

Lightbot and Functions

CSE 120 Winter 2017

Instructor:

Justin Hsia

Teaching Assistants:

Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

As Congress Repeals Internet Privacy Rules, Putting Your Options In Perspective

Passed by the Federal Communications Commission in October, the rules never went into effect. If they had, it would have given consumers more control over how ISPs use the data they collect. Most notably, the rules would have required explicit consent from consumers if sensitive data – like financial or health information, or browsing history – were to be shared or sold.

These rules wouldn't have applied to the likes of Google or Facebook – massive data collectors and digital advertisers. But consumer groups argue... you could abandon those companies in favor of other websites, if you disagree with their policies; switching Internet providers is not so easy.

- <http://www.npr.org/sections/alltechconsidered/2017/03/28/521813464/as-congress-repeals-internet-privacy-rules-putting-your-options-in-perspective>



Administrivia

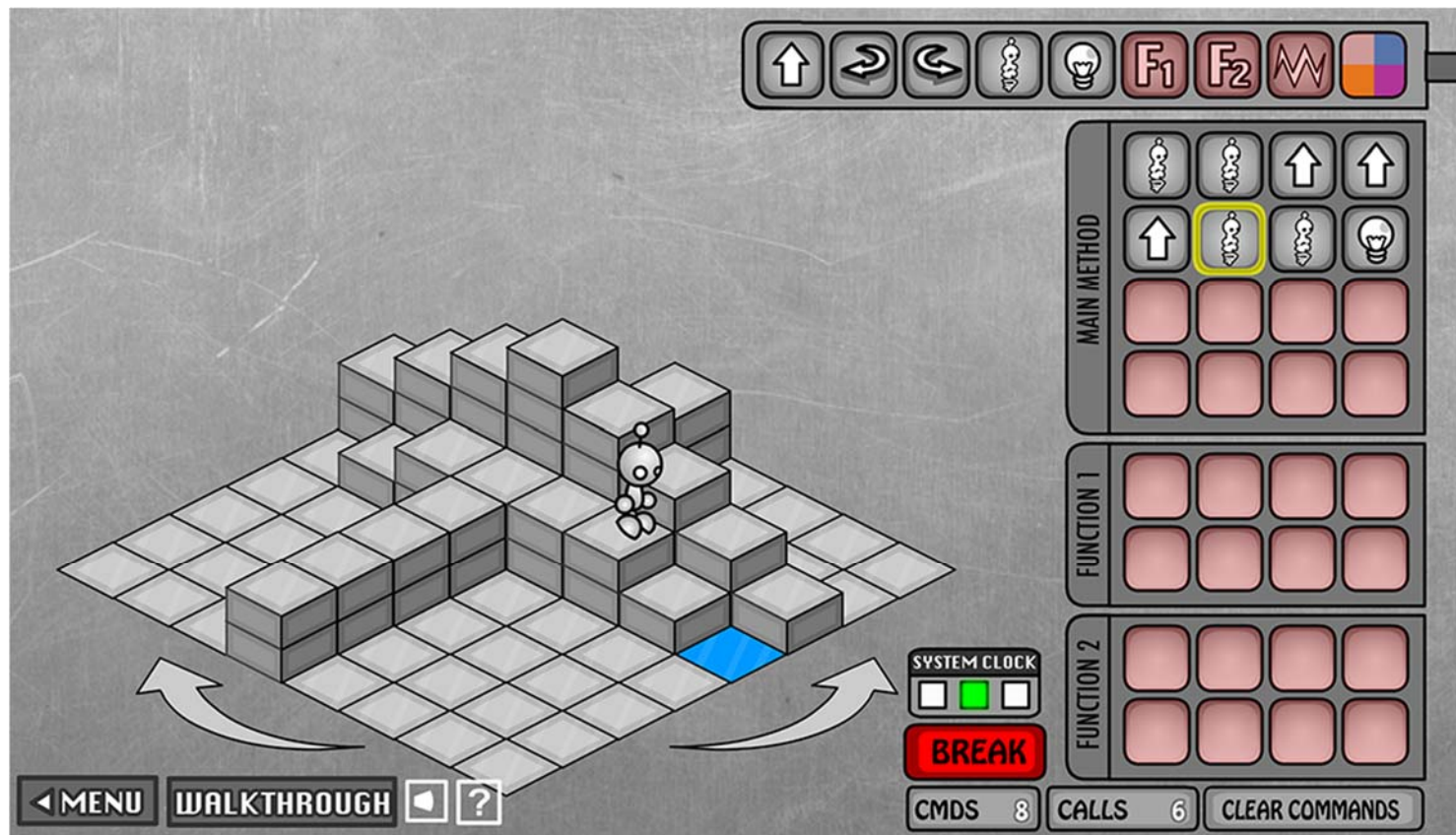
- ❖ Check-in
 - How is Exploring Lightbot going? (due tonight)
 - How is your portfolio setup going? (due Friday)
 - Sorry that we are not teaching you HTML/CSS
 - Any questions on course navigation?

- ❖ Reading Check 1 is on Canvas and due in the 24 hours preceding your Thursday lab
 - 20 minutes, short answer

- ❖ Personal Values assignment due Sunday (4/2)

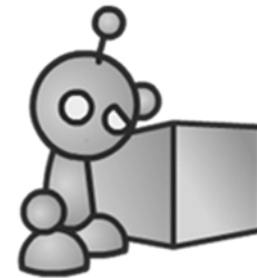
As Experienced Lightbot Players...

- ❖ What are you doing in Lightbot?
 - Commanding a robot through a world of blocks and switches



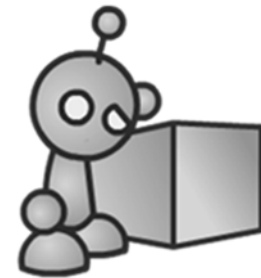
As Experienced Lightbot Players...

- ❖ What are you doing in Lightbot?
 - Commanding a robot through a world of blocks and switches
- ❖ Programming is *commanding* an *agent*
 - In this case, the agent is a robot
 - The agent is usually a computer, but could be a person or other device



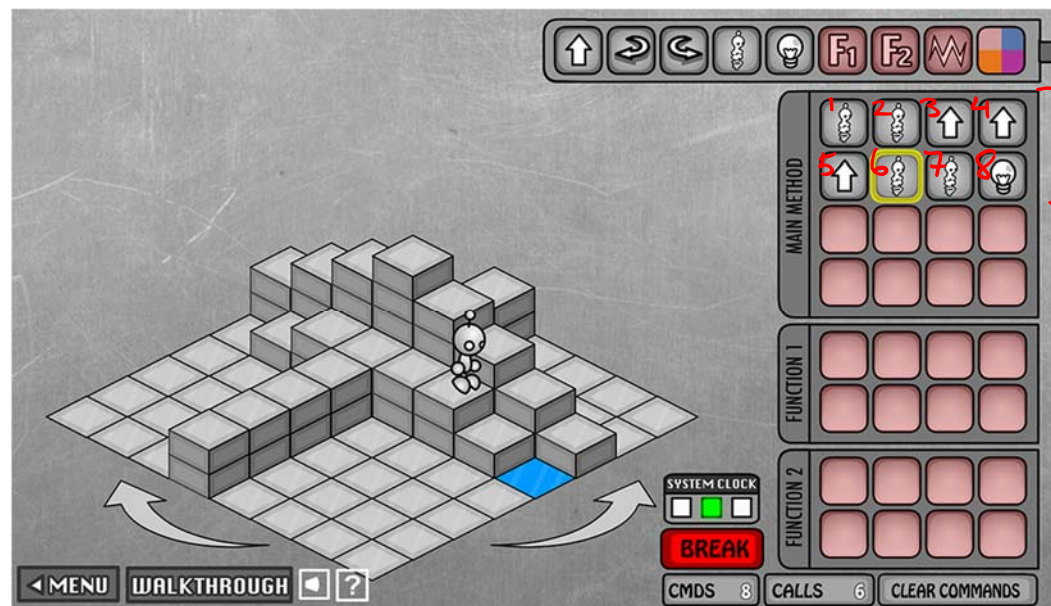
As Experienced Lightbot Players...

- ❖ What are you doing in Lightbot?
 - Commanding a robot through a world of blocks and switches
- ❖ Programming is *commanding* an *agent*
 - In this case, the agent is a robot
 - The agent is usually a computer, but could be a person or other device
- ❖ Direct an agent to a *goal* by giving it *instructions*
 - The agent follows the instructions flawlessly and mindlessly
 - ★ ■ The trick is to find the right instructions to match your *intent*



Order of Instructions

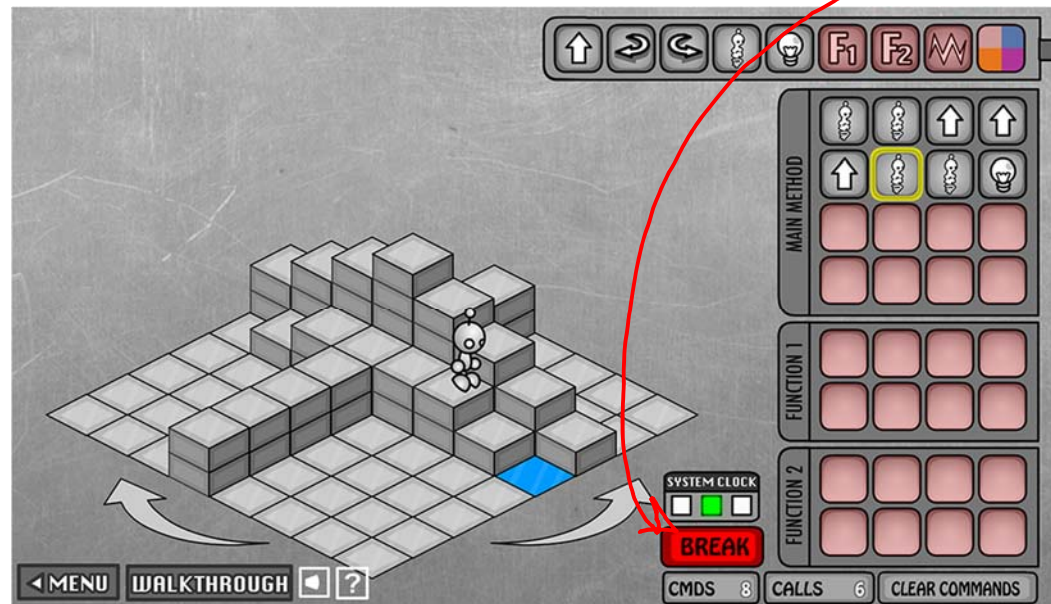
- ❖ Instructions are given in order (*i.e.* in a sequence)
 - The 1st instruction is completed, then the 2nd, then the 3rd, ...
- ❖ *You* issue the instructions and the agent follows them
 - When the agent is following your instructions, this is called **executing** the program, or **running** the program



sequence of instructions (program!)

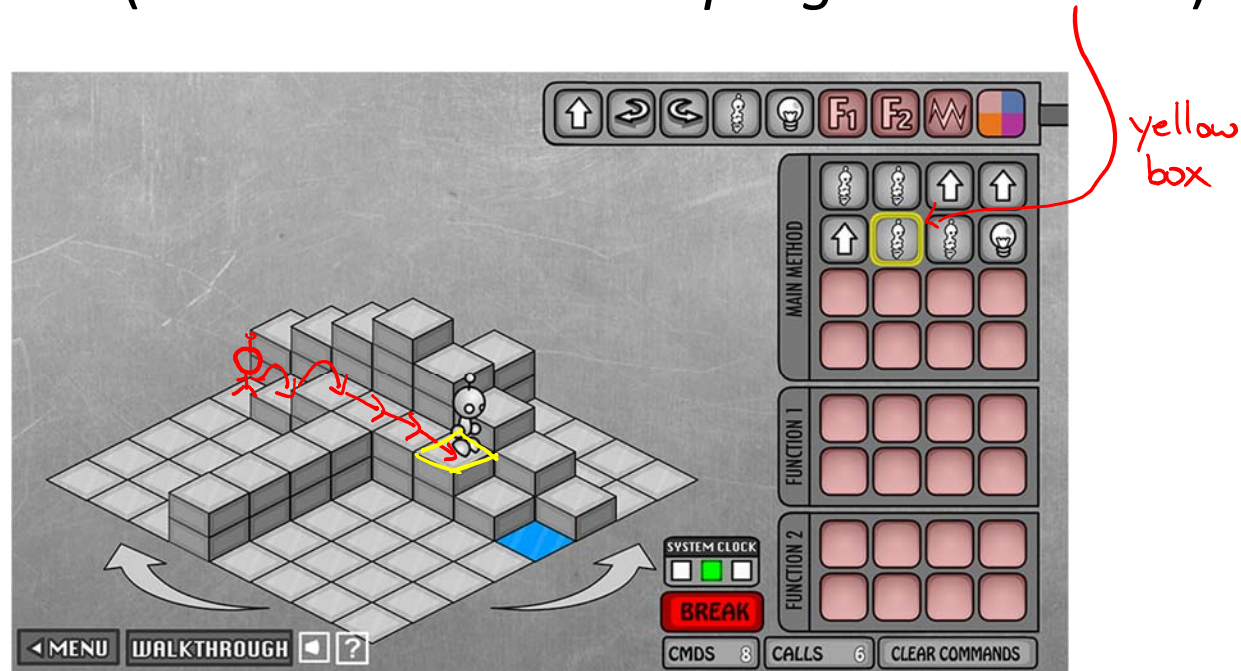
Order of Events

- ❖ The instructions are programmed *ahead of time*, and then executed *later*
 - The programmer cannot intervene until the program has finished executing or is terminated prematurely
- ❖ The instructions must be correct in order for the agent to achieve its goal



Point of View

- ❖ Programming requires you to take the agent's point of view
 - Because it is a sequence of instructions, you must account for everything that happened before (*i.e.* **trace** the program)
 - There is usually an indication of where you are currently in the program (sometimes called a *program counter*)

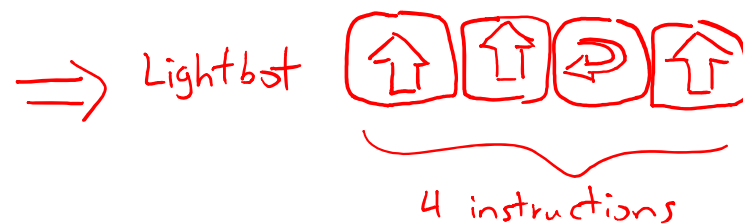


Limited Instructions

- ❖ The number and type of instructions is always limited



- The agent can only do certain pre-defined actions
- ❖ The agent can only execute one instruction at a time
 - Must learn how to specify complex tasks using just these simple actions



Limited Instructions

- ❖ Limited instructions is a reality of *all* computing
- ❖ A computer's hardware/circuitry can only execute a small number of instructions – usually about 100
 - Many are just different versions of the same idea

Amazing Fact:

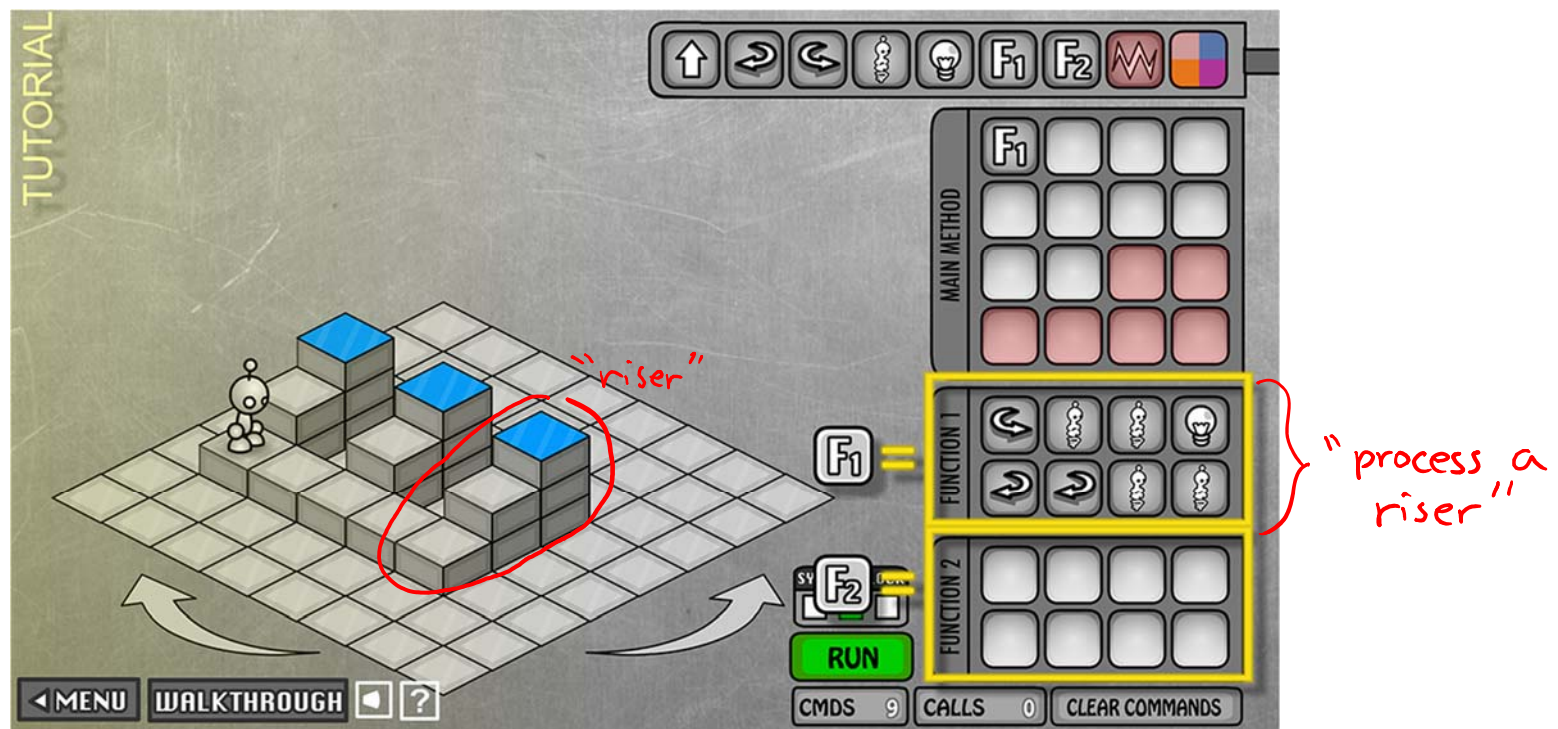
In theory, a computer with just SIX instruction types could compute all known computations!

Back in Reality

- ❖ Programming would be amazingly tedious if you could only use the basic instructions
 - No one would be a programmer no matter how much it paid!
 - The amazing applications we see today would not exist
- ❖ The early days of programming were like this
 - Tedious and error-prone

Solution: Functions!

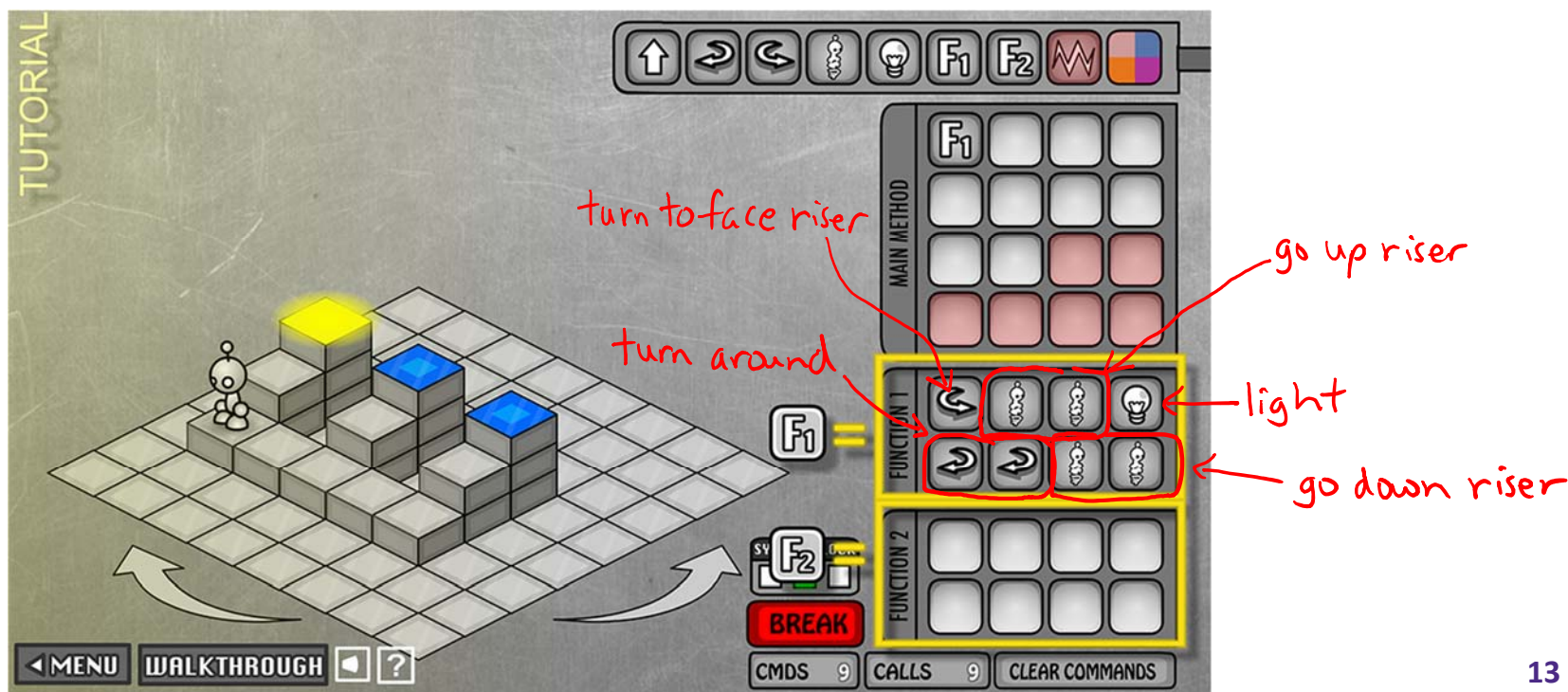
- ❖ **Functions** allow us to create new, more complex instructions for our agents
 - Below, F1 is a function to “process a riser”
 - We can **call** a function by name (F1) to execute its instructions



Choosing Functions

- ❖ The goal is to break down a complex problem into smaller/simpler ones
- ❖ Look for common patterns
 - detail removal
 - generalization
- “Process a riser” looks like a useful sub-problem because there are three of them [DEMO]

abstraction!



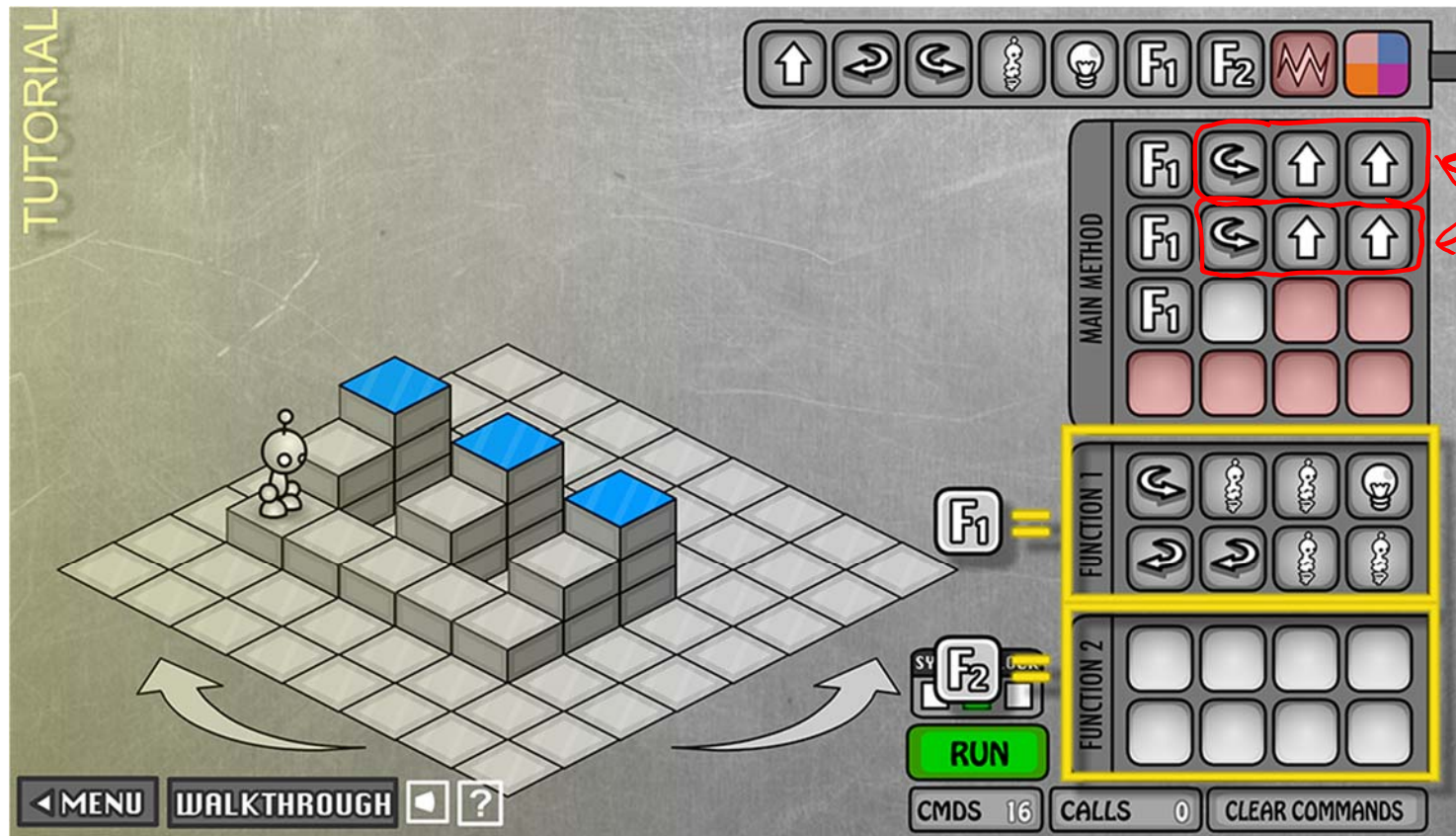
Choosing Functions

❖ One possible solution is shown below:

- 17 commands, 29 calls

size of program ↗

↖ *instructions executed*

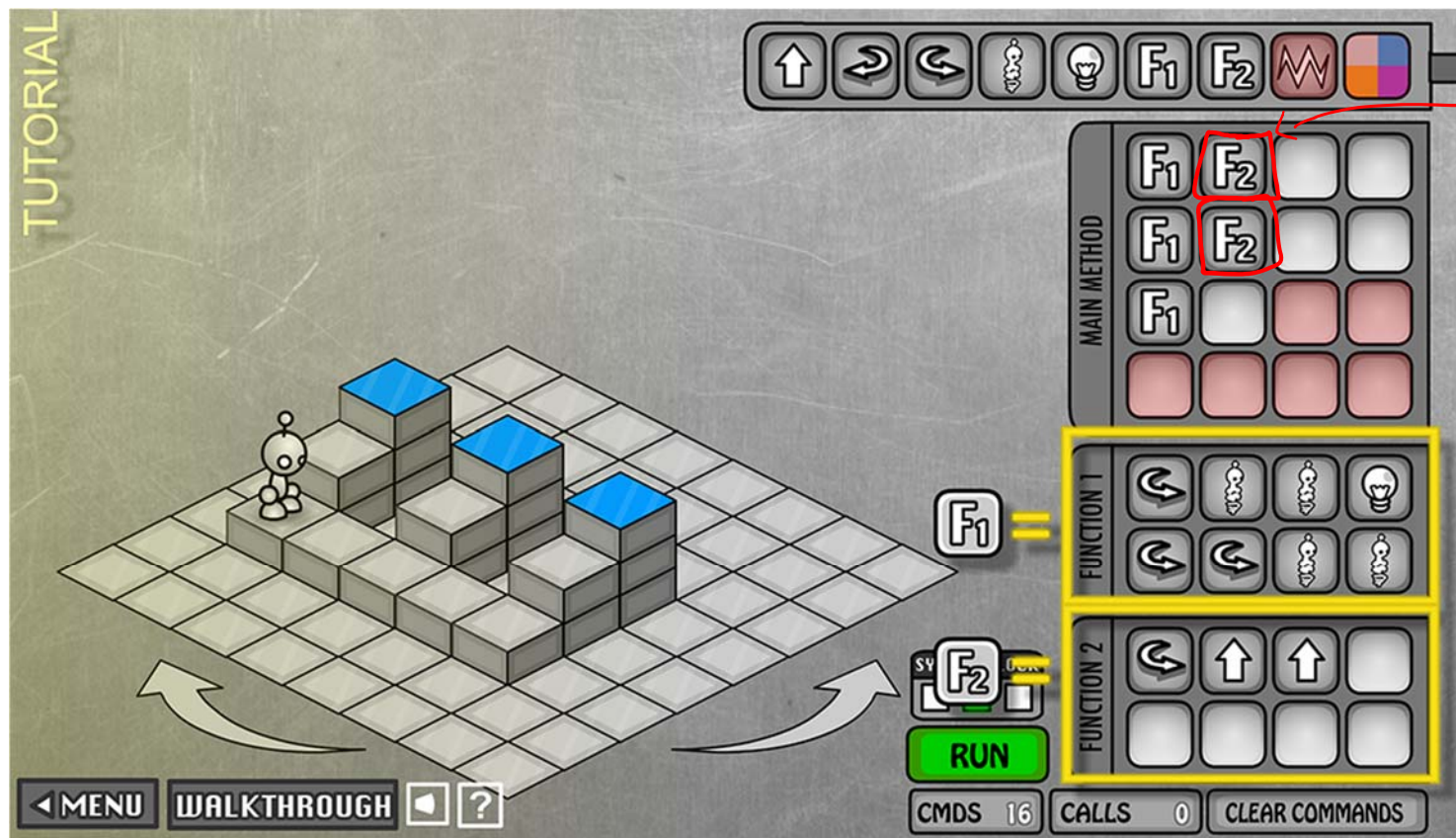


move to next riser

Choosing Functions

- ❖ Modified solution is shown below:
 - Now F2 is a function to “move to next riser”
 - 16 commands, 31 calls

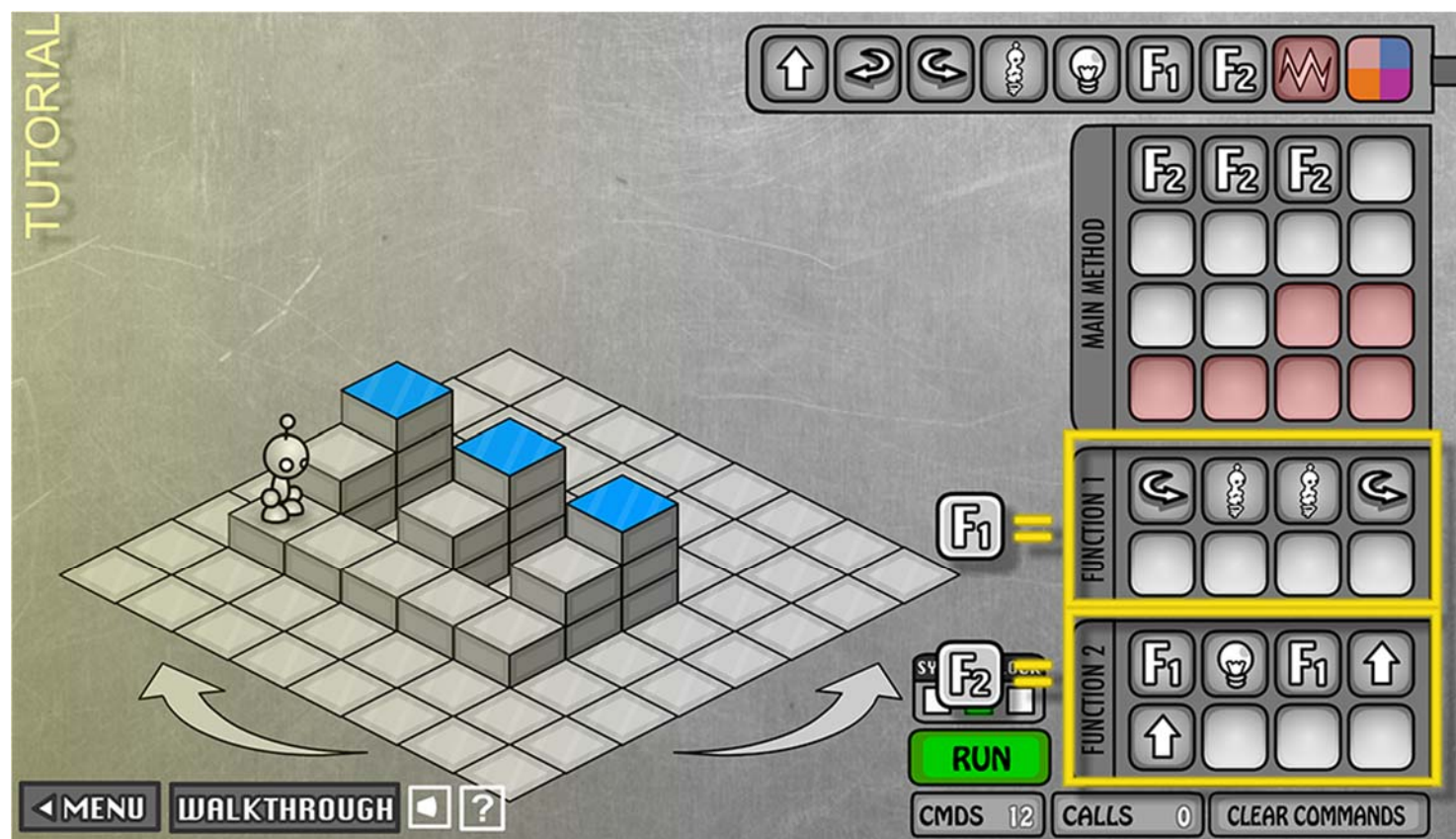
calling a function counts as a call



+2 from here

Choosing Functions

- ❖ Yet another solution shown below:
 - F1 is “traverse steps”, F2 is “process riser and move to next”
 - 12 commands, 35 calls



Peer Instruction Question

❖ Which of the following statements is TRUE?

▪ Vote at <http://PollEv.com/justinh>

A. An agent can learn new instructions

can create new functions, but not new instructions

B. It is the agent's fault if the goal is not achieved

C. All ways to decompose a problem into functions

it's only following instructions


are equally good no because of different metrics

→ execution time (calls)
→ program size (commands)
→ does it work?
→ how easy is it to understand?

D. None of the above

E. We're lost...

Functions Summary

- ❖ Functions may seem “obvious” to you, but they are a foundational idea of computer science
 - Abstraction in action!
- ❖ *Functional abstraction* helps us solve problems:
 - Reduce complexity: identify and solve a coherent activity or action (sub-problem) that can be reused
 - Associate these sub-problems with intuitive names
 - Solve the whole problem by composing functions
- ❖ There is no “correct” way to abstract! 

Looking Forward

- ❖ Continue to explore the concept of programming in the realm of Lightbot
 - Now symbolically – not restricted by computer or clicking
- ❖ Symbolic Lightbot
 - Start in lab tomorrow, due Friday (3/31)
 - Use handwritten symbols for instructions
- ❖ Lightbot Functions
 - Create functions using handwritten symbols
 $F.\text{turnaround}(\) \quad R, R.$