Catalog Description
Environments, stores, scoping, functional and imperative languages, modules, classes, data encapsulation, types, and polymorphism. Implementation of these constructs in a definitional interpreter. Three hours of lectures. One hour of computer laboratory.

Course Objectives
This course explores concepts underlying the definition, implementation, and use of programming languages. The goal is to provide you with an understanding of (and a vocabulary for) common language features, including how they are implemented, how other language-design choices affect them, and how they can be used effectively in program development.

Prerequisites

CIS 252: Familiarity with major ideas (e.g., recursion, modularity, data abstraction, pattern matching, higher-order functions) of functional programming

CIS 275: Familiarity with basic set theory, functions, predicate logic, and induction.

CIS 351: Familiarity with stacks, lists, and trees

Course Outcomes
After completion of the course, students should be able to:

• When given an informal but fairly precise English description, write a Scheme program that accurately captures the desired behavior. [ABET (a), (b), (c), (i) ]

• Write data-directed Scheme programs over lists, trees, and other inductively defined data structures. [ABET (a), (b), (c), (i), (k) ]

• When given a moderate-sized Scheme program and relevant input, calculate the result of that program. [ABET (a), (c), (i), (k) ]

• Describe, compare, and contrast various language features. [ABET (i), (j) ]

• When given a Scheme or \( \lambda \)-calculus expression, identify the free variables, identify the bound variables, and calculate the lexical addresses of the bound variables.

• When given a \( \lambda \)-calculus expression, identify all \( \beta \)-redexes and \( \beta \)-reduce the expression.

• When given a small piece of code, calculate its value under a variety of execution scenarios (dynamic or lexical scope, different parameter-passing mechanisms, etc). [ABET (c), (j)]

• Implement an interpreter for a simple language incorporating lexical or dynamic scope, side effects and state, environments, closures, and recursion. [ABET (a), (c), (i), (j), (k)]

Outcome Measurement
Outcomes are assessed through a combination of weekly homework assignments, weekly labs, and in-class exams or quizzes.
Course Topics

CAC Category Content
1  Data Structures
0  Algorithms
1  Software Design
0  Computer Organization & Architecture
2  Programming Languages