High Level Language Features:

* Names for almost everything
* Composite types (arrays, structs)
* Type declarations
* Type checking
* Easy ways to manage global, local, and heap storage
* Complex expressions are handled well
* Control structures (conditionals, switches, loops)
* Subroutines with their own private scope
* Abstract data types, modules, packages, classes
* Exceptions

How does the computer understand your instructions?

* Magic? NO!
* Programs must be translated into the machine language that the CPU understands
* Compilers!
  + Translate programming language to executable binary
  + Whole program is compiled before it can be run
* Interpreters
  + Understand a programming language and perform the actual computation
  + Line by line
* Hybrid
  + Java!
  + Combine compiler and interpreter
* Transpiler
  + Translate a programming language to another programming language

Overview of Program Interpretation

Source bytes go to Lexer, which generates Tokens sent to the Parser, which generates a Parse Tree and is verified by a Semantic Analysis, then marks tree as valid, sent to the Backend, which finally converts it to x86 Binary to be read by the CPU

What makes a program valid?

* Syntax
  + What does it mean to look like a valid program?
* Semantics
  + What does it mean for a program to be valid?
* Correctness?
  + Is the program the correct one for the job?

What is this course?

* Concepts common to a variety of programming languages
* Distinguish between “essence” and “arbitrary” of programming languages
* A firm basis for comprehensive new language

What will this course cover?

* How a programming language is described informally and formally
* How a programming language is implemented
* Be exposed to diverse programming languages