What is a program?

- A set of instructions to a computer

Algorithms

Problem1: How to look up Rei Diaz's phone number in a White Pages book?

- Algorithm 1:
    - Turn pages one by one until you hit "D"
    - Turn pages one by one until you hit "i" after the "D"
    - *etc.*
    - max search time for this algorithm in a book with 1,000,000 entries is 1,000,000 (if you're looking for the last entry)
    - This is a *linear search* algorithm
- Algorithm 2 (more efficient):
    - Open the book in the middle, compare the current page with "D",
        - if current page is further down, then split the earlier half in half next time (and "throw away" the latter half)
        - if current page is before the "D", then split the latter half in half next time (and "throw away" the earlier half)
        - if current page starts with a "D", take the 2nd letter and the search key and proceed similarly
    - repeat the process until you end up on the same page (e.g. starts with "Dia") – and finish the process on a single page in a similar manner
    - max search time for this algorithm in a book with 1,000,000 entries is just 20 steps! (You're throwing away half of the problem each time, so this algorithm is *logarithmic*)

Problem 2: How to count students in a room?

- Algorithm 1:
    - Count one by one
    - The so-called "running time" is, again, on the order of *n* if there are *n* students (*i.e.* linear)
- Algorithm 2:
    - Consider the algorithm in the lecture slides
    - Assume (in the world of computing, this is a reasonable) that steps 3-6 are, fundamentally, just one step (they form a *loop*). Then the performance of this algorithm is, again, *log(n)*.

- the branch of computer science that deals with algorithm efficiency studies the performance of different algorithms that achieve the same result in different amounts of time

- we measure "time", as related to algorithms, in the number of different instructions (or *steps*) that a computer needs to take to complete the program

- given the same computing resources, a faster algorithm may drastically improve the performance of a program

- a *bug* is a logical mistake in a computer program. An infinite loop, *i.e.* a loop that never reaches an exit condition, is a common example of a bug that will cause the program to "hang"

Programming Languages

- The algorithms we have seen thus far are written in what's called *pseudocode*, *i.e.* instructions resembling those of a computer program, but written in plain English
- Computers, however, cannot understand English (or any other *natural language*) due to its inherent imprecision and ambiguity, *i.e.* computers cannot make inferences or judgments
- Computers use specialized *programming languages* that have a strict logical grammar to which the programmer must conform

- Most modern programming languages use the same constructs:
  - *variable assignment*: assigning a value to a *variable* (or a place to store a value which can be changed at runtime), *e.g.*
    - a = 4 [assign the value 4 to variable a]
    - aVariable = anotherVariable [assign the value of one variable to another variable]
    - socks_on_feet++ [*increment* the value, *i.e.* add 1 to current value]
  - *looping*: repeating a set of instructions until a condition is met, *e.g.*
    - for (i = 0; i < 5; i++) (which is equivalent to "for all i such that i is more than or equal to 0 but less than 5)

      ```
      { do something }
      ```

    -
      ```
      while socks_on_feet != 2
      { do something }
      ```
  - *branching*: checking whether a condition is true or false, and following alternative paths in a program's execution depending on the outcome, *e.g.*

    -
      ```
      if you find a matching sock then
              { … }
      else
              { … }
      ```

    -
      ```
      if (i equals 5)
      then
              {do something}
      else if (i equals 10)
              {do something else}
      else
              {do something completely different}
      ```

- *indentation* may matter greatly in a programming language if it denotes the boundaries of constructs (many languages, however, do not distinguish between whitespace, in which case indentation is helpful for program readability)

- XHTML is not a *programming language*; it is a *markup language*, *i.e.* you don't direct the computer to execute specific instructions, but rather tell it how to present information

- *source code* is the original program written in a particular programming language's syntax. Computers do not understand source code, but they do understand binary, hence you need another standard program to translate your program from source code to *object code* (*i.e.* binary instructions)
- a *compiler* is a program performing such a translation
- every programming language has its own syntax and, hence, its own compiler

Schematically: [source code] → compiler → [object code]

- XHTML is not compiled, but rather *interpreted*, which means that a browser interprets it on the fly, without compilation
- Some "real" (*i.e.* non-markup, fully-fledged) programming languages are also interpreted, which in this case means that a special program called the *interpreter* executes the program instructions on the fly

- Compiled languages are very efficient (because the end result is already in binary), but they are not easily transferable between different platforms (operating systems)
- Interpreted languages, on the other hand, are more flexible in terms of platforms, but are generally slower because they incur the additional overhead of on-the-fly interpretation

- Some programming languages:
    - Compiled: C, C++, C#, Java, Visual Basic
    - Interpreted: Perl, PHP, Python, JavaScript

JavaScript

- This is the programming language of choice for client-side programs in web pages
    - Contrast this client-side paradigm with SSI (Server Side Includes), which are programs executed on the server side, and only transmitting results to the client
    - The flying bats on E-1's website around Halloween were written in JavaScript
- Despite its name, JavaScript has nothing to do with Java
- Dynamic HTML:
    - DHTML = HTML + CSS + JavaScript

Consider the following excerpt from name.html (available at http://www.fas.harvard.edu/~cscie1/distribution/lectures/11/interpreted/name.html):

```
<script language="JavaScript" type="text/javascript">
<!--
var name = prompt("Please enter your name.","");
document.write("<title>Welcome, " + name + "!</title>");
//-->
</script>
```

- the script is enclosed in XHTML comments (`<!--` and `-->`) to be considered a comment from the XHTML interpreter's (*i.e.* a browser's) point of view for compatibility with older browsers
- "prompt" is a *function* that takes 2 *arguments* (in parentheses, comma-separated, each flanked by quotes), the first of which denotes the wording of the prompt, and the 2nd the default initial value for the white box (here it's left empty). The *result* of "prompt" is what the user types in
- "var name" is a variable called name which (by way of the "=" operator) gets *assigned* the value returned from the prompt function

- Most of your problems with programming will probably arise from "silly" syntactical errors like typos
- When you are writing JavaScript at home or in section, check for such errors first