The University of Georgia
Department of Computer Science

Course Information Sheet
CSCI 4500
Programming Languages

Brief Course Description
(50-words or less)

In this course we will explore modern programming languages and the paradigm -- procedural or imperative, functional, and logic programming -- that each strives to accommodate. Projects involve at least three languages to get a feel for the language paradigms.

Extended Course Description / Comments

We will cover both past and present languages, with an emphasis on modern programming languages. We will look at a wide spectrum of styles of programming that include imperative, scripting, functional, logic and object oriented languages and evaluate their strengths and limitations. Specific topics include:

- Syntax and semantics.
- Names, binding and scope.
- Imperative, functional, logical based and object oriented paradigm.
- Types.
- Control flow.
- Programming: Functional, Scripting and Logical Programming.

Pre-Requisites and/or Co-Requisites

CSCI 1302 (Pre-Requisite)
Software Development in Java

CSCI 2720 (Co-Requisite)
Data Structures

Approved Textbooks
(if more than one listed, the textbook used is up to the instructor's discretion)

Recommended Textbooks:
Author: Michael L. Scott
Title: Programming Languages Pragmatics
Edition: 3 or later.
ISBN-13: 978-0123745149 or later.

Author: Robert W. Sebesta
Title: Concept of Programming Languages
Edition: 9 or later
ISBN-13: 978-0131395312 or later

Specific Learning Outcomes (Performance Indicators)

At the completion of this course students should be able to do the following:
1. Explain the differences between imperative, functional and logical paradigms.
2. Explain why it is important to understand these programming language paradigms.
3. Explain when (and why) one paradigm is more applicable than another paradigm.
4. Create a lexer (using a tool like flex or lex) for a simple language.
5. Create a simple parser (using a tool like bison) for simple language.
6. Create and design a program using a functional programming language.
7. Create and design a program using a logical programming language.

8. Create and design a program using a scripting language
9. Demonstrate comprehension of short programs written in functional, imperative and logic paradigms.
10. Explain and evaluate design and implementation features of programming languages.

Relationship Between Student Outcomes and Learning Outcomes

		Student Outcomes										
		a	b	c	d	e	f	g	h	i	j	k
Learning Outcomes	1		•							•	•	
	2									•	•	
	3		•							•	•	
	4		•	•						•		•
	5		•	•						•		•
	6		•	•						•		•
	7		•	•						•		•
	8		•	•						•		•
	9		•							•	•	
	10		•	•						•	•	

Student Outcomes

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
d. An ability to function effectively on teams to accomplish a common goal.
e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
f. An ability to communicate effectively with a range of audiences.
g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
h. Recognition of the need for and an ability to engage in continuing professional development.
i. An ability to use current techniques, skills, and tools necessary for computing practice.
j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the **tradeoffs** involved in design choices.
k. An ability to apply design and development principles in the construction of software systems of varying complexity.

Major Topics Covered (Approximate Course Hours) 3 credit hours = 37.5 contact hours 4 credit hours = 50 contact hours Note: Exams count as a major topic covered	Overview of Programming Languages (4 hours) Programming Language Paradigms (4 hours) Programming Languages Syntax and Semantics (4-hours) Scanning in Practice (4-hours) Parsing in Practice (4-hours) Functional Languages (lazy evaluation, evaluation order, higher order functions, currying, closures, static & dynamic scope, side-effects, introduction to LISP like languages, LIPS or Scheme and modern mainstream functional programming languages like Clojure, Groovy and Scala) (8 hours) Polymorphism (4 hours) Control Flow (4 hours) Names, Binding, Scope (4 hours) Scripting (4 hours) Data types (4 hours) Logical Languages (4 hours)
Assessment Plan for this Course	Each time this course is offered, the class is initially informed of the Course Outcomes listed in this document, and they are included in the syllabus. At the end of the semester, an anonymous survey is administered to the class where each student is asked to rate how well the outcome was achieved. The choices provided use a 5-point Likert scale containing the following options: Strongly agree, Agree, Neither agree or disagree, disagree, and strongly disagree. The results of the anonymous survey are tabulated and results returned to the instructor of the course. The course instructor takes the results of the survey, combined with sample student responses to homework and final exam questions corresponding to course outcomes, and reports these results to the ABET committee. If necessary, the instructor also writes a recommendation to the ABET committee for better achieving the course outcomes the next time the course is offered.
How Data is Used to Assess Program Outcomes	Each course Learning Outcome, listed above, directly supports one or more of the Student Outcomes, as is listed in "Relationships between Learning Outcomes and Student Outcomes".
Course Master	Dr. Maria Hybinette
Course History	05/2008 Course Approved into CAPA 02/2012 Course Information Sheet Prepared