

Lightbot and Functions

CSE 120 Winter 2018

Instructor:

Justin Hsia

Teaching Assistants:

Anupam Gupta, Cheng Ni,
Sam Wolfson, Sophie Tian,

Eugene Oh,
Teagan Horkan

Meltdown and Spectre: Bugs in modern computers leak passwords and sensitive data

“Meltdown and Spectre exploit critical vulnerabilities in modern processors. These hardware bugs allow programs to steal data which is currently processed on the computer. While programs are typically not permitted to read data from other programs, a malicious program can exploit Meltdown and Spectre to get hold of secrets stored in the memory of other running programs. This might include your passwords stored in a password manager or browser, your personal photos, emails, instant messages and even business-critical documents.”

- <https://spectreattack.com/>



Meltdown



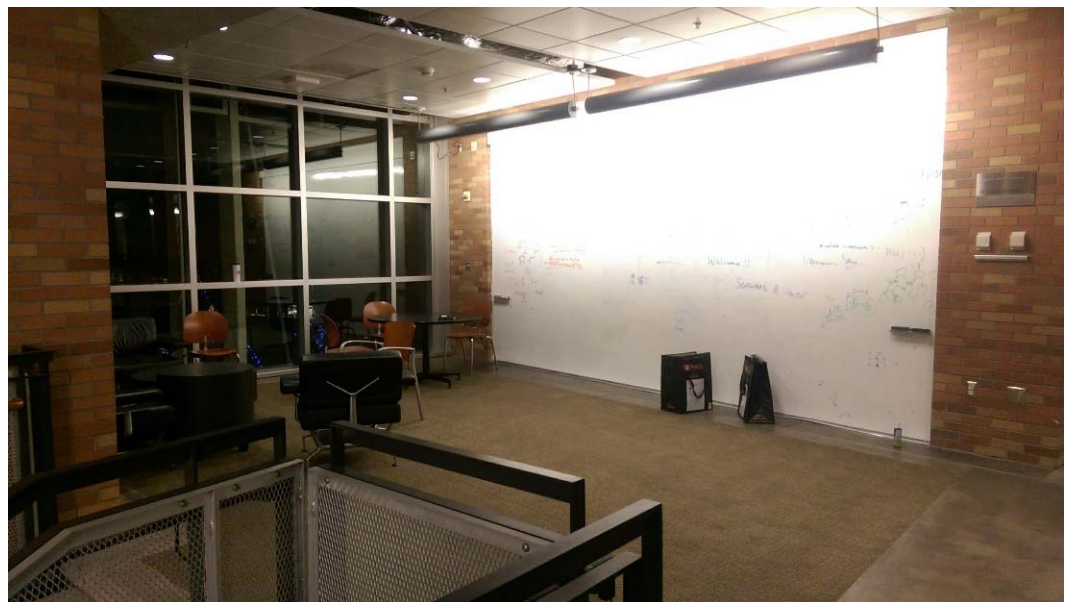
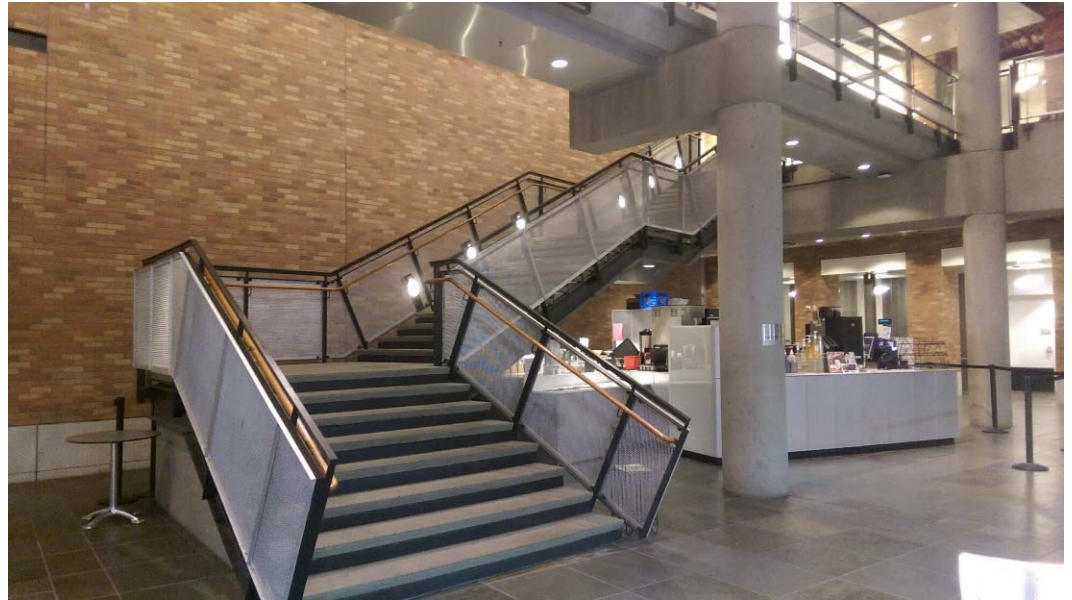
Spectre

Administrivia

- ❖ Website navigation [demo]
- ❖ Lab checkoffs
 - Labs done in pairs, have multiple attempts to complete
 - Generally due before next section; later labs are longer
 - Can get checked off by any staff member in Section or OH
- ❖ Homework submission
 - Submit via Canvas Assignments page
 - Make sure you look at **Canvas rubrics**
 - Lightbot Functions assignment due Monday (1/8)

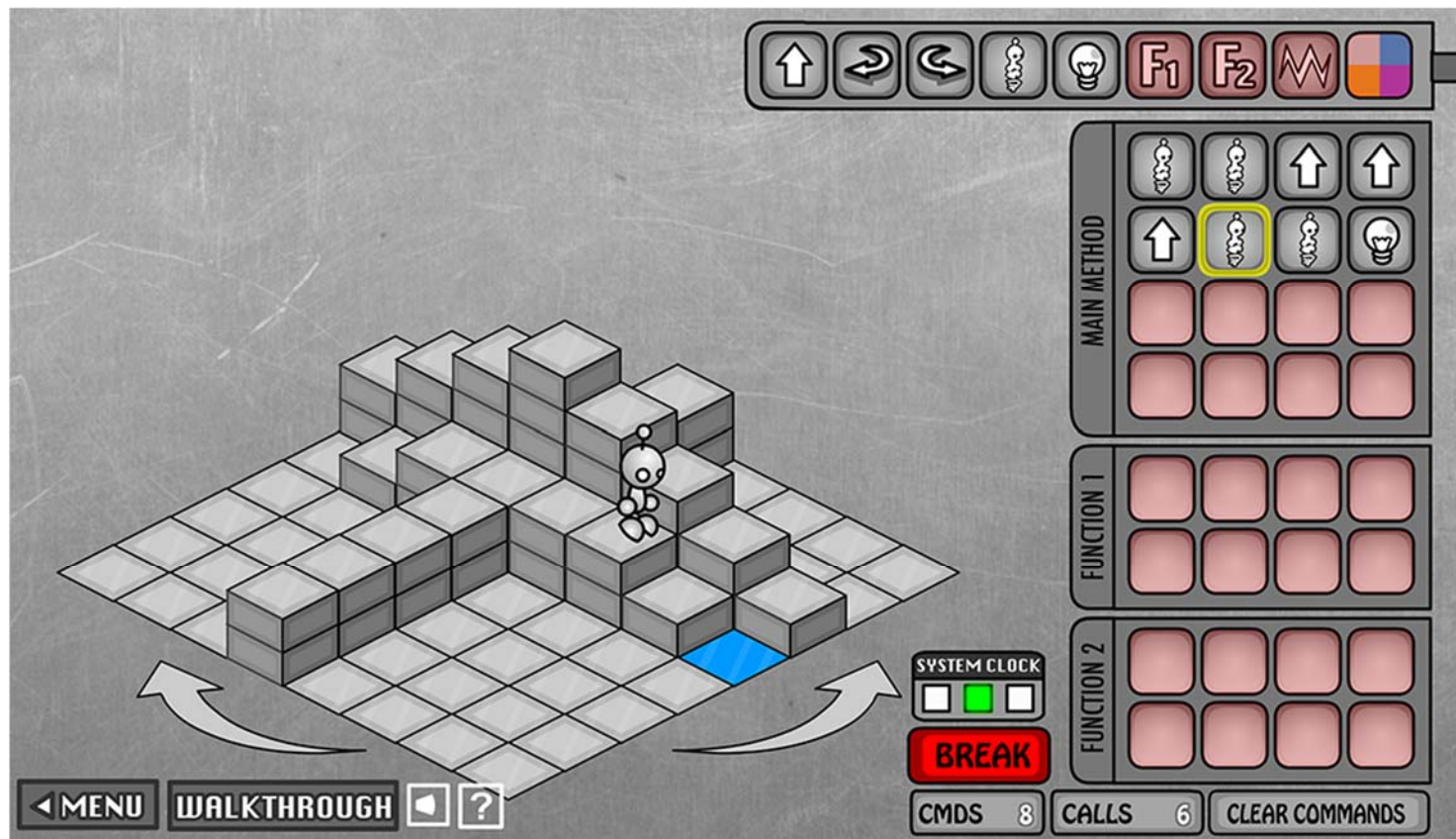
TA Office Hours

- ❖ CSE 2nd floor breakout
 - Up the stairs in the CSE Atrium (next to the café)
 - At the top of that first flight, the open area with the whiteboard wall is the 2nd floor breakout!



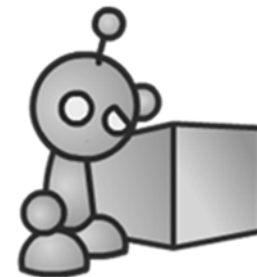
As Experienced Lightbot Players...

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

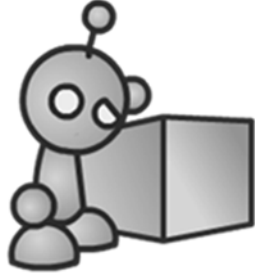


As Experienced Lightbot Players...

- ❖ 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

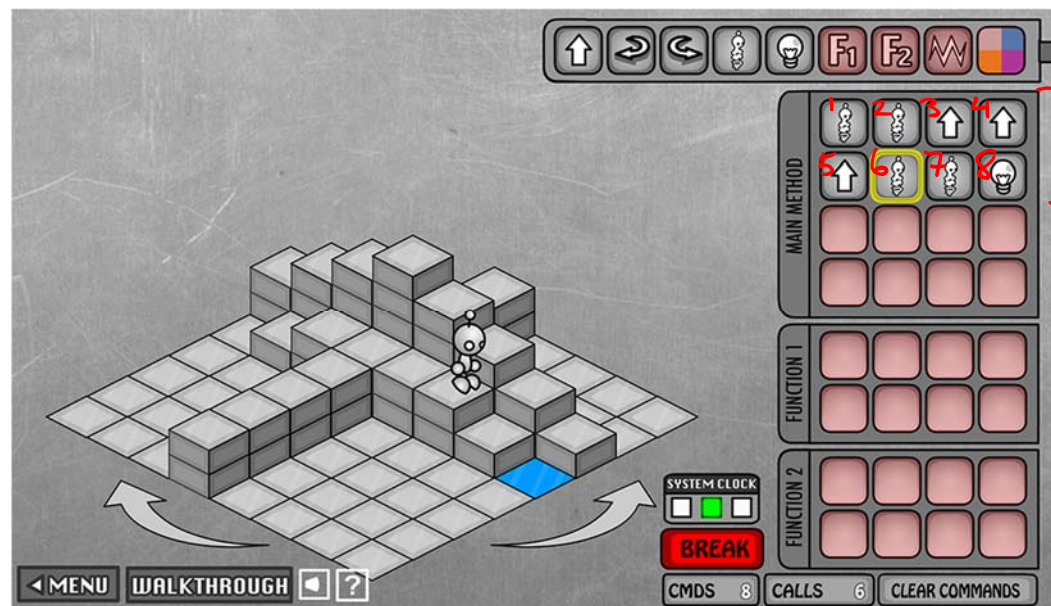


As Experienced Lightbot Players...

- ❖ 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*

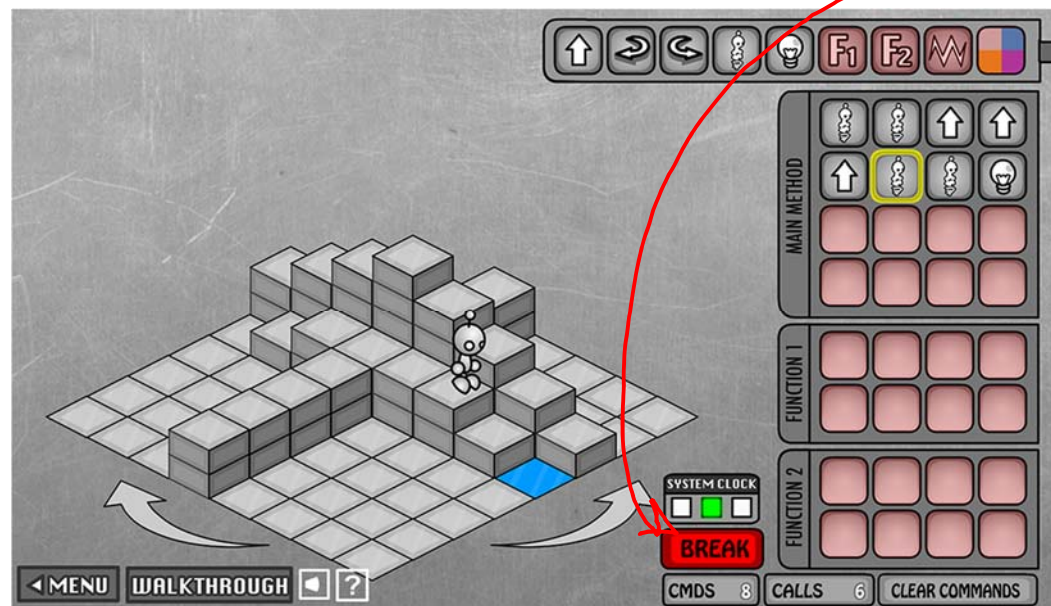
Order of Instructions

- ❖ 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**



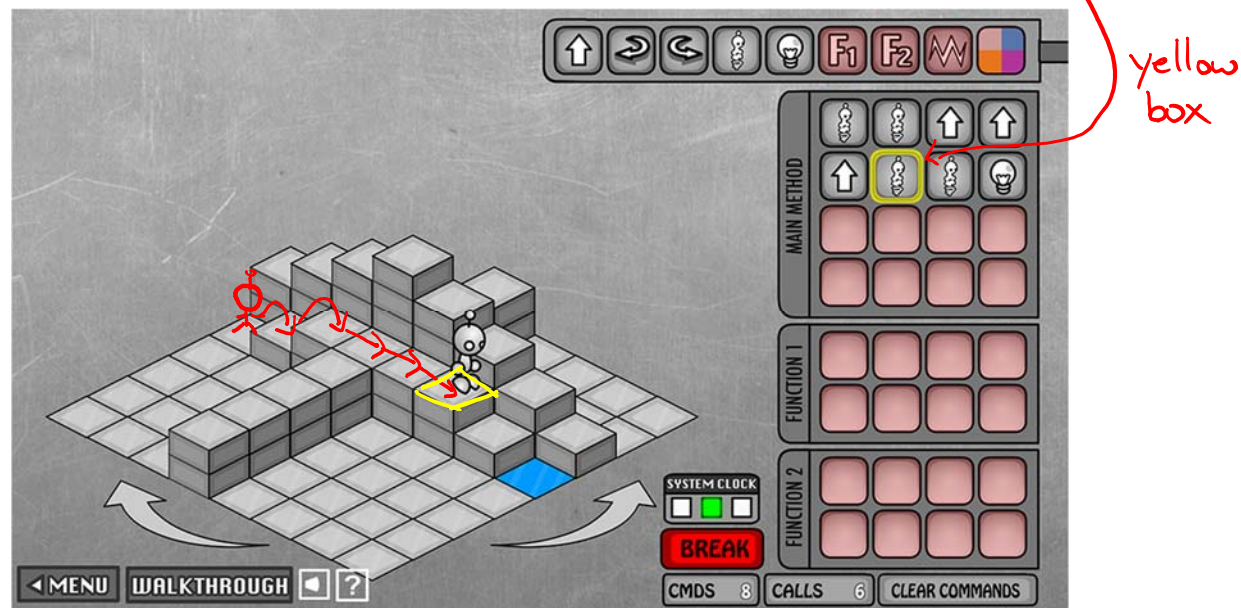
Order of Events

- ❖ 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*)

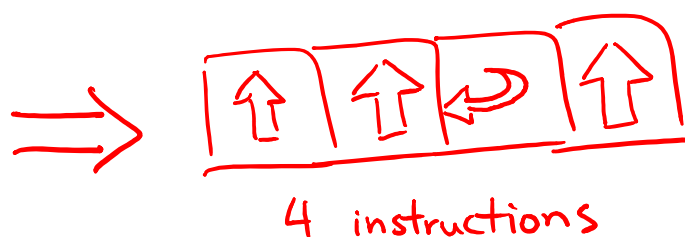


Limited Instructions

- ❖ 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



Light bot

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:

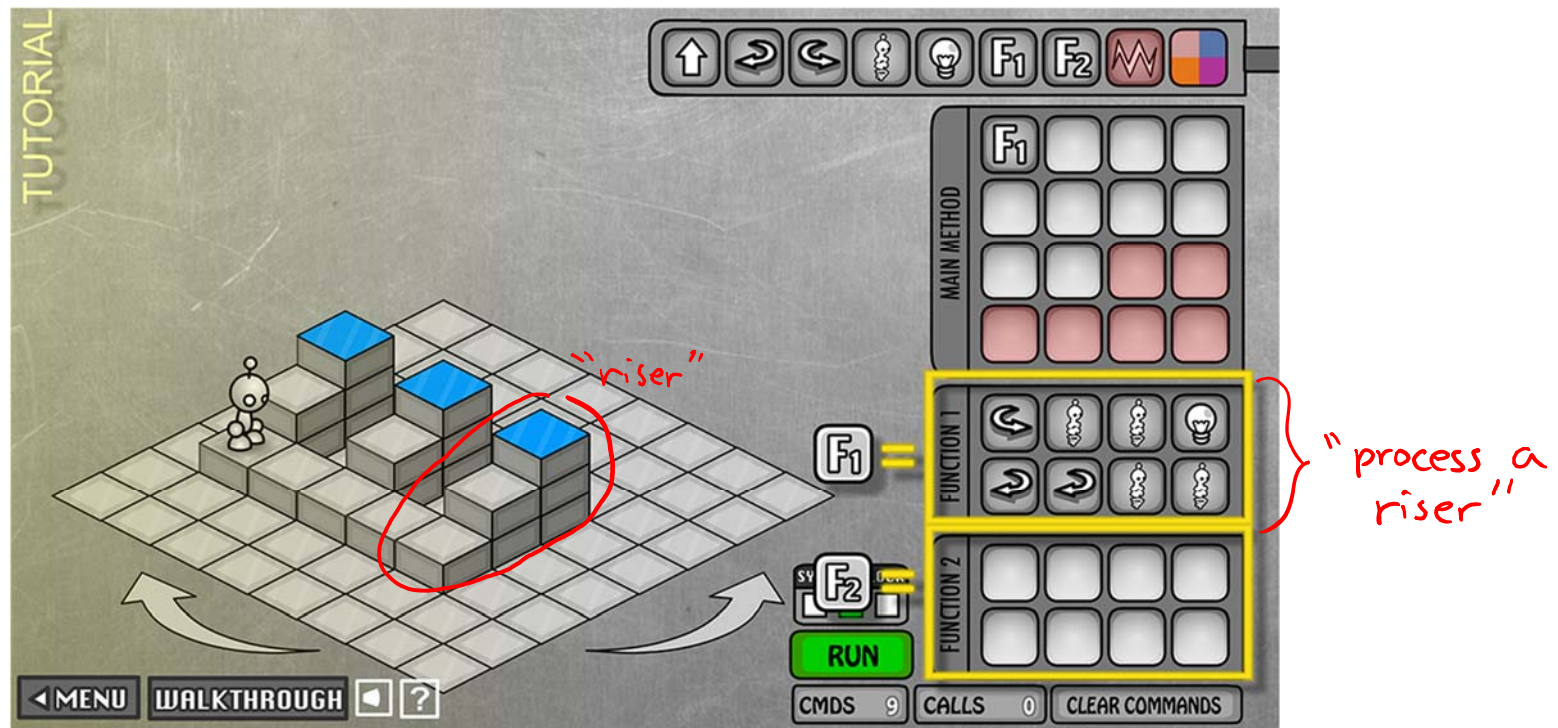
In theory, a computer with just SIX instruction types could compute all known computations!

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

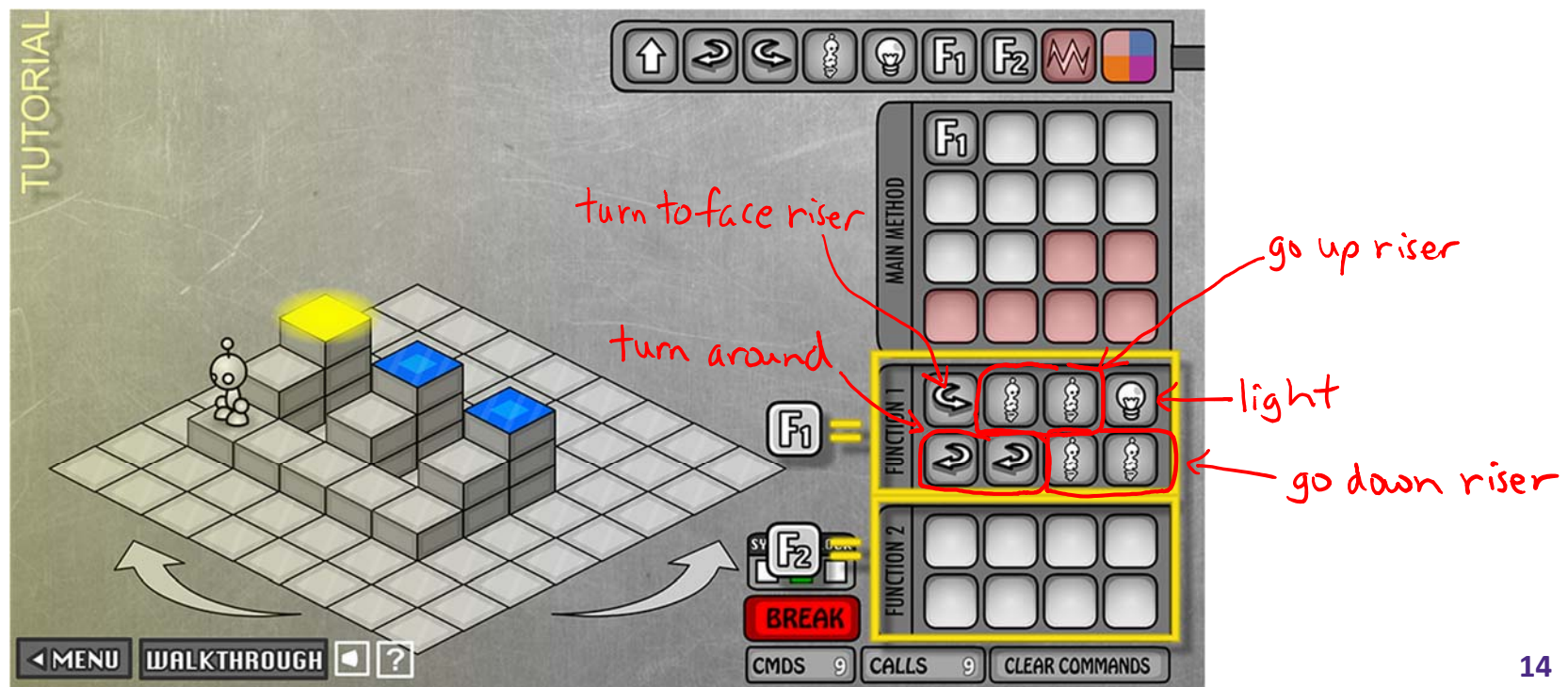
Solution: Functions!

- ❖ **Functions** allow us to create new, more complex subtasks 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 — detail removal
- ❖ Look for common patterns — generalization } abstraction!
- “Process a riser” looks like a useful sub-problem because there are three of them [DEMO]



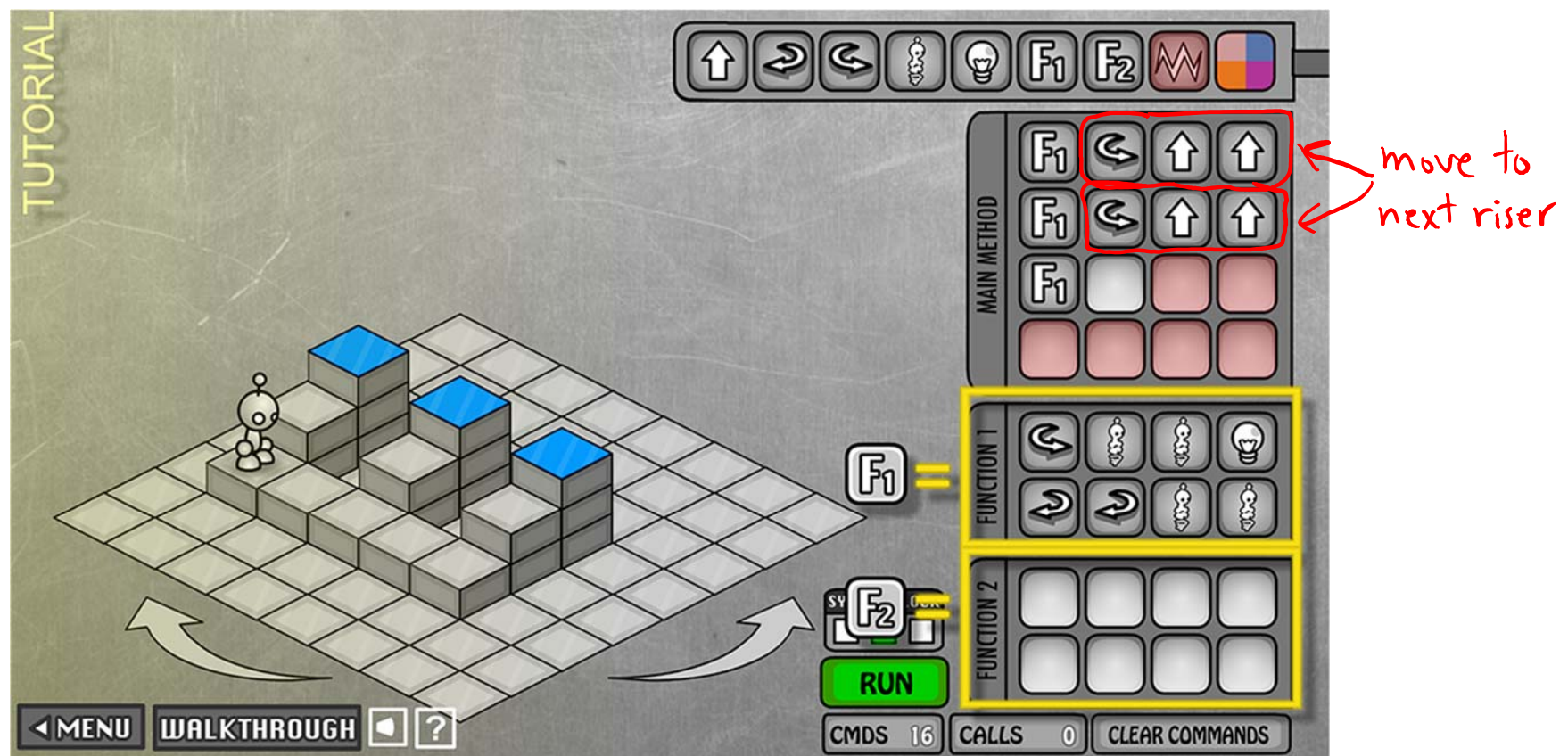
Choosing Functions

❖ One possible solution is shown below:

- 17 commands, 29 calls

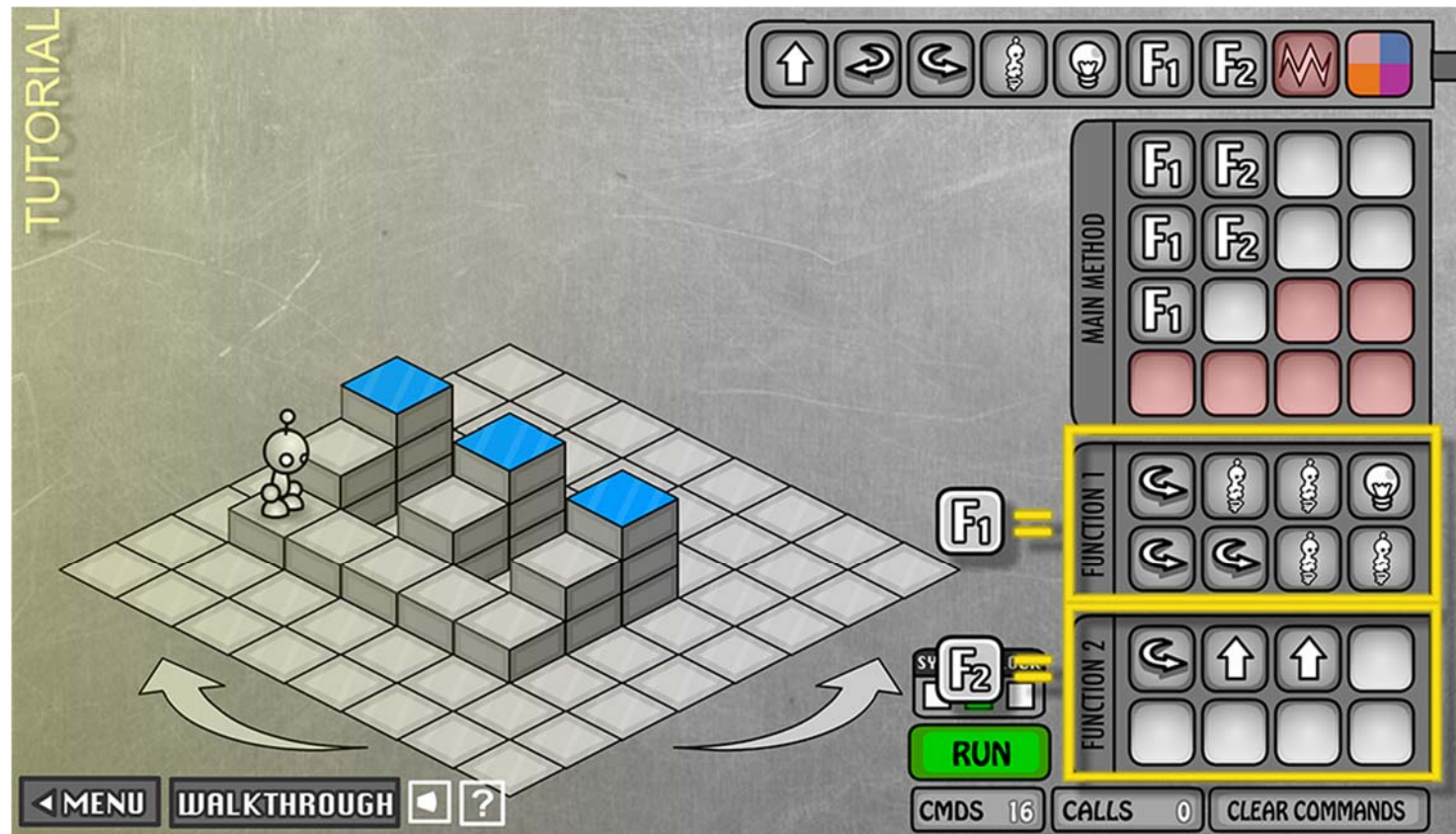
size of program ↗

↗ instructions executed



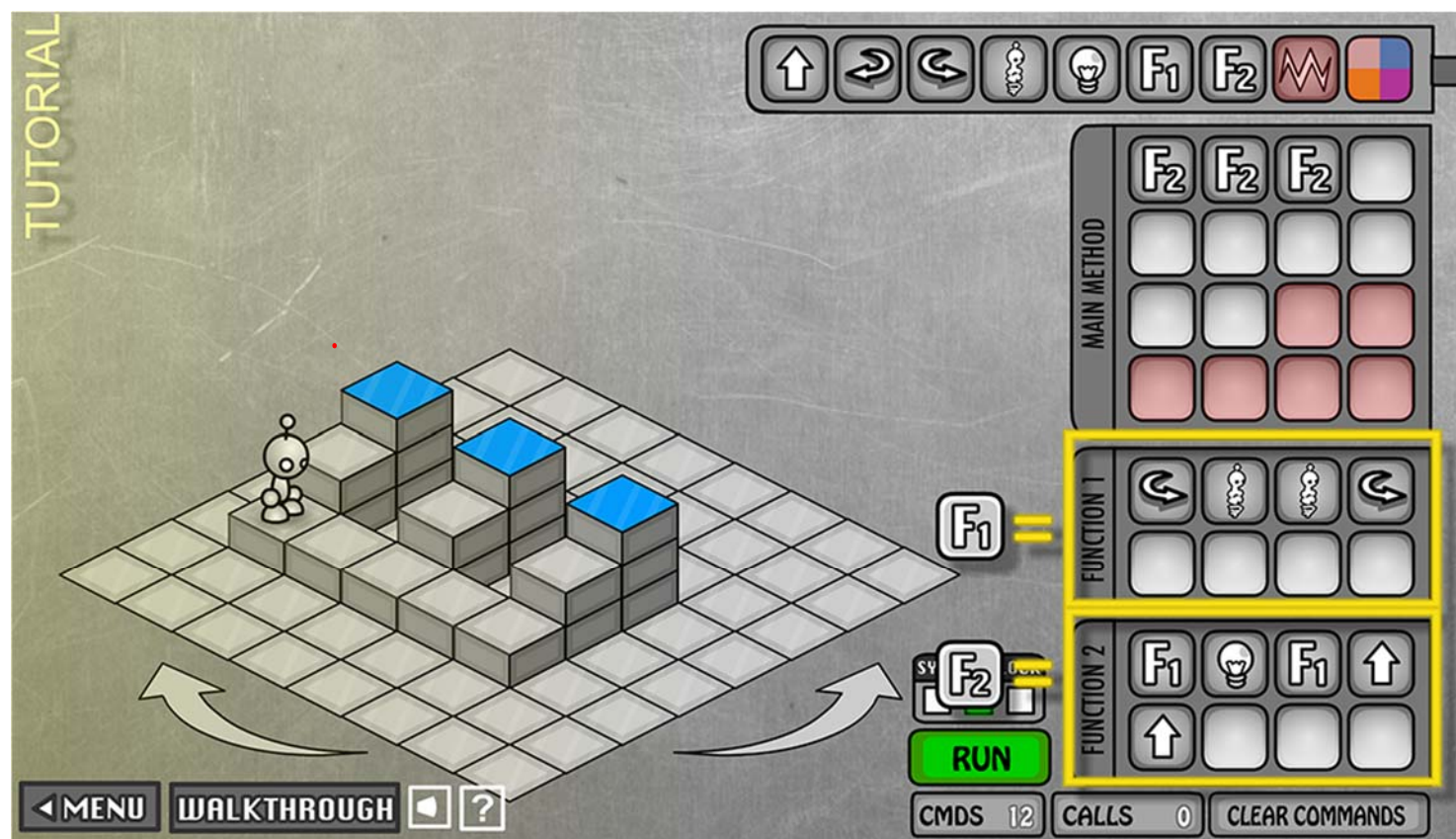
Choosing Functions

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



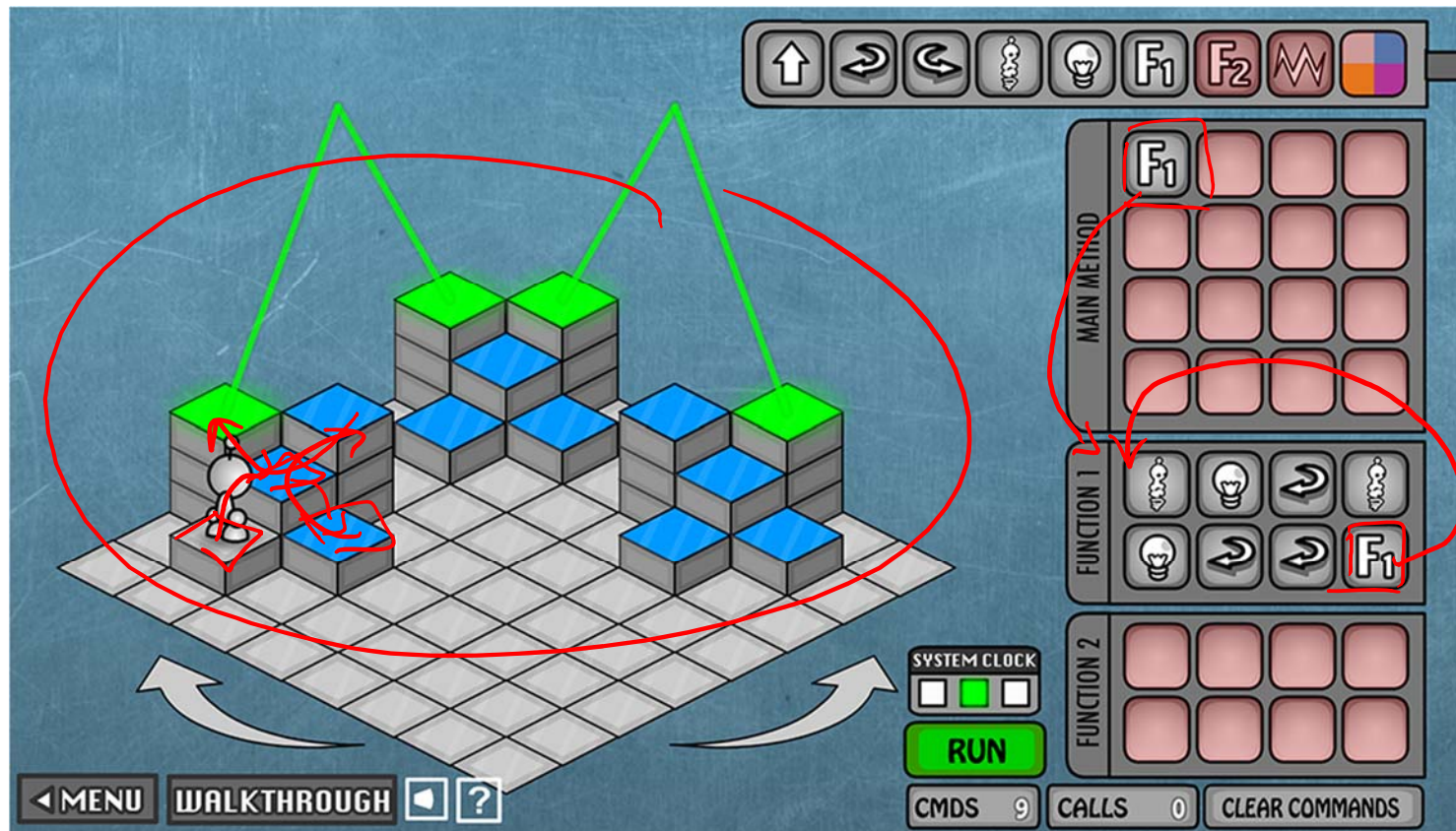
Choosing Functions

- ❖ Yet another solution shown below:
 - 12 commands, 35 calls



Recursion

- ❖ Special case where a function calls itself
 - “Conceptual unit” might apply again, immediately



Peer Instruction Question

❖ Which of the following statements is TRUE?

■ Vote at <http://PollEv.com/justinh>

A. An agent can learn new instructions

↙ no, fixed vocabulary
but can create new
functions/tasks

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

it's just following instructions

C. All ways to decompose a problem into functions
are equally good

no, because of

different metrics:

execution time (calls)

does it work?


program size (commands)

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
 - Lightbot (checkoff) due before lab on 1/9
 - Symbolic Lightbot (checkoff) due before lab on 1/9
- ❖ Lightbot Functions (submitted) due end of 1/8
 - Create functions using handwritten symbols

`F.turnaround() R,R.`