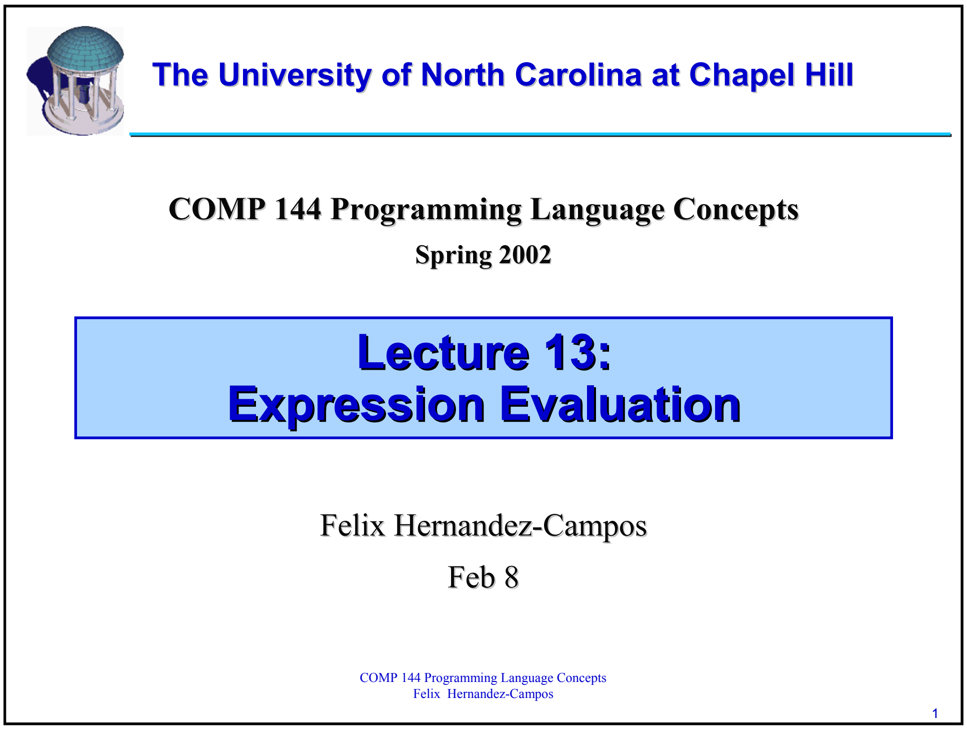


The University of North Carolina at Chapel Hill

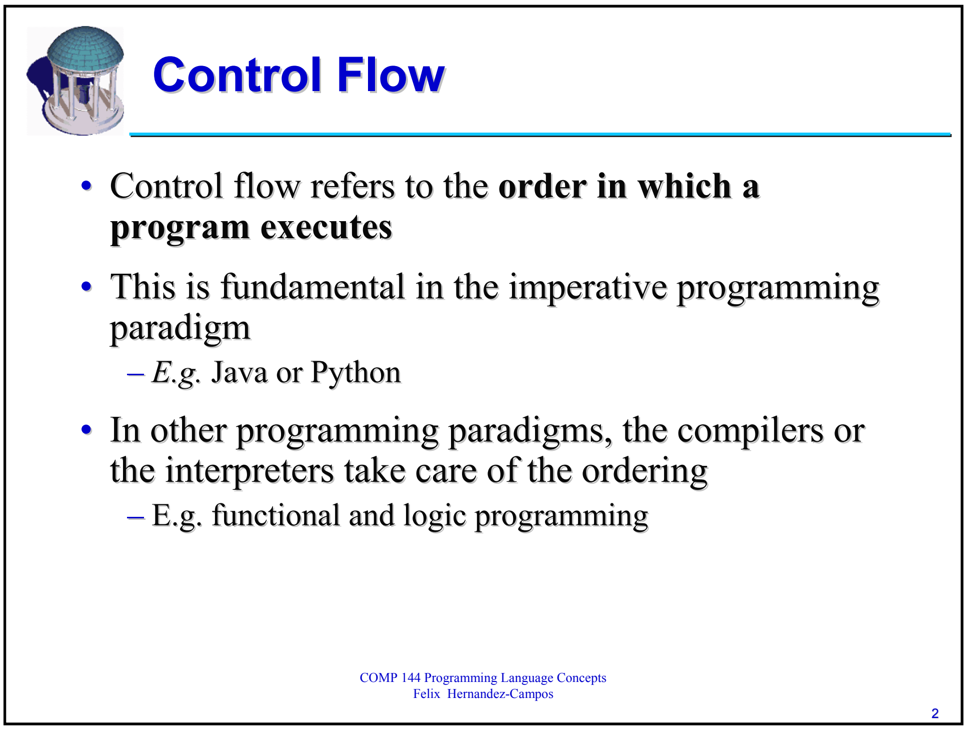
COMP 144 Programming Language Concepts

Spring 2002

Lecture 13: Expression Evaluation

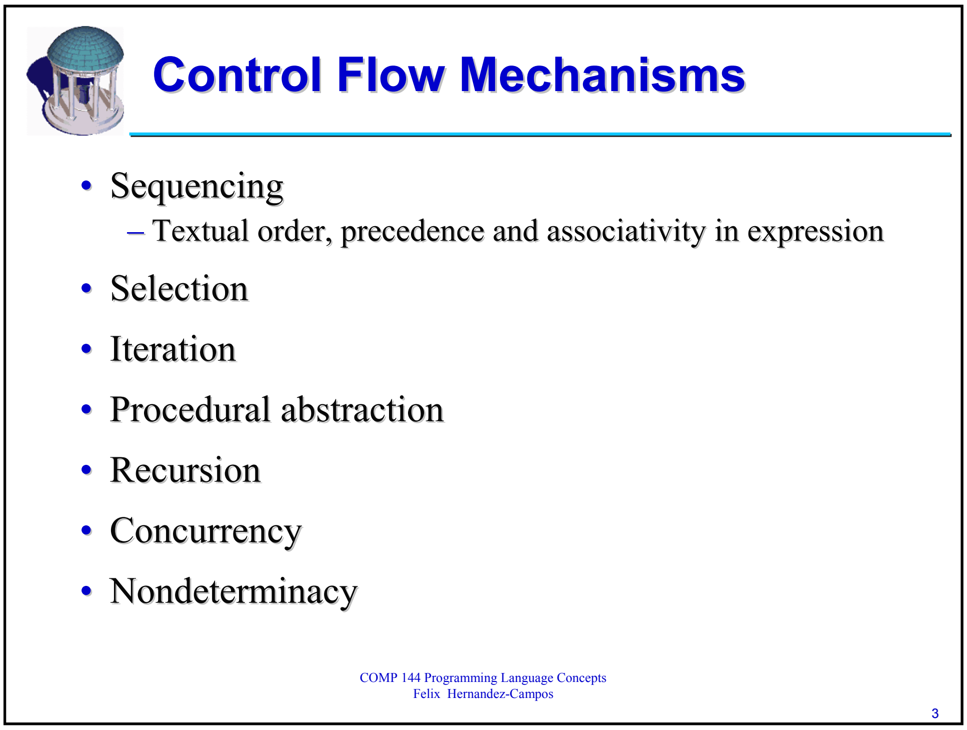
Felix Hernandez-Campos

Feb 8



Control Flow

- Control flow refers to the **order in which a program executes**
- This is fundamental in the imperative programming paradigm
  - *E.g.* Java or Python
- In other programming paradigms, the compilers or the interpreters take care of the ordering
  - E.g. functional and logic programming

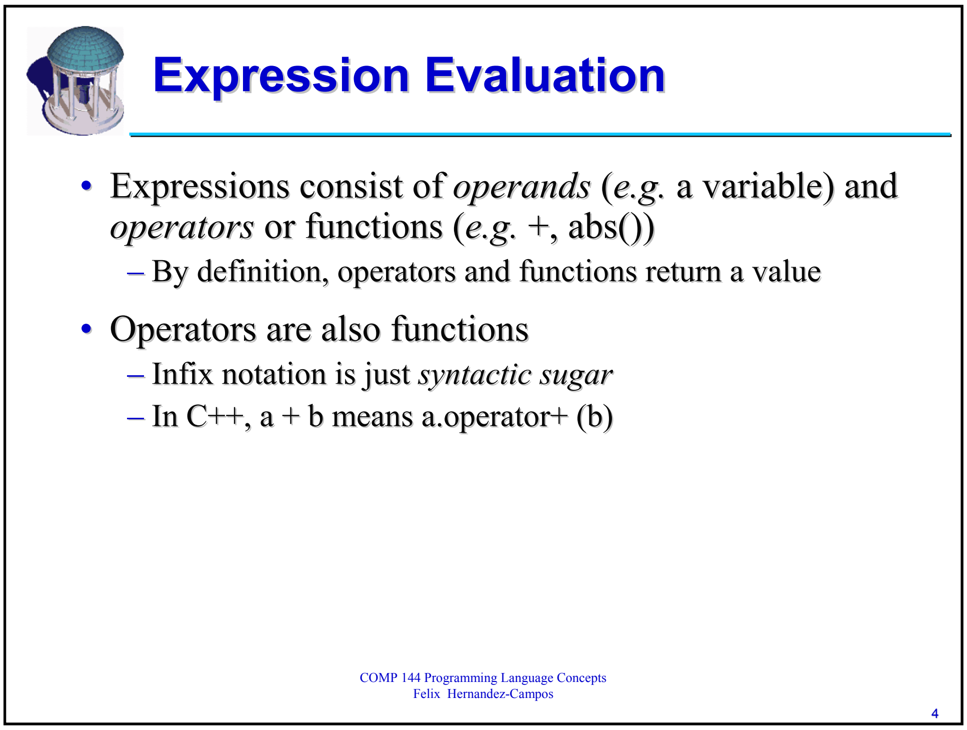


Control Flow Mechanisms

- Sequencing
  - Textual order, precedence and associativity in expression
- Selection
- Iteration
- Procedural abstraction
- Recursion
- Concurrency
- Nondeterminacy

COMP 144 Programming Language Concepts
Felix Hernandez-Campos

3



Expression Evaluation

- Expressions consist of *operands* (*e.g.* a variable) and *operators* or functions (*e.g.* +, abs())
  - By definition, operators and functions return a value
- Operators are also functions
  - Infix notation is just *syntactic sugar*
  - In C++, a + b means a.operator+ (b)

COMP 144 Programming Language Concepts
Felix Hernandez-Campos

4

Overloading Operators

- Python example

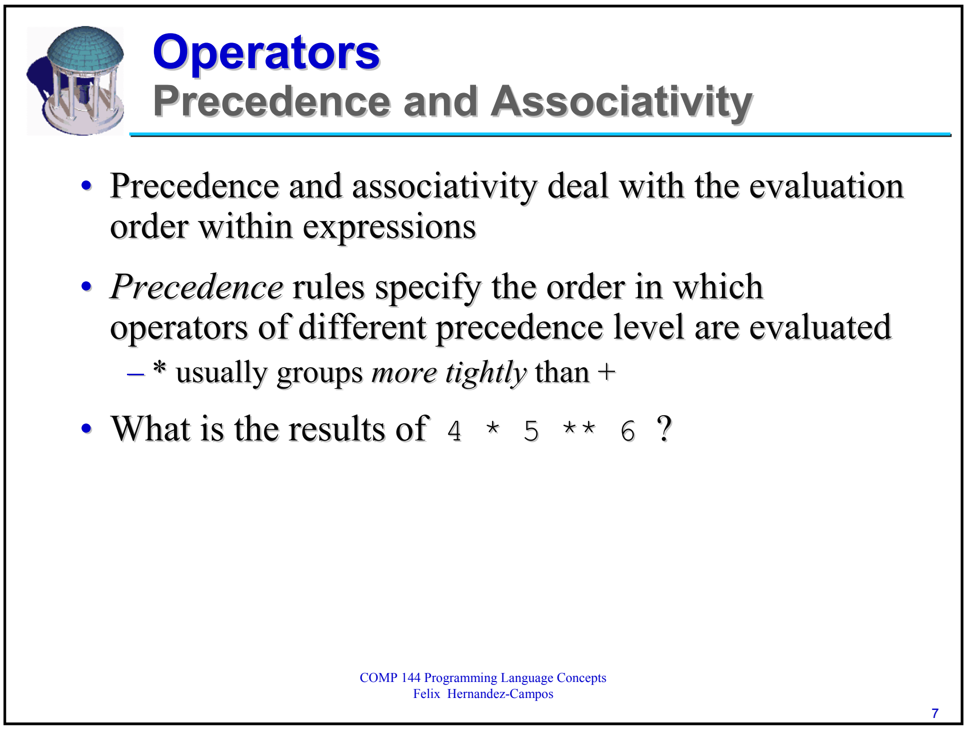
```
import time
class Time:
      def __init__(self, seconds):
            self.seconds = seconds
      def __repr__(self):
            return time.ctime(self.seconds)
      def __add__(self, x):
            return Time(self.seconds + x)
      __radd__ = __add__  # support for x+t
      def __sub__(self, x):
            if hasattr(x, 'seconds'):
                  # test if x could be a Time
                  return self.seconds - x.seconds
            else: return self.seconds - x
```

Overloaded Operator +

Overloaded Operator –

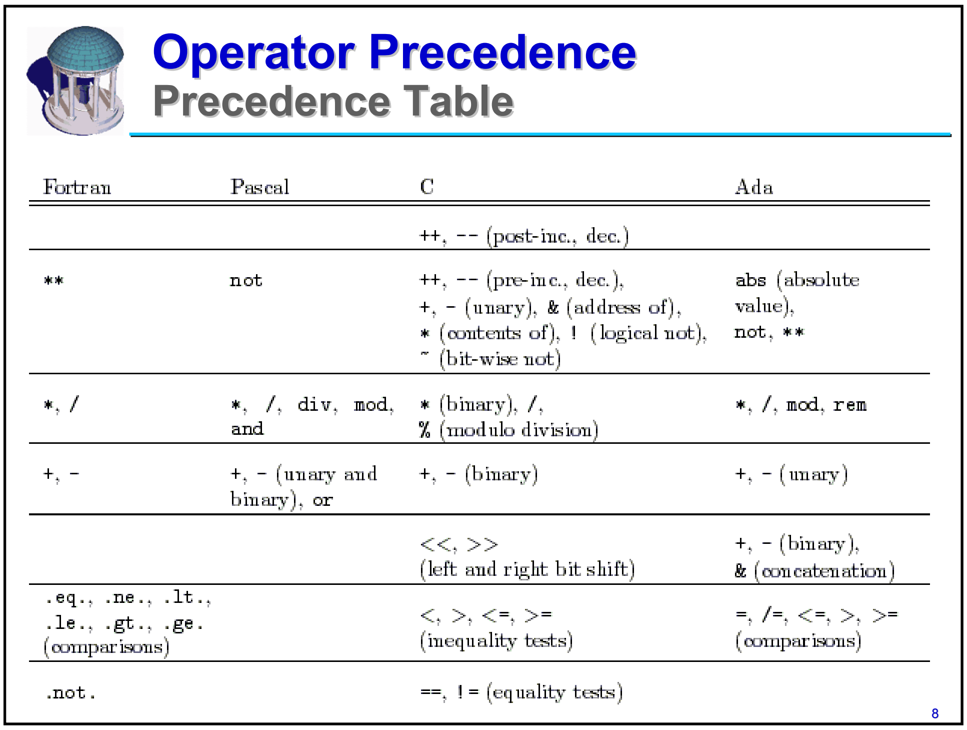
Operators

- Operators are used in
  - Prefix notation
    - » *E.g.* Expression `(* (+ 1 3) 2)` in Lisp
  - Infix notation
    - » *E.g.* Expression `(1 + 3) * 2` in Java
  - Postfix notation
    - » *E.g.* Increment `a++` in C
- Operators can have 1 or more operands
  - Increment in C is a one-operand operator: `a++`
  - Subtraction in C is a two-operand operator: `a-b`
  - Conditional expression in C is a three-operand operators: `(a == 3 ? 0 : 1)`



Operators
Precedence and Associativity

- Precedence and associativity deal with the evaluation order within expressions
- *Precedence* rules specify the order in which operators of different precedence level are evaluated
  - * usually groups *more tightly* than +
- What is the results of `4 * 5 ** 6` ?



Operator Precedence
Precedence Table

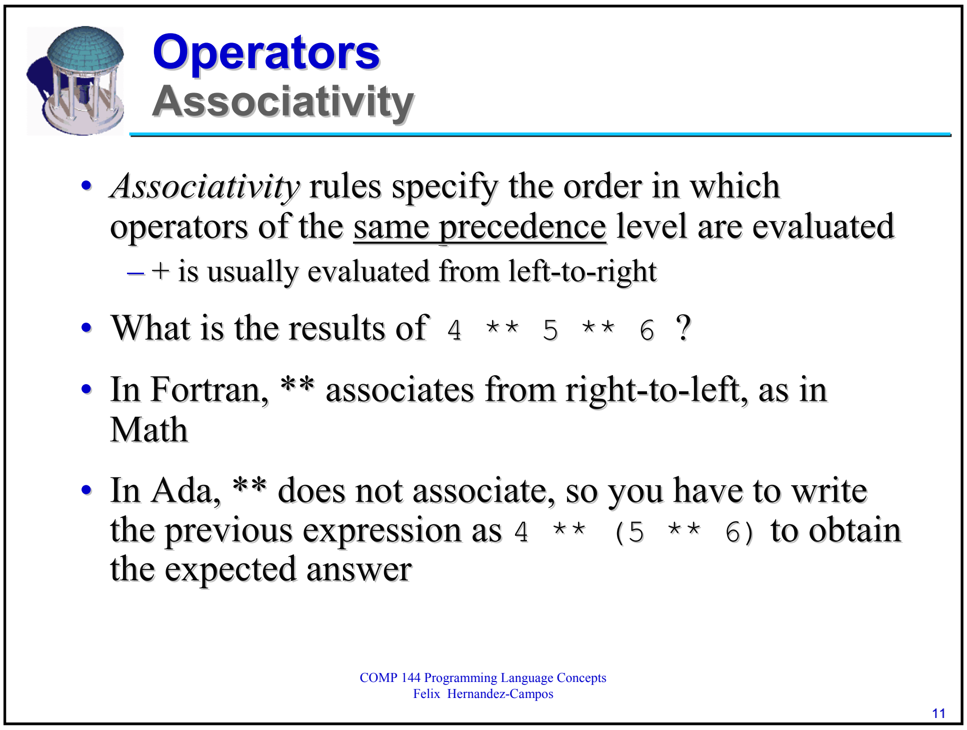
Fortran	Pascal	C	Ada
		`++`, `--` (post-inc., dec.)	
`**`	`not`	`++`, `--` (pre-inc., dec.), `+`, `-` (unary), `&` (address of), `*` (contents of), `!` (logical not), `~` (bit-wise not)	`abs` (absolute value), `not`, `**`
`*`, `/`	`*`, `/`, `div`, `mod`, `and`	`*` (binary), `/`, `%` (modulo division)	`*`, `/`, `mod`, `rem`
`+`, `-`	`+`, `-` (unary and binary), `or`	`+`, `-` (binary)	`+`, `-` (unary)
		`<<`, `>>` (left and right bit shift)	`+`, `-` (binary), `&` (concatenation)
`.eq.`, `.ne.`, `.lt.`, `.le.`, `.gt.`, `.ge.` (comparisons)		`<`, `>`, `<=`, `>=` (inequality tests)	`=`, `/=`, `<=`, `>`, `>=` (comparisons)
`.not.`		`==`, `!=` (equality tests)	

Operator Precedence
Precedence Table

	& (bit-wise and)	
	^ (bit-wise exclusive or)	
	| (bit-wise inclusive or)	
.and.	&& (logical and)	and, or, xor (logical operators)
.or.	|| (logical or)	
.eqv., .neqv. (logical comparisons)	?: (if...then...else)	
	=, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |= (assignment)	
	, (sequencing)	

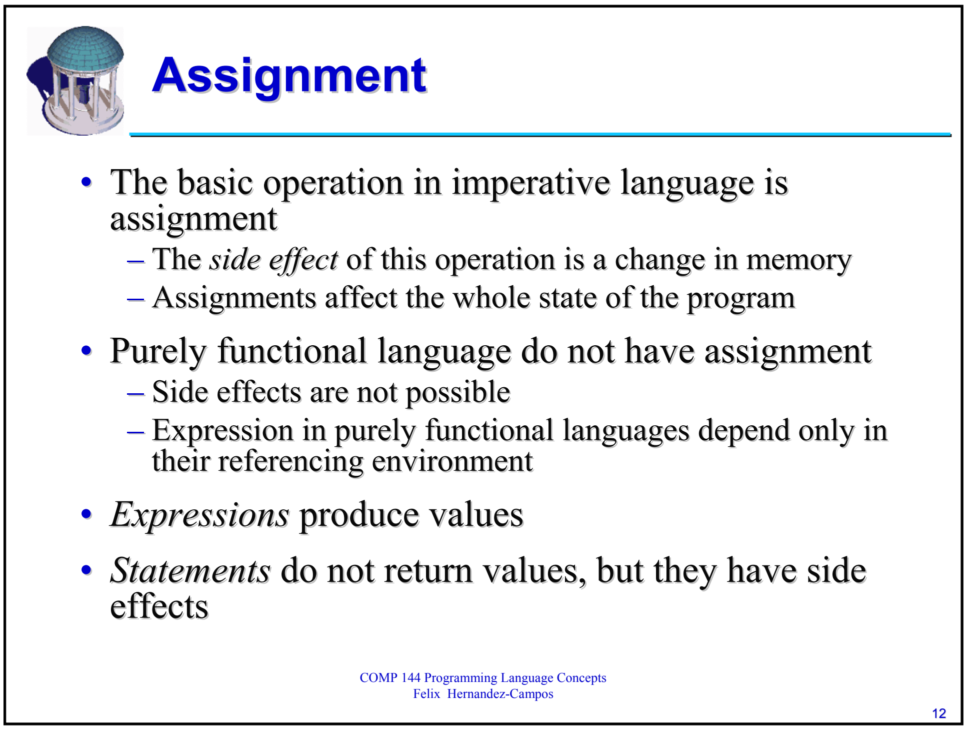
Operators
Precedence

- *Precedence* rules specify the order in which operators of different precedence level are evaluated
  - * usually groups *more tightly* than +
- What is the results of `4 * 5 ** 6` ?
- Precedence in Python
  - http://www.python.org/doc/current/ref/power.html
- Precedence in boolean expression is also very important
  - Pascal's `if A < B and C < D then (*ouch*)`



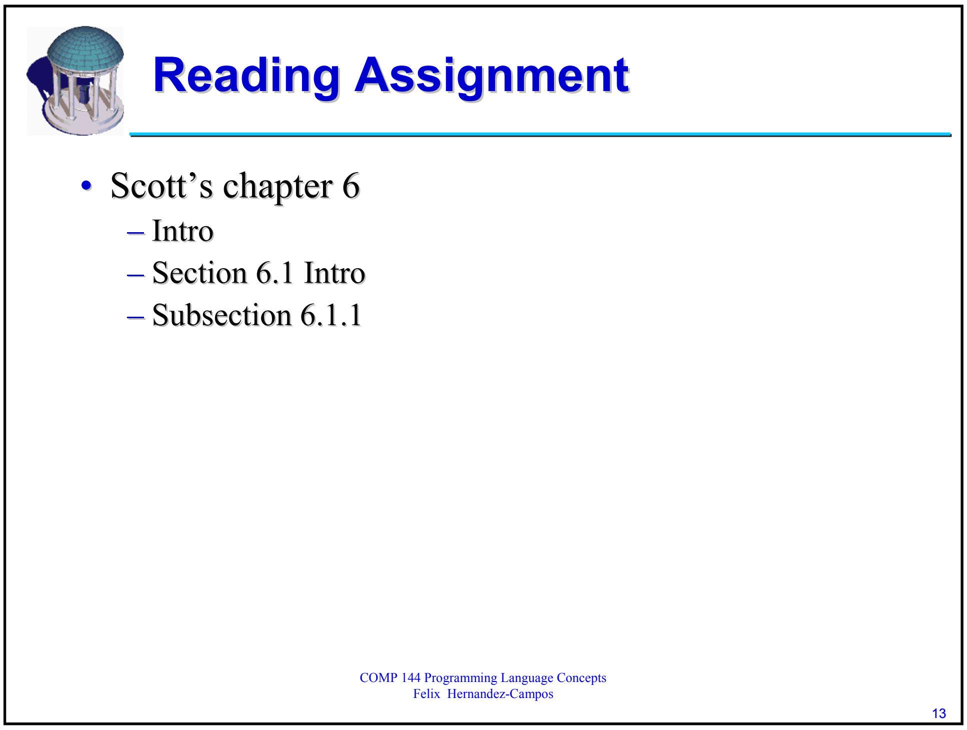
Operators
Associativity

- *Associativity* rules specify the order in which operators of the <u>same precedence</u> level are evaluated
  – + is usually evaluated from left-to-right
- What is the results of `4 ** 5 ** 6` ?
- In Fortran, ** associates from right-to-left, as in Math
- In Ada, ** does not associate, so you have to write the previous expression as `4 ** (5 ** 6)` to obtain the expected answer



Assignment

- The basic operation in imperative language is assignment
  – The *side effect* of this operation is a change in memory
  – Assignments affect the whole state of the program
- Purely functional language do not have assignment
  – Side effects are not possible
  – Expression in purely functional languages depend only in their referencing environment
- *Expressions* produce values
- *Statements* do not return values, but they have side effects



Reading Assignment

- Scott's chapter 6
  - Intro
  - Section 6.1 Intro
  - Subsection 6.1.1