Arithmetic Expressions

* Operand and Operator
  + Operator
    - Precedence Rule
      * In what order are operators run
    - Associativity Rule
      * In what order are operators of the same precedence run
  + Operand (Constants, variables, functions)
    - a = a + func(a)

Operator Overloading

* When one operator can have multiple functions depending on context and data types involved
  + +, -, &, \*

Type Conversions

* Type Widening
  + Usually safe from data loss
* Type narrowing
  + Usually more complex, not safe from data loss
* Type Casting
  + Implicit
    - int a; then float b = a + c;
      * a is implicitly converted to a float for the calculation
  + Explicit
    - float a; then int a;
      * a is explicitly converted to an int from a float

Relational and Boolean Expressions

* Relational
  + >=, >, <=, <, !=, ==
* Boolean
  + NOT, AND, OR, XOR
    - XOR is true if they are not both true or false

Short-Circuit Evaluation

* Not needing to evaluate the whole expression to know the outcome
* Example
  + (13 \* a) \* (b / 13 – 1)
    - If a is zero, there is no need to evaluate (b / 13 – 1) because outcome is zero
* Problem with non-short-circuit evaluation
  + index = 0
  + While (index < length) && (LIST[index] != value)
    - index++;
  + When index = length, LIST[index] will cause an indexing problem (assuming LIST is length – 1 long)
* C, C++, and Java use short-circuit evaluation for the usual Boolean operators (&& and ||), but provide bitwise Boolean operators that do not use CSEval (& and |)
* All logic operators in Ruby, Perl, ML, F#, and Python are short-circuit evaluated
* Short-circuit evaluation exposes the potential problem of side effects in expressions
  + Ex. (a > b) || (b++ / 3)

Assignment Statements

* The general syntax
  + <target\_var> <assign\_operator> <expression>
* a = a + b can be written as a += b in C-based languages
* Unary assignment operators in c-based languages allow for incrementing
  + sum = ++count (count incremented, then added to sum
  + sum = count++ (count added to sum, then incremented)
  + count++ (count incremented)
  + -count++ (count incremented then negated)
* Assignment operator is lower than relational operator (use parenthesis!!)