

Wright State University

CORE Scholar

Computer Science & Engineering Syllabi

College of Engineering & Computer Science

Summer 2013

CS 3180/5180: Comparative Languages

Krishnaprasad Thirunarayan

Wright State University - Main Campus, t.k.prasad@wright.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/cecs_syllabi



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Repository Citation

Thirunarayan, K. (2013). CS 3180/5180: Comparative Languages. .

https://corescholar.libraries.wright.edu/cecs_syllabi/608

This Syllabus is brought to you for free and open access by the College of Engineering & Computer Science at CORE Scholar. It has been accepted for inclusion in Computer Science & Engineering Syllabi by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

CS 3180/5180 Comparative Languages

- **Instructor** : T. K. Prasad
 - **Phone No.** : (937)-775-5109
 - **Email** : t.k.prasad@wright.edu
 - **Home page**: <http://knoesis.wright.edu/tkprasad/>
 - **Quarter** : Summer, 2013
 - **Class Hrs** : MW, 1:30 - 3:10pm, 145 Russ Center
 - **Office Hrs** : MW, 3:15 - 4:00pm, 395 JC (or by appt.)
-

Course Description

This course will introduce fundamental concepts and paradigms underlying the design of modern programming languages. For concreteness, we study the details of an object-oriented language (e.g. Java, C#, C++), a functional language (e.g. Scheme), and get introduced to multiparadigm languages (e.g., Python, Scala). The overall goal is to enable comparison and evaluation of existing languages. The programming assignments will largely be coded in Java and in Racket (formerly, Scheme), and optionally in Python or Scala.

Prerequisites

- **Data Structures and Algorithms.** (Equivalently, CS3100/5100.)
 - **Experience with programming in imperative languages** such as C/C++, Pascal, or Ada.
-

Course Material

1. [On-line Lecture Notes.](#)
2. [OOP Basics](#)

References

1. K. Arnold, J. Gosling, and D. Holmes: *The Java Programming Language*. Addison-Wesley Publishing Co., 4th Edition, 2005. ISBN 0-321-34980-6
 2. Michael L. Scott, *Programming Language Pragmatics*. Morgan Kaufmann Publishers, 2nd Edition, 2006. ISBN 0126339511
 3. [The Java Tutorial](#)
 4. Ravi Sethi, *Programming Languages: Concepts and Constructs*. Addison-Wesley Publishing Co., 2nd Edition, 1996. ISBN 0-201-59065-4
 5. R. Kent Dybvig, [The Scheme Programming Language](#), 4th Edition. The MIT Press 2009.
 6. [Scheme : Language Reference Manual](#)
 7. [Racket Download Site \(http://racket-lang.org/\)](http://racket-lang.org/)
 8. [Jython](#)
 9. [Python](#)
 10. [Scala](#)
-

Relevant Websites

- [The Teaching About Programming Languages Project](#)
-

Course Load

The course load includes programming assignments worth 30 points, a midterm worth 30 points and a final worth 40 points. Normally, graduate students are assigned additional homework problems and are expected to solve additional/different problems in the tests.

Grading

The letter grades will be assigned using the following scale: A[90-100], B[80-90], C[70-80], D[60-70], and F[0-60]. However, I reserve the right to adjust the scale somewhat to utilize the gaps in the distribution. Academic dishonesty will be "rewarded" with a grade of "F". "Sharing/reuse" of solutions to assignment problems is strictly prohibited.

Attendance Policy

All registered students are expected to attend all lectures. In case a student is absent from a lecture due to unavoidable circumstances, the student is still responsible for the material covered in the class, as it is typically available from the course web-page well in advance. Furthermore, the student is expected to find out about in-class announcements from their colleagues/instructor.

Class Schedule and Syllabus

(The following class schedule is based on normal semester classes which are 1hr 15min long. The content will be covered distributed in a different way and covered at a different rate in the summer semester when the classes are 1hr 40min long.)

Topic

Class 1	<u>Evolution of Programming Languages</u>
Class 2	<u>Syntax Specification : Grammars</u>
Class 3	<u>Object-Oriented Programming</u>
Class 4	(continue)
Class 5	<u>Symbolic Data; List Processing</u>
Class 6	<u>Styles : Functional vs Procedural</u>
Class 7	<u>Recursive Definitions (Examples)</u>
Class 8	<u>Abstraction : Higher Order Functions</u>
Class 9	<u>Scoping; Closures</u>
Class 10	<u>Java Design Goals</u>
Class 11	<u>Types, Values, Variables</u>
Class 12	<u>Arrays; Classes</u>
Class 13	<u>Midterm (June 10)</u>
Class 14	<u>Inheritance; Polymorphism</u>
Class 15	<u>Interfaces; Packages; Strings</u>
Class 16	<u>Exceptions</u>
Class 17	<u>Threads</u>
Class 18	(continue)
Class 19	<u>(Scripting vs Systems PL)</u>
Class 20	<u>Multiparadigm Languages: Python</u>
Class 21	<u>Multiparadigm Languages: Scala</u>
Class 22	<u>Polymorphic Types : SML</u>
Class 23	<u>Logic Programming</u>
Class 24	<u>SCHEME INTERPRETER (2/3 classes)</u>
Class 25	<u>Code (scm/txt)</u>
Class 26	<u>Hand Written Slides (83M pdf) (43M pdf)</u>
Class 27	<u>Specifying a Language : COOL</u>
Class *	<u>Parameter Passing Mechanisms</u>
Class *	<u>Implementing Subprograms</u>
	<u>Final (July 24)</u>

Assignments (Summer 2013)

- Assignment 1
- Assignment 2

Exams (Summer 2012)

- Midterm
- Final

T. K. Prasad