Announcements ...

- Please fill out the “pre-course” survey if you have not yet done so
We’re underway …

Following Lightbot

Lawrence Snyder
University of Washington, Seattle
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

- When you are commanding (programming), you direct an agent (by instructions) to a goal
  - The **agent** is usually a computer, but it can be a person, or other device (animated robot?)
  - The agent follows the commands a/k/a **instructions**, flawlessly, and mindlessly, doing only what it is asked
  - The program implements **human intent** – you are trying to get the robot to the Blue Tile goal – it’s the point of your instructions
Sequencing

- Instructions are *given* in sequence, i.e. in order
- They are *followed* in sequence, i.e. in order
  - YOU give the instructions ... it’s called **programming**
  - The AGENT follows them ... it’s called **executing** or **running** the program
  - A program **counter** marks the agent’s place
Order of Events

- The instructions are programmed *ahead of time*
- They are executed *later, w/o programmer’s intervention*
  - Each instruction makes *progress* towards the goal
  - The order *must be right* to achieve the goal
Point of View

- Programming REQUIRES you to take the *agent’s point of view* ... it’s a essential idea
Programming REQUIRES you to take the *agent’s point of view* ... it’s a essential idea

From this cell, a turn is required ... R or L?
Limited Instruction ‘Repertoire’

- The number and type of instructions is always limited – you need a solution using only them
  - Instructions ...
    - The agent can do only certain things ... nothing else
    - The Lightbot’s instructions
    - There is no JUMP_3
  - ... Lightbot’s even tougher than normal programming b/c in some LB games, some instructions are unavailable ... but it’s a game!
  - Executed the instructions one-at-a-time
The limited repertoire is a fact of all computing, but how limited?

A computer’s circuitry (the hardware) has very few instructions ... usually about 100, and many are just different versions of the same idea: `add_2_bytes`, `add_2_words`, `add_2_decimal_numbers`, etc.

In theory, a computer with only 6 instructions could compute all known computations.
If that were the end of the story

- Programming would be amazingly tedious if all programming had to use only the basic instructions – I mean REALLY REALLY REALLY tedious
  - No one would be a programmer no matter how much it paid
  - Apps as we know them would not exist
  - BTW programming was like this in the beginning
    - This is why they are called the “bad old days”
- Luckily, there are functions
Functions Package Computation

- We make new instructions using functions!

- \( F_1() \) packages actions: E.G. “process a riser”
Functions Package Computation

Just Do It!
We have a new instruction: Process_A_Riser

Call the function to use the new instruction
Functions may seem “obvious” but they are a HUGE idea ...

They allow us to solve problems by first creating some useful instructions, and then using them to get the agent to do our work

Sweet!

... Let’s see how this works
Because $F_1(\cdot)$ “processes a riser,” we think of the programming task as

- Process a riser( )
- Move to next riser
- Process a riser( )
- Move to next riser
- Process a riser( )
The Function Becomes A Concept

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

<table>
<thead>
<tr>
<th>Process a riser()</th>
<th>( F_1() )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to next riser</td>
<td>( F_1() )</td>
</tr>
<tr>
<td>Process a riser()</td>
<td>( F_1() )</td>
</tr>
<tr>
<td>Move to next riser</td>
<td>( F_1() )</td>
</tr>
</tbody>
</table>

- With \( F_1() \), we simplify the programming to just 5 conceptual steps rather than 21
The Function Becomes A Concept

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

  | Process a riser( ) | F1( ) |
  | Move to next riser | F1( ) |
  | Process a riser( ) | F1( ) |
  | Move to next riser | F1( ) |
  | Process a riser( ) | F1( ) |

- With $F_1(\ )$, we simplify the programming to just 5 conceptual steps rather than 21
- But, WAIT! What is “Move to next riser”?
  - It’s a concept ... make it a function!
  - Move_to_next_riser( )
Because F₁( ) “processes a riser,” we think of the programming task:

- Process a riser( )  F₁( )
- Move to next riser( )  F₂( )
- Process a riser( )  F₁( )
- Move to next riser( )  F₂( )
- Process a riser( )  F₁( )

With F₁( ), we simplify the programming to just 5 conceptual steps rather than 2₁

But, WAIT! What is “Move to next riser”?
- It’s a concept ... make it a function!
- Move_to_next_riser( )

Show that text is a function with parens.
A Five Instruction Program

Is this beautiful, or what?
Abstraction ...

- Formulating blocks of computation as a “concept” is **functional abstraction** [A better definition in a moment]
- What we did just now is important ...
  - We spotted a coherent (to us) part of the task
  - We solved it 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 then we did it again!
Collecting operations together and giving them a name is *functional abstraction*.

- The operations perform a coherent activity or action – they become a *concept* in our thinking.
- The operations accomplish a goal that is useful – and typically – is needed over and over again.

*Functions* implement functional abstraction: 3 parts
- A name
- A definition, frequently called a “body”
- Parameters – stuff inside the parentheses, covered later.
People Abstract All The Time

- Functional abstractions in which you are the agent, but someone taught you:
  - Parallel parking
  - Backstroke in swimming
- Functional abstractions you recognized and in which you are the agent
  - Doing a load of laundry
  - Making your favorite {sandwich, pizza, cookies, ...}
- Others?
Keep Using Abstraction ...

- If M.C. Escher handed us a problem ... what would we do?

It only simplifies our thinking; the bot still does all the work
The Function Is Just The Packaging

- Another way to use a function for the risers
Programming is **commanding** an agent

- **Agent:** usually a computer, person, or other device
- Agent follows **instructions**, flawlessly & mindlessly
- 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**

We’ll See It Again & Again