CS 3723, Spring 2010  
Lecture Notes #6: Language Paradigms

Reading:
- Mitchell 4.4, 15 (also relevant: Ch. 5, Ch. 10)

Objectives
1. be able to compare and contrast the declarative (functional/logic) and imperative (procedural/OO) approaches to programming drawing concrete examples from known functional, procedural, OO, and multi-paradigm languages
2. be able to compare and contrast the procedural and OO paradigms, drawing concrete examples from known languages
3. be able to compare and contrast the functional and logic paradigms, drawing concrete examples from known languages
4. be able to determine the value of Scheme expressions (whose evaluation terminates in a small finite number of steps)
5. be able to list some of the features commonly found in procedural languages but are not used in the functional paradigm (assignments or side-effects more generally, and while loops)
6. be able to list several key features of functional programming languages that facilitate programming in a functional style (first-class/higher-order functions, extensive polymorphism list types and operators, structured function returns, and constructors (aggregates) for structured objects)

Outline
1. paradigms
   (a) declarative
      i. functional (FP)
      ii. logic (LP)
   (b) imperative
      i. procedural
      ii. object oriented (OO/OOP)
   (c) concurrent?
      i. shared memory
      ii. message passing
iii. actors

2. what is functional programming
   (a) pure form: no "side effects" (very few practical pure-FP languages: Haskell)
   (b) “applied λ-calculus”
   (c) recursion, higher-order functions
   (d) expression rather than statement oriented
   (e) dynamically typed, or polymorphic static types

3. Benefits
   (a) correctness, readability, and reliability
   (b) local reasoning
   (c) referential transparency (can replace any variable/expression with its value in the
text of the program without affecting the meaning of the program)

4. Historical Developments
   (a) Lisp
   (b) Backus 77 Turing Award Lecture
   (c) ML
   (d) Haskell
   (e) modern "OO" languages (inner classes in Java, C# delegates, "lambda" in Python,
Scala), OCaML

Vocabulary
declarative, imperative, procedural, object-oriented (OOP), functional, logic, Lisp, Scheme,
Haskell, ML, OCaML, Prolog, Java, Scala, side-effect, mutation, referential transparency