COMP 102: Excursions in Computer Science

Lecture 8: Arrays and Algorithms

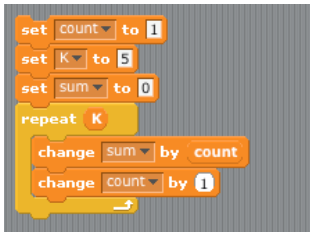
Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp102

Quick recap of loops and variables

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

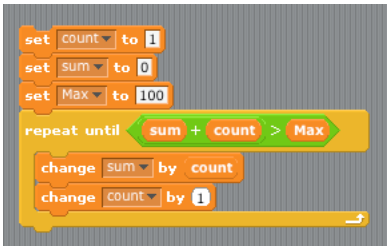
Using a "For" loop:



Summing integers (differently)

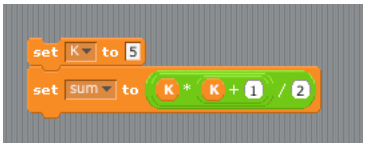
- Example 2: Stop when the sum reaches a maximum.

Summing integers until the total=100:

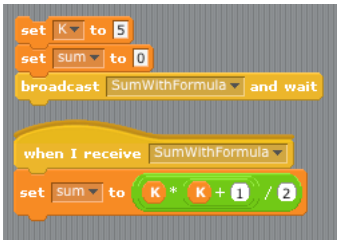


Sum of K integers the easy way!

Without a function:



With a function call:



Similar way to do this for other languages

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

```
SumIntegers(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;
```


A slightly harder problem

- Example 3: Calculate 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?

- Solution 1: Run our program multiple times:

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

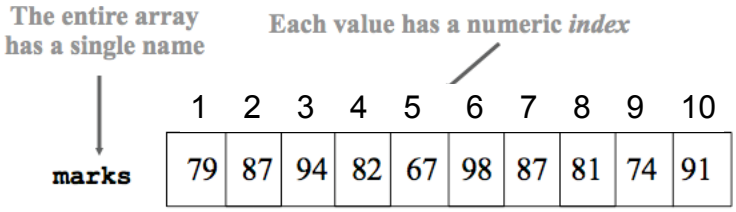
  Problem with this?

  Lots of extra work!

- Solution 2: Modify our program to return **many** variables.

Arrays

- An **array** is an ordered list of values. Sometimes called a **list**.



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.

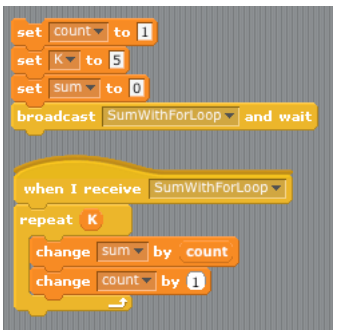
Declaring Arrays

- How do we tell the computer we want an array?
  - For single variables, need to specify 1 thing: type of data
    E.g. *integer x;*
  - In some programming languages (e.g.Java, C), you need to also specify: type of data AND # of data units.
    - The computer then reserves a sufficient block of memory (e.g. to store 5 integers.)
  - Other programming languages (e.g. Scratch) don't require you to specify type or size, just the array's name.
    - The computer automatically adjusts the amount of memory allocated as you add elements to the array.

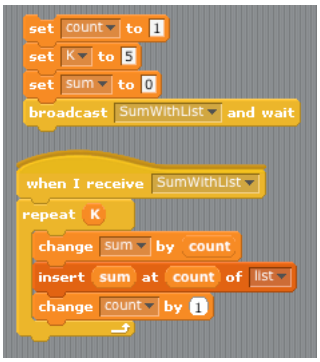
Back to our example

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

Standard "For" loop:



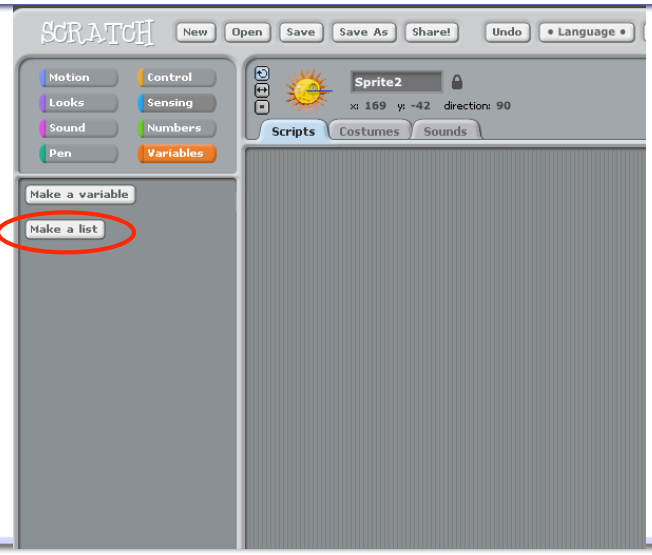
With a list:



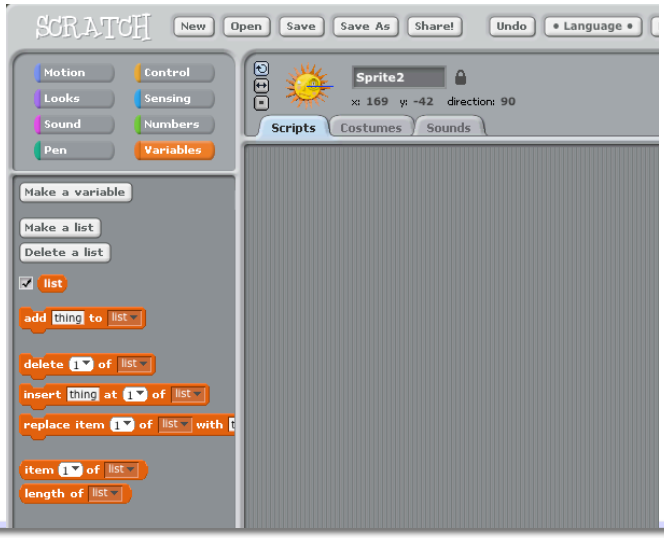
** Don't forget to create a list variable first! **

Creating a list variable in Scratch

Click here.



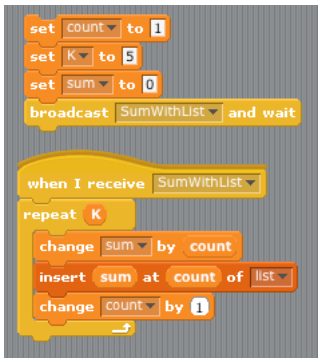
Creating a list variable in Scratch



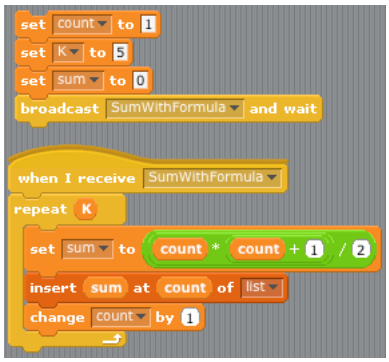
Back to our example

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

Using a "For" loop:



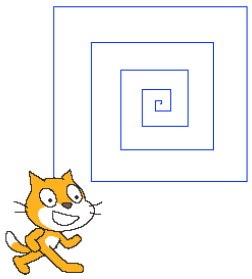
Using the formula:

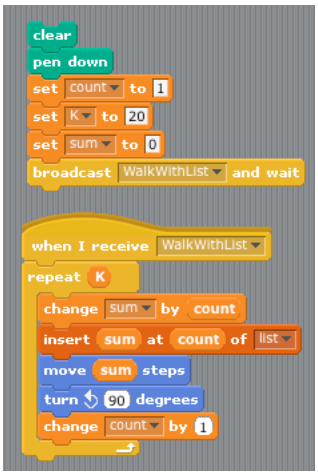


Using this array

- Get the cat to walk around in a spiral using the same values 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, …)

  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…)

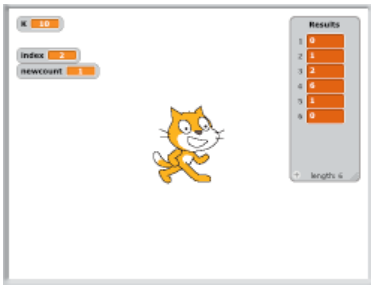
  Remember that the values don’t have to be numbers.

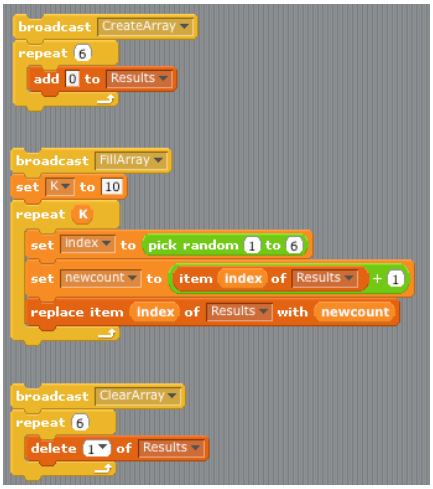
- **Sorting** data:
  - Alphabetical/numerical order, increasing/decreasing, etc.

- **Searching** for data:
  - Looking for a word in a dictionary, looking for a number in a phone book.

Keeping track of data in an array

Example 4: Suppose you are rolling a standard six-sided dice, and want to keep track of how many times each number has appeared.





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 solving problems.

Algorithm

- An algorithm is a definite procedure for solving a given problem or performing a given task.
- Origins of the word:
  - 9th century 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?

Example

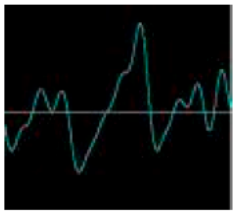
Task:

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.

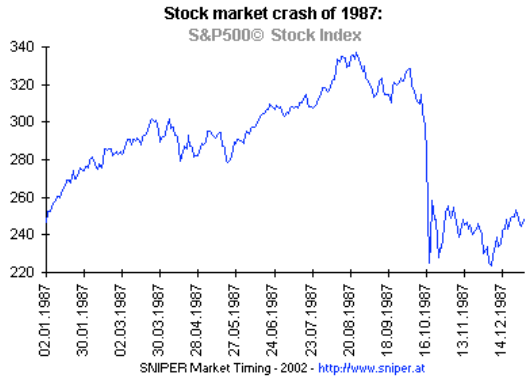
Why would I want to do this?

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



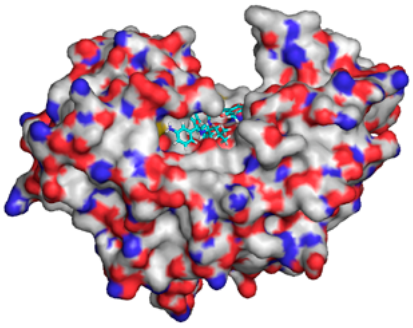
Why would I want to do this?

- 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

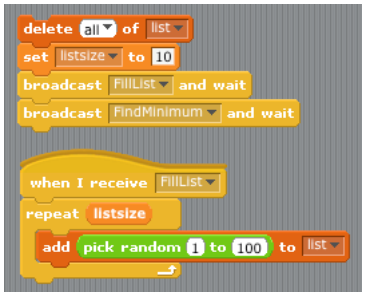
- Given $x_1, x_2, \ldots, x_K$, find i such that $x_i \le x_j$, $1 \le j \le K$.
- Input: $x_1, x_2, \ldots, x_K$
- Compute:

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

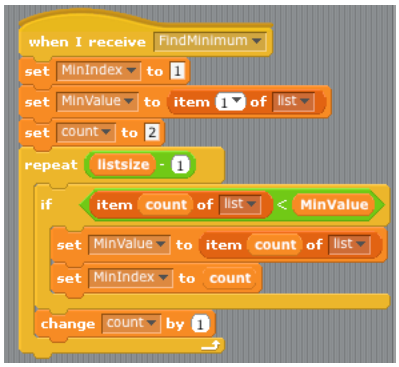
- Output: *MinIndex*, *MinValue*

Finding a Minimum - in Scratch

First fill the list:



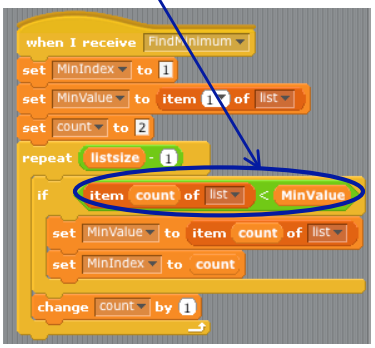
Then go through it to find the minimum:



Counting operations

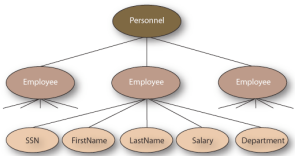
- Want to find the smallest number. How many comparisons?
- Want to find two smallest numbers.

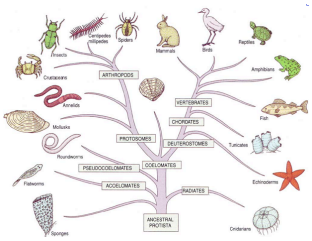
  How many comparisons?



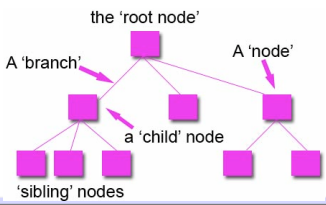
Other ways to organize data: Trees

- Good way to organize hierarchical data:

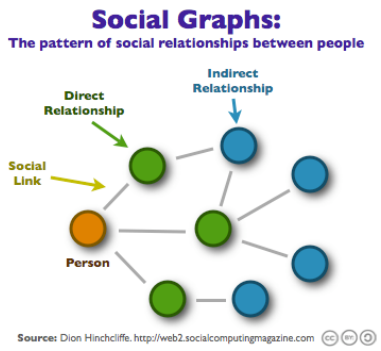




- Generic form:



Other ways to organize data: Graphs



Take-home message

- Understand the concept of array, 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 algorithms for calculating the sum of integers, and finding the minimum in a list.

Coming weeks:

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