

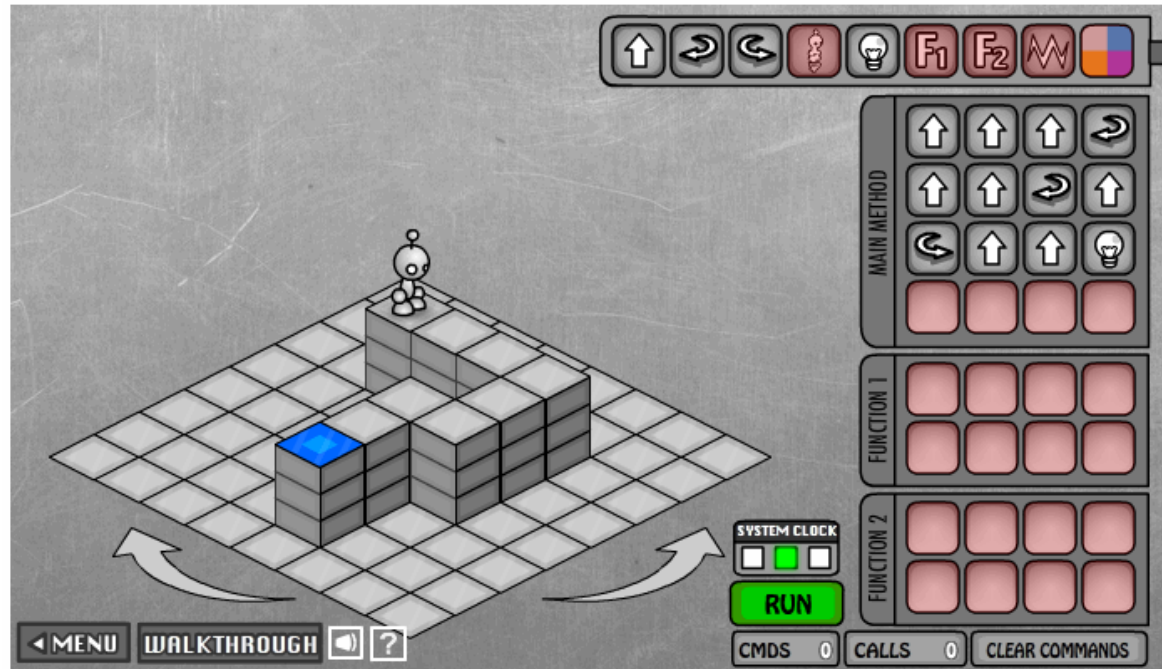
We're underway ...

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

A Lightbot 2.0 “Computation”

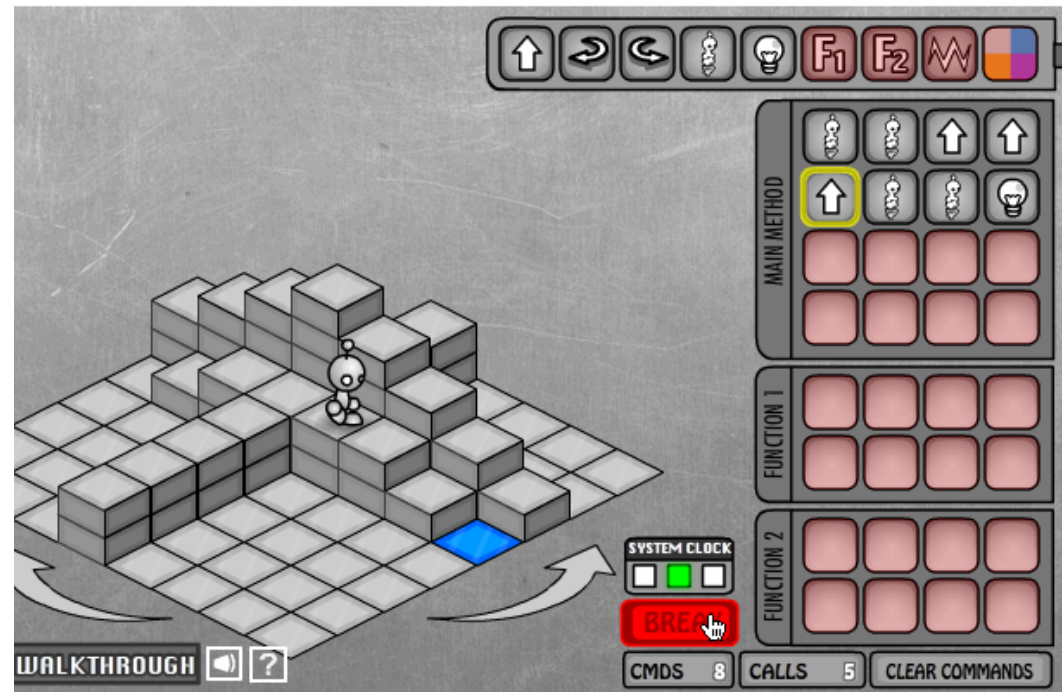
Just Do It!

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

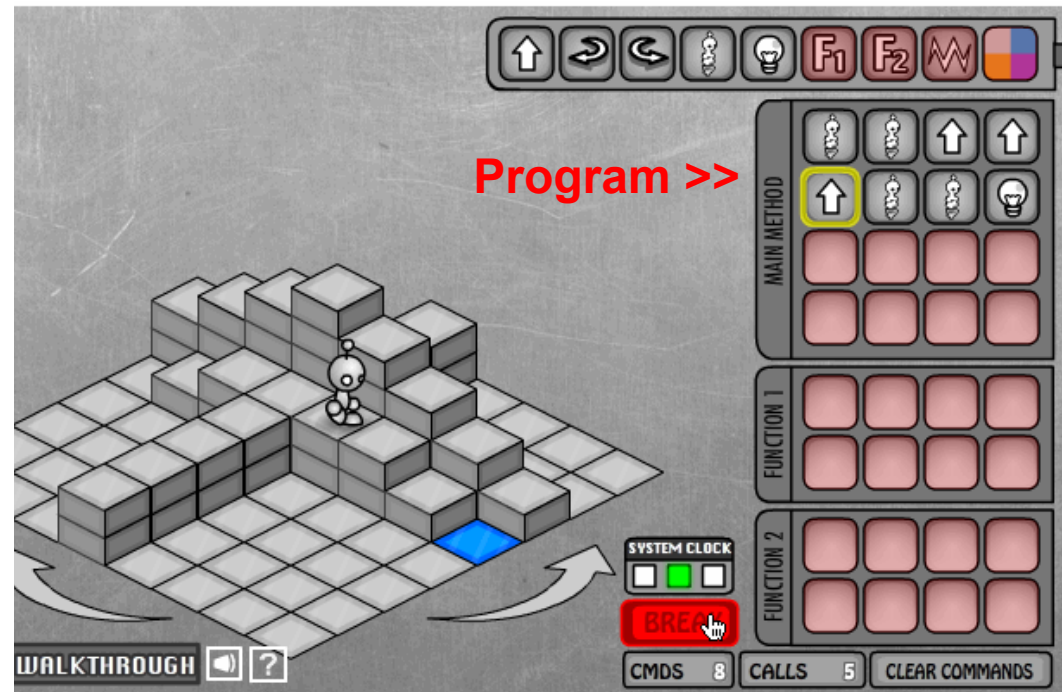
Sequencing

- 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



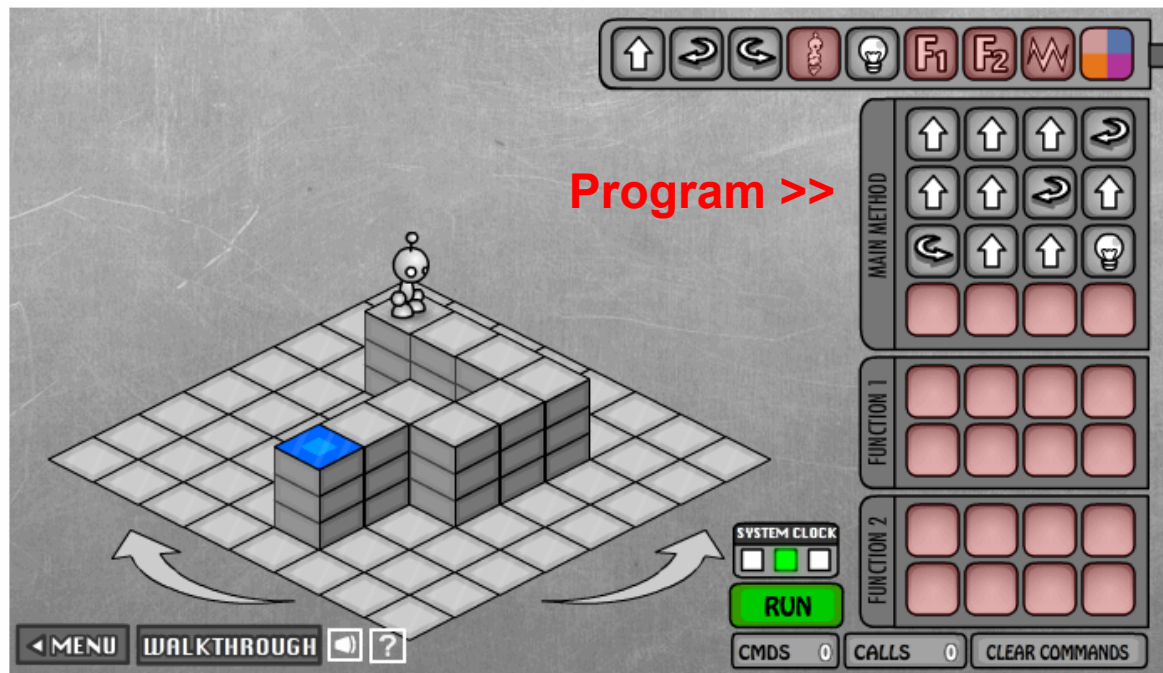
Order of Events

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



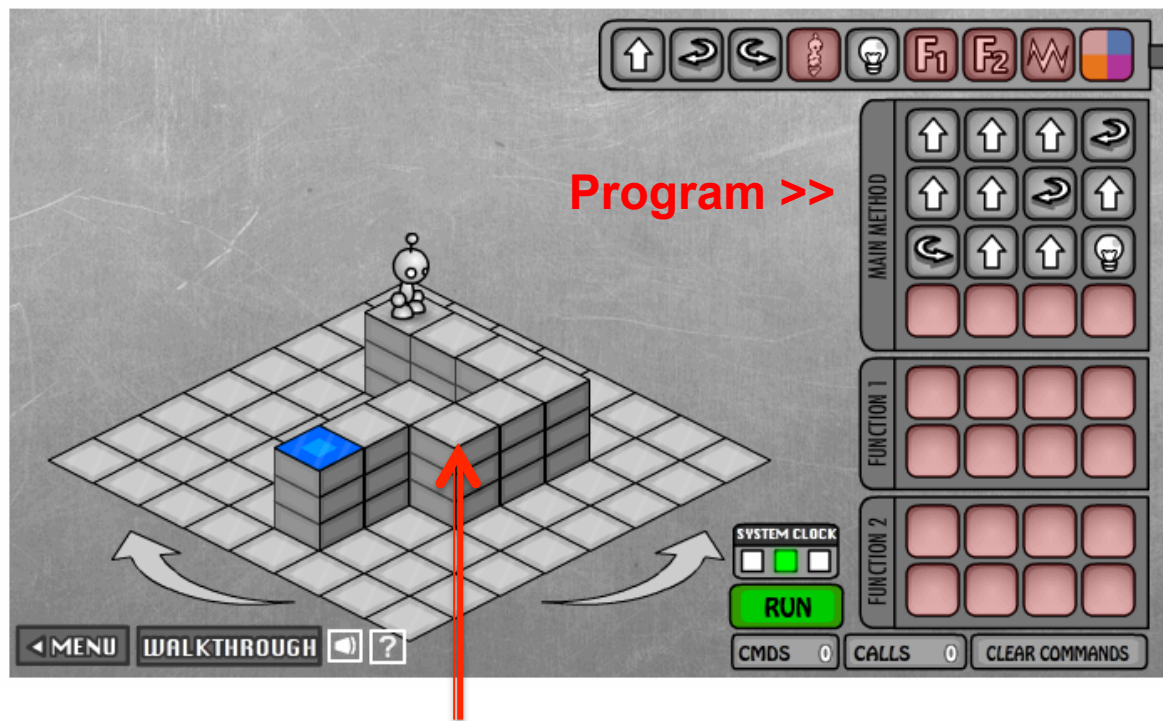
Point of View

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



Point of View


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



Program >>

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!
 - Execute the instructions one-at-a-time

An Amazing Fact ...

- 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_ints**, **add_2_decimal_numbers**, etc.

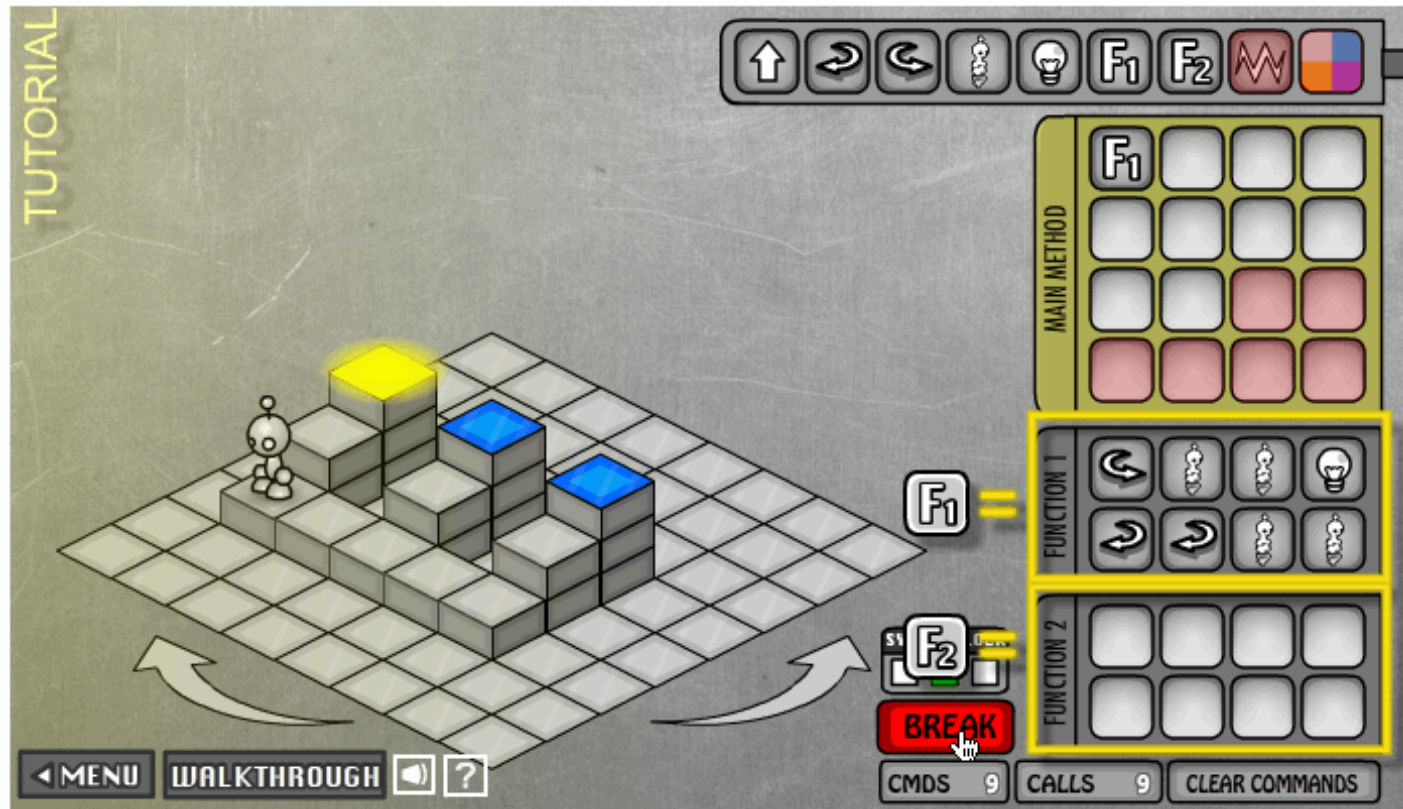
In theory, a computer with just 6 instruction types could compute all known computations

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 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!



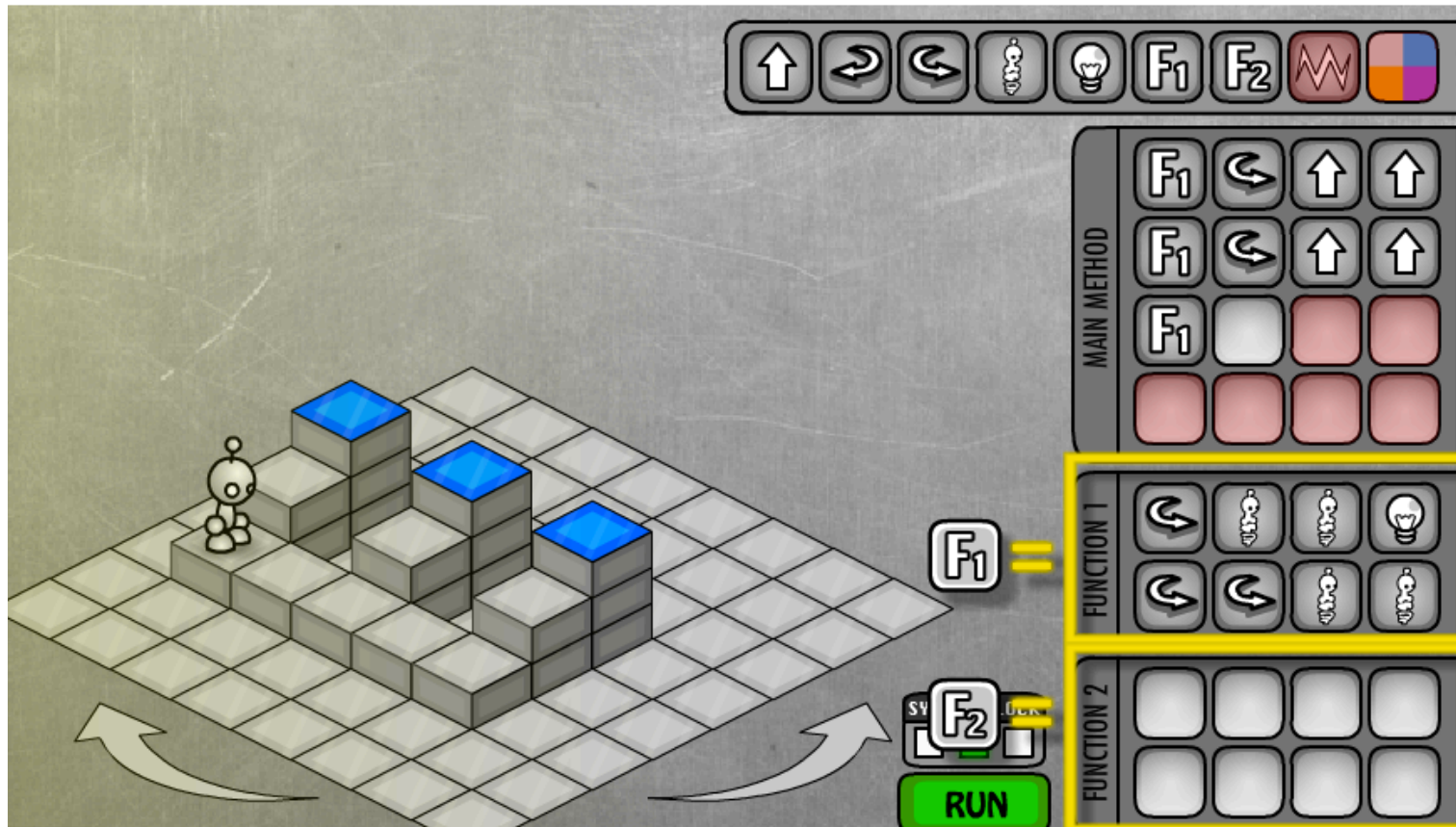
- $F_1()$ packages actions: E.G. “process a riser”

Functions Package Computation

Just Do It!

F₁(), A Process a Riser Instruction

- We have a new instruction: Process_A_Riser



- **Call** the function to use the new instruction

Where Do Functions Come From?

- The “process a riser” function was an obvious sub-problem of the overall task – we just saw it was a useful operation to perform
- But spotting common patterns is also another place to find “work” that could be turned into functions ... look how the “process a riser” code is used

Move To Next Riser

Perhaps the clearest solution

Control Panel:

- Top Row: Up arrow, Undo, Redo, Rotate, Lightbulb, F1, F2, Wave icon, Color palette.
- MAIN METHOD: 4x4 grid of buttons. Row 1: F1, F2, F1, F2. Row 2: F1, empty, empty, empty. Row 3: empty, empty, red, red. Row 4: red, red, red, red.
- FUNCTION 1: 2x4 grid of buttons. Row 1: Rotate, Rotate, Rotate, Lightbulb. Row 2: Rotate, Rotate, Rotate, Rotate.
- FUNCTION 2: 2x4 grid of buttons. Row 1: Rotate, Up arrow, Up arrow, empty. Row 2: empty, empty, empty, empty.

Character: A small robot on a 3x3 grid of blocks. The top row has 3 blocks, the middle row has 2 blocks, and the bottom row has 1 block. The top-right block of the top row is blue.

Buttons: A large F1 button with an equals sign, and a smaller F2 button with a 'SY' label. A green RUN button is at the bottom right.

Yet Another Solution

The image shows a 3D grid-based environment with a small robot on the left. The grid has several blue cubes stacked on top of grey cubes. To the right is a programming interface with a toolbar at the top containing icons for movement (up, left, right), a robot, a lightbulb, and function keys F1, F2, a red zigzag, and a color palette. Below the toolbar is a 'MAIN METHOD' grid with four columns and three rows. The first three columns have F2 buttons, and the last column is empty. The second and third rows have red buttons in the last two columns. Below this are 'FUNCTION 1' and 'FUNCTION 2' grids, both highlighted with a yellow border. FUNCTION 1 has a left arrow, robot, robot, and right arrow. FUNCTION 2 has F1, lightbulb, F1, and up arrow in the top row, and up arrow in the bottom row. A 'RUN' button is at the bottom right. A yellow callout box with the text 'Just Do It!' is in the bottom left. An arrow points from the 'RUN' button to the 3D grid.

Just Do It!

It's BIG!

- Functions may seem “obvious” but they are a HUGE idea ...
- They allow us to solve problems by solving parts, naming them (at least in our mind), and putting the part solutions together to solve the whole problem
- Sweet!
- Really sweet!