Overview: This is an introduction to computer science, which focuses on pro-
gramming (using the Python language). The topics include: design; algo-
rithmic thinking; recursion; object-oriented programming; ethics in com-
puter science; and some basics about computer systems: machine lan-
guage, interpreters, compilers, and data representation.

Meetings: Lectures: Mon+Wed+Fri 11-11:50 (CS-115A) or 12–12:50 (CS-
115B). Labs: various times on Thurs and Fri Exams: You must also reg-
ister for the course D110B for exams which will be held on Thursdays: 26
Sept, 24 Oct, and 21 Nov 5:00-6:05pm!

Email: To contact the instructors or TAs, please use Stevens email —not Moo-
dle mail. We will also use forums in Moodle.
**Prerequisite:** Substantial exposure in high school to C, C++, Java, or some other major imperative programming language — or else CS 105 or 110. Students with no prior programming experience are strongly recommended to take CS 105 or 110 first. (For CS majors, it should be CS 110). This course will cover programming in Python, from the beginning, without assuming any specific background — but it covers the basics at an accelerated pace.

Students with a great deal of programming experience and expertise may be allowed to take CS 181 instead of 115; contact the instructor of 181.

**Required textbook:** CS For All (http://www.cs.hmc.edu/csforall/), by Christine Alvarado, Zachary Dodds, Geoff Kuenning, and Ran Libeskind-Hadas.

**Software:** During the first lab session you will download the Python programming environment (details in lab).

*Warning!* Do NOT use the newer-sounding version Python 3.4. Both 2.7 and 3.4 are current branches of the language, but they have different feature sets. The reason we use 2.7 is that it has better support for graphical libraries and other support resources used through the course.

**Other resources:** We will use various material from the “CS5 Black” course at Harvey Mudd College (www.hmc.edu) (HMC) as well as our own additions. This and the textbook should be all you need.

If you want more, the Python Tutorial (http://docs.python.org/tutorial/) is a good introduction to the language features, but not an introduction to programming. The Python Standard Library (http://docs.python.org/library/) page has comprehensive documentation.

**Credits:** Thanks to Christine Alvarado, Zachary Dodds, Geoff Kuenning, and Ran Libeskind-Hadas — faculty in CS at Harvey Mudd (and U.C. San Diego)— for providing support and adapting their course material for our use.

**Coursework and grading**

Computer science centers on programming, which is learned by doing. The main focus of your work will be *programming assignments*:

- In-class exercises, some of which will be graded (also known as *pop quizzes*).
- Labs, to be completed and graded during lab. Sometimes you will be required to work in pairs in the lab.
- Homeworks, to be completed on your own time, usually due Monday just before midnight. The assignment, and your submission, is via Moodle.

Exam format: The first three exams consist of a one-hour, on-paper component (scheduled by the Registrar as course D110B) plus an in-class on-machine
programming problem on the following day. *The final exam is on-paper only, i.e., no computers.*

The course score is a weighted average of the following categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>homework &amp; in-class</td>
<td>20%</td>
</tr>
<tr>
<td>labs</td>
<td>20%</td>
</tr>
<tr>
<td>first exam</td>
<td>10%</td>
</tr>
<tr>
<td>second exam</td>
<td>15%</td>
</tr>
<tr>
<td>third exam</td>
<td>15%</td>
</tr>
<tr>
<td>final exam</td>
<td>20%</td>
</tr>
</tbody>
</table>

The lowest lab score will be dropped. Homework will include a few extra-credit problems, which can compensate for occasionally missing a pop quiz. Letter grades, with plus and minus, are assigned using the standard scale in Moodle.

**ADDITIONAL REQUIREMENT:** to pass the course you must average at least 60% on tests, 60% on labs, and 60% homework (incl. pop quizzes).

**FINAL OPT-OUT:** If your average on the first three exams is at least 80, you may skip the final and the exam average will be used in place of your final exam score when calculating the course grade.

---

**CS 115 Policies**

You, your instructor, and the teaching assistants are bound by the Stevens Honor Code. Students are responsible for reading and understanding the course policies in these web pages and for announcements made in class and in the course email list.

You will be permitted to use the textbooks and course notes for programming assignments (homework and labs). During exams, you are not permitted to use notes, books, computing or communication devices unless a different policy is specifically announced by the instructor.

**NO LAPTOPS** during Mon and Wed class session. You MUST BRING LAPTOP to lab and to Fri class session. This policy can be reconsidered if the class average on the first exam is at least 80%.

During lecture and lab sessions please refrain from talking on the phone, excessive texting, or otherwise being impolite.

**No make-up exams, labs, or quizzes**

You must go to your assigned lab session, unless given permission in advance by a TA.

There are no make-ups for pop quizzes or exams. The only possible exceptions are in the case of death in the student’s immediate family or near-death experience of the student; advance notice is required. Make-ups may be allowed for lab in case of documented illness.

---

1. The instructors reserve the right to give a higher grade than your course score, if your performance on later assignments and exams is very strong.

2. The fine print: For this purpose, the average of the first three exams will be weighted proportionally as for the course score: 25%, 37.5%, 37.5%. Also, if you are eligible to opt out, but choose to take the final, your score on the final will be counted in the course score.
Individual work

Except when groups are explicitly allowed, work must be done individually. You are encouraged to discuss the problems with your classmates but you must not share the details of the solutions. Not by email, not by text message, not by word of mouth, etc. If you are unsure whether you have shared too much, discuss the situation with the TA or instructor; it is your obligation to avoid even the appearance of cheating.

Late homework

We all have trouble meeting deadlines, and as a near-beginning college student you are confronted with many difficult deadlines. But homework doesn’t get easier to do if it’s late, and falling behind can snowball. Hence the following strict policy for homework: 2% penalty for each hour past the deadline.

CS 115 Week by week – preliminary

This gives a rough overview but consult Moodle for weekly assignments and updates. Along with the textbook readings listed here, you will be reviewing the lecture slides and supplemental material in the lab and homework assignments (posted in Moodle).

**week of Aug 25** intro to programming, pico-bot, Python, and CS 115
- Reading: chapt 1 (Introduction)

**week of Sept 1** lists and recursion
- no class Mon the 2th (labor day)
- Reading: 2.1-2.6 (Functional Programming)

**week of Sept 8** recursion; use it or lose it algorithms
- Reading: 2.7

**week of Sept 15** first-class functions; dictionaries; brief intro to cryptography
- Reading: 3.1-3.6 (Functional Programming, Part Deux)

**week of Sept 22** review; TBA: EXAM on Thurs+Fri?

**week of Sept 29** program design; computational biology; efficiency
- Reading: supplemental material only

**week of Oct 6** binary encoding and arithmetic
- Reading: 4.1-4.3 (Computer Organization)

**week of Oct 13** computer hardware for the computer scientist
- no class Mon (fall recess); Mon schedule on Tues
- Reading: 4.4-4.6
week of Oct 20  imperative programming
  TBA: EXAM on Thurs+Fri?
  Reading: 5.1-5.3 (Imperative Programming)

week of Oct 27  references; mutability; steganography and compression
  Reading: 5.4-5.8

week of Nov 3  object-oriented programming
  Reading: 6.1-6.6 (Fun and Games with OOPS: Object-Oriented Programs)

week of Nov 10  computer graphics; inheritance
  reading: 6.7-6.9

week of Nov 17  catch-up and review
  TBA: EXAM on Thurs+Fri?

week of Nov 24  Python revealed: everything is a class
  no class Wed–Fri (Thanksgiving)

week of Dec 2   wrap-up and review
  optional reading: chap 7 (How hard is the problem?)

CS 115 Goals and assessment

At a high level, the instructor’s goals for this course are to introduce you to fundamental concepts of computer science and to help you develop your ability to design, implement, and test programs. Several skills are needed to successfully write programs, including analytical thinking, systematic experimentation, persistence and patience, organization and time management, interpersonal communication, and effective use of reference material (reading technical documentation, searching the web). We focus on algorithmic thinking and problem-solving: Analyzing requirements, algorithm design, functions and procedural abstraction, pre- and post-conditions, data abstraction, and invariants. We will emphasize techniques for design, such as data driven programming and object orientation. We will touch briefly on topics that can be studied in advanced courses, including, ranging from tools for testing and secure coding practices to theories encompassing cryptography and the limits of what is computable.

Official course outcomes

encoding Explain binary encodings used by Python for integers, real numbers, characters, and images.

execution Demonstrate the dynamic behavior of Python programs that include array access, conditional execution, looping, object reference, and method invocation (including recursive invocation), by showing successive states of a computation.
exceptions Interpret the information provided by the stack trace of a thrown Python exception.

design Given a problem description, I am able to sketch a design as pseudo-code or flowchart.

coding Given a design, I am able to implement the design as a Python program.

class Write a non-trivial instantiable Python class.

state Explain the use of memory to implement static variables, instance variables, and local variables. Draw the state of the activation stack at any point in a computation.

inheritance Given a Python class, write a non-trivial extended class.

testing Write a unit test.