A High-Level Introduction to Algorithms

- A set of instructions that takes an input and produces an output.
- Must be unambiguous.

Computers carry out sequences of steps.

For example:

- Programs
- Low-level operations
- Translating high-level programs to low-level steps
- Exchanging info between one computer and another (anywhere in the world)
- Reading a html file and turning it into a visible webpage
- Examining each email you receive to check for viruses
- Accepting your cammands and converting them into actions in a game on a console

Knitting patterns tend to be pretty unambiguous, for example. They're a good set of instructions – if you know what it means, (and can perform the actions correctly,) you will produce the desired output.

Cooking recipes are kind of algorithms, but there's ambiguity. What's a "good splosh" of olive oil? What exactly is the "medium" in turn up the heat to medium? There's too much room for interpretation.

An algorithm is a set of instructions which state how a task is to be performed.

It is an ordered, deterministic, executable, terminating set of instructions.

An algorithm is a set of instructions that is:

- ordered
- deterministic
- executable
- terminating

Ordered doesn't just mean a simple sequence. A long division algorithm shows this:

- Layout the division problem
- While there are digits not yet brought down, do:
    - Bring down the next digit and put on end of target (or start a new target if there isn't one)
    - While the target number is less than the divisor and there are digits left to bring down, do:
        - add 0 to top above last digit brought down
        - bring down the next digit and put on end of target
    - Find the most number of times the divisor goes into the target, and write it above the last digit taken down
    - multiply the divisor by the number you just wrote, and write it below the target, aligned on the right
    - Subtract the new number from the target, and make the result the new target.
- When there are no digits left to bring down, the answer is then the top number, with the target as remainder

This is an *ordered* set of instructions using a "while loop".

The instructions must be unambiguous.

They may be clear to you, but are they clear to everyone?

A program must be written in a language to be executed on a computer. You must:

- understand the language
- know how statements will be interpreted
- express yourself clearly in that language

Terminating: has to have an end e.g. counting the number of heads when a coin is flipped *20 times*.

Executable: has to be possible e.g. can't "drive up the steps"

Programming is mostly about algorithms. If you understand the properties described above and can write algorithms with these properties, then you are halfway.

You still need to learn the language to communicate these instructions to the computer.

To talk about collections of data and be precise and unambiguous in algorithm descriptions, we use:

- sets, relations, functions, and logic

How do we talk about conditions, tests, complex instructions, etc., so that we can specify tasks for algorithms, and test that they work?

- sets, logic

Exercise, try the Origami.