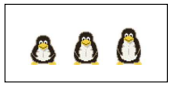
The CS 5 Times

Penguin Revenge "Rich," Gloats Computer Scientist

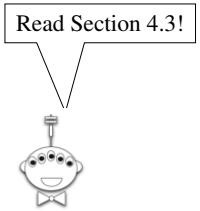


CS5 penguins chuckle at their friend's prank.

Upland (AP): A group of penguins celebrated in a beer garden here after one of their number played an irreverent prank on a rival. "You should have seen it," chirped one. "Those uppity physics rhinos think they're better than us because they have spaceships and stuff, but who writes the guidance software? Those guys are all wet—especially now!"

Burping delicately, another bird commented, "Maybe we should insert an, um, insect into their rockets. Who says every booster has to go up?"

A computer science professor in their midst was more restrained. "I did appreciate the retaliatory stunt," he commented, "but mostly because of the robust flavor of the beer involved."



Representing Data

Last time…

1. Representing numbers in different bases
2. Converting between bases
3. Arithmetic in different bases

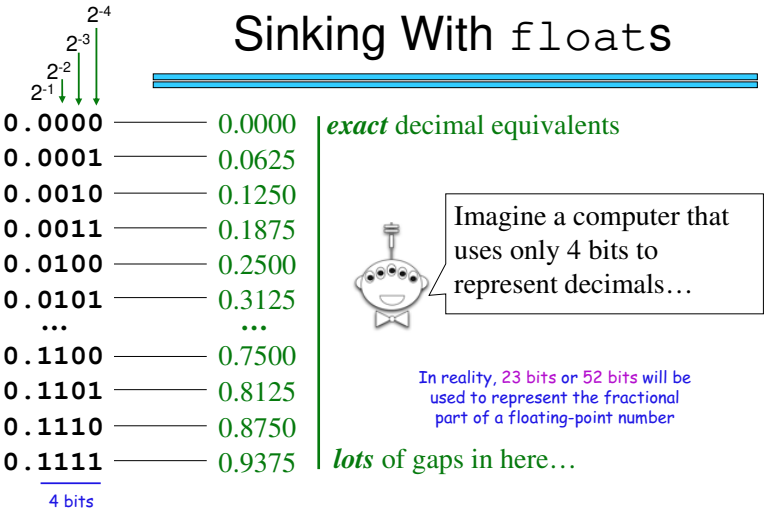
Today…

1. Representing non-integer data
2. Boolean functions
3. The Minterm Expansion Principle
4. Building an adder!

What's Up With This!?

```
>>> .01*10 == .01/.1
False
```

Sinking With `floats`




```
>>> x = 0.1
```

Representing Symbols

ASCII (American Standard Code for Information Interchange)

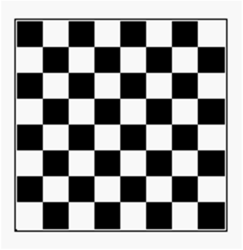
```
BINARY      HEX     DESCRIPTION
0000 0000   00      Null character
0000 0001   01      Start of header
0000 0010   02      Start of text
0000 0011   03      End of text
0000 0100   04      End of transmission
…
0000 1101   0D      Carriage return
…
0010 0000   20      Space
0010 0001   21      !
0010 0010   22      "
0010 0011   23      #
0010 0100   24      $
…
0011 0000   30      0
0011 0001   31      1
0011 0010   32      2
…
0100 0001   41      A
0100 0010   42      B
0100 0011   43      C
…
0110 0001   61      a
0110 0010   62      b
0110 0011   63      c
…
```

8 bits are called a "byte". How many different symbols can be represented with this 1-byte-per-symbol system?

In Python…

```
>>> print "\x21"
!
>>> print "\x63"
c
>>> print "\x63\x61\x62"
cab
>>> print "\u03c6"
φ
```

Binary Images



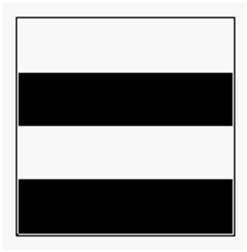
Binary Image

```
'01010101
 10101010
 01010101
 10101010
 01010101
 10101010
 01010101
 10101010'
```

Encoding as raw bits

Just one big string of 64 characters

Binary Image Compression



Binary Image

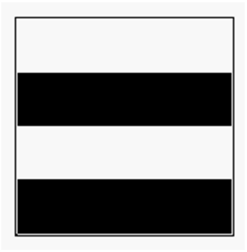
```
'11111111
 11111111
 00000000
 00000000
 11111111
 11111111
 00000000
 00000000'
```

Encoding as raw bits

Just one big string

Can we represent this more compactly?

Binary Image Compression



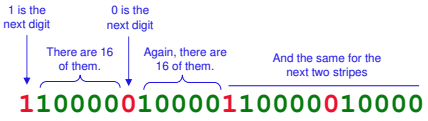
Binary Image

```
'11111111
 11111111
 00000000
 00000000
 11111111
 11111111
 00000000
 00000000'
```

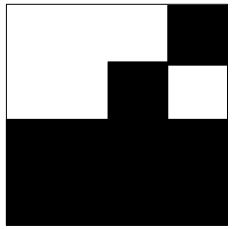
Encoding as raw bits

Just one big string

An idea:



HW4: Quadtrees!



Binary Image

```
'11111100
 11111100
 11110011
 00000000
 00000000
 00000000
 00000000'
```

Encoding as raw bits

Just one big string

New idea: [White, [W, B, B, W], Black, Black]

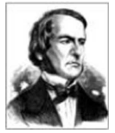
Algebra with Only 0s and 1s!

Real-valued variable

A variable assigned any real number (e.g. *x = 3.1234567)*

Boolean variable

A variable assigned either 0 or 1 (e.g. *x = 0* or *x = 1)*



George Boole
1815-1864

0 and 1 are also called "False" and "True" or "No" and "Yes"

Real Operators

\+
−
×
÷
…

Boolean Operators

NOT
AND
OR

Functions

Describing a real-valued function…

Words

f is a function of TWO real variables s.t. the output is the sum of their squares

Table

x	*y*	*f(x,y)*
0	*0*	*0*
1	*2*	*5*
1.2	*1*	*2.44*
1	*1*	*2*

Formula

$f(x) = x^2+y^2$

That table seems to be missing a "few" entries!

Boolean Functions

A Boolean function just takes Boolean arguments and gives a Boolean result.

Example: f(True, True) = True

The most common Boolean functions take 1 or 2 arguments.

There are exactly four 1-argument Boolean functions!

There are 16 two-argument functions, but only 5 are commonly used!

Boolean Functions

Describing a Boolean function (inputs and outputs: 0 and 1)

Words

f is a function of TWO binary (Boolean) variables s.t. the output is *1* if and only if exactly one of the two inputs is *1*

Table

x	*y*	*f(x,y)*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Formula

?

NOT, AND, OR

x	NOT *x*
0	1
1	0

Also written $\overline{x}$

x	*y*	*x* AND *y*
0	0	0
0	1	0
1	0	0
1	1	1

Also written *xy*

x	*y*	*x* OR *y*
0	0	0
0	1	1
1	0	1
1	1	1

Also written *x+y*

Playing with Functions…

Describe these functions in English:

xx $x\overline{x}$ $x+\overline{x}$ $(xy + \overline{x}\overline{y})$

Worksheet!

Playing with Functions…

How about Boolean formulae ("formulas") for:

- A function of two variables *x,y* that evaluates to 1 iff *x* and *y* are not equal
- A function of two variables *x,y* that evaluates to 1 iff $x \geq y$

XOR

x	*y*	*x* XOR *y*
0	0	0
0	1	1
1	0	1
1	1	0

Python uses ~, &, |, and ^ to represent NOT, AND, OR, and XOR, respectively.

Properties of Boolean Functions

All the "usual" Boolean functions commute:

f(x, y) = f(y, x)

AND, OR, and XOR associate:

f(f(x, y), z) = f(x, f(y, z))

e.g., (x AND y) AND z = x AND (y AND z)

What about the *Un*usual ones?

The Alien's Life Advice

Don't get bogged down in a single project.

Get bogged down in at least two!

Digital Logic Gates

x	NOT *x*
0	1
1	0

Also written $\overline{x}$

NOT is often shown as just the small circle on another gate.

x	*y*	*x* AND *y*
0	0	0
0	1	0
1	0	0
1	1	1

Also written *xy*

x AND *y*

x	*y*	*x* OR *y*
0	0	0
0	1	1
1	0	1
1	1	1

Also written *x+y*

x OR *y*

Finding the Formula!

The Minterm Expansion Principle

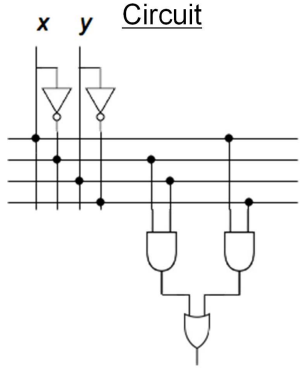
Consider this function…

Words	Truth Table			Formula
	x	*y*	*x* XOR *y*	
A function of TWO binary inputs *x,y* where the output is 1 iff $x \neq y$	0	0	0	
	0	1	1	$\bar{x}y$
	1	0	1	$x\bar{y}$
	1	1	0	

$$f(x,y) = \bar{x}y + x\bar{y}$$

From Formula to Circuit!

Words	Table			Formula
	x	*y*	*f(x,y)*	$\bar{x}y + x\bar{y}$
f is a function of TWO binary (Boolean) variables s.t. the output is *1* if and only if exactly one of the two inputs is *1*	*0*	*0*	*0*	Circuit
	0	*1*	*1*	
	1	*0*	*1*	
	1	*1*	*0*	



You Try It!

The Minterm Expansion Principle

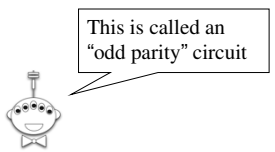
Consider this function…

Words	Truth Table	Formula
A function of TWO binary inputs *x,y* where the output is 1 iff $x \geq y$		Circuit

Try This One…

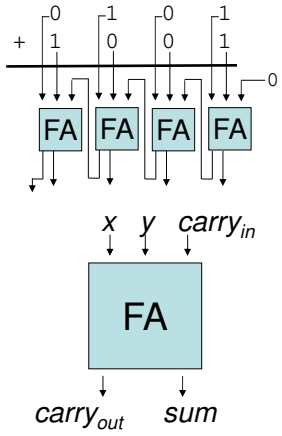
Consider this function…

Words	Truth Table	Formula
A function of THREE binary inputs *x,y,z* where the output is 1 iff the number of 1's is odd		Circuit



A Circuit for Adding!

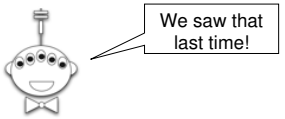
Base 2 Addition



x	*y*	*carry*$_{in}$	*sum*	*carry*$_{out}$
0	0	0	0	0
0	0	1	1	0

A Cool Thing About XOR

x XOR y XOR y = x



Try this on for size:

```
x = x ^ y
y = x ^ y
x = x ^ y
```

A Cool Thing About XOR

Try this on for size:

```
x = x ^ y
y = x ^ y
x = x ^ y
```

Suppose x, y =	0,0	0,1	1,0	1,1
Now x,y =	0,0	1,1	1,0	0,1
	0,0	1,0	1,1	0,1
	0,0	1,0	0,1	1,1

A Cool Thing About XOR

We can prove this if we call the original values x_0 and y_0:

```
x = x ^ y       x1 = x0 ^ y0
y = x ^ y       y1 = x1 ^ y0 = (x0 ^ y0) ^ y0
x = x ^ y       x2 = x1 ^ y1
```

A Cool Thing About XOR

We can prove this if we call the original values x_0 and y_0:

```
x = x ^ y        x1 = x0 ^ y0
y = x ^ y        y1 = x1 ^ y0 = x0
x = x ^ y        x2 = x1 ^ y1 = (x0 ^ y0) ^ x0
```