Course Description

Syntax and semantics. Compilers and interpreters. Programs as data. Regular expressions and context free grammars. Programming paradigms, including procedural, functional, and object-oriented programming. Type systems, including dynamic and static typing disciplines. Binding, scope, data abstraction, and modularity. Denotational, operational, and axiomatic semantics. Introduction to program correctness.

Prerequisites

This class requires CS 202, 251, 311; passed with grades of C or better. These courses provide you with the background that you will need to understand the course material and develop solutions to the assignments that you will be given. It is assumed that you have experience programming in a high-level, object-oriented language (specifically, C++ or Java, from CS 202 and earlier classes). If you have not completed the prerequisites, then you may not have the background that you need to pass this class. In this case, by default, you will not be able to continue in the class and you should contact the instructor at the earliest opportunity to avoid an administrative drop.

Course Objectives/Student Learning Outcomes

Upon the successful completion of this course students will be able to:

1. Explain the distinction between language syntax and semantics.
2. Describe the similarities and differences between interpreters and compilers.
3. Explain the phase structure of a typical compiler and the role of each phase.
4. Use regular expressions and context-free grammars to describe the syntax of simple programming languages.
5. Illustrate the features and characteristics of different programming paradigms, including procedural, functional, and object-oriented programming.
6. Explain the concepts of binding, scope, block structure, and lifetime, and apply them to resolving variable uses to their binding sites in a variety of languages.
7. Describe and apply the basic concepts of type systems, including primitive types, compound, and recursive types, abstract data types, and type equivalence models.
8. Describe the strengths and limitations of static and dynamic typing disciplines.
9. Describe and apply the basic concepts of data abstraction, encapsulation, object-oriented classes, and modules.
10. Explain basic approaches and applications for the formalization of programming language semantics.

Outline of Course Content

<table>
<thead>
<tr>
<th>Week</th>
<th>Lecture Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Course overview. Syntax and Semantics. Program equivalence, and the distinction between syntactic</td>
</tr>
</tbody>
</table>
and semantic equivalences. Representation of programs as abstract syntax trees. Distinctions between concrete and abstract syntax and between static and dynamic semantics.

Programs that manipulate programs as data. Basic principles in the function and construction of interpreters, compilers, program generators, and program analysis tools.

Techniques for describing language syntax. Regular expressions, context-free grammars, and common usage patterns.


Midterm exam. Review.


Type checking. Association of types to variables and values. Dynamic and static typing disciplines. Type safety. Decidability issues. Interpreting type error messages.

Techniques for formalization of programming language semantics, including denotational, operational, and axiomatic approaches.

Introduction to program correctness. Program specification using predicate logic. Application of formal semantics, and the role of tools for mechanized reasoning.

Course Requirements and Method of Evaluation

The course will be assessed by a combination of homework and exams.

Students will receive eight homework assignments (Weeks 1-4 and 6-9; no new assignments will be set in Weeks 5 and 10 to allow for review ahead of the midterm and final exams, respectively). The majority of these assignments will involve programming, and will span a range of different languages and paradigms across the term.

There will be one midterm exam (in Week 6) and one final exam (in Finals week), both of which will be closed-book. Exams are scheduled in advance and, unless a prior arrangement is made, a grade of zero will be recorded for missed exams.

There may be some assignments that are not formally assessed. This might include written exercises, additional reading, or watching prepared videos, etc. These will be designed to help you master the subject, to keep pace with the lectures, and to prepare for the exams, so it will be very important for you to keep up to date with those exercises.

The weightings that will determine final grades are as follows: homework assignments: 40%; midterm: 25%; final: 35%.

Required Texts and/or Required Reading List

The slides and other materials that are presented in lectures and on the course web site are required reading.

There is no required textbook for this class, but all students are strongly encouraged to do background reading in parallel with the coverage of topics in class. There are many textbooks on the subjects of this
Of course, there are plenty of good resources on the Internet of direct relevance to this course that can be found on sites like Wikipedia or simply by searching on Google.

Computing Facilities

The assignments for this course will require the use of a variety of programming language implementations. Students may develop their solutions on any machine and operating system on which these implementations are available. At a minimum, all of the systems that we use will be installed on the departmental Linux systems that can be accessed in person in the lab area, or remotely via ssh/putty.

Academic Integrity

We follow the standard guidelines for academic integrity. It is permissible to discuss assignments with other students, but you must develop the solution yourself (although you can consult the tutors for help in debugging). Do not, under any circumstances, copy any part of another person's solution and submit it as your own. Writing code for use by another, or using another's code in any form (even with their permission) will be considered cheating. Cheating on an assignment or exam will result in an automatic zero grade for that piece of work, and the initiation of disciplinary action at the University level. Please refer to http://www.pdx.edu/dos/codeofconduct for details of the general PSU Student Code of Conduct.

Disabilities and Accommodations

If you are a student with a documented disability who is registered with the Disability Resource Center, please contact the instructors immediately to arrange any needed academic accommodations, and let us know as soon as possible if you feel that your needs are not being met. If you have accommodations that include taking tests at the University test center, you should take steps to make the necessary reservations at the earliest possible opportunities; the times and dates for both the midterm and final exams are already fixed for 2/9 and 3/16, respectively, as described above.

Subjective Matters

To ensure consistency, all questions about matters that are potentially subjective—such as possible grading errors or determining whether an exceptional circumstance warrants an extension of a deadline—should be addressed to the primary instructor for this class.