We’re underway …

Following Lightbot

Lawrence Snyder
University of Washington, Seattle

Announcements …

- Please fill out the "pre-course" survey if you have not yet done so
- "Bring" a digital picture of yourself to Lab on Thursday …

Two Paths Diverge in the Lectures

- As noted, this class is about principles, and about learning to use computational thinking to solve your problems
- I will use a 2-thread class structure …
  - One thread will cover principles and key knowledge that everyone should know about CS
  - The other thread will focus on "doing stuff" – reasoning, analysis, abstracting, programming, problem solving, creating, etc.

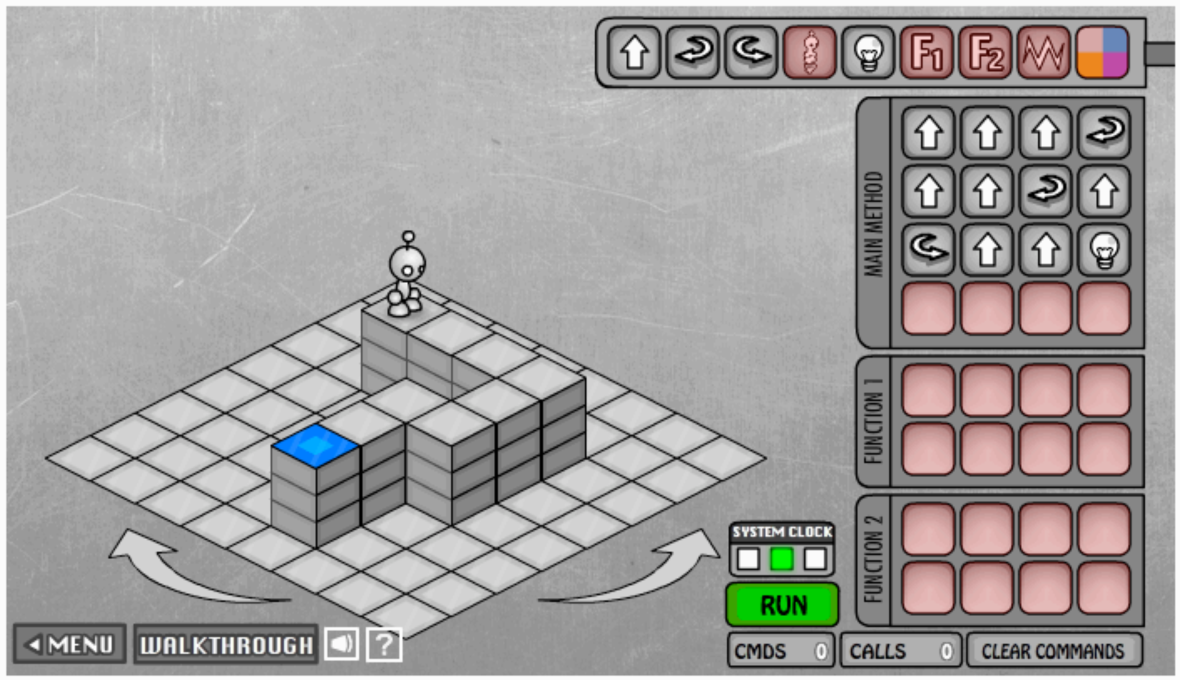
Think of the first as *concepts*, the second as *capabilities*

Today – They Are Merged

- Topic: The act of directing a computer to do something … called *programming*
- The Lightbot 2.0 exhibited many properties of programming, so to launch both threads we will review what those properties are. (I have a complete list at the end.)

As Experienced Lightbot Hackers

- What are you doing in Lightbot?



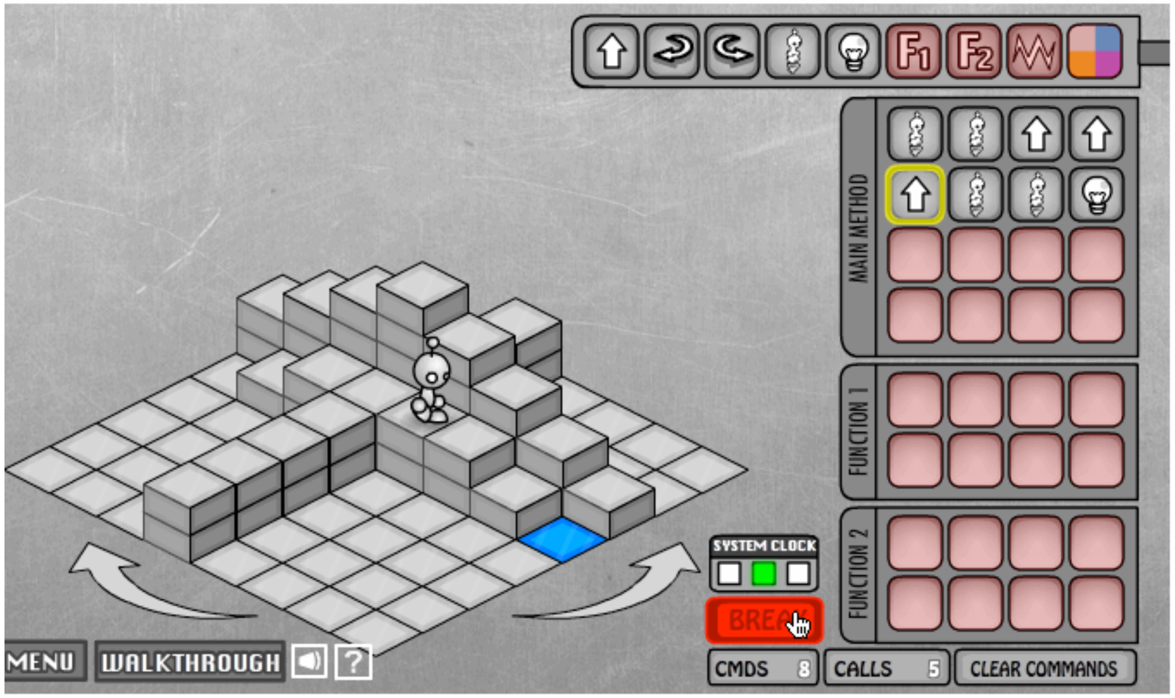
- Commanding a robot through a "blocks world"
- Programming is **commanding** an agent

Agent, Instructions, Intent

- Other aspects of “commanding”
  - The **agent** is usually a computer, but it could be a person, or other device (animated robot?)
  - The agent follows the commands a/k/a **instructions**, flawlessly, and stolidly, doing only what it is asked
  - The program implements human intent – **you** try to get the robot to the Blue Tile goal – it’s the point of your instructions

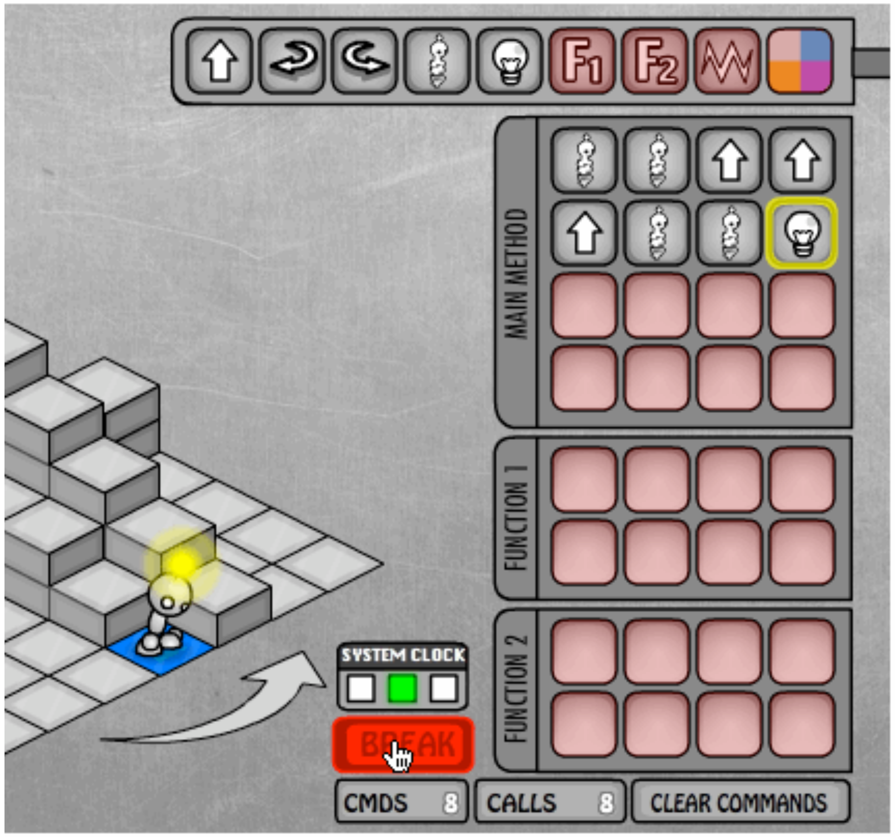
Sequencing

- 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 instr. is a series of steps



An Instruction *abstracts* those steps

Abstraction

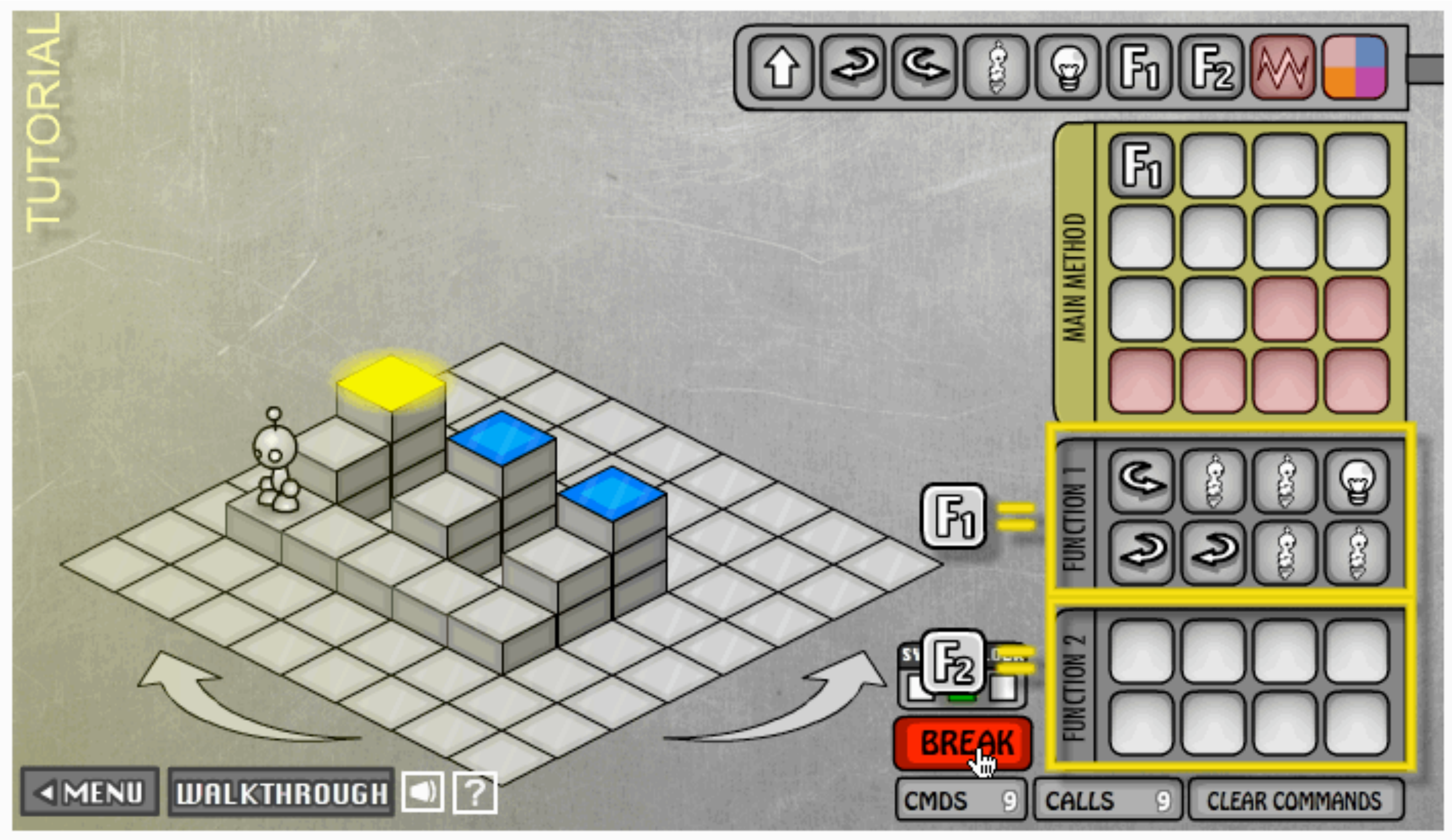
- The word “abstraction” is used a lot in computing, and in this course
- The instruction example just given illustrates *functional abstraction* meaning that we have given a **name** to a **series of operations** that perform a coherent (and to us meaningful) activity; the name is the instruction, the 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
  - Giving that function a name is *functional abstraction*
  - It doesn't seem like a big deal … and if it wasn't AMAZINGLY powerful, it wouldn't be
  - 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

Functions Package Computation

- F1( ) packages actions: E.G. "process a riser"



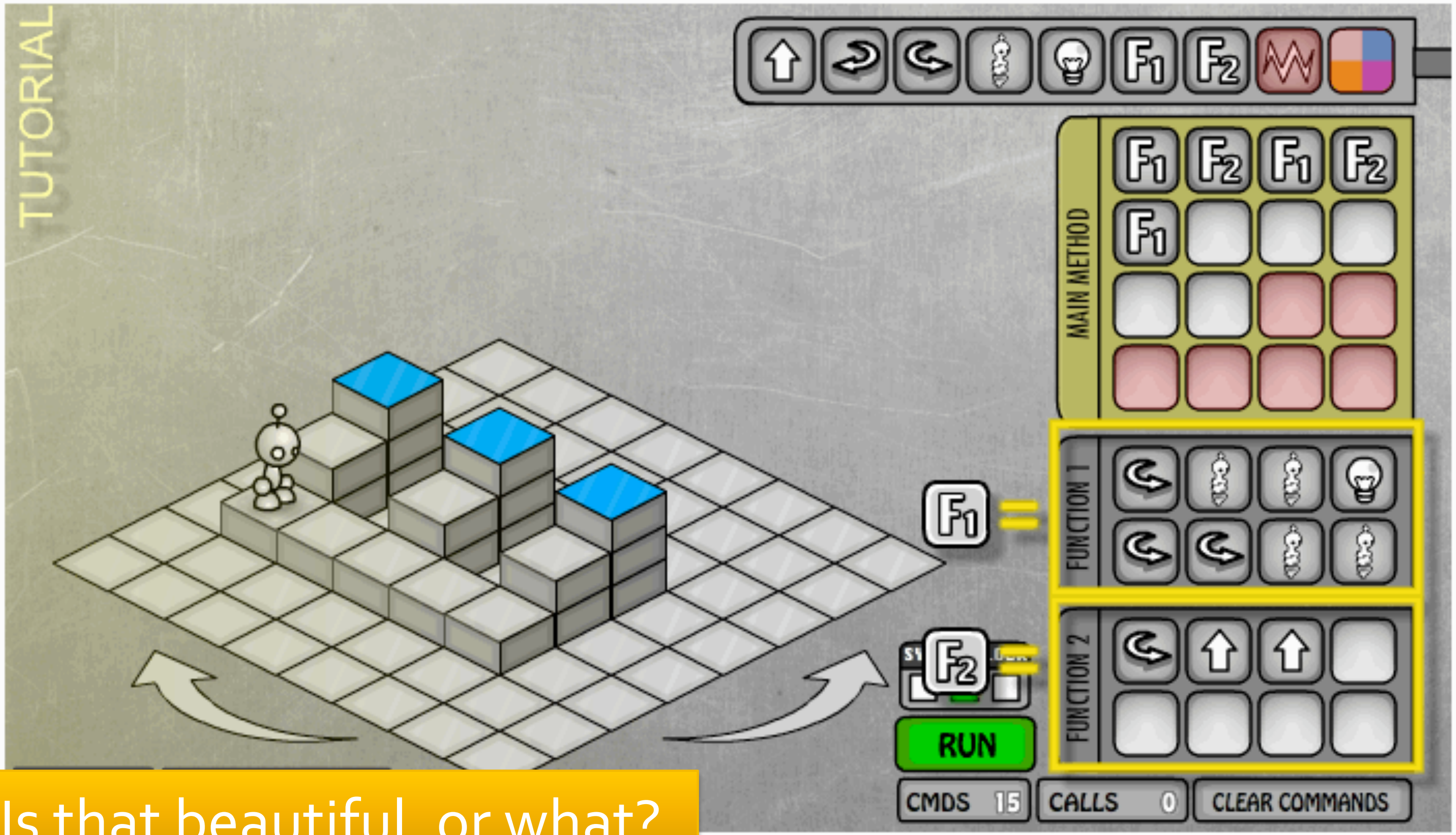
The Function Becomes A Concept

- Because F1( ) "processes a riser," I think of the programming task as

```
Process a riser          F1
Move to next riser
Process a riser          F1
Move to next riser
Process a riser          F1
```

- With F1( ) 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



Is that beautiful, or what?

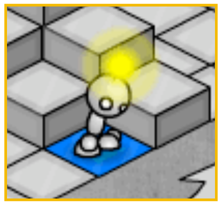
Here Is What Is Beautiful …

- Did everyone see 1 idea, 2 applications?

Slide 8

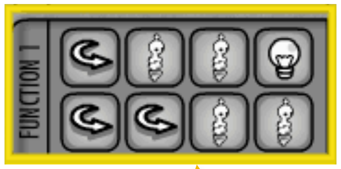
•To a programmer the instruction is monolithic (one thing)

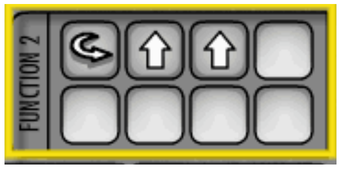
•To an agent each instruction is a series of steps

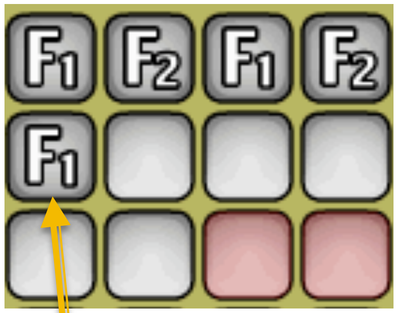


Slide 11

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







It is one concept here, but

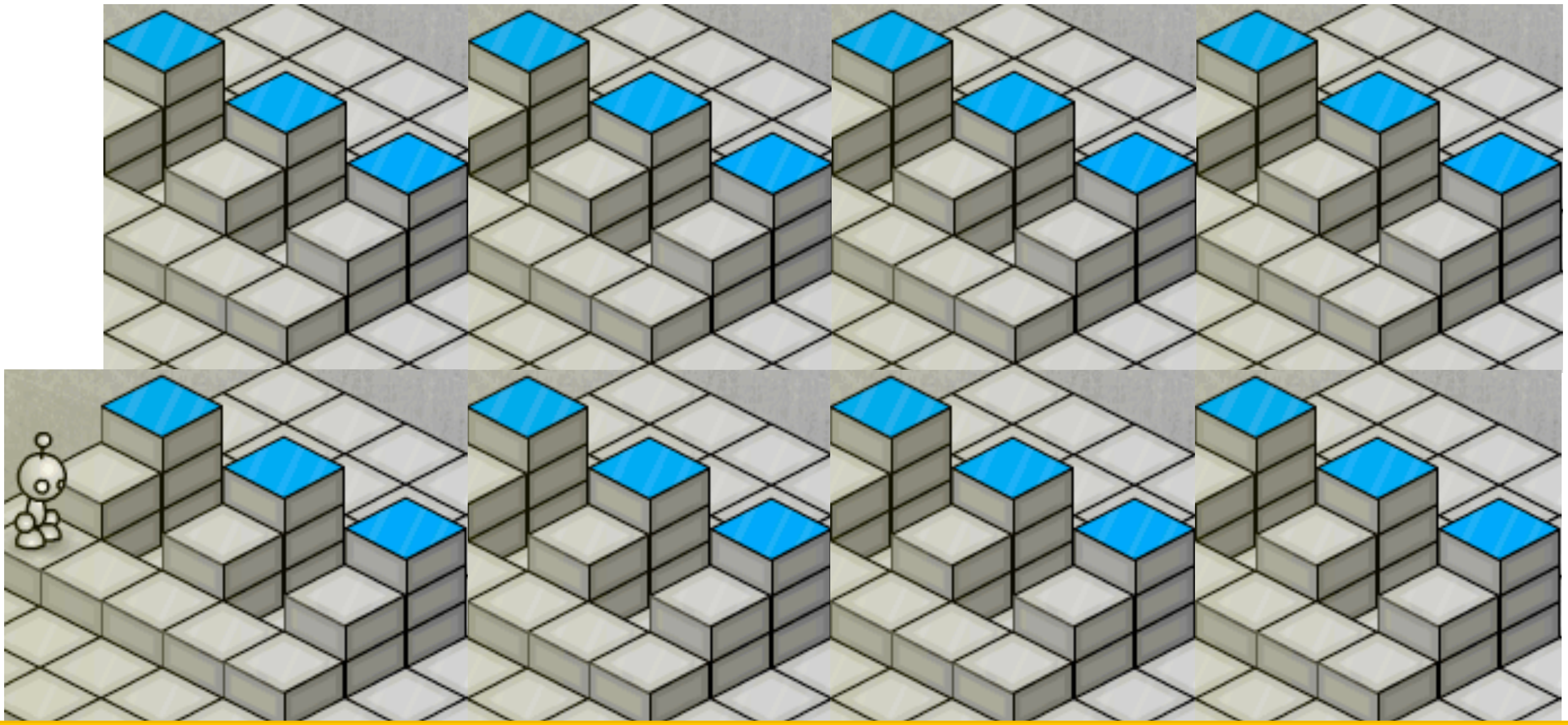
here it is eight instructions

Abstraction …

- Formulating blocks of computation as a "concept" is **functional abstraction**
- What we did is important here …
  - 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 probably repeat this approach at the next higher level

Keep Using Abstraction …

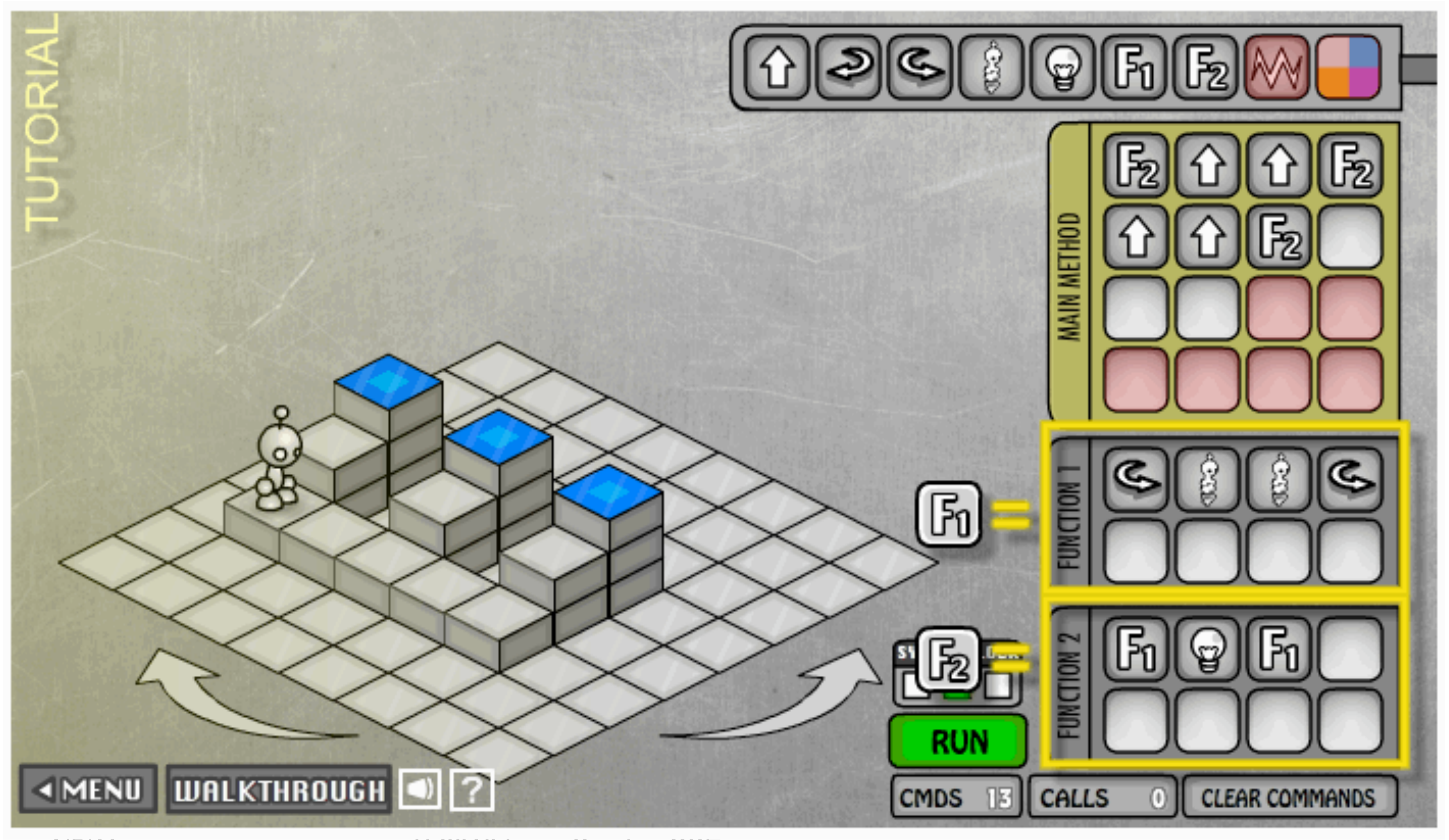
- If M.C. Escher handed us a problem … what would we do?



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



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**