Abstract Syntax Trees
what is an abstract syntax tree

• Tree representing the structure of the program
• Leaf nodes represent elements like variables and literals
• Interior nodes represent higher-level constructs:
  • Arithmetic operations and other complex expressions (e.g., function calls)
  • Assignment statements
  • Control statements
  • Lists of statements
why abstract syntax trees?

• Parse trees are tied to specific grammar used to recognize a language
  • Lots of extraneous information we don’t need (semicolons, parentheses, intermediate constructs)
  • May be structured oddly to deal with parser constraints (capturing order of operations, dealing with left-recursion)
• Abstract syntax trees are, well, **abstract**
  • Tree nodes represent operations without necessarily being tied to specific concrete syntax (can change keywords without changing AST structure)
  • Only preserve information needed for correct analysis and code generation (tree structure captures order of operations, rather than grammar structure)
ast vs parse tree

\[ a + b \times c \]
next: ASTs for expressions