CSE 341 Programming Languages

Credits
4.0 (3 hrs lecture, 1 hr section)

Lead Instructor
Daniel Grossman

Textbook

- Depends on instructor, often relying on free language user's guides, online tutorials, and/or instructor-written reading notes and/or videos

Course Description
Basic concepts of programming languages, including abstraction mechanisms, types, and scoping. Detailed study of several different programming paradigms, such as functional, object-oriented, and logic programming. No credit if CSE 413 has been taken.

Prerequisites
CSE 143

CE Major Status
Selected Elective

Course Objectives

- To understand fundamental programming-language concepts
- To become fluent in non-imperative programming paradigms
- To become able to learn new programming languages efficiently

ABET Outcomes
(a) an ability to apply knowledge of mathematics, science, and engineering
(c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability
(e) an ability to identify, formulate, and solve engineering problems
(i) a recognition of the need for, and an ability to engage in life-long learning
(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

Course Topics

- The following topics are always covered:
  - functional programming (avoiding mutation; exploiting recursion and higher-order functions; closures; anonymous functions)
  - algebraic datatypes and pattern-matching
  - essential object-oriented programming (late-binding / dynamic dispatch, subtyping vs. subclassing)
  - language support for abstraction, such as modules, abstract types, and dynamic type-creation
  - syntax vs. semantics
  - static vs. dynamic typing
  - parametric polymorphism / generics
  - object-oriented extensibility vs. functional extensibility
- In any particular offering, most of the following are covered:
  - tail recursion
  - currying
  - equality vs. identity
  - macros
  - type inference
  - lazy evaluation and related idioms such as streams and memorization
  - code-as-data concepts, such as reflection and eval/apply
  - lexical vs. dynamic scope
  - subtyping issues such as structural vs. named subtyping and sumpsumption vs. coercion
  - object-oriented concepts such as multiple inheritance, multimethods, and metaclasses
  - bounded parametric polymorphism
  - forms of parameter passing
  - subtype polymorphism and bounded polymorphism
  - logic programming
- A small number of these topics might also be covered:
  - language-design principles
  - history of programming languages
  - programming environments
  - debugging support
  - compilers vs. interpreters
  - continuations
  - continuation-passing style
  - coroutines
  - iterators
  - language support for concurrency