The act of directing a computer to do something ... called programming.

The Lightbot 2.0 game exhibited many properties of programming.

What are you doing in Lightbot?

Commanding a robot through a “blocks world”

Programming is commanding an agent.

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

When the program executes, it doesn’t always do what you intended for it to do. Rather, it does precisely what you told it to do. No ambiguity!

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

An Instruction abstracts those steps

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 instruction is a series of steps

An Instruction abstracts those steps

Abstraction

The word “abstraction” is used a lot in computing.

As a general definition, abstraction eliminates details to focus on essential properties

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
Collecting the operations together and giving them a name is functional abstraction.
- The group of operations perform some function but we ignore all of the details.
- Giving it 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.

The group of operations perform some function but we ignore all of the details.
- Giving it 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.

Because F1( ) “processes a riser,” I think of the programming task as

- Process a riser
- Move to next riser

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 ()

Is that beautiful, or what?

Did everyone see 1 idea, 2 applications?

- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

Here Is What Is Beautiful ...

- 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

Functions Package Computation
- F1( ) packages actions: E.G. “process a riser”

A Five Instruction Program
- It is one concept here (monolithic, but here it is a series of eight instructions
If M.C. Escher handed us a problem … what would we do?

It only simplifies our thinking, the bot still does all the work.

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 & stolidly

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