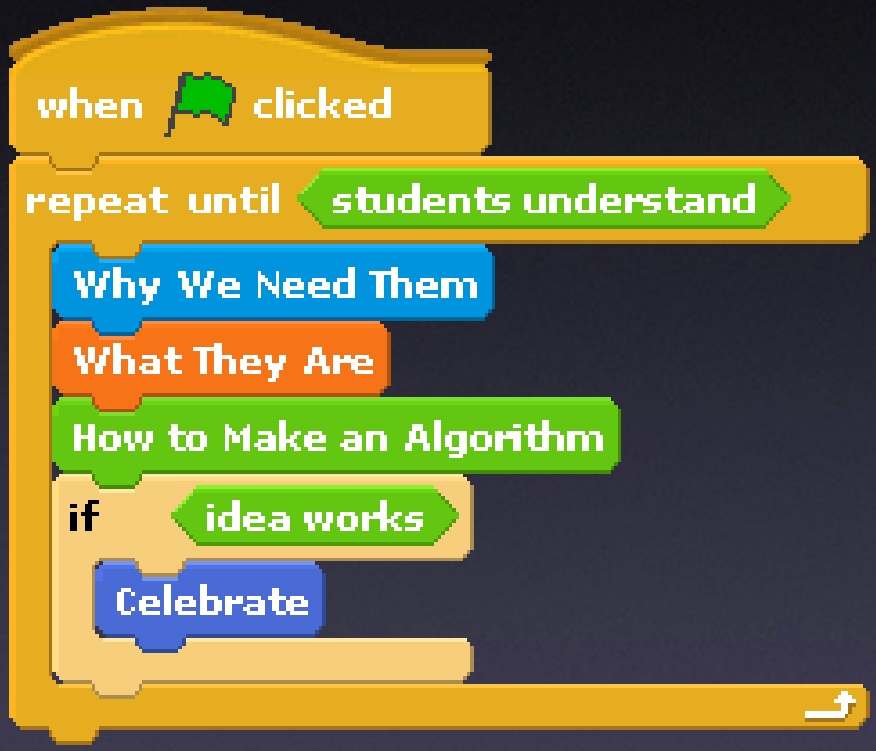
Algorithms

Algo-what?! Why?

Algorithms

The Plan

- Why We Need Them
- What They Are
- How to Make an Algorithm
- Testing Your Idea



Computers are really, really fast.

How fast, you ask?

A reasonably powerful computer in 1961 could perform about 56 multiplication operations in one second.

The most powerful supercomputer in the world today can perform:

a) < 500,000 ops per second

b) Between 500,000 and 5,000,000 ops per second

c) Between 5,000,000 and 100,000,000 ops per second

d) > 100,000,000 ops per second

Really, really fast

233,000,000

operations per SECOND

Really, really fast

2,330,000,000

operations per SECOND

Really, really fast

23,300,000,000

operations per SECOND

Really, really fast

233,000,000,000

operations per SECOND

Really, really fast

2,330,000,000,000

operations per SECOND

Really, really fast

23,300,000,000,000

operations per SECOND

Really, really fast

233,000,000,000,000

operations per SECOND

Really, really fast

2,330,000,000,000,000

operations per SECOND

Doors have opened and there's work to be done.

We've got three ways to get our work done faster:

1. Make more computers.
2. Make new computers faster.
3. Make what we're doing faster.

What is an algorithm, anyway?

An algorithm is any **well-defined computational procedure** that takes some value or set of values as **input** and produces some value or set of values as **output**.

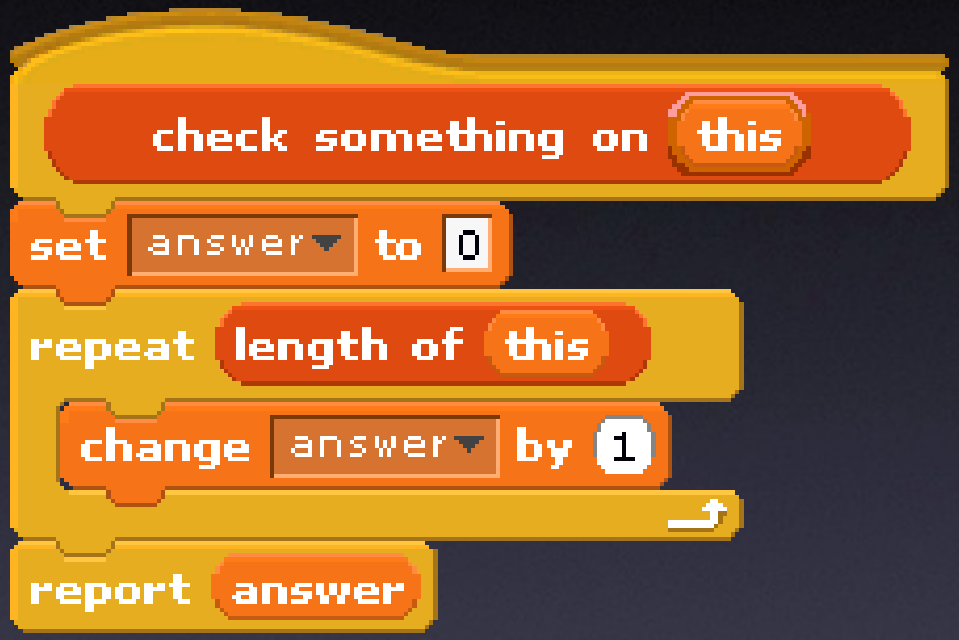
Algorithm for Calling a Friend

Input: friend's phone number

Output: blissful satisfaction

Wait! What <u>assumptions</u> have we made?

Head Scratch-er



What does the Scratch algorithm on the left do to the list *this*?

a) sort the items

b) sum the items

c) reverse the order of the items

d) count the items

How to Make an Algorithm
Part 0 / 4: Bad News

Problem solving Steps from "*How to Solve It*" by George Polya

How to Make an Algorithm
Part 1 / 4: Understand the Problem

What is the general problem
you're trying to solve?

How to Make an Algorithm
Part 2 / 4: Make a plan

There can be many successful strategies for solving the same problem. Some of my favorites are:

Guess and check

Looking for a pattern

Solving a simpler problem

Drawing a picture

How to Make an Algorithm
Part 3 / 4: Carry out the Plan

Now put your plan into action. Stick with it unless you have a reason to believe it will no longer work.

Patience is your best friend on this one.

How to Make an Algorithm
Part 4 / 4: Reflect

Understanding algorithms and where they work best is tough work. Practice is hugely important.

Reflecting on your successes and mistakes will make you improve faster.

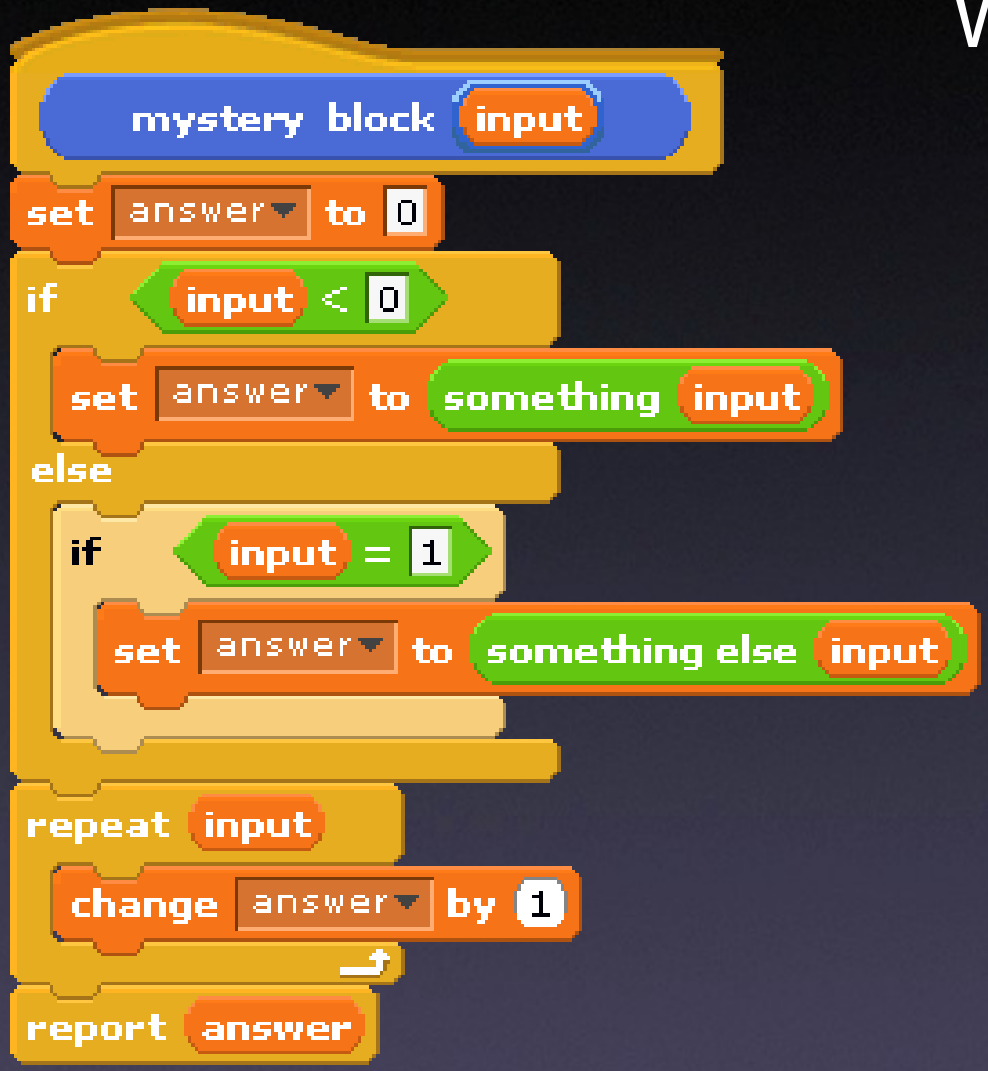
Testing Your Idea

If there is anything wrong with your algorithm / program, your users WILL find it!

Testing is Really Complex

You want to test as many “paths of execution” as possible!

The Best Test



Which of the following sets of values for *num* would make the best test case for this block?

a) 1, 2, 3

b) -1, 1, 3

c) 12, 100, -3

d) -1, -2, -3

e) no testing needed