Marshall University Syllabus

Course Title/Number	Programming Languages/ CS 300
Semester/Year	Spring/2014
Days/Time	TR/11.00 - 12.15 PM
Location	GH 211
Instructor	Venkat N Gudivada
Office	GH 207A
Phone	304 - 696 - 5452
Email	gudivada@marshall.edu
Office/Hours	Tuesday and Thursday: 1:00 - 2:00 PM; Friday: 10:00 - 12:00 Noon and 1:00 - 3:00 PM
University Policies	By enrolling in this course, you agree to the University Policies listed below. Please read the full text of each policy be going to www.marshall.edu/academic-affairs and clicking on "Marshall University Policies." Or, you can access the policies directly by going to http://www.marshall.edu/academic-affairs/?page_id=802 Academic Dishonesty/ Excused Absence Policy for Undergraduates/ Computing Services Acceptable Use/ Inclement Weather/ Dead Week/ Students with Disabilities/ Academic Forgiveness/ Academic Probation and Suspension/ Academic Rights and Responsibilities of Students/ Affirmative Action/ Sexual Harassment.

1 Course Description: From Catalog

Comparative study of the concepts found in contemporary programming languages. Emphasis is on design and evaluation of a language in terms of its features and their implementation. (PR: CS 210)

This course begins with an overview of programming languages syntax and semantics, computational models (i.e., programming paradigms), and concepts that span across languages. Focus is on concepts and fundamental abstractions that characterize various programming languages. In the process, you will gain a beginning-level programming expertise in: Python, Haskell, and C/C++.

Substantial writing is required in this course. Writing assignments comprise 60% of the course grade. **Informal**, **low-** and **medium-stakes**, as well as **high-stakes** assignments are

designed to effect learning through writing. Students are exposed to concepts, principles, techniques of programming languages through these writing assignments.

One informal, ungraded writing activity takes place in every class meeting. Medium-stakes writing takes place at least once in two weeks. High-stakes writing is spread through the entire semester and involves document revision based on instructor and peer feedback.

2 Course Student Learning Outcomes

The table below shows the following relationships: How each student learning outcome will be practiced and assessed in the course.

Course Student Learning Outcomes	How **students will practice each outcome** in this Course	How **student achievement of each outcome will be assessed** in this Course
Students will enhance their **writing skills and strategies** by developing several documents and computer programs in the context of learning programming languages concepts and principles – contributes to BSCS degree program goal f (see Section 11)	Informal in-class writing	six formal writing assignments
Students will enhance their **critical thinking skills through various forms of low, medium, and high stakes writing** – contributes to BSCS degree program goal f (see Section 11)	Informal in-class writing	six formal writing assignments
Students will **engage actively with the subject matter through various form of writing** – contributes to BSCS degree program goal f (see Section 11)	Informal in-class writing	six formal writing assignments
Students will be able to **explain** the fundamental principles that underlie all programming languages – contributes to BSCS degree program goal i (see Section 11)	Informal in-class writing, in-class exercises	Two formal writing assignments

Students will be able to **demonstrate understanding** of the basic structure of programming languages including syntax, semantics, data and control abstractions, pointers, subprograms, modules, concurrency, exceptions, and polymorphism – contributes to BSCS degree program goals c and i (see Section 11)	Informal in-class writing, in-class exercises, writing small pieces of code, exploring and experimenting with programming solutions	Programming projects, exams
Students will be able to **explain** the concepts and fundamental abstractions that underlie the following languages and **demonstrate** beginning-level programing expertise in them: Haskell (pure functional), Python (multiple paradigms), and C/C++ (imperative, procedural, OO) – contributes to BSCS degree program goals c and i (see Section 11)	Informal in-class writing, in-class exercises, writing small pieces of code, exploring and experimenting with programming solutions	Programming projects, exams
Students will be able to **understand**, **analyze**, and **document** features of a programming language – contributes to BSCS degree program goals f and h (see Section 11)	Informal in-class writing, in-class guided discussion, reading papers	Formal writing assignments, exams

3 Required Texts, Additional Reading, and Other Materials

[1] Michael L. Scott. *Programming Language Pragmatics*. Third. Morgan Kaufmann, 2009.

Web Resources

- **TIOBE** Programming Community Index. Access it here.
- Brian Heinold. **An Introduction to Programming Using Python**. Available for free download under Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License here
- Bryan O'Sullivan, Don Stewart, and John Goerzen. **Real World Haskell**. Available for free download/reading under the Creative Commons Attribution-NonCommercial 3.0 Unported License in PDF and HTML formats here.
- **Learn You a Haskell for Great Good!** is a great book for learning Haskell. Available for free for online reading here.

- Allen B. Downey, **Think Python: How to Think Like a Computer Scientist**, OReilly Media, 2012. Available for free download/reading under the Creative Commons Attribution-NonCommercial 3.0 Unported License in PDF and HTML formats here.
- Peter Seibel, **Practical Common Lisp**. Free online textbook available here.
- M. Ben-Ari. **Understanding Programming Languages**. Free online textbook available here.
- J.R. Fisher **Prolog** :- **Tutorial**. Available here.
- William E. Shotts, **The Linux Command Line**. Available for fee download in PDF format under a Creative Commons license here.
- Learn **Prolog** Now — an introductory course to programming in Prolog available here.
- **Project Euler** is a series of challenging mathematical/computer programming problems. Challenge yourself here.
- **Logic** is a means mathematical proof and reasoning about correctness of computer programs. Explore Logic in Action Open Course Project here.
- Find out how **Prolog** is used in the real-world here.

4 Grading Policy

Activity	Weight
Writing assignments	30%
Programming (writing) assignments	20%
Midterm exam	20 %
Final exam	30 %

Based on my previous experience with the course, students who copy solutions to assignments by searching on the Web or from students who took the course in the past, did not do well on exams. It is important that you approach assignments as an opportunity to learn and

internalize course topics.

Course grade is awarded based on the following scheme:

Score	Letter Grade
>= 90	A
>= 80 & < 90	B
>= 70 & < 80	C
>= 60 & < 70	D
< 60	F

5 Attendance Policy

Attendance will be taken at the start of class. However, attendance does not count towards the course grade. Only University excused absences will be accepted for missing the class on exam days.

6 Course Schedule

- Week 1
  - Computational models for programming (imperative and declarative)
  - Overview of syntax, semantics
- Weeks 2 - 3
  - Syntax
- Weeks 4 – 5
  - Names, scopes, and bindings
- Week 6
  - Python
- Week 7
  - Control flow
  - Midterm exam
- Weeks 8 - 9
  - Functional programming with Haskell

- Week 10
  - Control flow
  - Data types
- Week 11
  - Data types
- Weeks 12 - 13
  - Imperative (procedural and OO) programming with C/C++
- Week 14
  - Subroutines and control abstraction
- Week 15
  -
  - Final exam (includes all topics from weeks 1 through 14). Will be held on 8 May 2014, 10:15 AM - 12:15 PM.

7 Writing Assignments

Substantial technical writing is required in this course. In fact, you will be writing on a regular basis. Writing assignments comprise 50% of the course grade. You approach writing as an effective means to learn the subject matter, rather than as a practice to improve your writing abilities. However, it is expected that your ability to write well will also improve as a byproduct because of the frequency and the amount of writing involved.

Technical writing is precise, concise, and comprehensive. Typically sentences are short and written in active voice. Long sentences are hard to read and also prone to multiple interpretations. Brevity and accuracy are the hallmarks of technical communication. Variation in sentences is good only when it makes sense. Convoluted means to introduce sentence variation just for the sake of it is considered a bad practice. We will do **two types of writing**: **informal** and **formal**.

7.1 Low-stakes, Ungraded Informal Writing

In every class we will do one or more *explain to the village idiot* type informal, ungraded writing. Village idiot is a person known for ignorance and lack of sophistication. We will write about technical concepts in plain English in a way that even a village idiot can understand. We take turns in sharing our writing with the class. Each informal writing activity is designed to take no more than five minutes of class time. Focus of writing varies from concepts, task procedures, to metacognitive reflection.

7.2 Medium-stakes Formal Writing Assignments

There will be four graded, medium-stakes programming assignments. On average, there will be one medium-stakes assignment in a three-week period. A medium-stakes assignment typically requires several hours (in the range of 8 to 15) for completion. They are completed outside the class period. We will write programs using the following languages:

- Haskell
- Prolog
- Python
- C/C++

7.3 High-stakes Formal Writing Assignments

This course requires two graded, high-stakes writing assignments:

- The primary goal of the first assignment is to assess your ability to explain basic concepts of the imperative programming paradigm. The second goal is to assess your ability to categorize programming languages using various facets.
- The goals for the second assignment is to assess your ability to understand, analyze, illustrate, and summarize a functional programming language.

Details will be provided in separate handouts.

You will be required to upload the final versions of both the high-stakes writing assignments to the General Education Assessment Repository (GEAR) for the purpose of assessing the university's Communication Fluency outcome and preparing for the HLC accreditation visit in 2015. Instructions for upload will be provided later in the semester.

7.4 Rubric for Evaluating Writing Assignments

Learning outcome: Students will develop cohesive written, technical communications tailored to software engineers. The rubric for evaluating high-stakes writing assignments from writing traits perspective is shown in Table 2.

8 Classroom Etiquette

- Students are expected to show up for class on time and participate in the class constructively.
- The practice of sitting with legs on the table is prohibited. This practice runs against the classroom dignity and decorum.
- Attendance will be taken at the beginning of the class. However, attendance has no bearing on students' course grade. Students are not penalized for not attending classes in any form. However, they are responsible for turning in assignments and projects on time and taking exams as scheduled. Due dates will be posted on muOnline.

Perf Level Trait	Intro	Milestone	Capstone	Advanced
Context/ Audience	**Identifies** potential contexts/ audiences for communication.	**Selects** a specific context/ audience for communication.	**Appraises** audience and tailors the communication with their needs/ culture in mind.	**Engages** the audience in novel ways.
Design/ Organization	**Identifies** and uses basic organizational principles.	**Applies key** design/organizational principles in communication.	Fully **develops** the design/organization of the communication in a cohesive manner.	**Creates** innovative designs of communication.
Diction	**Chooses** commonplace vocabulary that conveys the intended meaning of his/her communication.	**Chooses** vocabulary that conveys the intended meaning of his/her communication.	With the audience in mind, **chooses** a varied vocabulary that conveys the intended meaning of the communication.	**Chooses** lively, imaginative, memorable, and compelling vocabulary, skillfully communicating meaning to the audience.
Communication Style	Communication has only a few (but noticeable) errors in style, mechanics, or other issues that might distract from the message.	Communication is virtually free of mechanical, stylistic or other issues that might distract from the message.	**Uses** complex and varied sentence styles, concepts, or visual representations.	**Creates** a distinctive communication style by combining a variety of materials, ideas, or visual representations.

Table 2: Rubric for evaluating writing assignments

- ❍ No Internet browsing is allowed in the class. During the class and exams, students should turn off all types of electronic gadgets including mobile/smart phones, iPhones, iPods, blackberries, laptops. These devices must remain out of sight for the entire duration of the class. Students who violate this policy will be asked to leave the classroom. Repeating offenders of this policy will be administratively withdrawn from the course.

9 muOnline

It is important to visit muOnline regularly for up-to-date information about the course. It hosts all the course materials including assignments, handouts, lecture notes, and reading

materials.

10 Policy for Students with Disabilities

Marshall University is committed to equal opportunity in education for all students, including those with physical, learning and psychological disabilities. University policy states that it is the responsibility of students with disabilities to contact the Office of Disabled Student Services (DSS) in Prichard Hall 117, phone 304-696-2271, to provide documentation of their disability. Following this, the DSS Coordinator will send a letter to each of the student's instructors outlining the academic accommodation he/she will need to ensure equality in classroom experiences, outside assignment, testing and grading. The instructor and student will meet to discuss how the accommodation(s) requested will be provided. For more information, please visit http://www.marshall.edu/disabled or contact Disabled Student Services Office at Prichard Hall 117, phone 304-696-2271.

11 BSCS Degree Program Goals

a. an ability to apply knowledge of computing and mathematics appropriate to the discipline, including the ability to analyze and evaluate performance tradeoffs of algorithms, data structures, and hardware solutions;

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, including software systems of varying complexity, 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, both written and oral, with a range of audiences;

g. an ability to analyze the local and global impact of computing on individuals, organizations, and society;

h. a 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, including the ability of expressing algorithms in at least two of the most important computer languages currently in use in academia and industry.

12 Bibliography

[1] Richard Reese. *Understanding and Using C Pointers*. http://it-ebooks.info/book/2265/. O'Reilly, 2013.

[2] TIOBE Software. *TIOBE Programming Community Index*. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. 2013.

[3] Carlos A. Varela. *Programming Distributed Computing Systems*. The MIT Press, 2013.

[4] Allen B. Downey. *Think Python: How to Think Like a Computer Scientist*. O'Reilly Media, Inc., available for free under the Creative Commons Attribution-NonCommercial 3.0 Unported License in PDF and HTML formats at http://www.greenteapress.com/thinkpython/, 2012.

[5] David Griffiths and Dawn Griffiths. *Head First C*. http://it-ebooks.info/book/704/. O'Reilly, 2012.

[6] Brian Heinold. *An Introduction to Programming Using Python*. Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License, http://faculty.msmary.edu/heinold/Introduction_to_Programming_Using_Python_Heinold.pdf, 2012.

[7] Mark J. Johnson. *A Concise Introduction to Programming in Python*. Chapman & Hall/CRC, 2012.

[8] Ben Klemens. *21st Century C: Tips from the New School*. O'Reilly, 2012.

[9] Huw Collingbourne. *The Book of Ruby: A Hands-On Guide for the Adventurous*. No Starch Press, 2011.

[10] Conrad Barski. *Land of Lisp: Learn to Program in Lisp, One Game at a Time!* No Starch Press, 2010.

[11] Vernon L. Ceder. *The Quick Python Book*. Manning Publications Co., 2010.

[12] Mark Lutz. *Programming Python*. Fourth. O'Reilly Media, Inc., 2010.

[13] Bruce A. Tate. *Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages*. The Pragmatic Bookshelf, 2010.

[14] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[15] David A. Black. *The Well-Grounded Rubyist*. Manning Publications Co., 2009.

[16] Jeri R. Hanley and Elliot B. Koffman. *Problem Solving and Program Design in C*. Sixth. Addison Wesley, 2009.

[17] Mark Lutz. *Learning Python*. Fourth. O'Reilly, 2009.

[18] Michael L. Scott. *Programming Language Pragmatics*. Third. Morgan Kaufmann, 2009.

[19] David Flanagan and Yukihiro Matsumoto. *The Ruby Programming Language*. O'Reilly Media, Inc., 2008.

[20] Daniel P. Friedman and Mitchell Wand. *Essentials of Programming Languages*. Third. The MIT Press, 2008.

[21] Kent D. Lee. *Programming Languages: An Active Learning Approach*. Springer, 2008.

[22] Bryan O'Sullivan, Don Stewart, and John Goerzen. *Real World Haskell*. O'Reilly, 2008.

[23] M. Ben-Ari. *Understanding Programming Languages*. http://www.freetechbooks.com/understanding-programming-languages-t657.html, 2006.

[24] David Ascher, Alex Martelli, and Anna Ravenscroft. *Python Cookbook*. Second. O'Reilly, 2005.

[25] Peter Prinz and Tony Crawford. *C in a Nutshell*. O'Reilly Media, Inc., 2005.

[26] Peter Seibel. *Practical Common Lisp*. Apress, 2005.

[27] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, 2004.

[28] Peter Van Roy and Seif Haridi. *Concepts, Techniques, And Models Of Computer Programming*. The MIT Press, 2004.

[29] Brian W. Kernighan and Rob Pike. *The Practice of Programming*. Addison Wesley, 1999.

[30] Harold Abelson and Gerald Jay Sussman. *Structure And Interpretation Of Computer Programs*. Second. The MIT Press, 1996.

[31] K.N. King. *C Programming: A Modern Approach*. W.W. Norton, 1996.

[32] Leon S. Sterling and Ehud Y. Shapiro. *The Art of Prolog: Advanced Programming Techniques*. Second. The MIT Press, 1994.

[33] Brian W. Kernighan and Dennis M. Ritchie. *C Programming Language*. Second. Prentice Hall, 1988.