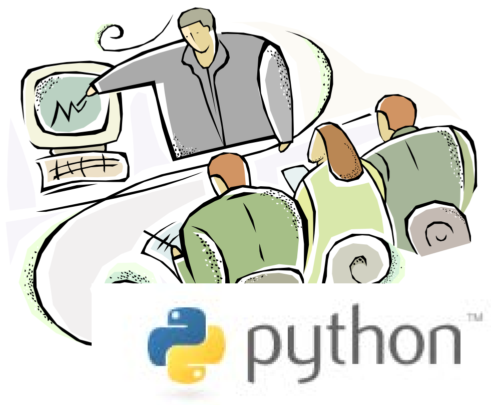
CS 115: Introduction to Computer Science, Spring 2015



Dr. Brian S. Borowski
Email: bborowsk@stevens.edu
Voicemail: 201-216-5358
Office Hours: Lieb 311, M 3 – 5 P.M. and by appt.

Course Overview:
This is an introduction to computer science, with an emphasis on programming (using the Python language). The topics include: design; algorithmic thinking; recursion; object-oriented programming; ethics in computer science; and some basics about computer systems: machine language, interpreters, compilers, and data representation.

Meetings:
Lectures: MWF 12-12:50
Labs: LA 10-11:40, LB 12-1:40, LB 3-4:40 on Thursdays

Assistants:
Office hours are held in Lieb 318 (3rd floor lounge).

LA:	Ben Knutson (bknutson@stevens.edu)
	Andy Wiggins (awiggins@stevens.edu)
LB:	Ashley Bromiley (abromile@stevens.edu)
	Ammar Mustafa (mmustafa@stevens.edu)
LC:	Greg Goldshteyn (ggoldsht@stevens.edu)
	Zach Caldarola (zcaldaro@stevens.edu)

The current hours will be posted in Canvas.

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 LibeskindHadas.

Software:
During the first lab session you will download the Python programming environment, Eclipse, and PyDev (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 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 Wednesday just before midnight. The assignment, and your submission, is via Canvas.

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

- homework & in-class assignments 20%
- labs 20%
- first exam 10%
- second exam 15%
- third exam 15%
- final exam 20%

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 Canvas.

ADDITIONAL REQUIREMENT: to pass the course you must average at least 60% on tests, 60% on labs, and 60% homework (including 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[1].

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.
- 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 (advance notice is required) or near-death experience of the student. Make-ups may be allowed for lab in case of documented illness.

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.

Communication

- As in all of my courses, you are more than welcome to ask me questions as often as you want, and I will always be happy to help.
- The amount of help provided will be directly proportional to the amount of time left before the deadline. Please don't wait until the day before an assignment is due to see

[1] 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.

me; it'll be too late for me to provide help and too late for you to truly learn the material.

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 psuedocode 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.

Approximate Weekly Schedule:

Week	Topics Covered
1	Elementary concepts of computer programming
2	Simple Python data types, list concept
3	Definition of Python functions, if/then/else concept
4	Recursion on lists
5	Filtering, map/reduce

6	Functions as values
7	Hardware representation of basic data types
8	Assembly language programming using HMMM simulator
9	Iteration
10	Representation of data: atomic vs. composite, mutable vs. immutable
11	Sorting
12	Object oriented programming: class concept
13	Object oriented programming: inheritance
14	Case study: pygame