Advanced Introduction to Programming

STM3910C

Class Details

Credit Hours: 3

Days Class Meets: M Th 8:30-9:45a, plus additional 1.25 hr slot TBD

Room 310

Instructor Information

Clinton Staley ("Clint")

Email: cstaley@uaustin.org or clintstaley@gmail.com

Office Hours: M T Th 9:45a-10:30a or till 11a if needed.

Text: 805 835-5024 (working hours only, pls)

Texts

None. We will use online reference material. Here are several links to bookmark for the course

Main Python Documentation

Python docs on Virtual Environments and venv

Numpy Documentation

Top Level JDK 17 Docs

JDK 17 API Documentation

Description

This course assumes moderate prior programming background, preferably in Python or Java. We'll reinforce, in Python, the basics of variables, arrays/lists, data types, if-statements, loops, functions and the like, assuming these are review topics, not new to you. Then we will cover classic data structures and related algorithms, in Java. The combination will give you all content normally offered in STM 3910B and the subsequent data structures (STM 3911?) course.

Outcomes and Objectives

Upon successful completion of this course, students should be able to do the following. The basic goals are required to pass the class. The stretch goals expand on the basic goals to reach the typical goals of the next course. Completion of these as evidenced in programming projects and tests will permit you to skip that second course.

Basic Goals

- Write and debug simple programs in Python, using programming fundamentals such as variables, I/O operations, control structures, functions and parameters, lists/arrays, basic objects, exceptions.
- Understand basics of computer hardware and software design.
- Use a Numpy-class Python library for basic numerical or array operations
- Understand references/pointers and dynamic data organization
- Understand a set of standard data structures and their use for various purposes. These include stacks, queues, BSTs, heaps, and hash tables.
- Intuitively evaluate the order of complexity of data structures and algorithms

Stretch Goals

- Implement software using classic data structures, adjustiing them as needed
- Use inheritance both for interface and implementation purposes, and use important OO design patterns.
- Test and debug software thoroughly and effectively.

Course Difficulty

Given the stated goals, and a 3-unit class, this class will obviously be a bit of a "rocket sled".

1. Be focused in class. We'll move quickly, and I'll be asking you to think about the ideas and answer questions regarding them during lecture.

2. We'll find a third meeting time during the week to offer extra coverage and support. Please think about times you might have free for this. (Tuesday 8:30a is a leading candidate at this point given our present schedule)

3. Follow through with reading and homework. I cannot cover all the concepts needed in just 2.5 (or even 3.75) meeting hours per week. Some will necessarily come from reference material. This is good professional practice anyway since that's how you'll learn in a constantly changing field after you graduate from UATX.

4. If you are confused by a concept, a coding bug, etc, *seek help in office hours*.

Grading Policy

Task	Weight
Class Participation and Quizzes	30%
Midterm (Week 6)	25%
Final	25%
Programming Projects	20%

In-Class Participation and Quizzes

Course instruction will be heavily Socratic, in that lectures will frequently include questions extending on the content. I'll call on people at random for such questions, and your response is graded. However, any reasonable answer that shows attention to the lecture and a sensible effort to answer will gain credit. The one "F" answer is "What was the question again?".

Quizzes will be unannounced, and held in the first 3-4 minutes of class. Arrive promptly if you want the full 3-4 minutes; there are no makeups. Quizzes will focus on concepts from the prior lecture or two, so the best way to do well is to carefully study the code and concepts from lecture, and arrive on time.

Programming Projects

Class homework takes the form of programming projects that fill in missing parts of code from lecture, fix bugs in it, and extend it. These projects will have up to three levels, labeled bronze, silver and gold. To be eligible for 70 or better, you must complete all bronze projects, for 80 or better, all

bronze and silver projects, and for 90 or better, all bronze, silver and gold. Note this is just to establish eligibility; your quiz, class participation, and test scores must also justify the grade.

I will label all projects pertaining to the second term data structures content as gold-only. This means you need not complete them at all to be eligible for up an 89.

And, importantly, I accept only fully correct programs, with proper style. You need to demo the program for me and get a style check (or possibly use an automated system I'll arrange) for credit. But, if you fail the demo, you can revise and try again, as many times as needed. Late penalties will accrue if you need to keep trying past the deadline, but ultimately you get 100% minus late penalties for completed projects. And for purposes of the must-complete rule above, any completion will count, even if late.

Midterm and Final

Be prepared for challenging examinations, focused on conceptual mastery, plus good vocabulary memorization. Test scores typically average 50%, but this does not imply failure. Measure your score against class average, not an absolute scale.

Respecting UATX policy, when entering grades in Populi I will renormalize them to place the class average at 85% and 1/2 of class average at 70%.

Accessibility Statement

Please review the University Accessibility Statement in the student catalog. Students having special needs should contact the Polaris Center or email Accomodations@uaustin.org.

Disability Support Services: The university will make reasonable accommodations for students with disabilities in compliance with Section 504 of the Rehabilitation Act and the Americans with Disabilities Act. The purpose of accommodation is to provide equal access to educational opportunities for eligible students with academic and/or physical disabilities.

Course Requirements

The course requires basic knowledge of programming.

Electronics

You are welcome to use laptops to take notes and try out code during class, but we will often have closed-electronic sessions, focusing on reading and debugging code examples on the class display. Phones are a different matter: please silence these and set them aside during class.

Later courses will not only permit but require use of AI assistance in coding. But, your introductory coding experience must be *without AI assistance* so you can build your own skill before relying on an AI for assistance. No use of AI for course projects in this class.

Academic Misconduct

Instructors at UATX have the authority to assess possible plagiarism, unauthorized use of artificial intelligence, and other forms of cheating in their courses. Normally, cheating will result in failing the assignment. Students may appeal such decisions to the Disciplinary Council, where they may exercise their right to a public hearing by writing to the Dean of the Center responsible for the course.

Syllabus Changes

The course instructor reserves the right to make changes to this published syllabus if it is in the best interest of the educational development of this class. Any such changes will be announced as soon as possible and, insofar as practical, after consultation with the whole class.

Attendance and Tardiness Policy

Attendance is mandatory. Each student may be tardy 1, 2, or 3 times per course per term for a 1.5, 3, or 4.5 credit course, respectively, without penalty. Each additional unapproved late arrival results in a 0.5% final grade penalty. Being over 25 minutes late for a class is an unapproved absence.

Approximate Schedule of Class

Week 1 - 3	Overview of programming basics in Python, including basic Numpy
Week 4	Introduction to Java
Week 5	Stacks and Queues
Week 6	Linked Lists
Week 7	Binary Search Trees
Week 8	Heaps, DFS/BFS
Week 9	Hash Tables
Week 10	Sorting and Recursion