What did you think of Lightbot?
As Experienced Lightbot Hackers

- What are you doing in Lightbot?

- Commanding a robot through a “blocks world”
- Programming is **commanding** an agent
Agent, Instructions, Intent

- Other aspects of “commanding”
  - The **agent** is usually a computer, but it could be a person, or other device (e.g. a robot)
  - The agent follows the commands, the **instructions**, without making a mistake, and without stopping, doing only what it is asked
  - The program implements your intentions – it does what you tell it to do
  - **You** try to get the robot to the Blue Tile goal – that’s the point of the instructions that you gave it
Sequencing

- Instructions are *given* in sequence
- They are *executed* in sequence – essential
  - Instructions ...
    - From a limited repertoire
    - All are within agent’s ability; no JUMP_3
    - Executed one-at-a-time
  - A “program counter” keeps track of agent’s progress
Instructions Formed of Simpler Instructions

- Check out this screen shot of the Lightbot
- It is partway through an instruction ... its beacon is lit, but not the tile
- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

An Instruction *abstracts* those steps
Abstraction

- The word “abstraction” is used a lot in computing
- Remember: it was one of the 7 big ideas
- *Abstraction is a way to understand and solve problems*
- As a general definition, abstraction eliminates details to focus on essential properties
- The instruction example just given illustrates *functional abstraction* meaning that we have given a name to a series of operations that perform a coherent (and to us meaningful) activity; the name is the instruction, the series of operations are the bot’s actions to implement it
Abstracting

- Collecting the operations together and giving them a name is *functional abstraction*
  - The group of operations perform some function but we ignore all of the details
  - Giving it a name is *functional* abstraction
  - This is AMAZINGLY powerful
  - What makes it powerful, is we can forget about the operations and think only about the function they do; more about this later

- Let’s do some functional abstraction
Example: Abstraction in Everyday Life

Get Dressed

Dress Bottom Half
  - Put on Sox

Dress Top Half
  - Put on Pants
  - Put on Shirt
Functions abstract by packaging Computation

- \( F_1() \) packages actions: E.G. “process a riser”
The Function Becomes A Concept

- Because $F_1()$ “processes a riser,” I think of the programming task as

  - Process a riser
  - Move to next riser
  - Process a riser
  - Move to next riser
  - Process a riser

- With $F_1()$ as a concept, I simplify the programming to just 5 steps rather than 21.

- It also suggests another concept:
  - Move_to_next_riser()
A Five Instruction Program: Define F1, F2, interleave

New way of thinking about the problem
Did everyone see 1 idea, 2 applications?

Slide 8

- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

Slide 11

F1( ): Process Riser
F2( ): Move To Next Riser

It is one concept here (monolithic, but here it is a series of eight instructions)
Abstraction …

- Formulating blocks of computation as a “concept” is functional abstraction
- What we did is important and we do it all the time in everyday life (otherwise we couldn’t deal with life’s complexity)...
  - We break a task down into (one or more) subtasks
  - We solve a subtask using a sequence of instructions
  - We put the solution into a function “package”, gave it a name, “process a riser,” and thus created a new thing, a concept, something we can talk about & use
  - Then we used it to solve something more complicated … and probably repeat this approach at the next higher level
- This lets us do more complicated things
It only simplifies *our thinking*, the bot still does all the work.
First Try: Recursion 2
Second Try: Recursion 2. Just keeps going....
Third Try: Recursion 2: Finds stop condition!
Programming Rule 1: Try, try again!!

It only does what you tell it to do
Recursion 3!: First Try
Recursion 3!: First Try
Recursion 3: OOPS!! Second Try
Recursion 3: Third Try...... Wow!
Recursion 4: Whoa!!! What is F2 subroutine?
What do you think?
Recursion 4: Whoa!!! What is F2 subroutine?
Recursion 4: Whoa!!! What is F2 subroutine?
Recursion 4: We keep trying
The Function Is Just The Packaging

- Another way to use a function for the risers
Summary From Lightbot 2.0

- Programming is **commanding** an agent
  - **Agent:** usually a computer, person, or other device
  - Agent follows **instructions**, flawlessly & stolidly
  - The program implements human intent

- Instructions are *given* in sequence

- ... and *executed* in sequence
  - Limited repertoire, within ability, one-at-a-time
  - “Program counter” keeps track current instruction

- Formulating computation as a “concept” is **functional abstraction**
Summary: Abstraction

- An instruction (of the Lightbot or any other computer) is abstracted into the command name; functions abstract useful and meaningful operations, functions abstract functions built of functions, etc.

- Layer upon layer, we build software solutions
Homework 2: Lightbot in Symbolic Form

**Goal:** The purpose of this assignment is to look at textual ways of programming the bot.

When we solved the Lightbot 2.0 exercises last time, we programmed the bot using a tiny list of instructions presented iconographically, that is, as pictures. But, these could just as easily be written symbolically, that is, as text. So, our instruction list

```
   Step   Right Left   Jump Power   F.
```

name

can be expressed symbolically,

```
   Step   Right Left   Jump Power   F.name
```

as shown. When we program symbolically, we don’t have to call the functions F1 and F2. We can give them actual names like `F.turn_around` for a function that causes the bot to turn around. So, we can give the symbolic solution to
Also for Tuesday. READ Blown to Bit Chapter 2.
HW 3: More Lightbot. Due Thursday. Could do along with HW2. Then....

Homework 3: Lightbot Moon Walk

Goal: The purpose of this exercise is to learn how to express a function symbolically.

Abstract: You will write three functions in this assignment; for the last one, the bot does the Moonwalk.

This assignment is a continuation of the last assignment when you programmed the Lightbot symbolically, that is, with text instructions instead of iconigraphic instructions. Recall the following association:

- Step
- Right
- Left
- Jump
- Power
- F.name

We also introduced iteration, that is, repeating operations, as in 4:Step. (Check the earlier assignment if you have forgotten.)

Symbolic Function Definitions

When we define functions symbolically, we use a special form. For the function definition, that is, saying how it works, we write the name, a pair of parentheses, the operation sequence, and a final period. (The purpose of the parentheses will be clear later.) For example, this is a function definition for a function that turns the bot around:

F.turn_around() Right, Right.

The name of the function is the part between the dot and the open parenthesis, and the part after the closing parenthesis to the period, is called the body. It defines how the function works.

When we use a function, it’s called the function call or function invocation, we also have a special form. We give the name, followed by the parentheses, as in turn_around(), which instructs the bot to do the instructions in the body of the function. For example, to program the bot to “turn around and jump”, we would write the program turn.around(), Jump.
Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- Free to download and open source
- Interactive programs using 2D, 3D or PDF output
- OpenGL integration for accelerated 3D
- For GNU/Linux, Mac OS X, and Windows
- Projects run online or as double-clickable applications
How do Computers represent data?
The 1890 U.S. Census

- The 1880 census had taken seven years to tabulate,
- By the time the figures were available, they were obsolete.
- Immigration leads to rapid growth of the U.S. population from 1880 to 1890
  - Estimated that the 1890 census would take approximately 13 years to complete
  - U.S. Constitution mandates a census every ten years to apportion taxation between the states and to determine Congressional representation
  - A faster method was necessary.
The 1890 U.S. Census

- Late 1880s: Herman Hollerith, inspired by conductors using holes punched in different positions on a railway ticket to record traveler details such as gender and approximate age, invented the recording of data on a machine readable medium.
- First serious advance in digitization: punch cards
- Mechanical method – sensing a hole in a card or not

Hollerith Card, Courtesy IBM
Hollerith’s Tabulating Machine
A mechanical machine can “read” a card with ... a “metal brush”
Sensing Punch Allows Some Action

- When the brush touches the source of current, the circuit closes,
- The electrical impulse can cause a mechanical action
Suppose Hollerith coded men as 0, women as 1.

How many men and women in the population?

Machine Reads Cards, Puts women in this slot, Puts men in this slot, ... producing 2 piles. Run each pile through again just to count them -- done.
After processing based on reading cards, a machine can “save its work” by punching cards.
Making Data Digital

- “Digitizing” makes information discrete, it’s either there (1) or not (0),
- A machine can determine that fact using mechanical or electronic means.
- Once data is digital, it can be processed in many different ways
- Processing power can grow and grow
The first electronic computers: codebreaking during WW II

- [http://www.youtube.com/watch?v=NbhbsXWDAE](http://www.youtube.com/watch?v=NbhbsXWDAE)
- COLOSSUS in the U.K.
- ENIAC at Penn
- Female mathematicians were the first programmers!
Another of the key ideas
The Human Brain

- Contains approximately ten thousand million basic units, called **neurons**.
- Each neuron is connected to many others.
- Neuron is basic unit of the brain.
- Neurons specialize in communication
  - Information passes among neurons that result in brain activity
Important Features of the Brain to Model

- The human brain: capable of immensely impressive tasks.

- Fault tolerant
  - Distributed processing
  - Many simple processing elements sharing each job
  - Can tolerate some faults without producing errors

- Graceful degradation
  - With continual damage, performance gradually falls from high level to reduced level
  - Only catastrophic failure (e.g., death) results in zero performance
The Neuron

- **Soma** is the body of neuron
- Attached to soma are long filaments: **dendrites**.
- Dendrites act as connections through which all the inputs to the neuron arrive.
- **Axons** serve as output channel
Activating a neuron

- Axon: electrically active. Serves as output channel of neuron.
- Axon is non-linear threshold device. Produces **electrical pulse**, when resting potential within the soma rises above some threshold level.
- Axon terminates in synapse which couples axon with dendrite of another cell.
- Activating gates, when open allow charged ions to flow. Passes pulse to next neuron.
McCulloch and Pitts 1943

- “A Logical Calculus of the Ideas Immanent in Nervous Activity”
- Simple model of human brain, neuron
- Influenced the design of the first computers
- Starts with Logic
- Ideas from logicians: Carnap, Russell and Whitehead.
How a neuron processes input

- **Excitatory input**
  - Neurotransmitter increases membrane potential

- **Inhibitory input**
  - Neurotransmitter decreases membrane potential

- **Temporal summation**
  - Initial input not sufficient to trigger action potential, but 2\textsuperscript{nd} input, following within a short time, can

- **Spatial summation**
  - Several inputs (excitatory or inhibitory) originating from different neurons can have cumulative effect
Important features to model

- Neuron accepts many inputs, which are all added up (in some fashion).
- If enough active inputs received at once, neuron will be activated, and fire. If not, remains in inactive quiet state. Either ON or OFF.
- The output depends only on the inputs. A certain number must be on at any one time in order to make the neuron fire.
- The efficiency of the synapses at coupling the incoming signal into the cell body can be modelled by having a multiplicative factor (i.e. weights) on each of the inputs to the neuron.
- More efficient synapse has correspondingly larger weight.
McCulloch & Pitts Neuron

- **Input** = \((W_1 \times X_1) + (W_2 \times X_2) \ldots + (W_n \times X_n)\)
- **Output** = If Input > Threshold then 1, otherwise 0
Neuron can model most Logical Functions

- McCulloch and Pitts then proceeded to show (prove mathematically) that such a model of the neuron could calculate most logical functions

- Let's say $P = \text{“Today is a sunny day”}$
- Let's say $Q = \text{“I am happy”}$

- When is $(P \text{ and } Q)$ true?
Truth Table for *And* *(using True and False)*

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
### Truth Table for *And* *(using 0 and 1)*

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
McCulloch Pitts Neuron for AND

\[ X \lor Y \land -2 \]
How the neuron computes logical AND

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y-2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
</tbody>
</table>
McCulloch Pitts Neuron for OR

\[ X + Y - 1 \]

\[ X \text{ OR } Y \]
### How the Or Neuron computes logical OR

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y-1</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
<td>P and Q</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>
These are the fundamental units of computers
It All Works Because of Digital

- Key principle: information is the presence or absence of a phenomenon at a given place and time!

- Card example:
  - Phenomenon – hole in the card
  - Present – detected by brush making elec contact
  - Absent – brush insulated from electrical source
  - Place – there are several on the card; devices can know the positions
  - Time – hole is permanent representation of info

- Neuron example:
  - Phenomenon – activation on the axon
  - Present – action potential greater than threshold, neuron fires
  - Absent – action potential not greater
**Bits**

- PandA is a *binary representation* because it uses 2 patterns.
  - Bit -- it’s a contraction for “binary digit”
  - a position in space/time capable of being set and detected in 2 patterns
A General Idea

- Digital Information: Detecting the presence or absence of a phenomenon at a specific place and time: PandA
- Phenomena: light, magnetism, charge, mass, color, current, ...
- Detecting depends on phenomenon – but the result must be discrete: it was detected or not; there is no option for “sorta there”
- Place and time apply, but usually default to “obvious” values
So computers can do logic, using simple gates (like neurons) for each logic function....

But there are more key ideas
Bits

- PandA is a *binary representation* because it uses 2 patterns
  
  Bit -- it’s a contraction for “binary digit”
  
  -- a position in space/time capable of being set and detected in 2 patterns
Bytes: Standard encodings of meaning

- A byte is eight bits treated as a unit
  - Adopted by IBM in 1960s
  - A standard measure ever since
  - Bytes encode the Latin alphabet using ASCII -- the American Standard Code for Information Interchange

\[
\begin{align*}
0101 & 0101 \\
0101 & 0111
\end{align*}
\]
<table>
<thead>
<tr>
<th>ASCII</th>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
<th>0100</th>
<th>0101</th>
<th>0110</th>
<th>0111</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N U</td>
<td>S H</td>
<td>S X</td>
<td>E X</td>
<td>E T</td>
<td>E Q</td>
<td>A K</td>
<td>B L</td>
</tr>
<tr>
<td></td>
<td>D L</td>
<td>D 1</td>
<td>D 2</td>
<td>D 3</td>
<td>D 4</td>
<td>N K</td>
<td>S Y</td>
<td>E I</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
</tr>
<tr>
<td>0000</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>0001</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>0010</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
</tr>
<tr>
<td>0011</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>0100</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>0101</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
<td>p</td>
</tr>
<tr>
<td>0110</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
</tr>
<tr>
<td>0111</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td>}</td>
<td>~</td>
<td>d</td>
</tr>
<tr>
<td>1000</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>1001</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
</tr>
<tr>
<td>1011</td>
<td>Ö</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
</tr>
<tr>
<td>1100</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
<td>Å</td>
</tr>
<tr>
<td>1101</td>
<td>Ð</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
</tr>
<tr>
<td>1110</td>
<td>á</td>
<td>â</td>
<td>ã</td>
<td>â</td>
<td>ã</td>
<td>â</td>
<td>ã</td>
<td>â</td>
</tr>
<tr>
<td>1111</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
<td>Ñ</td>
</tr>
</tbody>
</table>
With 8 places how many different letters?
UTF-8: All the alphabets in the world

- Uniform Transformation Format: a variable-width encoding that can represent every character in the Unicode Character set
- 1,112,064 of them!!
- UTF-8 is the dominant character encoding for the World-Wide Web, accounting for more than half of all Web pages.
- The Internet Engineering Task Force (IETF) requires all Internet protocols to identify the encoding used for character data
- The supported character encodings must include UTF-8.
Encoding Information: There’s more!

- Bits and bytes encode the information, but that’s not all
- **Tags** encode format and some structure in word processors
- **Tags** encode format and some structure in HTML
- **Tags** are one form of meta-data
- **Meta-data** is information about information
Professor Marilyn Walker
Department of Computer Science
University of California Santa Cruz
1156 N. High, SOE-3
Santa Cruz, Ca. 95064
Phone:831 429 1058
Natural Language and Dialogue Systems Lab
maw@soe.ucsc.edu

DATA and SOFTWARE DOWNLOADS

PROFESSIONAL EXPERIENCE
Professor of Computer Science, Natural Language and Dialogue Systems Lab, University of California, Santa Cruz, 2009 to present
Professor of Computer Science, Head of Cognitive Systems Group, University of Sheffield, Sheffield, England, 2003 to 2009
Principal Research Staff Member, ATT Labs - Research, Florham Park, N.J., Speech Processing Software and Technology Research, 1996 to 2003
Research Scientist, Mitsubishi Electric Research Laboratories, Cambridge, Ma., Interactive Learning and Entertainment, 1993 to 1996
Consultant, Hewlett Packard Laboratories, Bristol, England, on dialogue systems, speech technology, and personal information systems: 1989-1993
Researcher, Dialogue Modeling Department, Electrotechnical Laboratory, Tsukuba City, Japan: Summer 1991

EDUCATION
B.A. Computer and Information Science, With Honors, University of California Santa Cruz, 1984.

http://users.soe.ucsc.edu/~maw/
More kinds of meta data all the time

- “The semantic web”
- An “ontology” i.e. “taxonomy” of the kinds of things there are in the world
  - People, place, thing, animal, organization, country
- New tags within web pages
- Could be put there by people
- Could be programmatically identified by text processing algorithms (like what Watson Jeopardy uses)
Representing Information: Summary

- Bits encode numbers using the binary representation 1110 0111
- Bits encode letters using ASCII for North American and Western European languages

This suggests an principle we will soon argue:
- All information can be represented with bits