NORTHERN ARIZONA UNIVERSITY
College of Engineering, Forestry & Natural Sciences
Department of Electrical Engineering and Computer Science

Course Syllabus

# CS 396 – Principles of Languages		**Spring 2016**
Credits: 3 credits	**Pre-reqs:** CS 249 with C or better	**Co-Reqs:** N/A
Section#: 1	**Co-convened with:** N/A	**Cross-listed with:** N/A

Academic Catalog Description: Intensive study of the underlying linguistic principles, constructs, and mechanisms associated with diverse programming paradigms.

Course Purpose: This course is part of the theoretical core of the computer science program, focusing on theory and implementation of programming language design. Upon successful completion of this course, you should have an understanding of the abstract design principles and implementation considerations both common to all programming languages, and associated with specific languages and paradigms. You should be able to draw on this broad understanding to critically and comparatively evaluate older, current and future programming languages; and be able to rationally choose the most appropriate language or paradigm for tackling a given software project.

ABET Program Learning Outcomes supported

Outcomes	Achievement Assessment
2.2 -- Familiarity with a broad range of programming languages and paradigms. Provides an abstract framework for understanding conceptual issues underlying all programming languages, covers all four major paradigms, and provides hands-on exploration of several languages and their implementation.	**Exams:** Broad understanding of languages, paradigms, and evolution. **Programming:** Deep understanding of paradigm function and implementation.
2.3 -- Ability to apply knowledge of formal software development concepts to select and apply software development processes, programming paradigms, and architectural models. The course has a strong focus on not only surveying various languages that exist, but also on exploring the relationships between languages and delineating pros and cons from each.	**Exams:** Pros and cons of various languages and paradigms.
2.4 -- Motivation and skills needed for lifelong learning. A major emphasis in this course is on the evolution of programming languages and paradigms over time to match advances in hardware technology and changing application contexts.	**Exams:** Understanding of evolutionary nature of languages; need to always develop/learn new tools.

Detailed Information for this offering

Time and Location: 2:20 – 3:35, Monday and Wednesday. Rm 224 Engineering

Course Website: http://www.cefns.nau.edu/~edo/Classes/CS396_WWW/

Readings and Materials:

Course Textbook: Concepts of Programming Languages, by Robert W. Sebesta

Instructor's Name: Dr. Eck Doerry

Office Building/Room Number: Rm 259 Engineering

Email: Eck.Doerry@nau.edu

Instructor Availability:

Office hours:	**MW 13:00 – 14:20** **TH 14:30 -- 16:00**
Other:	• Although you should try hard to make it to scheduled office hours, I am also available at other times by appointment. To schedule, see me before or after class or send email. • Email is appropriate for **short** questions; longer questions/discussions should be handled in person. • If my office door is open or cracked outside of office hours, feel free to knock - I may be available (but no guarantees). Talking to and helping students is what makes my job fun, so if you are having problems please find some way to come see me as soon as you can.

Course Structure and Evaluation Mechanisms:

General expectations: The internet and the course web site is where all the action is. In my classes, all homeworks, programming assignments, solutions, announcement, etc. appear on the class website. This means specifically:

- You will very rarely have anything handed to you in hard copy. If you need it, it's on the course website!
- Although there is usually redundancy, some important announcements may appear only in email! I expect you to check your email at least once a day, preferably more often.

Evaluation Mechanisms: There are three mechanisms by which your course grade will be determined. Programs allow you to develop and practice your programming skills and gain hands-on experience emulating the programming paradigms discussed in class. The exams allow you to explore the extent to which you understand the theory and principles that underlie these paradigms. This does not imply, however, that exams won't require you to read/write the occasional bit of code.

Programs: Three to four programs will be assigned to more deeply explore some of the concepts discussed in class. Programming assignments must be done individually.

Exams: There will be one midterm during the semester and one final exam.

Class participation: Interaction -- both with the instructor and with other students -- is crucial for understanding and integrating the ideas presented in lecture. To emphasize this, a significant number of points are set aside to reward students who take an active role in the class. Participation points are also used to credit quizzes (in-class or on BBlearn) and other formative exercises.

Grading System:

Weighting of Deliverables:	**Grading Scale**:
The following percentages are used in weighting total points earned on programming, exams, and participation: • Programming Assignments = 40% • Midterm = 25% • Participation/Quizzes = 5% • Final Exam = 30%	90-100% = A 80-89% = B 70-79% = C 60-69% = D under 60% = F
Notes: • You must perform satisfactorily in both theoretical (exams) and practical (programs) parts of the course. If you score an "F" in either part, your total final grade percentage will be reduced by an additional 10%. • Simply completing what is required is enough to earn a "C". To get an "A" or a "B" you must show additional (i.e. above average or outstanding, respectively) analytic insight, clarity of presentation, and creativity. For more detail on what is expected for each grade level, refer to the ASEE's Guidelines for Engineering Grading and Written Presentation Evaluation Rubric linked on the course website.	

Class Outline or Tentative Schedule:

See Tentative Course Schedule outlined on Class Website

Class Policies:

Attendance: Attendance is required. You are responsible for all material covered during the lectures whether you attend or not. If you must miss a class, be sure to get the notes from another student. Late arrivals are disruptive --- plan to arrive five minutes before the start of class.

Electronic Device usage: All cell phones, PDAs, music players and other electronic devices must be turned off (or in silent mode) during lecture, and may not be used at any time. Laptops are allowed for note-taking only during lectures; no surfing or other use is allowed. I devote 100% of my attention to providing a high quality lecture; please respect this by devoting 100% of your attention to listening and participating.

Late work and Make-ups: Unless otherwise noted, all assigned work is due at the beginning of class on the date they are due! Programming assignments will be submitted electronically as well as in hardcopy. The following specific policies apply:

- **Quizzes:** No make-ups, no late work accepted
- **Exams:** Make-ups only when scheduled/approved in advance or with proper documentation (note from physician, etc.), as required by NAU policy.
- **Written Homeworks:** No late work accepted, no make-ups.
- **Programming Assignments:** Late work accepted with 10% deduction per half-day late, within limits. See Late work and Make-up Policies on course website for details.

Grade Challenges: Although I try hard to grade as accurately and fairly as I can, mistakes do occur. If you feel that I owe you some points, or would like to discuss an evaluation, I encourage you to stop by office hours. To avoid loss of context, any grade disputes must be brought to my attention no later than 5 business days after the assignment was returned.

Homework Submission and Format: All homework must be submitted in hardcopy (no email submissions accepted!). Submissions should be clearly collated, with annotated content, and stapled. For details on formatting and submission requirements. See Formatting Guidelines on the class website. For programming assignments, electronic submission of the code/application may also (in addition to packet) be required for grading. The official submission time for programming assignments is based on the timestamp generated by the electronic upload.

Academic Dishonesty: Cheating will not be tolerated and may result in serious sanctions, including immediate failure in the course. Serious incidents of academic dishonesty will also for brought to the attention of the university and may result in expulsion. All work in this class is meant to be an individual effort by the person receiving the grade. Any variation from this is considered cheating and all parties involved (giving or receiving) will be sanctioned.

Additional clarification on cheating for coding assignments: Sometimes students are not clear on what is or is not cheating in regards to programming. The programs you turn in for grading should have been written and typed in by you without referencing another's work (you may refer to books, of course --- but you may not copy substantial sections of code from any book, the internet, or any other medium). It is often helpful to verbally discuss problems with others to clarify the problem to be solved and discuss possible solutions; this is acceptable and even encouraged. But you should never at any time give another student a copy of your code or accept code from another student, either physically or electronically. Some examples of cheating include but are not limited to:

- Turning in someone else's code (perhaps with minor modifications) as your own. This includes code that you found on the internet, reference texts, or elsewhere.
- Turning in fabricated output for a non-functional program (or editing output from a partially-functional program) in an effort to fool the grader into thinking the program works as specified.
- Using fragments (e.g. functions/methods) from another's code in your code, passing them off as you own.

Occasionally, it is acceptable to use a peripheral utility code written by someone else but you should always (a) clearly document the source of the code fragment and (b) check with me first.

Other Important Course Information:

Vital Skills and other Hints for success in CS396

Be active, not passive.
One of the skills you should develop is an ability to read difficult material on your own - you will exercise this skill in the present course. Try to read at least lightly through the material in the assigned reading before the related lecture. You are expected to read the material more thoroughly before (and preferably well before) any exams. Also read introductory sections, summaries, and chapter notes, and skim exercises and problems to gain an idea of what's going on and where to find material!

Avoid hacking.
Writing a program can be a funny thing. Some people are very fast programmers, others are quite slow. One common complaint I hear is that someone spent XX hours on a program and still couldn't finish. Others might spend only a couple of hours. This disparity emphasizes the difference between programming and merely "hacking". It is important to make sure you understand what has to be done, the concepts associated with the assignment, and that you have a plan or outline for your program -- before you write a single line of code. All of this will help to decrease the time you spend in front of the computer wondering why something doesn't work.

Strive for elegance.
Because of the course emphasis upon programming in an actual language, and because of the overall purposes of our program and the pedagogical placement of this course within the program, I emphasize good style including modularization (e.g., class design), pretty-printing, and appropriate commenting in my evaluation of your programs. Unpleasantness like global variables, inefficiency, and just plain ugly code will be penalized. In short, it is important for you to learn to produce clean, efficient, and easily-maintainable code, not just code that works.

Don't procrastinate.
Don't delay programming assignments until the last minute. Unlike writing a paper for an english class (which is done when you want it to be), a program is a living thing, full of errors that must be corrected before it is done. How long this takes is completely unpredictable. So be sure to leave yourself enough time --- time to think carefully about the design before you program, and time to resolve whatever problems do arise. Remember, whatever can go wrong will and at the least opportune time. My willingness to help you with your assignment outside of office hours greatly diminishes in the hours immediately before it is due.

University Policies:

This course is conducted in accordance with all applicable university policies which can be found at: http://nau.edu/OCLDAA/_Forms/UCC/SyllabusPolicyStmts2-2014/ including Safe Environment, Students with Disabilities, Academic Contact Hours, Academic Integrity, Research Integrity, Sensitive Course Materials, and Classroom Disruption Policy. The student handbooks are also valuable resources for other policies. The undergraduate student handbook is at http://nau.edu/student-life/student-handbook/.