# Syllabus

## Course Meeting Times

Lectures: 2 sessions / week, 1 hour / session

Recitations: 2 sessions / week, 1 hour / session

Tutorials: 1 session / week, 1 hour / session

## Course Objectives and Expected Outcomes

### Learning Objectives

Upon completion of 6.001, students will be able to explain and apply the basic methods from programming languages to analyze computational systems, and to generate computational solutions to abstract problems. In particular, students will:

- Be able to explain and apply the major mechanisms for control of complexity in large programming systems:
    - Building Abstractions

    - Controlling Interaction through Conventional Interfaces, and
    - Designing New Languages

- Be able to discuss issues of programming style and programming aesthetics.
- Be able to read and modify a substantial (20 pages) Scheme program, if it is written in exemplary style.
- Be able to design and implement programs in Scheme that demonstrate the concepts covered in the course, specifically:
    - Recursive and Iterative Processes and Procedures
    - Higher Order Procedures
    - Object Oriented Methods
    - Data Abstractions
    - Procedures with State
    - Dispatch on Type

- Be able to understand and modify an interpreter for a Scheme-like language, either at the level of a register machine description or at the level of a higher order language description.

## Measurable Outcomes and Assessment Methods

Students completing 6.001 will have demonstrated an ability to:

- Design, implement and test short (1/2 page) recursive programs in Scheme that demonstrate the following concepts (measured by problem sets and quizzes):
    - Building Abstractions:
        - Computational Processes

        - Higher-order Procedures
        - Compound Data
        - Data Abstractions

    - Controlling Interactions:
        - Generic Operations
        - Self-describing Data
        - Message Passing
        - Streams and Infinite Data Structures
        - Object-oriented Programming

    - Meta-linguistic Abstraction:
        - Interpretation of Programming Languages
        - Embedded Languages

- Analyze the operation of a Scheme-like interpreter using the substitution and environment models (measured by problem sets and quizzes).
- Design, implement and test at least one substantial (5 pages) program (measured by project).

## General Information

### Instructors

In Charge:

Prof. Eric Grimson

Lecturers:

Prof. Trevor Darrell
Prof. Peter Szolovits

## Scheduled Exams

Quiz 1: After lecture 8

Quiz 2: After recitation 19

Final Exam: At the end of the term during finals period.

## Subject Meetings

6.001 is a large subject, but we have tried to design it so that you can receive a lot of personal attention. In particular, you are expected to participate actively in recitations and tutorials, and the quality of your participation will be a major factor in your grade. It is even more important that you attend and participate in these, because there will be less "in person" interaction in the lectures, as described below.

## Lectures

In previous years, the entire class met twice a week for lectures, in "intimate" groups of 250-400. Over the past few years, we have engaged in a major educational experiment in which the lectures are replaced with online lectures, to which students were expected to listen, either in the 6.001 lab, or on their own computer. This term, based on positive feedback from last term's experiment, we are using the following plan.

Lectures will be given live on Tuesdays and Thursdays. Most of the lectures will also be available online, but may not (and probably will not) be identical to the live lectures. You should treat these online lectures as a kind of audiovisual textbook - they expand on the material presented in lecture, and as opposed to a live lecture, you can listen to them (if you wish) on your schedule and at your pace.

Lectures are the primary vehicle for introducing and motivating new material, some of which is not in the book. It is essential that you listen to the lectures (whether live - which we prefer - or online), as the recitations will assume you have already heard the material, and will build upon it. The online versions of lectures can be accessed via the 6.001 online tutor system, which will be described in the related resources[1] section.

The first time you use the tutor, it will ask you to register to set up a username and password. Once registered, you will find a button "Choose a Lecture" on the main tutor page that will take you to the lecture page. As well, you can use the "Set Preferences" button to set your choice of how you want to view the lectures. We plan to allow viewing of the lectures by Microsoft® PowerPoint®, as a Javascript™, or as an HTML® presentation. If you prefer not to listen to the audio, the latter two methods also will provide text annotations to go with each slide. Details on the use of the tutor are described below.

## Recitations

During the first lecture you will be filling out a form, which we will use to assign you to a recitation section. In order to keep sections balanced, we ask you to attend that section. If for some reason you cannot, you **must** contact the course secretary by e-mail, asking to switch sections. She will try to find a new section to accommodate you, but because of the need to keep sections reasonably balanced in size, she may not always be able to do so. Please do not contact her until after the first day of recitations.

We will also be using the information form to schedule you for a regular weekly tutorial. Details on this assignment will be handled during recitation.

The class is divided into sections of about 25 students. Each section meets twice each week (on Wednesday and Friday). Recitations expand upon the material currently being introduced, as well as introducing supplementary material that is not directly covered in lecture. They also give you a chance to practice working with the material in an interactive setting. Recitations are the primary source of interaction with the staff, and **your attendance at recitation is essential to good performance** in this class. This is your opportunity to ask questions about the lecture material, to work through examples that reinforce the material, and to explore variations on topics introduced in lecture.

## Tutorials

Tutorials will be scheduled in your recitation section during the first week of classes. Tutorials are one-hour small group meetings held once each week, on Mondays or Tuesdays. They provide you an opportunity to obtain individual help, to review homework assignments, and to have your progress in the subject checked. **Attendance at tutorial is essential!** If you are unable to attend a tutorial, you must contact your tutor in advance to make alternate arrangements for that week.

## Course Materials

The textbook for the course is: Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*[2]. 2nd ed. Cambridge, MA: MIT Press, 1996. ISBN: 9780262011532. The book is also available (without the discount) from the MIT Press bookstore. In addition, an online version is available, and can be found through the link on the readings[3] section.

Your major source for subject materials is the 6.001 web page. On it you will find general announcements, a detailed syllabus and lecture schedule, downloadable implementations of Scheme (the programming language used in 6.001) for use on personal computers, copies of the projects, guidelines for preparing homework, specific information associated with each recitation section, and useful documentation.

Included in the tools[4] section (under "Scheme documentation") is the document: "Don't Panic: An Introductory Guide to the 6.001 Computer System." **You should browse through this handy reference manual before going to the lab**, as it provides useful descriptions of the 6.001 system, Scheme and Edwin (an Emacs-like editor used in 6.001). It also contains useful information about how to use the 6.001 lab.

## Assignments

Problem sets are released weekly, onto the 6.001 online tutor system. This tutor is found by clicking the appropriate link in the related resources[5] section.

Each problem set will include questions intended to reinforce the material covered by the lectures. This includes two different types of problems. Some are short "finger exercises" intended to directly reinforce material from the each lecture. Others are longer problems intended to allow you to explore ideas introduced in the lecture. These problems are due on the date listed, typically Tuesday at midnight.

As you will see, the tutor system provides you with an opportunity to check your answers. This will provide you with feedback as to which problems you have understood, and which still need work. Please note that once you press the **"submit"** button, the answers for that question will be locked in. This is equivalent to your turning in your answers to the TA, and thus you cannot change them. Your answers must be submitted by the due date listed on each problem set. Your work will be reviewed by your TA, who will discuss it with you in tutorial.

In addition to the problem sets, there will be several substantial projects this term, each involving extensive programming that uses the 6.001 computing facility or a personal computer. The programming assignments have been planned on the assumption that you will do the required reading and other preparation before you start programming. It is generally much more efficient to test, debug, and run a program you have planned before beginning than to try to do the planning online. Students who have taken the subject in previous terms report that failing to prepare ahead for programming assignments generally ensures that the assignments will take much longer than necessary. Not only is it more efficient to begin work on each problem set soon after it is distributed, but it is advantageous to complete your computer work early. If you use the 6.001 Lab, you'll find that there is larger demand for the laboratory facilities and for help from the laboratory assistants just before assignments are due.

**Late homework will not be accepted**.

In case of illness or absence from MIT, make arrangements to complete assignments with your recitation instructor and tutor.

## Grades

Your grade in 6.001 will be determined by the following approximate weighting (though the staff reserves the right to consider other factors such as participation in adjusting this formula):

Grading Table

| ACTIVITIES | PERCENTAGES |
|---|---|
| Two Mid-term Quizzes | 25% |
| Final Exam | 25% |
| Projects | 30% |
| Problem Sets | 10% |
| Course Participation in Recitations and Tutorials | 10% |

However, **you must do the projects to pass the course**; a passing grade based on the other parts may be converted to a failing grade if you do not turn in all the projects, where turning in a project means including a serious attempt to complete each project.

- Homework: You are expected to do all the homework. While performance on exams is an indication of basic competence, performance on homework is your major opportunity to demonstrate outstanding achievement in 6.001. Mediocre homework performance will result in a lower grade, even if performance on exams is good. It is virtually impossible to get an A in 6.001 unless all homework assignments have been turned in. Missing more than a couple of the homework assignments may result in a failing grade for the semester, regardless of performance in exams, tutorials, and recitations. This applies to the weekly problem sets and to the programming projects.

- Participation in Recitations: You are expected to participate actively in recitation and tutorials. Short review problems will often be given in lecture, and these are to be worked on and prepared for the following recitation where you may be called upon to provide your answer to that problem. **Recitation participation is essential and counts for 5% of your grade!**
- Participation in Tutorials: You may be asked to explain or to expand upon your written homework solutions in order to demonstrate your mastery of the material. **Tutorial participation is essential and counts for 5% of your grade!** Missing tutorials will have a major impact on grades; students who repeatedly miss tutorial should drop the course.

## 6.001 Policy on Collaborative Work

Most people learn more effectively when they study in small groups and cooperate in various other ways on homework. This can be particularly true in programming assignments, where working with a partner often helps to avoid careless errors. We are very much in favor of this kind of cooperation, so long as all participants actively involve themselves in all aspects of the work - not just split up the assignment and each do only a fraction.

We are structuring the work this term into two types: problem sets that you should turn in weekly and projects that you should turn on the listed dates. Please abide by the following guidelines with regard to these different types of work. Problem sets (all the on-line work) are designed to reinforce key concepts. These should be completed by each student individually, though seeking tutoring help from Lab Assistants or other staff is perfectly appropriate. Projects are designed to be larger scale activities, in which group activity in brainstorming and design is often a key component. For these projects, we encourage you to work with one or two other people. When you turn in your project, you must identify with whom you worked. We expect, however, that you are involved in all aspects of the project, that you write and comment your own set of code, and that you **write**

**up your results separately**. When you hand in material with your name on it, we assume that you are certifying that this is your work and that you were involved in all aspects of it. Do not just turn in a copy of a single file; write your own version. This means that you create this file yourself, and not just annotate a copy that you received from someone else. We know that this may sound like replication of work, but an important part of learning the material is making the process an active one, particularly with respect to editing, executing, and debugging software, which you do by ensuring that you create and explain your solution.

Here is an example scenario of how a good collaboration might work:

- Both (all) of you sit down with pencil and paper and together plan how you're going to solve and test things. You go together to a cluster and sit at adjacent machines. You check after each problem to make sure that the others have working implementations and are all caught up. When one of you has a problem, the others look over your shoulder. For example, your partner has a bug on one part, and you are able to point out where the bug is and how to fix it. On each part of the problem, you write your own code and solution, seeking help from the others when you have difficulties. On the write-up, each of you lists the names of all of your collaborators.

Here is an example of an inappropriate collaboration:

- You send your friend a copy of your code so far. She works on it to complete the procedure you had not finished, and she fixes a bug in another procedure. You each submit this shared code and solution. Even though you list the names of each other as collaborators, this is inappropriate collaboration because you were not both involved in all aspects of the work - you did not each write your own implementation even if to a common plan, and you shared a common set of code and writeup.

Not listing the name of a collaborator will be deemed cheating. Similarly, remember that copying another person's work and representing it as one's own work is a serious academic offense and will be treated as such.

In general, we strongly encourage you to work as a group. It is a very effective way of catching conceptual and other errors, and of refining one's thinking and understanding. Also note that if you are having trouble solving a problem, please take advantage of the Lab Assistants. Part of their responsibility is to answer questions and provide advice and guidance on the course material.

See the discussion of the use of "bibles" below for additional information on use of materials in completing problem sets and projects.

## The 6.001 Programming Laboratory

The laboratory facility for 6.001 contains 40 personal computers (donated to MIT by Intel), laser printers, and network servers. There are sign-up sheets in the lab that you can use to reserve machines, in the event that they become difficult to access.

The 6.001 laboratory is staffed by undergraduate lab assistants (LA's) who are there to help with the assignments and answer questions about the system. Perhaps more so than the availability of machines, a key benefit to your working in or visiting the lab is the opportunity to get help from the lab assistants. Lab coverage hours will be posted. Please do not attempt to cajole lab assistants into staying beyond these hours - they work long hours as it is, and they need occasional sleep.

The 6.001 Lab is reserved for the exclusive use of students in 6.001. You'll find it convenient to work here because the Lab Assistants are available to help you, and you can share the warmth and camaraderie of your classmates while you work on problem sets. If you want to use your own computer, the 6.001 staff provides implementations of Scheme for Linux® and Windows® (sorry, no Macs). See the tools[6] section for software and installation instructions.

The problem sets and projects are designed so that they should run on all of these implementations, and you can move your work between them if you find this useful. For example, you might start problem set at home, then spend some time debugging it in the lab where the Lab Assistants can help you, and then finish things up at home.

6.001 is not officially supported on the MIT server. The implementation of Scheme on the MIT server that runs on Linux® machines should be compatible with the 6.001 implementations, although we do not guarantee that it can run all the problem sets. Other versions of Scheme on the MIT server are likely not to be compatible with 6.001.

The lab machines are set up as the MIT server machines in the following manner, however. They will automatically save all your work in your MIT server home directory. You can perform most of the MIT server operations on these machines. Please note that this is not a supported MIT server platform, so don't bug I/S employees with problems. Send all bug reports, comments, and questions about the lab setup to the TA.

## Using your own Computer

You'll find it convenient to work in the 6.001 lab because the lab assistants are available to help you, and you can share the warmth and camaraderie of your classmates while you work on problem sets. In addition, you can send email if you are in need of technical assistance with the Scheme system.

If you want to use your own computer, the 6.001 staff supports implementations of Scheme on machines running the MIT server-Linux®, GNU/Linux®, Windows® 95, NT, 2000 or XP. We do not support Macintoshes®.

Furthermore, if you use the MIT server, you can only run Scheme from an IBM or a Dell computer. Finally, you cannot run Scheme from a dialup MIT server machine. If you have a PC capable of running Scheme, we suggest that you install it there, since it will be convenient for you to work at home. The 6.001 lab is probably the best place to work if you want help, however, since that is staffed by knowledgeable and friendly Lab Assistants.

See the tools[7] section for software and installation instructions.

One way you can use the MIT server as a repository for your own working files, and for transferring the files between the 6.001 Lab and your personal machine. See the 6.001 problem set web pages on the MIT server for information on transferring files.

The 6.001 project code is available in the projects[8] section. (Most of the problem sets will not require that you load pre-designed code, but the projects will generally build upon code that we provide for you.)

## Workload

6.001 is time-consuming - but the assignments are not intended to require excessive amounts of time. In past subject evaluation surveys, students have typically reported that they spend very close to the expected 15 hours per week on the subject. Spending enormous amounts of time in 6.001 is often the result of simply not asking for help when you need it. If you find yourself spending more time on 6.001 than you think you should, please speak to your recitation instructor. It is also possible to spend an excessive amount of time programming. This is often the result of failing to prepare properly, i.e., not reading the assigned text and exercises, understanding the distributed code, developing plans to solve the questions, etc.

In addition, please be aware that prolonged computer usage combined with poor posture or improper typing habits can result in conditions such as repetitive strain injury. Remember to take frequent short breaks and to consult the medical department for more information.

## Getting Help

The 6.001 staff is always willing to help you. If you feel that you are getting lost, cannot understand the statement of a problem, have a gripe about the way things are being done, or have any other problem that we might be able to help with, please come see us. We hope that you will develop a good relationship with your tutor and recitation instructor as the term progresses. Your recitation instructors will have posted office hours; often these are underutilized, and you should take advantage of these to overcome confusions or deepen your understanding. Do not hesitate to call for help.

## 6.001 Bibles

Collections of past 6.001 homework assignments and solutions are available in various living groups. A database of 6.001 problem sets and solutions is also available on the MIT server. This material must be used **very** carefully. It is legitimate to use bibles as a source of supplementary problems for additional practice, to try to test and increase your understanding of the material.

It is **not** legitimate to use bibles a source of code or solutions to any of this year's assignments. Doing so is not only likely to hinder your learning the material, it is intellectually dishonest and a form of cheating. **In recent terms several students made use of bibles in this way, resulting in disciplinary action taken against them. Do not copy from bibles**. Solve the problems on your own, work on the projects in an appropriate fashion consistent with the collaboration policy, or get assistance from the teaching staff.

1. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/related-resources
2. http://mitpress.mit.edu/books/structure-and-interpretation-computer-programs
3. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/readings
4. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/tools
5. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/related-resources
6. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/tools
7. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/tools
8. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/projects