Introduction to Languages

Core Instructions

- What can my computer do for me?
  - Basic math (+, -, *, /)
  - Store and move data in memory (registers)
  - Bitwise operations
  - Comparisons (<, >, ==, !=, <=, >=)
  - Control flow (IF THEN, JUMP, LOOP)

Types of Languages

- Low Level
  - Languages that have little to no abstraction from the machine architecture
    - Language Generations 1 & 2
- High Level
  - Languages that have greater abstraction, are human readable, and use more advance data structures
    - Language Generations 3 +

Language Generations

- 1st Generation (Machine Level) 1930's – 1950's
  - Cables, switches, vacuum tubes, machine code
    - Binary, Assembly
- 2nd Generation (Machine Abstraction) 1950's – 1960's
  - First group of languages that brought logical operations to software development
    - FORTRAN, LISP, COBAL

Language Generations

- 3rd Generation (Human Friendly) 1960's – 1980's
  - Further abstraction from structured programming. Geared towards human readability. Introduced ADT (Abstract Data Types), and OOP (Object Oriented Programming)
    - SmallTalk, C, C++, BASIC, SQL
- 4th Generation (Frameworks) 1980's – Today
  - Languages aimed a cross platform compatibility. Focuses on process and function, not THE process and function. Many run on virtual machines or frameworks such as .NET, Java Runtime, etc.
    - Visual Basic, C#, Java, Ruby

Types of Languages

- Procedural Language
  - AKA Top down language
  - Languages that contain variables, functions, and primitive data types
    - FORTRAN, COBAL, C, BASIC
- Object Oriented Language
  - Contain the same structures as Procedural Languages, however these variables, functions and primitive data types can be encapsulated into objects
    - C++, Visual Basic, Perl, C#, Java

Types of Execution

- Compiled
  - Source code is converted to machine language by a compiler
- Interpreted
  - Source code is converted to machine language during runtime by an external program called an Interpreter
- Framework / Virtual Machine
  - Source code is converted to a CRL (Common Runtime Language), which contains standardized instructions to be processed by the VM or framework.
- Emulated
  - Usually machine code that is executed on a platform that it was not natively compiled on (simulates hardware)
  - Major types: Platform, Console, Terminal

The End?