CMPSCI 119
Spring 2021
Introduction to Programming with Python

Professor William T. Verts

Class:

Monday, Wednesday, Friday 1:25PM–2:15PM Eastern Time. Zoom links will be emailed via SPIRE the day before. Zoom recordings of the lectures will be posted the same day.

Office Hours and Email:

M/W/F 11:15-12:00 & by appointment. Zoom links will be emailed via SPIRE the day before.
I am retired and will not be participating in UMass activities on Tu/Th, but will check email.
`verts@cs.umass.edu` Personal, for asking questions. Put CMPSCI 119 in the subject line.
I read all email daily, but do not expect a speedy reply. I might not reply at all if the question is something I can address in class. Do NOT email attachments to me; they will be deleted. Do not call me at home.

Book:

Computer Science Companion, 5TH Edition, 2020 Printing, ISBN 9781792446696, $28 (paper) or $15 (digital, 180 day subscription), by me. We will get all other reference materials from the Web itself. The ***Computer Science Companion*** is a required text for COMPSCI 119, 120, and 145.

Web:
`http://people.cs.umass.edu/~verts`
`http://people.cs.umass.edu/~verts/cmpsci119/cmpsci119.html`
`http://people.cs.umass.edu/~verts/cmpsci119/quizzes/quizzes.html`

Social Media:

Please do not "friend" me on Facebook, Linked-In, or other social networks. I reserve Facebook for relatives, hiking buddies, and friends from high-school.

Course Scoring (percentages may change according to number and type of assignment):

Midterm #1	15%	late February or early March. Open notes.
Midterm #2	15%	early April. Open notes.
Final Exam	20%	Final's Week in May. Open notes.
Programming Projects	40%	Throughout semester: Late penalties will apply as appropriate.
Homeworks	10%	Throughout semester. On-Line.

Letter grades will be assigned according to final computed course score:

A ≥ 90%, A- ≥ 88%, B+ ≥ 86%, B ≥ 80%, B- ≥ 78%, C+ ≥ 76%, C ≥ 64%, C- ≥ 62%, D+ ≥ 60%, D ≥ 50%, F < 50%. Missing either of the midterms, or the final exam, incurs an automatic F for the course. Fractional final course scores are ⌈ ceilinged ⌉ up to an integer.

Do not ask for extra work after the end of the semester to boost an undesirable grade. It is unfair to other students in the class and I never grant such requests.

Please contact me directly if you have any concerns about the running of the course, the TAs, grading, etc.

Computer:

You are expected to do all work on your own personal computer. For the lectures I may switch between PCs running Windows 10 and Macs running OS/X, arbitrarily, or as my demonstrations require. While this course is largely platform agnostic, there may be some (free) programs I will have you use that are designed to work exclusively on a Windows PC or exclusively on a Mac. I may also provide some free software that will run on either platform. You may use either a Windows PC or an Apple Macintosh.

The programming environment we use is Python 3 under IDLE. There are versions that run on both PCs and Macs. Download and install the appropriate Python and IDLE environment from **`http://www.python.org/`** (Mac users also have an older version of Python already installed, accessible from the Terminal application, but we won't be using that).

Server-Side Python:

We *may* also run Python programs on a UNIX server, but at this time doing so is not certain. Accessing the server will require a UNIX account (which I will give you). It also will require special free software that I will tell you about at the appropriate time.

Course Expectations:

I expect that few students coming in to COMPSCI 119 have any experience in writing computer programs (in any language). Some do, and that's cool, but I am assuming that students have no relevant experience. At the end of the course, however, I expect students to be given any reasonable problem description and then write a Python program that solves that problem. The resulting program does not have to be the most efficient or well-written, but it must run to completion without crashing. Students must be able to modify an existing program to do new tasks and must be able to find and correct errors in that program.

Note that this course will require a substantial amount of work from and a serious commitment on the part of each student, and some students may spend many hours writing, debugging, and running their programs before those programs are ready to be turned in and graded. Partial, incomplete, or broken programs don't count: a program that crashes isn't ready to be turned in.

Final Notes:

1. **DO YOUR OWN WORK, INCLUDING HOMEWORK AND LAB WORK.** You may discuss homework and lab assignments with other students, but you may not share files or code. Upon discovery of duplication, I will contact you for a conference, as required in the guidelines set out by the University of Massachusetts Academic Honesty Policy, and we will resolve the issue according to those guidelines. See:

   `http://www.umass.edu/dean_students/academic_policy/`
   `https://www.umass.edu/honesty/`
   `https://people.cs.umass.edu/~verts/class_documents/AcademicHonestyPolicy.html`

2. Students who are registered through Disability Services should arrange for accommodations as soon as possible. See:

   `http://www.umass.edu/disability/`
   `https://people.cs.umass.edu/~verts/class_documents/DisabilityStatement.html`

Course Outline:

Here is a general sense of the topics we will cover. Many of these topics can be covered in a lecture or two; others may take several weeks.

1. Basic data types (`int`, `float`, `bool`, `complex`) and variables,
2. Python from the command line,
3. Flowcharts and program structure,
4. Simple sequential Python programs, assignment, and the `input` and `print` statements,
5. Predefined functions and parameter passing, `import` of predefined libraries,
6. Type conversion functions (`str`, `int`, `float`, etc.),
7. The `if` (`if`, `if-else`, `if-elif-else`) and `while` statements, and how they differ, and indentation,
8. Interactive loops, error checking, and exception handling (`try-except` and `try-finally`),
9. User-defined functions (the `def` and `return` statements), parameter passing, custom library modules,
10. Intermediate statements (`pass`, nested functions, `global` variables, etc.),
11. Advanced data types (lists, tuples, strings, dictionaries), string and list slicing, byte arrays,
12. Advanced techniques for building lists (`range`, list comprehensions, multiplication),
13. Advanced control structures (`for`, `lambda`, passing functions as parameters, `break`, `continue`),
14. File access (reading and writing text files and binary files),
15. Image and sound creation (`.WAV` and `.BMP` files, sine waves, DTMF tones, colors, lines, circles, etc.), Note that Computer Graphics will be represented by a *major* number of programming examples and will span several lectures.
16. Approaches to program design, evaluating alternative data structures and design techniques,
17. Debugging, debugging, debugging, (oh, and did I mention debugging?),
18. Really advanced topics (recursion, object-oriented Python), if there is time.