CSCI 1133 - Introduction to Computing and Programming Concepts
(4.0 cr; Prereq-MATH 1271 or Math 1371 or Math 1571H; fall, spring,
summer, every year)
Main entry point into the computer science major.
Fundamental programming concepts using the Python language.
Problem solving skills, recursion, and object-oriented programming.
Algorithm development techniques. Use of abstractions and modularity.
Data structures and abstract data types. As part of the course, students will develop substantial application programs to solve real-world problems. The course will provide students with both programming skills and a broad exposure to some of the major ideas in computer science. Integral weekly lab.

SLO
Student in the course:

- Can identify, define, and solve problems

Please explain briefly how this outcome will be addressed in the course. Give
brief examples of class work related to the outcome.

CSci 1133 is focused on computational problem solving. Specifically,the
course labs, homework assignments, and exams all ask the students to
solve various problems. For example, for a given problem students must engage
in problem-solving tasks such as clarifying any ambiguous aspects of the
problem definition, decomposing the problem into subproblems, deciding which
computer-related problem solving strategies (such as recursion) might be
useful in solving the problem, constructing a solution, implementing the
solution as a computer procedure, and verifying that the solution is correct
(including modifying it when it is not).

How will you assess the students' learning related to this outcome? Give brief
examples of how class work related to the outcome will be evaluated.

This SLO will be assessed through labs, homework, exams, and in-class
exercises. Each of these types of student work is problem-based. They will
often focus on well-defined problems students need to solve; however, they
will also involve some open-ended problems where students need first to
identify, define, and/or clarify what the problem is before solving it.

Sample course syllabus

CSCI 1133 - Introduction to Computing and Programming Concepts

3 hours of lecture per week. 2 hours of lab per week.

CSci 1133 is one of the entry points into the computer science major.
Students should take it in their freshman year. It is one of the courses
required for admission to the computer science major.

CSci 1133 does not assume any previous programming knowledge; however, it
does have a co-requisite of Calculus I. This means you should either have

completed Calculus I successfully, or should be taking Calculus I. Some material from Calculus I, such as differentiating polynomials, may be used in 1133; moreover, the mathematical and logical reasoning skills used in Calculus I also play a heavy role in this class.

CSci 1133 offers an introduction to the fundamental principles of programming and to different programming paradigms, with emphasis on the design of abstract data types and object-oriented programming.  Specifically the course covers:

* The concept and properties of algorithms;
* The role of algorithms in the problem-solving process;
* Fundamental design concepts and principles (abstraction, program decomposition, encapsulation and information hiding, separation of behavior and implementation);
* Fundamental data types and structures (numbers, strings, tuples, lists, dictionaries) and abstract data types (stack, queue, binary tree);
* Strategies for choosing the appropriate data structure.

Upon successful completion of the course students should be able to:

1. explain what a computational process is and be able to express computational processes in the Python programming language.
2. trace the execution of a variety of code segments and explain their computations.
3. explain and use basic principles of program design, such as abstraction to hide implementation details, and problem decomposition to control the intellectual complexity of the problem.
4. use the Python language to implement, test, and debug algorithms for solving simple problems using strings, lists, stacks, and dictionairies.
5. explain what an abstract data type is, how to create new abstract data types, and how to express them in Python.
6. design appropriate data structures and algorithms to solve a given problem,
7. compare and contrast the costs and benefits of dynamic and static data structure implementations.
8. implement a coherent abstract data type, with loose coupling between components and behaviors.

Grading:

| | |
|---|---|
| 15% | In-class Exercises |
| 6% | Quizzes |
| 12% | Midterm Exam 1 |
| 12% | Midterm Exam 2 |
| 20% | Final |
| 25% | Labs |
| 10% | Homeworks |

Using the above weights, a total of 90% and up will earn you some level of A, 80% and up at least some level of B, 70% and up at least some level of C, 60% and up at least a D.

Class Webpage: All future handouts, assignments, announcements, and any additional material will be available through the 1133 class web page in the directory http://www.cselabs.umn.edu/classes/

Labs: Weekly labs include a variety of problems designed to help you

acquire programming skills in the Python language and to develop good
software design practices. The labs will also expose you to interesting
problems and will give you a sense of the breadth of applications of
computing.
Labs will be graded on correctness, completeness, and style. Correctness
and completeness refer to how well the program works. Style includes
good design, readability (indentations, descriptive names for variables
and procedures, and appropriate use of blank spaces), and useful comments.
Testing your program for correctness is very important and you are
expected to submit your own test cases along with the results. You should
include test cases that verify where your code works as well as cases that
show where it fails.

Scholastic Conduct: Although you are free to discuss assignments with others,
the work you submit for grading must represent your own efforts only. Exams
are closed book and note and are to be completed using only your own knowledge
of the course material. The experience gained from the labs will be very
helpful for the exams and future labs may build on previous ones, so put in
the effort needed to fully understand the solutions. Cheating on quizzes or
exams is a serious offense, and will be dealt with as such. Additionally, labs
are done in groups of two, but collaboration with others outside your group of
two is prohibited. Homework assignments are individual efforts. You may
discuss (in a general way) labs and homework problems with others, but you may
not collaborate by writing code or specific solutions with others (with the
exception of your lab partner in the case of labs). Copying other's answers,
or letting another person copy your answers (either intentionally or as a
result of negligence) is a serious situation and can result in failing the
course. For further information on academic misconduct please see
http://www-users.cs.umn.edu/ barry/intro/acad-conduct.html. If you have any
questions about what is and is not allowable in this class, please ask the
course instructor.

Incompletes: Incompletes will be given only in very rare instances when an
unforeseeable event causes a student who has completed all the coursework to
date to be unable to
complete a small portion of the work (typically the final assignment or exam).
Incompletes will not be awarded for foreseeable events including a heavy
course load or poorer-than-expected performance. Verifiable documentation must
be provided for the incomplete to be granted, and arrangements for the
incomplete should be made as soon as such an event is apparent.