COMP 102: Computers and Computing

Lecture 8: Of Arrays and Algorithms

Instructor: Kaleem Siddiqi (siddiqi@cim.mcgill.ca)

Class web page: www.cim.mcgill.ca/~siddiqi/102.html

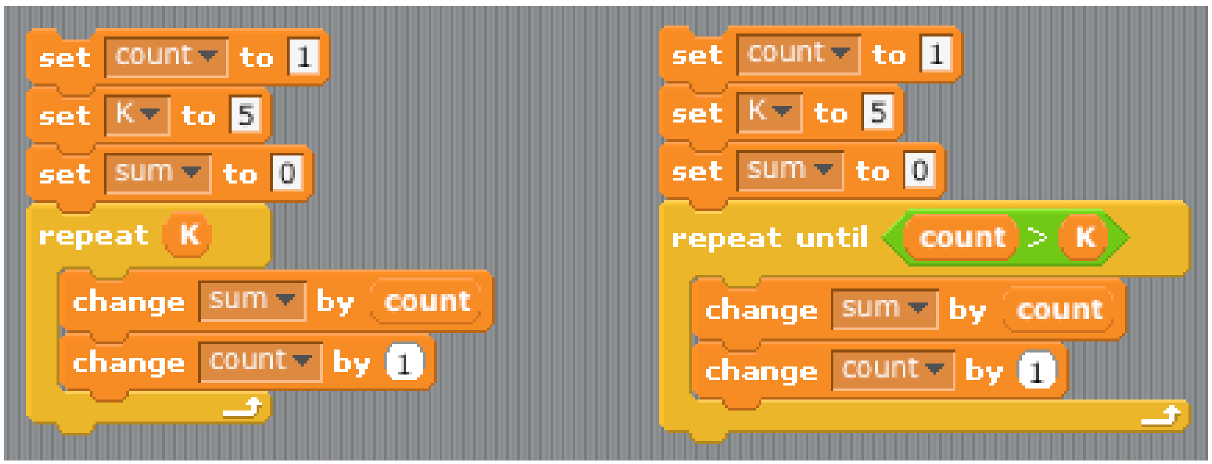
Quick recap of loops and variables

- Example: Calculate the sum of (integer) numbers from 1 to k.

- Can you do this:
  - Using for / while loops?
  - Without using a loop?
  - Using recursion?

Sum of K integers using a loop

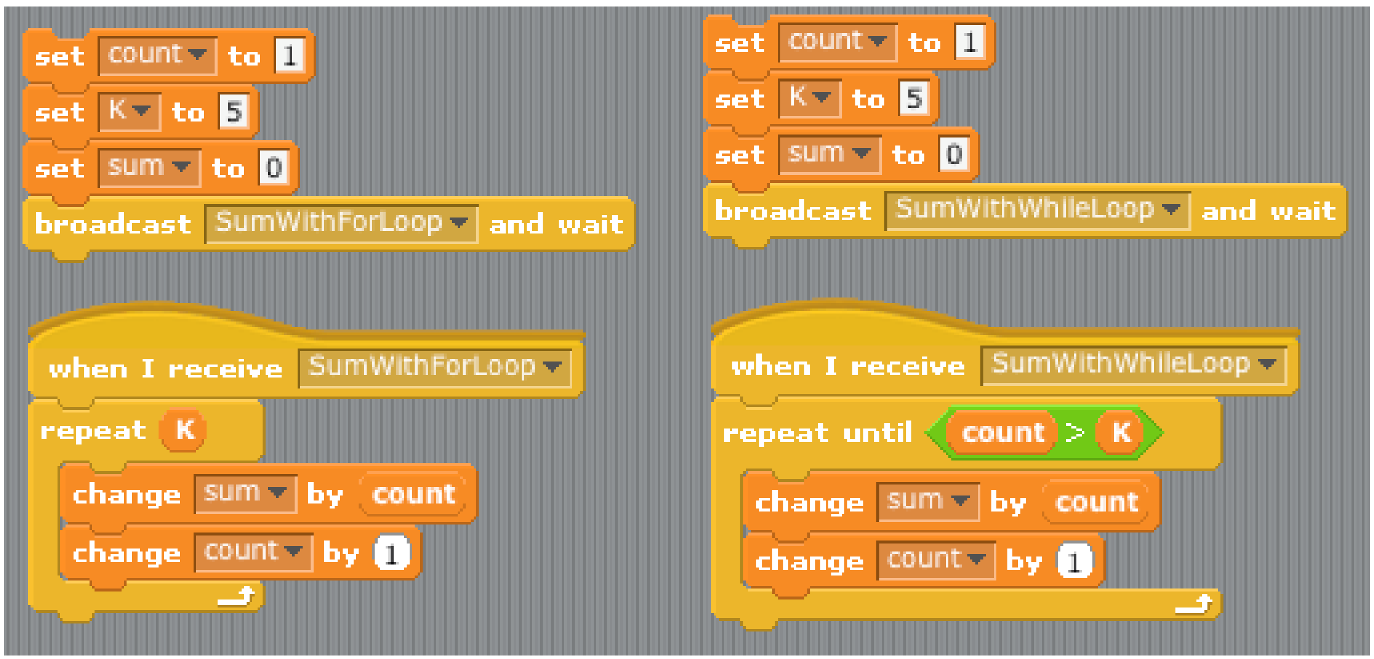
Using a "For" loop: Using "While" loop:



Sum of K integers with a loop

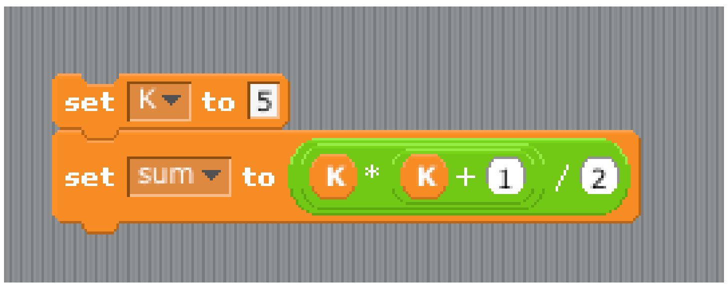
- Can also separate the main calculation into a function:

Using a "For" loop: Using "While" loop:

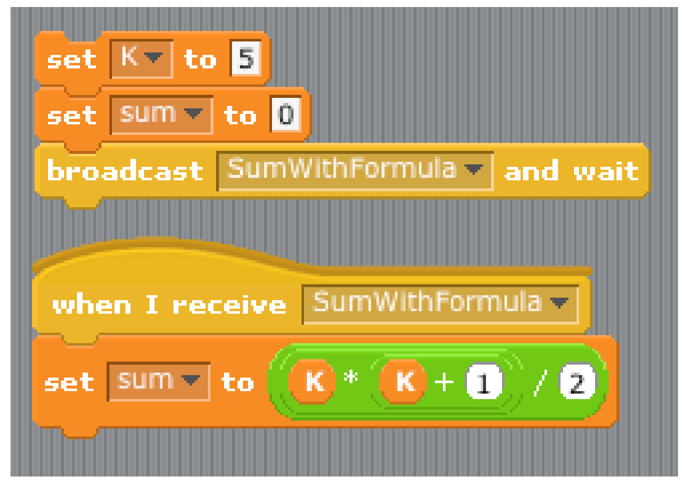


Sum of K integers the easy way!

Without a function:

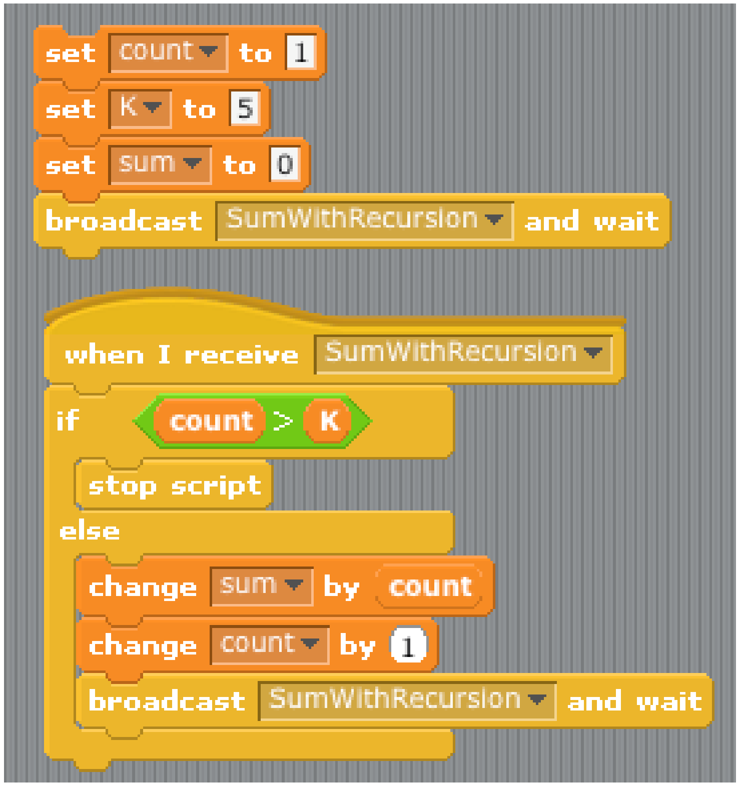


With a function call:



Sum of K integers using recursion

- In this one we definitely need a function call.



Similar way to do this for other languages

Here is how these programs would look in the C programming language:

```
SumWithForLoop(K)
  integer sum, count;
  sum = 0;
  for ( count=1, count<=K, count++)
    sum = sum + count;
  }
  return sum;
```

```
SumWithFormula(K)
  integer sum;
  sum = K * (K+1) / 2;
  return sum;
```

```
SumWithWhileLoop(K)
  integer sum, count;
  sum = 0;
  while ( count<=K )
    sum = sum + count;
    count = count + K;
  }
  return sum;
```

```
SumWithRecursion(K)
  integer sum;
  if ( K==1 ) {
    x = K;
  }
  else {
    sum = K + SumWithRecursion(K-1);
  }
  return sum;
```

A slightly harder problem

- What if we wanted to know this sum for each integer K (from 1 to K)?

  E.g. Sum(5) = 1, 3, 6, 10, 15.

  Does that remind you of anything?   *Babbage's difference engine!*

  How can we do this with modern computers?

- <u>Solution 1</u>: Run our program multiple times:

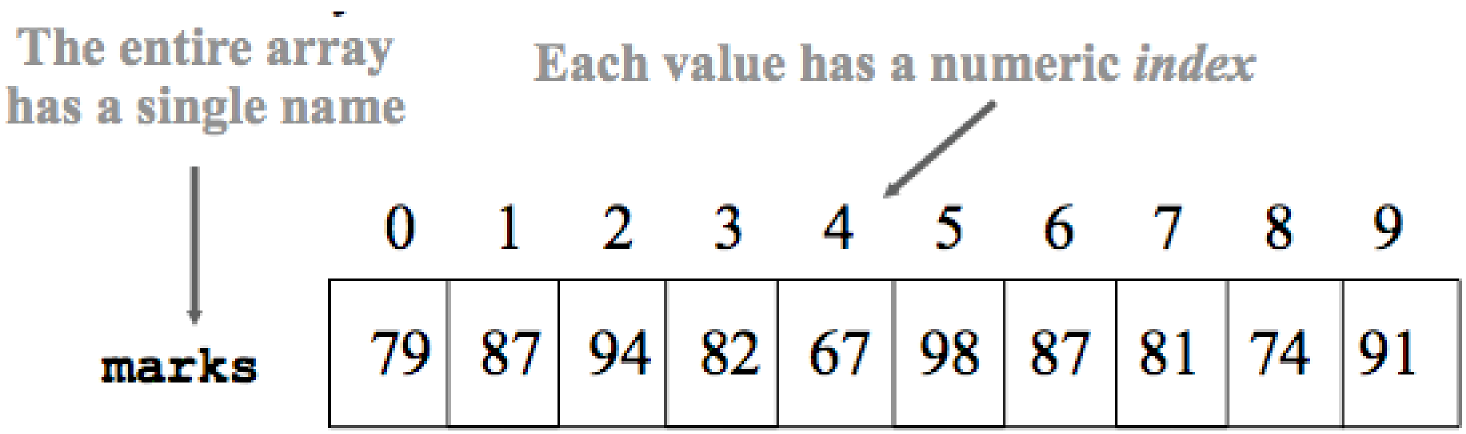
  E.g. Sum(1) = 1, Sum(2) = 3, Sum(3) = 6, …

  – Problem with this? Lots of extra work!

- <u>Solution 2</u>: Modify our program to return many variables.

Arrays

- An array is an ordered list of values.



An array of size N is indexed from 1 to N.

This array holds 10 values that are indexed from 1 to 10.

(In some programming languages, arrays are indexed from 0 to N).

Arrays

- An array stores many values of the same type.

  E.g. integers, real numbers, characters

- An array is given a name.

- A particular value in the array can be accessed, e.g. to read or modify the value.
  - To access the value, we need to call the array name and the index of the particular element we are interested in.

Declaring Arrays

- How do we tell the computer we want an array?
  - Recall for <u>single variables</u>, we need to specify 1 thing: <u>type of data</u>

    E.g. *integer x;*

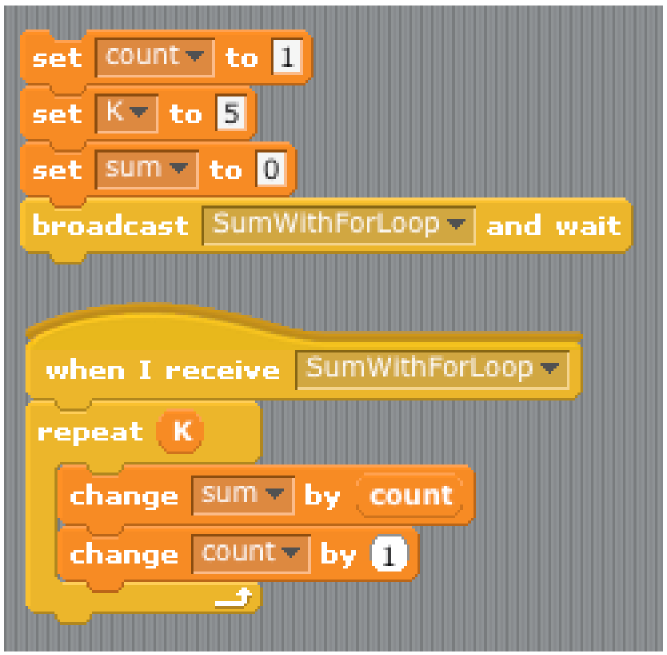
  - For <u>arrays</u>, the computer needs to know 2 things: <u>type of data</u> and <u># of data units</u>.
    - E.g. Reserve a block of memory, sufficient to store 50 integers.

- This only apply to *some* programming languages (e.g. Java, C).
- Other programming languages (e.g. Scratch) don't require you to specify the <u>size</u> or <u>type</u> of the array, only its <u>name</u>.
  - The computer automatically adjusts the amount of memory allocated as you add elements to the array.

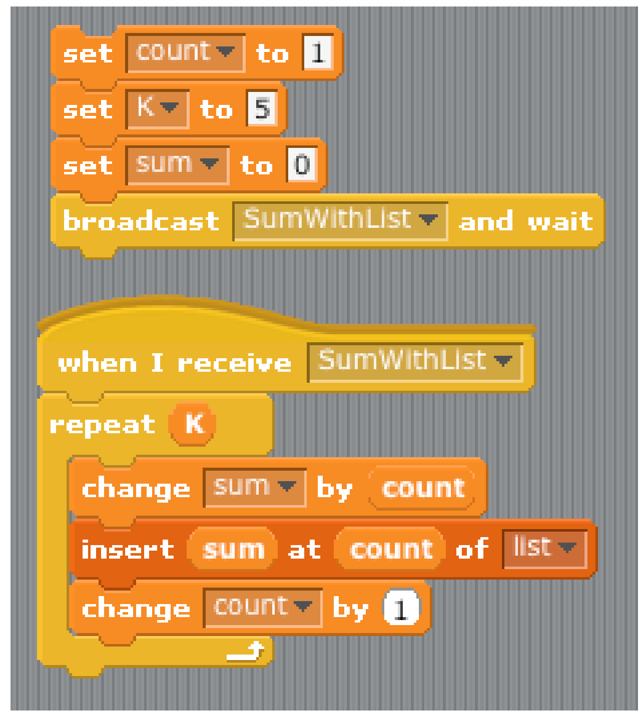
Back to our example

- Calculating the sum of integers 1 to K, and storing the result for each integer.

Standard "For" loop:



With a list:

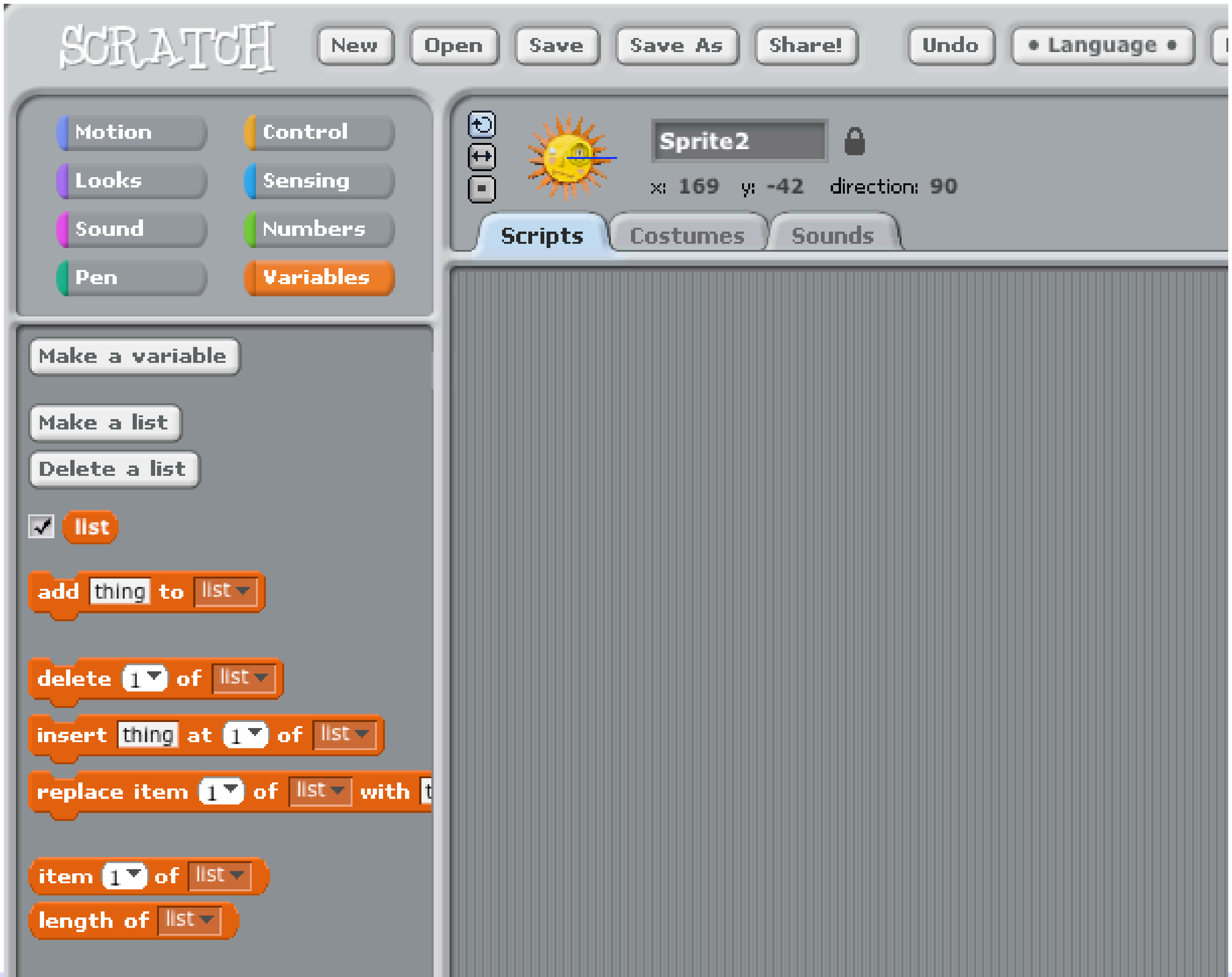


** Don't forget to create a list variable first!

Creating a list variable in Scratch

Click here.

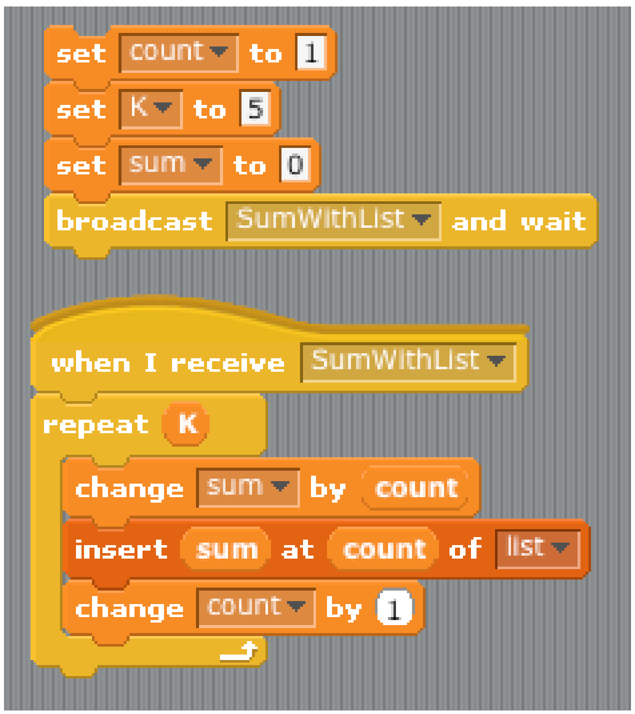
Creating a list variable in Scratch



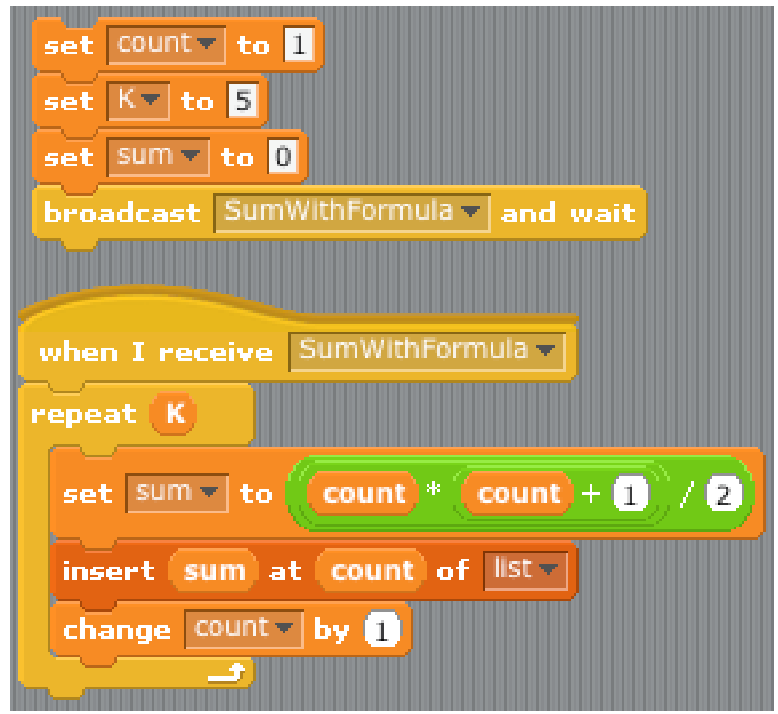
Back to our example

- Can we do this using the formula? Sure! But is it worth it?

Using a "For" loop:



Using the formula:

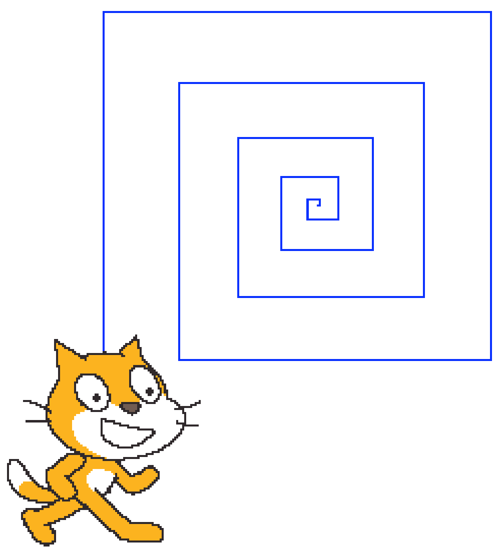


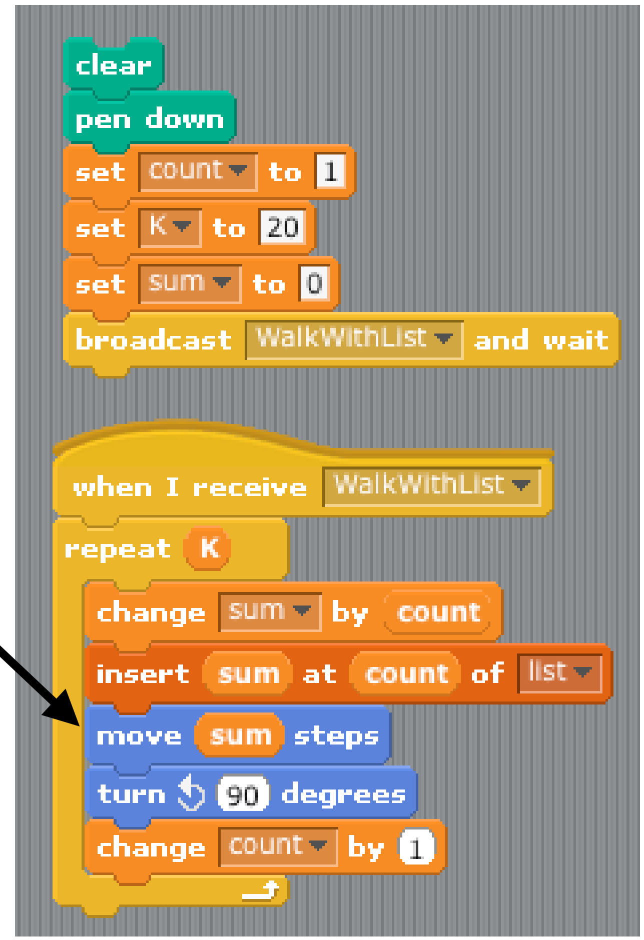
Using this array

- Get the cat to walk around in a spiral

  by <u>accessing the values</u> in the list:

If you run the code,
you'll see this output:





Many uses of arrays

- Storing data (e.g. grades, census information, appointments, …)
  - Remember that the values don't always have to be numbers.

  E.g. List of names: [alice, bob, clara, daniel, ella, fred, gina]

  List of characters: ['a', 'e', 'i', 'o', 'u']

  List of lists… (this gets a little more complicated…)

- Sorting data:
  - Alphabetical/numerical order, increasing/decreasing, etc.

- Searching for data:

  E.g. Looking for a word in a dictionary, looking for a number in a phone book.

What about more complicated tasks?

There are many tasks involving arrays

- Database of course grades.
- Matrix multiplication.
- 3D brain imaging.
- Etc.

For many of these, we need multi-dimensional arrays. This is a little more complicated, but not much.

But for now let's focus on <u>solving problems</u> involving lists.

Algorithm

- An algorithm is a definite procedure for solving a given problem or performing a given task.

- Origins of the word:
  - 9th century Muslim mathematician Abu Abdullah Muhammad ibn Musaal-Khwarizmi whose works introduced Arabic numerals and algebraic concepts.
  - The word algorism originally referred only to the rules of performing arithmetic using Arabic numerals.
  - Evolved via European Latin translation of al-Khwarizmi's name into algorithm by the 18th century.

Algorithm Design

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process.

- May be described:
  - Abstractly, using human language (we call this *pseudocode*) to describe the steps for carrying out some procedure using a computer.
  - Using a programming language of your choice.
  - By providing a set of machine instructions to be executed.

Algorithm Design

- **Pseudocode** is a programming language independent description of the sequence of steps necessary to solve a problem.

- Algorithms that are written in pseudo-code may be then translated into a particular programming language to make a computer program.

- A programmer may come up with his/her own algorithm, or (s)he may implement an existing algorithm

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process.

- Is the following an algorithm?

  Calculate 1/3 exactly

- No, because 1/3 = 0.333333... and this algorithm does not terminate.

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process.

- Is the following an algorithm?

  Find the minimum

- No, because it is ambiguous: minimum what?

Algorithm

- An **algorithm** is an ordered set of unambiguous, executable steps, defining a terminating process

- Is the following an algorithm?

  Find the third largest number in the list {3, 5}

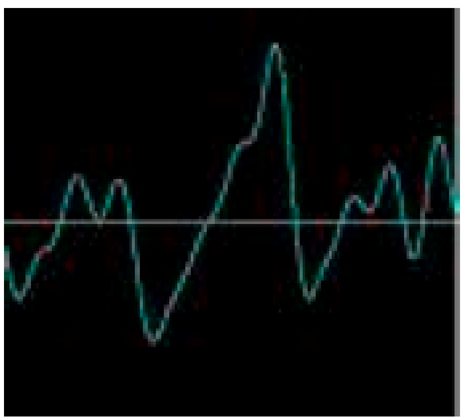
- No, because it is not executable.

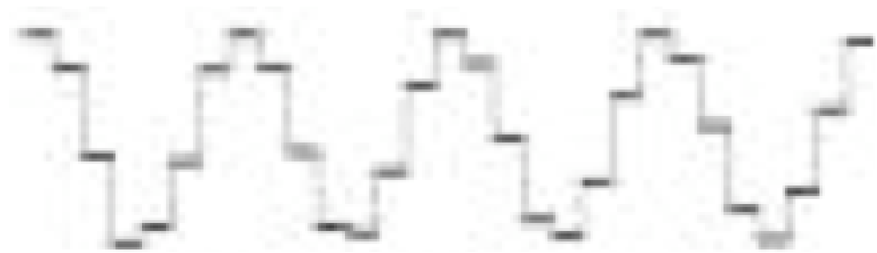
Example

- Given a list of numbers, find the smallest one and its position in the list.
- This is a precise problem.
- We can write an algorithm to do this.

Why would I want to do this?

- Consider finding the minimum (and maximum) of a sound signal to calibrate the signal (e.g. re-scale to match preset max/min values).



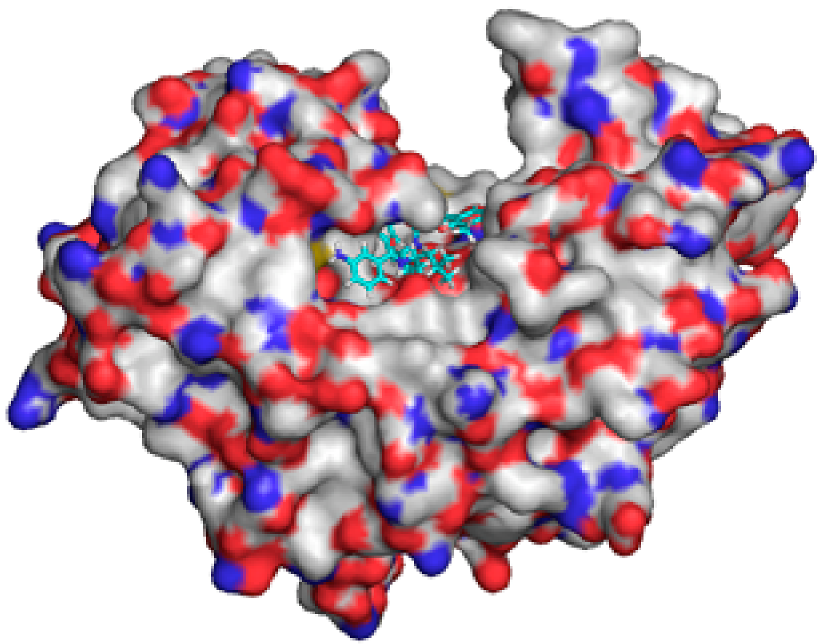


Why would I want to do this?

- Analyze stock market, to estimate minimum stock price over a given time period.

Why would I want to do this?

- Finding the best site for molecular docking is an important aspect of drug development.



http://www.macresearch.org/molecular_docking_on_openmacgrid_part_i

Finding a Minimum - in pseudo-code

- Given $x_1 x_2 \ldots x_K$, find i such that $x_i \le x_j$, $1 \le j \le K$.

- Input: $x_1 x_2 \ldots x_K$

- Compute:

  MinIndex = 1 — Variables

  MinValue = x_1

  for i = 2 to K do — Loop

      if x_i < *MinValue* — Conditional

          MinValue = x_i

          MinIndex = i

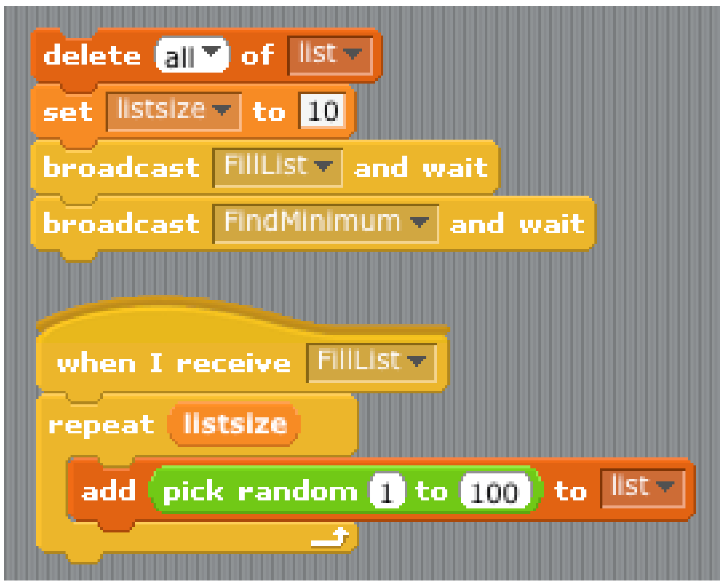
      End if

  End for loop

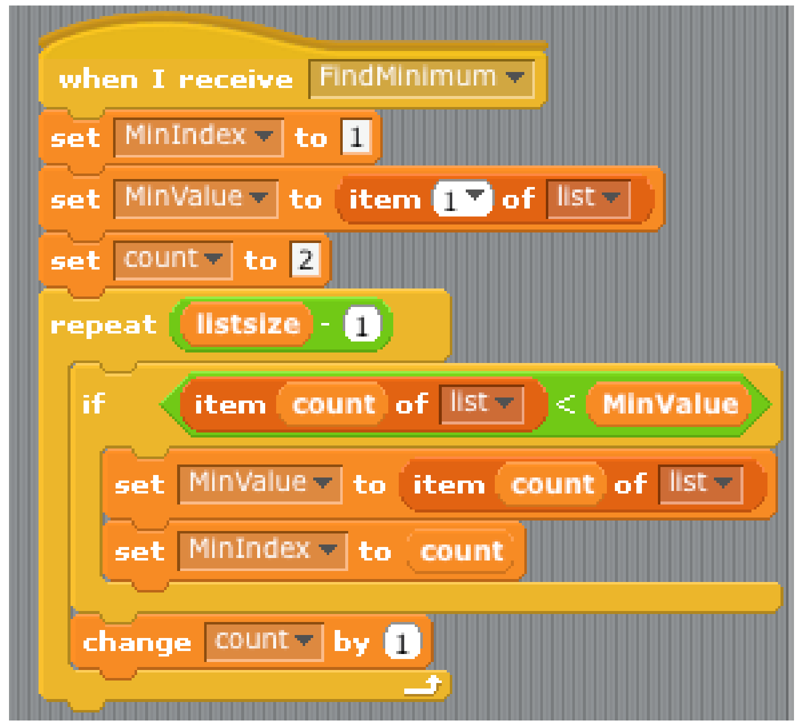
- Output: *MinIndex*, *MinValue*

Finding a Minimum - in Scratch

First fill the list:



Then go through it to find the minimum:



Take-home message

- Understand the concept of list, how it is defined, what it contains.
- Understand the basic notion of an algorithm.
- Know the difference between an algorithm and a program.
- Understand the algorithm for finding the minimum in a list.

Final comments

- Coming weeks:
  - Study examples of problems (and their algorithms) for searching, sorting, making graphs, encoding text, playing games, …

- Some material from these slides was taken from:
  - *http://www.cs.mcgill.ca/~crepeau/COMP102/*
  - *http://www.cim.mcgill.ca/~sveta/COMP102/Lecture16.pdf*