How does computer programming work?

Magic.
Computer Components

- Hardware
- User Programs
- Operating System

Software

Hardware
What are Programs?

- Programs provide instructions for computers.
- Similar to giving directions to a person who is trying to get from point B from point A. A program may say...
  
  if starting at location B
  go north 3 blocks
  then go East 2 blocks
Programming Language Popularity

StackOverflow Questions Tagged vs. Projects on Github

Corr = 0.78
Programming History

• First programmer:
  • Ada Lovelace

• Different paradigms
  • Functional (lisp, scheme, etc.)
  • Procedural (C, PHP, etc.)
  • Object Oriented (Java, C++, etc.)

  - Note: Most object oriented languages can also be procedural
What are Programs?

- Input
- Processing
- Output
High vs. Low Level Languages

- Low level programming languages -

- High level programming languages -
High vs. Low Level Languages

- Low level programming languages – tied to hardware

- High level programming languages - provides a layer of abstraction that allows for the programmer to only have to learn a hardware independent language
Pros/Cons of High Level Languages

- **Pros**
  - Faster development (libraries/etc.)
  - Easier to understand (abstraction)
  - Can be platform independent

- **Cons**
  - Usually larger executables
  - Abstraction layer usually adds overhead for processing resulting in slower executables
Assembly Example (Low Level)

mov ax, cs
mov ds, ax
mov ah, 9
mov dx, offset Hello
int 21h
xor ax, ax
int 21h
Hello:
  db "Hello World!", 13, 10, "$"
Python Example

(High Level – Scripting Language)

print("hello World!")
Java Example
(High Level – Strongly Typed Language)

class HelloWorld {
    static public void main( String args[] ) { 
        System.out.println( "Hello World!" ); 
    }
}

Anatomy of a Program

Traditional Language

Source Code

Compiler

Compiled Code

Linker

Executable

Virtual Machine Language

Source Code

Interpreter and/or Bytecode Compiler

Bytecode

Virtual Machine

Execution
Bytecode vs. Executable

- **Bytecode**
  - Cross platform
  - Allows for replacement of small components without recompiling entire programs
  - Generally slower performance

- **Executable**
  - Runs on one platform
  - Programs generally compile down to larger executables
  - Generally faster performance
Algorithms
Algorithms

- A series of instructions that solve a problem
Algorithms

• A series of instructions that solve a problem. Example, sorting a list of numbers:

```plaintext
repeat:
    swapped = false
    for i = 1 to length(numbers) – 1:
        /* if this pair is out of order */
        if numbers[i-1] > numbers[i] then
            /* swap and remember change */
            swap( numbers[i-1], numbers[i] )
            swapped = true
        end if
    end for
until no swaps occur
```
Natural Languages

- Syntax: punctuation and spacing
- Semantics: The meaning of the words

- Example: The quick brown fox jumped over the lazy dog.
Programming Languages

- Syntax: punctuation (e.g. parentheses, colons, spacing)

- Semantics: The meaning of the words
  - What does “System.out.println()” mean?
Coding Style

• Just as in natural languages the style of coding matters.

• Style determines the level of readability and maintainability

• Several things make up a coding style:
  • Formatting, naming schemes, comments, design
Pseudo-Code

- The expression of programming logic in a language independent nature.
- Good for design phase of coding
- Example: the last slide!
Bugs
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in different varieties:
  • Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in different varieties:
  • Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
  • Runtime errors (crashes): Errors during execution Example: input file may not be readable
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in different varieties:
  • Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
  • Runtime errors (crashes): Errors during execution Example: input file may not be readable
    – Logic errors: A problem that is caused by a flawed algorithm or set of instructions used to solve a problem
Basic Programming Concepts

- Keywords
- Statements
- Operators
- Data Types
- Variables
Keywords

• Keywords - reserved words that have special meaning in a particular programming language. These words can not be used for any other purpose (e.g. variable names). Java keywords are:

  abstract, continue, for, new, switch, assert, default, goto, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp**, volatile, const*, float, native, super, while
Statements

• Instructions telling the computer what to do
Basic Data Type (primitive data types)

- Integers (Real numbers): short (16 bits), int (32 bits), long (64 bits)
- Single text character: char
- 32bit decimal numbers: float
- 64bit (more precise) decimal numbers: double
- True/False (similar to a bit): boolean
- 8 bits: byte
- Strings (are actually objects): store a series of characters
Operators

- Different data types have different operators
- Operators are part of a statement
- Operators act on the operands around them
- Unary operators – take 1 argument
  - e.g. setting a number to a negative value: -3
- Binary operators – take 2 arguments
  - e.g. arithmetic operators: 1 - 3
- Ternary operators – takes 3 arguments
  - // if x < y set largest to y, else set it to x
  - largest = x < y ? y : x;
Operators on Numbers

In order of increasing precedence, with respect to dividers (i.e. + and – have the same precedence, but execute after *, /, %.)

+ Additive operator (also used by strings)
- Subtraction operator
* Multiplication operator
/ Division operator
% Remainder operator
() Dictate order of operations
Comments

• Provides:
  • Documentation
  • Clarifying what specific code is doing
  • Make code easy for the author or other programmers to understand

• Line comments start with the // characters
  // this is a comment

• Block comments start with /* and end with */
  /* this is a block comment
    comment, it spans multiple lines */
Variables

Variables are labels refer to the storage of some data.

Variables give us access to data that is stored in memory.

Two types of variables:

- Primitives (char/int/float/boolean/etc.)
- Objects/Arrays (Strings/Integer/Float/etc.)
Variables

• Variable name in java:
  • Can be long
  • Are case sensitive (Alpha != alpha)
  • Can contain letters, numbers, and underscores ( _ )
  • Must not start with a number
  • Can not be a keyword
Variables

• The variable's data type (boolean, int, float, double, char, etc.) must be declared when created!
• The variable's data type will never change (i.e. it will always store that type of data)
Variables

- Primitives refer to the actual value and are stored in the “stack”
- Objects/Arrays store a reference to the location in memory where the data is stored in the “heap”
Variables Names (Examples)

thisVariableNameIsValid = "valid naming conventions"
this_variable_name_is_valid = "also valid"
names = "names are valid"
Names = "not the same as names"
numbers_between_0_and_9 = "another valid"
0_9_can_not_start_a_name = "invalid"
if = "keywords are not allowed"
special_chars(!_).allowed = "invalid"
Variable Assignment Statements

// constant PI
static final float PI = 3.14159265;

// variables radius/area_circle
float radius = 5;
float area_circle = PI * radius * radius;
Constants

- A constant is a stored value that doesn't change
- Used for things that will remain constant throughout the program. (e.g. pi, conversions between metric and standard units, etc)
- When representing data that doesn't change it's often a good idea to use constants
- Generally constants are in all CAPITAL letters