CS 110: Introduction to Computer Science

Syllabus of Practice

Northern Arizona University • College of Engineering, Forestry & Natural Sciences • Department of Electrical Engineering & Computer Science

Prepared by Steven M. Jacobs, Lecturer
Department of Electrical Engineering & Computer Science
For the purpose of certifying CS110 as compliant for the NAU First Year Learning Initiative
http://www2.nau.edu/fyli-p/wordpress/

Course Information

Catalog Description: Introduction to algorithmic thinking and object-oriented programming for students with little or no prior programming experience. Emphasis on graphical programming, object-oriented problem solving, and basic Java and Python concepts.

Broad Topics: Intro to Computer Science, algorithms, object-oriented and structured programming, Alice, and Java with Greenfoot, Python

Prerequisites: None

Corequisites: MAT 125 (Pre-calculus) or better

Skill Level: Introductory

Credit Hours: 3


Class Web Site: http://www.cefns.nau.edu/~smj93/cs110

2nd Web Site: We will use the BbLearn web site for this class to submit some homeworks, provide access to some course material, and post grades.

Instructor Information

Instructor: Prof. Steven M. Jacobs

Office Hours: Engineering Bldg. Rm 254. Check Prof. Jacobs' web site Other times by appointment.

Email: Steven.Jacobs (at) nau.edu

Phone: (please email)

NAU Address: Box 15600 Flagstaff, AZ 86011-5600
Overview

CS110 is a blended class; in that significant student learning will occur before and after class time with the use of online resources for automatic grading, immediate feedback and student accountability. The comprehensive set of materials for each individual class meeting in CS110 includes:

• **Learning outcomes** for each lecture. Four or five learning outcomes per 50 minute lecture. These are critical to guiding all other efforts in the course design. The learning outcomes are provided to the students to guide their study efforts.

• **Guided readings** on specific pages of the book or internet research with notes on important concepts, equations, and figures. These readings are to be completed before the lecture. A reward system is in place to produce the homework early, so that the students are prepared with questions. The extra-credit-for-early-work technique motivates students to read the textbook before lecture. Another option is pre-lecture quizzes, but, this instructor has seen success with first-year students with this approach.

• **Quizzes** on Blackboard Learn to reinforce concepts outside of class. These are low stakes quizzes that are graded automatically. These are very simple questions to give students a basic introduction to the material.

• **In-class activities** with “review sheets” (worksheets) that are carefully designed to help students achieve the learning outcomes. These problems include conceptual problems as well as creative thinking problems. An example is developing a language to command a robot to move from Point A to Point B, with or without obstacles in front. Visualize the Rumba (carpet cleaning) robot being controlled by the student using a new programming language. Another exercise is a problem with dice, where a computer language is developed using multiple dice. Think of that like Morse code, there each symbol has a specific meaning and translation, and the relative position of the dice can have meaning. One example is dice touching each other can be a letter, while separated dice can represent a number. Students work in groups to encourage discussion and deeper thinking.

• **Web links to videos** will be shown in class because computer science is a technology-driven subject, covering the latest computer science news and technology. Students find it intriguing to know how computers are shaping the world we live in.

• The NAU computer science faculty speak to the students of CS110 each semester. Their purpose is to recruit students into the major and answer questions about student expectations.

• **Homework assignments** that are closely tied to the learning outcomes and include reading assignments and internet research. These problems will serve as a formative assessment to help students to identify their progress in achieving each learning outcome. The same homework assignments will be used for all sections in a semester.
• **Programming projects** will be assigned, building a scaffolding of successively more difficult programming problems. One programming project will utilize pair programming ([http://en.wikipedia.org/wiki/Pair_programming](http://en.wikipedia.org/wiki/Pair_programming)) to ready students for future CS classes and industries that use this methodology.

• **Project presentation** requirement: students will be asked to demo at least one project to the class --- the first being a computer game. The students demo the game, explain the game design (game objective, how the game works, how you progress to a higher level of game proficiency, how you win), discuss the game code (project solution), and answer student questions. Students are also asked what features they would add to the game program if they had more time to do so. Being able to present a computer program design and creatively express future enhancements in front of a group is a key skill required by industry. Past experience with this has shown the students are very proud to demonstrate their work and find the class presentation assignment quite fun.

• **Midterm Exams** that will be given two times per semester and coordinated between sections for consistent evaluation and grading.

• **Final Exam** will be cumulative over the entire semester to encourage students to review the material one final time.

The objective is to make the interaction between instructor and students more of a discussion. Students will already have an introduction to the material through the pre-lecture reading when they come to class. Students work together to answer questions and solve problems, discussing solution methods and points of confusion. The problems are designed to explain the same concepts presented in a traditional lecture, but the students are deriving the results themselves. Class time is a discussion of the topic while students are engaged in problem solving. All students are engaged. Since students will already have an introduction to the material through the pre-lecture reading, the face-to-face time is a high-energy, collaborative learning experience. The ultimate goal is to make an undergraduate class more like a graduate student experience.

The out-of-class activities for the redesigned class are totally different than in a traditional lecture course. The pre-lecture activities (guided reading) are at a basic, introductory level. Because on-line quizzes are computer-graded, the students are held accountable. Problems are worked in class with the professor available for questions and feedback.

In general, students must have a consistent weekly effort with this structure.

These methods in general provide a significant amount of formative assessment to students. The pre-lecture reading and in-class worksheets are low stakes assessments at an introductory level that encourage students to assess their learning. Summative assessment measures of student learning include programing projects and exams. Student attitudes will be assessed through surveys.
All course materials will be made accessible as needed by individual students with disabilities. The course materials will be developed in accordance with the Quality Matters Program rubric (http://www.qmprogram.org/files/QM_Standards_2011-2013.pdf).

Benefits

The anticipated benefits of this project are as follows.

1. Improved learning in this class. Part of this is due to the spacing effect, in that students must study on a consistent basis because of the weekly pre-lecture assignments. Part is due to more time spent on solving problems at the appropriate level (scaffolding) with increasing difficulty and stakes from pre-lecture to lecture to homework then exam. Some homework will be on paper to help students develop problem solving skills and presenting their work.

2. Improved student skills and abilities in this foundational CS course will improve performance in follow-on classes, especially in CS 126.

3. Lower DFW rate due to higher student engagement with the material, with other students, and with the instructors.

4. A pair-programming (team of two) project will insure that every student has peers of differing abilities and backgrounds. This is valuable interaction in that it improves the student's communication skills, cultural mindfulness, and ability to work in a diverse environment, as well as their skills and abilities in this particular class.

5. First Year Learning Initiative (FYLI) guidance will improve course content and student success.

5. Uniform course design for all sections.

Emphasis on student success

It is the opinion of this instructor that students need to be informed of the expectation and culture of student success in the classroom. The first day lecture, office hours, open class discussion boards, rapid email response to student questions, and a syllabus statement that student participation and attendance will lead to success are all contributing factors to the students doing well in this course.

Student success is a joint responsibility – that the instructor is there to facilitate their success, but students need to come to class and do the work.

A short list of success factors for students in this course include, but are not limited to:

Habits of Highly Successful Students
1. Attend class
2. Listen
3. Read the book
4. Ask questions
5. Get help when you need it
6. Make friends with someone in class
7. Do not miss assignments
8. Manage your time
9. Practice what you have learned
10. Start programming projects early

The above list should appear in the syllabus and should be discussed the first day of class.

A welcome message is always sent to CS110 students using the NAU “classList” email tool (reference: http://classlists.nau.edu/faculty) that sends email to students who are enrolled or who join the class late. This shows from the start the instructor commitment to student success and engagement.

**Teaching the Course**

For the instructor, this course is significantly different than a traditional lecture course. The students are expected to look over material in the book and complete pre-lecture homework before lecture. During lecture, a small amount of material is presented then students work in groups of two or three on an activity, followed by a class discussion of the answer and methods. The instructor can expect to spend less time lecturing and more time answering questions from individual students or groups as they work on an activity. The instructor's focus should be "What are students able to do" rather than "What I am going to talk about".

This course is organized into three parts, with an exam at the end of each of the first three parts. There is a comprehensive final exam. Each part is composed of the three programming languages discussed in this course: Alice, Java, and Python. Each topic has textbook reading assignments associated with it. The Alice text is fundamental and required. Other readings are made available to students to learn Java and Python for the reasons that it is readily available via the Internet, and to keep student costs down by requiring only one textbook.

The following materials are provided in the learning management system (presently Blackboard Learn or BbLlearn):

1. Detailed pre-lecture readings and learning objectives for each topic.
2. Pre-lecture homework and/or quizzes
3. PowerPoint slides of lectures and guest (other CS faculty) lecture presentations
4. Code examples to facilitate student learning

5. Student-graded review sheets with solutions to be provided to the students after recording for participation points.

6. Web links to current videos

Copies of the lecture slides should be available to the students well before class starts. The instructor provides copies of review sheets. These should be collected at the end of class and used for participation points. It is recommended that the faculty member look these over at the beginning of the semester and note any missing work to let students know that their work is being evaluated. There is no need to grade the review sheets as we go over them in class. Returned review sheets are provided after class so the students can use these to study for homework and exams.

The pre-lecture quizzes and/or homework are motivation to look at the material before class. Extra credit is used as a motivator to hand in work one class period early. Options for on-line quizzes must be carefully specified in four places: date available, date due, date no longer available as well as repeating the due date in the text box description of the assignment, quiz or project. Remember in BbLearn to make the quiz only available until the due date, as the annotation for late quiz taking is not as clear as late assignments submittals.

Exams should cover as many of the learning objectives as possible and all questions should be directly tied to a learning objective.

Attendance should be taken in each class. You may include attendance as part of the grade if you want.

If you have any questions If you have any questions about this course and the way it is being taught, please contact Steve Jacobs, steven.jacobs (at) nau.edu.

**BbLearn Shell and web site design**

The Blackboard Learn Shell is organized as shown below, as well as a screenshot of a recent class web page. It is up to the instructor to tailor this for their particular class.
Welcome to CS110

Intro to Computer Science
CS 110 – TuTh 11:10 to 12:25 – Spring 2013
explore the art and science of computational problem solving

Class Web Site
Link to class web site

CS 110 Spring 2013 syllabus.doc

PowerPoint slides plus
PowerPoint plus material presented in class

Assignments including projects and some quizzes
Project assignments will appear here. Homework will be uploaded here. Some quizzes may appear here, too. Tests are usually conducted in class.

Web Links
Links to web sites discussed in class.

Alice code examples
Alice .java files discussed in class.

Greenfoot
Availability: Item is not available.
Please do not share Greenfoot textbook data outside of this class.

Python
Availability: Item is not available.
Python course materials
# CS 110 Web Page / Fall 2012

## Assignments & Announcements

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Due Date and Time</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Exam Study Guide</td>
<td>Fri 10/12, 10:00 AM - 12:00 PM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 14: Retrospective</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 13: Introduction to Python</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 12: Python Basics</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 11: Greenfoot scenarios</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 10: Greenfoot reading</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 6: Robot Algorithm</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 9: Greenfoot reading</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 8: Greenfoot reading</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 7: Aisle to Java</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 5: Introduction to Greenfoot</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 6: Aries</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 5: Programming</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 4: Interactive Object</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 3: Adventure Game</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 4: Alice &amp; Objects</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 3: Bruce Patent</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 2: Object Movement</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Project 1: Xen Editing</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 2: Low-Level Basics</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
<tr>
<td>Homework 1: Algorithms</td>
<td>Fri Dec 7, 10:00 AM</td>
<td>Page 114, 3-40</td>
</tr>
</tbody>
</table>

Welcome to CS 110!
The course syllabus is included below as Part 1. Items highlighted in red should be completed by the instructor and, of course, turned into regular black font after completion.

**Part 1. Syllabus material**

**Course Information**

Catalog Description: Introduction to algorithmic thinking and object-oriented programming for students with little or no prior programming experience. Emphasis on graphical programming, object-oriented problem solving, and basic Java and Python concepts.

Broad Topics: Intro to Computer Science, algorithms, object-oriented and structured programming, Alice, and Java with Greenfoot, Python

Prerequisites: None

Corequisites: MAT 125 (Pre-calculus) or better

Skill Level: Introductory

Credit Hours: 3


Class Web Site: [fill in URL of class web site here, if applicable. Otherwise, point students to BbLearn and delete reference below to 2nd web site below]

2nd Web Site: We will use the BbLearn web site for this class to submit some homeworks, provide access to some course material, and post grades.

**Instructor Information**

Instructor: Insert instructor name here

Office Hours: Insert office hour times here. Office location is: Engineering Building (Bldg. 69) Rm xxx. Check instructor web site for office hour times. Other times by appointment.

Email: Instructor.name (at) nau.edu

Phone: Instructor phone number (928) 523-xxxx

NAU Address: Box 15600

Flagstaff, AZ 86011-5600

**Course Description**

*Computer Science is no more about computers than astronomy is about telescopes.*

--Edsger Dijkstra
Computers are - without a doubt - one of the most important, most pervasive, and yet least understood and grossly underutilized inventions of the human race. Their raw power is staggering - they have the ability to perform billions of calculations on billions of numbers per second.

What do you do with that kind of power? You solve problems. Math problems, including physics, chemistry, biology, and astronomy. Data management problems, including data storage, retrieval, and transmission. Analysis problems, including statistics, data mining, projections, and simulations.

Few people know how to take advantage of that computation power directly, instead relying on software that others have written - that computer scientists have written, to be exact. Computer scientists aren't interested in computers per se - we're interested in using computers as tools to solve mathematical, organizational, and analytical problems. Computer scientists are problem solvers.

Being a computer scientist requires thought and imagination. You must be able to comprehend the logical implications of every programming decision, to anticipate what the structure of a data model will be like after every step of a long sequence of operations, and to develop new techniques to solve new problems. These are skills that anyone can learn. Developing and honing these skills is an important part of becoming a Computer Scientist.

This course exposes you to the modern object-oriented programming languages of today and teaches you how to use them. At the same time it cultivates the critical thinking skills that can be used to understand and solve problems of all kinds. All of this is set in a context of digital animation, gaming, and simulation.

By the end of the semester you should be able to:

- Understand what Computer Science is.
- Understand object-oriented programming, including the concepts of classes, objects, methods, and object interaction.
- Understand structured programming, including the concepts of sequencing, selection, and iteration.
- Be able to logically break a problem down into equivalence categories, formulate algorithms to solve each category of problem, and write a program that includes the solution for each possible category.
- Create programs in the Alice and Java languages.

Liberal Studies

The mission of the Liberal Studies Program at Northern Arizona University is to prepare students to live responsible, productive, and creative lives as citizens of a dramatically changing world. To accomplish the mission of Liberal Studies, Northern Arizona University provides a program that challenges students to gain a deeper understanding of...
the natural environment and the world's peoples, to explore the traditions and legacies that have created the dynamics and tensions that shape the world, to examine their potential contributions to society, and thus to better determine their own places in that world.

CS 110 supports this mission by helping you:

- Gain a deeper understanding of the tools and processes that enable and drive our technologically-oriented society.
- Explore the history and culture of Computer Science, a field that significantly contributes to all other science.
- Understand the nature and purpose of computer programming and be able to apply it to solve various problems you encounter in life.

CS 110 is a course in the Science Distribution Block and supports the intent of that block by:

- Teaching you a programming language or two.
- Teaching you how to program and problem-solve with a programming language.
- Cultivating highly logical and algorithmic thinking.
- Exposing you to the common algorithms and techniques that are the basic building blocks of all programming.

Through the program students acquire a broad range of knowledge and develop essential skills for professional success and life beyond graduation. In addition to discipline-specific skills, this course will emphasize critical thinking, an essential skill defined in the University's Liberal Studies Program. By completing all the coursework in the class, you will meet all three learning outcomes specifically linked to critical thinking:

- You will be able to convey, to an intended audience, the meaning of a statement. You will be able to take an existing program and describe what it does and, just as importantly, take an algorithm you wish to employ and write it as a logically sound series of computer commands.
- You will be able to assess the validity of a claim, taking into account different conceptual schemes, contextual factors, and evidence. With computer programming, every statement and every action is either manipulating or evaluating the state of the problem or the solution. Computer "bugs" are the result of logical fallacies - actions that fail to take into account all the possible inputs or consequences and analysis that doesn't include essential bits of information.
- You will be able to evaluate an argument by determining whether the conclusion would be probable if the premises were true. This is necessary when developing algorithms - once you've identified a scenario or "case" to solve, you must be able to employ, with confidence, instructions that will correctly implement the solution.
Schedule

Week # (Monday of that week) Topics covered

Week 1  Computer concepts
Week 2  Algorithmic thinking
Week 3  Alice - objects and methods
Week 4  Alice - control structures and functions
Week 5  Alice - properties and techniques
        Test 1
Week 6  Alice - custom methods
Week 7  Alice - object-oriented thinking
Week 8  Alice - object state
Week 9  Greenfoot - Getting to know Greenfoot
        Greenfoot - First Greenfoot programs
Week 10  Test 2
Week 11  Greenfoot - More sophisticated programming
Week 12  Greenfoot - Interacting objects
Week 13  Python - Introduction
Week 14  Python - Input, Processing, Output
Week 15  Python - Functions and wrap-up
Finals Week  Final Exam

Assessment and Grading System

The coursework includes the following assignments and tests:

- 11 Homework Assignments at 10 pts each

  Homework assignments involve research using book and online resources to answer specific questions. They help to fully prepare you for and familiarize you with the current lecture topics. Points are awarded for correct answers.

- 8 Programming Projects at 20 pts each

  Programming projects are where you put your knowledge you've learned into practice, transitioning from the theoretical to the practical with hands-on experience. Points are awarded based on the completeness and quality of your work and the thoroughness of your project report.

- 2 Tests: 100 pts each
  1 Final Exam: 100 pts
The tests and the final are an incentive for you to ensure you fully understand the topics being covered - as well as demonstrating that fact to the instructor. Points are awarded for correct answers. Total points on the tests and final may vary. Regrade requests of test questions (or homework) may include regrade of entire test or homework.

• Participation points including in-class review sheets, in-class discussion, and online discussion board

• Total: 645 pts (may vary depending on class progress)

Your class grade is based on the standard scale of points earned: 90%=A, 80%=B, 70%=C, 60%=D, below 60%=F. No grades are curved or dropped, though there are opportunities for extra credit. Projects are individual effort.

Assignments are due in-class on the due date. Graded assignments and tests are handed back during class.

**Extra Credit**

There will be a number of extra credit possibilities that involve doing extra work on assignments. Besides that, you can also get +4 points on any project by turning it during class in at least a full class period early.

**Late Policy**

Project and homework assignments are accepted with a 10% late penalty per school day, i.e. 50% penalty may be imposed for work one week late and 100% penalty for work submitted over two weeks late.

If you miss a test or know you will miss a test, discuss the matter with me as soon as possible.

**Communication with professor**

Outside of class, please contact your professor by attending an office hour or via regular email: professors.name (at) nau.edu (not BbLearn email) for any questions, e.g. requesting an excused absence, assignment content, or your status in the class. Please include "CS110" in body or subject of the email message.

**Attendance & Absentee Point Reductions**

Regular attendance is expected. Attendance is taken. Don't be late, and don't leave until class is dismissed. While class attendance is expected, please be cautious about attending class if you are feeling ill. Please inform me by email if you are feeling unwell; if you are experiencing flu-like symptoms, you should not attend class; please take precautions not
to infect others, and seek medical attention if your symptoms worsen. Remember, unless you are ill or have a family emergency, it is unwise to not miss any classes. Recall that absences do not include institutionally documented and approved absences. Besides illness, absences are also permitted other medical reasons, or family matters, if discussed in advance of the missed class. If attendance is poor, I will use my judgment at the end of the semester to drop one letter grade for poor attendance.

**Failed Final Policy**

If you score less than 50% on the final exam, your final class grade will be reduced by one letter grade.

**Lectures and the Book**

The lecture topics follow the same general outline as the book. However, the lecture complements the book rather than being a mirror of it. If you only read the book or only pay attention to the lecture you're likely to end up missing some key concepts. To get the most from the class, read each chapter before we discuss the corresponding topic in the lecture, then use the lecture as an opportunity to reconsider the key points of the material and ask questions on anything you're confused on.

**Plagiarism and Cheating**

Students are to work independently and without consultation with other students unless the assignment specifically states that you may collaborate. Grades are a way to motivate students and to evaluate students' mastery of a subject and their ability to get work done. The grades you get are not themselves truly important, but instead are representative of your knowledge, capabilities, and work ethic, and those are the things that matter.

If you plagiarize source code, fabricate results, make fraudulent claims, or attempt to cheat in any way, you are misrepresenting yourself, your level of understanding, your capabilities, and your ability to accomplish things. It is dishonest and unethical.

Anyone who plagiarizes, copies, fabricates, or cheats will at the least receive a zero on that assignment or test.

Consulting with others and using their advice on projects is fine. However, the work you submit should be your own work that you thoroughly understand and are entirely responsible for.

**Pre-requisites and dropping the course**

If you have not completed the prerequisites for a course as stated in the academic catalog or if you are absent from class during the first week, you may be administratively dropped from the course before the twenty-first day of the term. Do not rely on your
instructor to drop you from the courses that you want to drop. You are responsible for changing your own course schedule.

**Web Page**

Most assignments and handouts will only be available on the class web page at [put class web site here, either a custom web site as most computer science courses do and/or refer to the class BbLearn web site] --- they will not be handed out in class. Projects will be posted at least a week before they're due - usually the next project will be posted on the same day as the old project is due. Homework assignments will be posted by the Friday of the week before they're due. Any clarifications, corrections, and announcements will be posted on the web page.

**Computer Access**

The projects and many of the homework assignments are programming problems requiring a computer to solve. You can use your NAU computer account to access lab computers and our Unix computer remotely.

We will not be spending any class time in the computer lab. You are responsible for going to the lab on your own time or working from home to complete the assignments. You can use the general lab in room 106, the PC lab in room 317, the computers in the building's Internet Cafe, or any of the various PC labs around campus.

The Engineering Building's computer lab hours are as follows: M-Th 10am-8pm, F 10am-5pm, Sun 4pm-10pm. The Internet Cafe is open 24/7 with keypad access from the outside. If you are an Engineering student you can get a keycode at the Engineering front desk - I'm afraid this option isn't available to non-Engineering majors.

Most of your questions on general computer setup can be answered at the following web pages:

- CEFNS Information Technology:
  [http://www.cefns.nau.edu/it/](http://www.cefns.nau.edu/it/)

- NAU ITS Academic Computing Virtual Help Desk:
  [http://www4.nau.edu/achd/](http://www4.nau.edu/achd/)

**Habits of Highly Successful Students**

1. Attend class
2. Listen
3. Read the book
4. Ask questions
5. Get help when you need it
6. Make friends with someone in class
7. Do not miss assignments
8. Manage your time
9. Practice what you have learned
10. Start programming projects early

University Policies

There are a number of university policies that govern your education and safety that all students should be aware of. These are:

- Safe Working and Learning Environment
- Students With Disabilities
- Accommodation of Religious Observance And Practice
- Institutional Review Board (And Use Of Human Subjects)
- Academic Dishonesty
- Medical Insurance Coverage For Students
- Classroom Management
- Evacuation Policies

You will find a complete description of each policy here:

http://www4.nau.edu/avpaa/UCCPolicy/plcystmt.html

Also, please review the latest version of the on-line NAU Student Handbook here:

http://nau.edu/Student-Life/Student-Handbook/

Resources for Student Success

Successful university students take advantage of services and resources designed to boost learning and achievement. NAU recommends that you begin with:

- MyFoundations - use this online tool to assess and develop required university skills at your own pace (free for first-time freshmen at NAU Flagstaff)
- Supplemental Instruction - attend these course-specific review sessions whenever offered; proven to reduce the chances of earning a D or F
- Student Learning Centers - free drop-in, online, and individual tutoring appointments for math, writing, and over 100 courses; available Monday through Friday
- ResourceConnect - your online central navigation point for all NAU student resources

For a full-listing of University College services visit: http://nau.edu/University-College/
MyFoundations

Need to fill a gap? Brush up on your skills? Whether you need to get up to speed for your calculus class or brush up on your essay writing skills, the MyFoundations Self-Assessment and Development tool gets you on track for university-level academics. Free to all incoming first-year NAU Flagstaff students - topics include:

- Math
- Reading
- Writing
- Study Skills

How it works

1. Self-Assess: Complete a path builder assessment in the topic area of your choice, which creates specific modules for your personalized learning path based on your demonstrated needs for improvement or development
2. Self-Develop: Complete the learning paths for mastery

- Instant feedback
- Choose activities that fit your learning style
- Work at your own pace

Where to find it - "MyFoundations" is in your course list in BbLearn