Sarah Lawrence College — Department of Computer Science

Introduction to Computer Programming

Course Syllabus[1] — Fall 2018

http://science.slc.edu/msiff/courses/cs1/

Meeting times

Lectures: Tuesdays *and* Friday, 11:05–12:30, Titsworth Lecture Hall

Labs: Mondays 11:05–1:00 *OR* Fridays 1:30–3:25, Science 104

Students will attend one of the labs, not both. Official lab placement will be made after the first lecture (Tuesday, September 11), prior to the first day of lab (Friday, September 14).

Instructor

Michael Siff, Faculty of Computer Science
Science Center 213
siff.michael@gmail.com
914/395-2490

Required text

The required text, available at the SLC bookstore, is:

William F. Punch and Richard Enbody. *The Practice of Computing Using Python (2nd Edition).*

Please note the book is the **second edition** (not the more recent third edition).

DRAFT SYLLABUS

Python

The course relies on version 3 of Python (which is similar, but definitely not the same as version 2). Students should use a 3.6.x release of Python.

The software can easily be installed using the Anaconda Python distribution (https://www.anaconda.com/download/). It is a fairly straightforward installation process regardless of whether the machine is running Linux, macOS, or Windows. The installation includes an interactive development environment (IDE) for Python called Spyder (https://docs.spyder-ide.org/). It allows users to write and execute programs from within the same piece of software. Spyder is installed on all the iMacs in the computer lab (Science 104). Time will be set aside during the first lab meetings (Friday, September 14 and Monday, September 17) to assist students with installing Python on their own machines if they have not already done so.

[1]Draft syllabus as of September 2, 2018, for consideration during registration.

Examinations

There will be two in-class examinations, one in the middle of the semester (Tuesday, October 30) and one at the end of the semester (Tuesday, December 18). The exams will not ask students to write code (except, perhaps, to complete a line or two); but they will ask them to *read* code. *Important: No make-up exams will be given.*

Homework

Regular reading. Students are expected to keep up with assigned readings. There will be an assigned reading for just about every class (lectures and labs). The readings will not be long, but they are fairly technical which means to get the most out of the readings it is best not to rush, and, ideally, to sit near a computer that has Python installed so that the reader may experiment as she reads.

Programming assignments. There will be eight programming assignments; they are the primary form of written homework. Programming assignments are due Thursdays at 5pm. (The first assignment is due September 20; the last December 20.) Programming assignments, unless explicitly stated otherwise, are to be completed independently (see more under the Policy section).

Problem sets. There will be two problem sets (the first due Friday, October 19, the second Tuesday, December 11). The problem sets serve as untimed practice for the two examinations. Each problem set will have two due dates. By the start of class on the first date, students need to have seriously attempted each of the assigned problems; time will be set aside during that class to answer questions concerning some of the problems (and hints may be offered to the solution of others). A typed, clearly organized, well written set of solutions to the problems is due on the latter date. Work on problems sets should be performed independently; all questions should be addressed directly to the instructor.

Late submissions will not be accepted.

The calendar for written assignments is as follows:

#	due date	theme
0	09/20	getting started
1	09/27	fruitful functions
2	10/04	ifs and loops: text-based REPL game
3	10/11	strings: anagram identification
	10/19	*first problem set*
4	11/01	2D-arrays: Conway's Game of Life
5	11/15	dictionaries: randomly generated poetry using Markov chains
6	11/29	event handling: game project, part I
	12/11	*second problem set*
7	12/20	objects & inheritance: game project, part II

DRAFT SYLLABUS

Laboratories

Laboratory sessions will be held weekly (except immediately preceding October Study Days and immediately following Thanksgiving — there are no labs on October 19, October 22, November 23, nor on November 26). The first labs will be Friday, September 14 and Monday, September 17; the last pair of labs will be Friday, December 14 and Monday, December 17.

Each student will be placed in one lab section according to their schedule constraints and available space in that section (due to physical limitations of the computer classroom — Science 104 — labs are capped at 15 students).

Although it may seem counterintuitive at first glance, there is a logic behind the Friday-Monday lab pairing — that way both labs occur between lectures and all students should have been exposed to the same concepts prior to their lab meeting.

The purpose of the labs is to offer instructor-supervised, hands-on experience writing Python programs, to reinforce concepts covered in class during the past week, and to prepare students for upcoming programming assignments. (New material will rarely be introduced during lab meetings.) At the beginning of each lab, students will be directed to a set of programming exercises to work on individually or with a partner. If assigned a partner, the pair should collaborate equally on typing at the keyboard and writing, testing, and debugging code. The lab problems will not be formally graded, but students are expected to finish them on their own time if necessary; several of the exercises will pertain directly to subsequent homework assignments. Familiarity with the solutions to the labs will improve comprehension of general programming principles and practices, streamline future labs and programming assignments, and prepare students for problem sets and examinations.

DRAFT SYLLABUS

The calendar for laboratories is as follows:

#	**dates**	**topics**
0	9/14, 17	intro to Python programming using Spyder
1	9/21, 24	fruitful functions
2	9/28, 10/1	ReadEvalPrintLoops (REPLs)
3	10/5, 8	loops, strings, slices
4	10/12, 15	files and functions
	10/19, 22	*no lab (Study Days)*
5	10/26, 29	lists, matrices
6	11/2, 5	randomness
7	11/9, 12	sorting and searching
8	11/16, 19	sets, dictionaries
	11/23, 26	*no lab (Thanksgiving)*
9	11/30, 12/3	event-driven programming
10	12/7, 10	objects, classes
11	12/14, 17	inheritance, more OO

Attendance

As stated on p. 22 of the *Student Handbook*: "Consistent attendance at all classes and conferences is expected." Unexcused absences or persistent lateness, whether in class, lab, or group conference, may lead to reduced credit. In this era of ubiquitous connectivity, an absence for which a student did not inform the instructor of in advance is *unexcused*. (That is *not* to say that merely informing in advance will excuse an absence, but it will often help.) As this is a lecture course, students are responsible for making sure they are marked present. If they arrive late or hide silently in the back row (not that the instructor recommends ever doing either), they should check in with the instructor at the end of class.

Checking email

Students are expected to check Sarah Lawrence email account frequently. I stress your `@gm.slc.edu` account because that is what I will use to communicate with you. Please reply in a timely fashion to course-related inquiries.

Syllabus

The combination of what is communicated in lecture, lab, the course Web page and course-related email forms the only authoritative syllabus. (In other words: *The course may depart from this document.*)

DRAFT SYLLABUS

Academic integrity and mutual respect

Familiarize yourself with the sections in the *Student Handbook* on **Undergraduate Policy on Academic Integrity** (pp. 20–22) and **Freedom of Expression and Mutual Respect** (pp. 129–130). I take both quite seriously.

Plagiarism

Copying code, whether electronically (as in downloading, file sharing, copying and pasting) is plagiarism. Retyping someone else's solution is also plagiarism.

Paraphrasing someone else's code by using different spacing and variables names is still plagiarism.

Web searching (whether using Google, StackOverflow or other sources) for solutions to assigned problems on line (for example, googling "python counting letters in a string") is prohibited.

If you are at all unsure if how your work habits are in keeping with the spirit of "working independently" be sure to check with the instructor.

Additional assistance

Many students may find find some of the material daunting. They should not fear. If a student finds it difficult, most likely so does the vast majority of the rest of the class. Anyone who feels the course is moving too fast or if is having trouble understanding a particular topic should talk to the instructor in class or out, in person or via email.

Grades

The instructor for this course does not put much stock in grades. Discouraged questions:

> "What kind of grade am I looking at now?"
>
> "How much is this assignment worth?"

Encouraged questions are those that deal with course content and learning fundamental concepts:

> "Why do programmers count from zero?"
>
> "Can you suggest exercises for me to feel more comfortable with recursion?"

Possibly acceptable:

> "Am I at risk of losing credit?"

(However, if students are in academic trouble in this course, the instructor will inform them.)

A formula for grading exists, but it will likely seem inscrutable. It is of this form (only more complicated):

```
(sum(D6:N6) - min(E6:J6) + sum(M6:N6))/12.75 + O6
```

Elements that factor into your evaluation:

- programming assignments
- examinations
- problem sets
- participation (to the extent possible in a lecture course)
- attendance
- effort and demonstrated improvement

The course is *not* graded on a curve. In principle 100% of the class may receive an A — in practice, that will not happen, but there is no predetermined distribution for grades.

To rephrase: Focus on learning, forget about grades.

DRAFT SYLLABUS

Calendar

week	date	topics	reading
1	09/11	What is C.S.? Why Python?	0.1–0.5; 1.1–1.5
	09/14	operations, functions	1.6–1.10
2	09/18	conditional statements	2.1
	09/21	loops	2.2.1–2.2.9
3	09/25	read-eval-print loops	2.2.10–2.3
	09/28	what makes for a good program?	3.1–3.5
4	10/02	strings: working with text	4.1–4.3
	10/05	iterating over strings	4.4–4.8
5	10/09	files and exceptions	5.1–5.7
	10/12	functional abstraction	6.1–6.4
6	10/16	lists and mutable data	7.1–7.3
	10/19	tuples (and more lists)	7.4–7.6
7	10/23	*no class — October Study Days*	
	10/26	*review*	
8	10/30	*first examination*	
	11/02	randomness	*tba*
9	11/06	binary search	*tba*
	11/09	sorting and searching	*tba*
10	11/13	dictionaries	9.1–9.3
	11/16	sets	9.4–9.5
11	11/20	recursion	16.1–16.5
	11/23	*no class — Thanksgiving*	
12	11/27	event-driven programing I	*tba*
	11/30	event-driven programing II	*tba*
13	12/04	objects and classes	11.1–11.4
	12/07	encapsulation and inheritance	12.1–12.6
14	12/11	modules	*tba*
	12/14	*review*	
15	12/18	*second examination*	
	12/21	*game demonstrations*	

DRAFT SYLLABUS

Week 1 **What is computer science?**
abstraction, abstraction, abstraction
algorithms, pseudocode and programs
primitive, conditional, iterative operations
desirable properties of algorithms
Python v. other languages
Interactive Development Environments (IDEs) — Spyder
numbers and basic arithmetic operations
syntax v. semantics; expressions v. statements
why are introductory algorithms so often numerical in nature?
text (strings), rudimentary input/output (I/O)
what's in a name? variables, assignments, symbolic abstraction
functions and functional abstraction
fruitful functions (return statement).

Lab 0 **Getting started**
programming with blocks
simple arithmetic calculations in shell
using variables to compactly express long strings
introduction to graphics.

Week 2 **Computers $\geq$ calculators**
repetition via definite (for) loops
simple conditional statements
multi-way decisions
Program 0 (warm up) due
formal parameters, actual arguments
function composition.

Lab 1 **Fruitful functions**.

Week 3 **With power (and repetition) comes responsibility**
indefinite (while) loops
basics of Boolean logic
truth tables
a bit more about algorithms
what makes for a good program?
Program 1 (fruitful functions) due
ReadEvalPrintLoops (REPLs)
some very rudimentary games.

Lab 2 **REPLs**.

Week 4 **Sequences and collections**
text processing
strings as sequences of characters
Program 2 (text-based REPL game) due
string slicing
linear algorithms.

Lab 3 **Looping and slicing**.

DRAFT SYLLABUS

Week 5 **Files and functions**
making programs more general and more robust
reading and writing text files
exceptions and error handling
Program 3 (anagram finder) due
functional abstraction.

Lab 4 **Files and functions**.

Week 6 **Generalized sequential data**
lists
mutable data
built-in list operations
tuples
first problem set due.

Week 7 (*October Study Days*)

Lab 5 **Lists and matrices**.

Week 8 **That's so random!**
our first exam
Program 4 (Conway's Game of Life) due
randomness v. pseudorandomness
deterministic v. nondeterministic computation
randomness v. unpredictability.

Lab 6 **Randomness**.

Week 9 **Searching and sorting**
what is computational complexity?
linear v. binary search
Siff's Law of Logarithms
linear v. quadratic behavior.

Lab 7 **Searching and sorting**.

Week 10 **Non-sequential data**
dictionaries
Program 5 due
sets
hashes.

Lab 8 **Sets and dictionaries**.

Week 11 **Recursion is recursion**
(*Thanksgiving*).

Week 12 **Event-driven programming**
sequential processing v. thinking out of order
basics of event-driven programming
timers
Program 6 due
introduction to concurrent programming.

DRAFT SYLLABUS

Lab 9 **Event-driven programming**.

Week 13 **Object-oriented programming**
objects
classes
encapsulation
polymorphism
inheritance.

Lab A **Objects and classes**.

Week 14 **Modules**
second problem set due. beyond programming in the small
where to go from here.

Lab B **Inheritance**.

Week 15 **Wrapping up**
our second exam
game demonstrations
Final programs due.

DRAFT SYLLABUS