Floating points (floats)
  - Fractional numbers
  - Why? (more in a moment)
- String
  - Ordered set of alphanumerics (“characters”)
- Boolean
  - True/False

Strings

- List of alphanumeric characters
- Individual characters:
  - Represented many ways - common is ASCII
  - American Standard Code for Information Interchange
  - Chart on the right is the original 1972
  - Constantly expanding now
- Some complexities when we “string” them together (ex - when does it end?)
- Note: strings are *big*

b_7 b_6 b_5 → / Bits ↓					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	—	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Numbers

- Integers:
  - Whole
  - Positive or signed
  - Simple binary counting

What happens when we have fractional components?

- Floating point numbers:
  - Essentially - scientific notation
  - Size (board exercise)
  - Math gets *hard*
  - FLOPS

Boolean

Why are they useful?

- Tiny!
- Simple!
- Logical (more on this)

Lists

What happens when we have more than one value?

- Lists!
- Arrays in other languages
- Imply order
- In Python, do *not* have to all be the same type
- “Indexing” board exercise
    - Same types
    - Different types
    - Strings *are* lists

Dictionaries

What happens when we want to associate two values in an orderly way?

- We want to draw a “map” between two things
- In Python, called a Dictionary
  - Why?
- The “index” in a Dict is called the “key”, the return is the “value”
- Dictionaries in Python are
  - *Not* ordered
  - Extremely common
  - One of the most useful data constructs

Python Operations

```
len(list)
type(variable)

“ “.join([list, of, strings])
“{}”.format(variable)

.keys()
.values()
.items()

interactive: dir(variable)
```

What can we *do* with our data?

- Compare it
- Do math on it
- Display it
- Change it
- Delete it
- Make copies of it

Variables

"=" means "Assignment"

This is important.

= does *not* compare equality.

Arithmetic (math basics)

- "Operator"
  - The thing we're doing
- "Operands"
  - The things we're doing the operator *on*
- Order of operations:
  - PEMMDAS
- Left-associated
- Return a number

Op	Definition
+	Add
-	Subtract
*	Multiply
/	Divide (always gives a float)
%	Modulus (remainder)
**	Exponent

Compariators

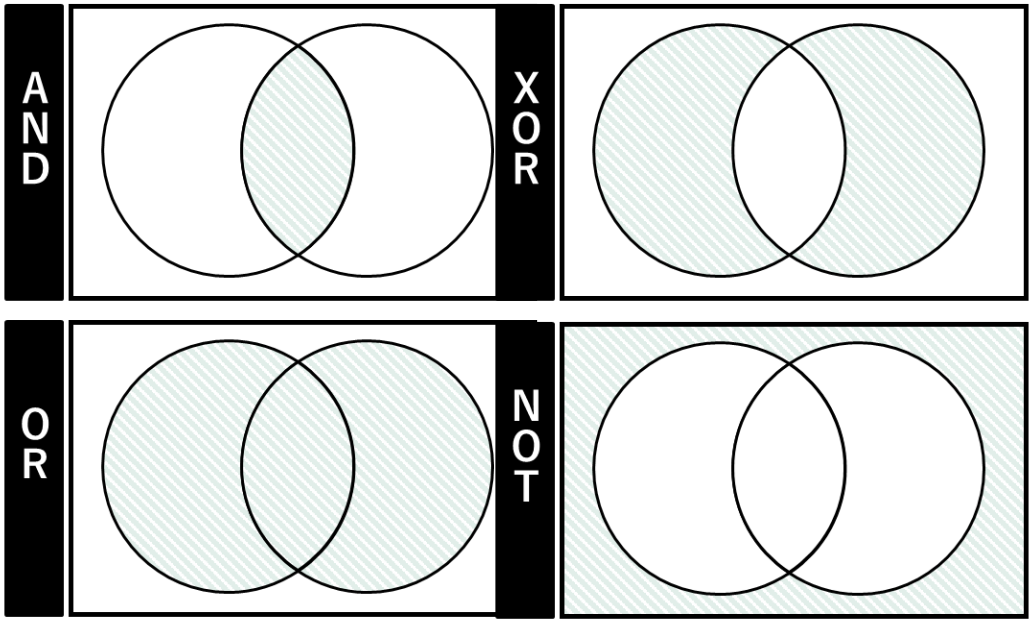
- Return True/False
- Parenthesis *matter*

Op	Meaning
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

Boolean Logic

- Return True/False

Op	Meaning
and	AND (both True)
or	OR (either is True)
not	NOT
^	XOR (only one is True)



Board examples:

```
a =     5
b =     10
c =     -10
d =     10
e =     1
g =     2
h =     0

t = True
f = False
```

```
a + b
a * b
b ** a
b ** a - c
d / g - e
d / (g - e)

a > b
b > c
a > b > g
b > a > g
(a > b) > g
(a > b) > c
```

```
h = h
h == h
g >= a
a <= c
d == c

h == f
e == t
t > f
t >= f
t > g
f > g
```

```
t & t
t | t
t | f
t & f
f & t
f ^ t
t & f | t
t & f | f

t ^ f
t ^ f | f

!t
!f
!t == f
!f > 0
```