DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF TENNESSEE - KNOXVILLE

COSC 101 Introduction to Programming
Fall 2025 Syllabus (Draft)

Time and Location:
Tuesday & Thursday 8:00 am - 9:15 am, Dougherty Engineering Building 416

Instructor:

- Dr. Scott Emrich
  - Office: 608 Min Kao; 865-974-3891; semrich@utk.edu (Discord for quick questions to everyone, email me for formal matters).
  - Tentative office hours: Mon 2:00pm-3:00pm, after class and by appointment.
  - If my office door is open, you are welcome to come in and ask questions.

An Invitation

My goal is for you to start learning the fundamentals of programming, get comfortable starting to think like a computer scientist, and along the way enjoy the process. This course balances a positive, supportive learning environment with developing real, applied skills that will be useful for future classes and beyond. We'll have active learning experiences and in-class engagement (aka mini "labs") to reinforce the relevant concepts by "learning-by-doing." **No prior programming experience is required**; we'll start from the ground up but expect to be challenged.

Course Webpage: https://web.eecs.utk.edu/~semrich/cs101/

Short Course Description:

In this course you will be introduced to computational thinking and structured programming using Java. Skills learned include designing a program to solve a problem, developing simple algorithms, writing and testing code, correcting errors, and documenting code. Topics include problem solving, algorithm development, conditional statements, loop-based iteration, and function calls. Students will be introduced to common data structures such as arrays and ArrayLists.

Prerequisites: Ability to use Canvas; NetID and UT email address. Programming experience will be helpful but is not required.

Programming Language for Course: Java (https://java.com/en/)

Textbook & Materials

Primary: Free online textbook via CodeHS: AP Computer Science A (link provided on Canvas). No need to buy a textbook for this course.

Additional free lecture notes, slides, and other resources will be provided.

Optional Recommended Readings: If you want a physical book

1. *Learn Java in One Day and Learn It Well* – Jamie Chan (~$17 on Amazon)
2. *Java: Programming Basics for Absolute Beginners* – Nathan Clark (~$15 on Amazon)

Equipment and Software Requirements

- A personal laptop meeting Tickle College of Engineering (TCE) requirements: Windows 10 or MacOS (Chromebooks, iPads, phones not supported). See: TCE Computing Requirements
- We will use **VSCode** as our Integrated Development Environment (IDE) and install Java during the first class period.
- You may use the lab systems in Min Kao if you have laptop issues or financial hardship. Lab machines will allow you to pursue introductory computer science while you are on campus.

Course Outcomes

By the end of the course, you will be able to:

1. Understand how a Java program "flows" from beginning to end.
2. Compile source code and understand basic limitations of a programming language.
3. Apply variables, data types, arithmetic operations, and type casting.
4. Use and manipulate objects, including String and Math classes.
5. Write and evaluate Boolean expressions and conditional statements.
6. Use loops (while, for, nested) to perform iterative tasks.
7. Write and use classes with constructors, accessors, mutators, and static members.
8. Create, traverse, and manipulate arrays and ArrayLists.

Major Topics & Approximate Hours

- **Week 1:** Programming fundamentals & program flow
- **Weeks 2–3:** Primitive types, variables, and expressions
- **Weeks 4–5:** Using objects
- **Weeks 6–7:** Boolean expressions & if statements
- **Weeks 8–9:** Iteration
- **Weeks 10–11:** Writing classes

- **Weeks 12–13:** Arrays & ArrayLists
- **Week 14:** Special topics & review
- **Week 15:** Collaborative review & final prep
  (Topic timing may shift based on class progress.)

Grading: Assignments are due before midnight on Tuesdays. Late submissions are accepted with a **10% per day deduction** (pro-rated for the first day).

You may submit **one updated assignment** for re-grading ("redo") by the end of the semester. This replaces the original grade. Extensions are only granted in documented cases (e.g., official university travel, serious illness). Requests without documentation will count as your redo request. This will not allow you to redo a later (higher point) one.

If extra credit is offered, it will be available to the entire class, not individual students.

All assignments will have an accompanying rubric so it should be clear what is required. On rare occasions TAs can make mistakes and you can email me (Dr. Emrich) to make an appeal; however, these appeals must document their grievances in a professional and respectful manner citing the pre-posted rubric. We will grade assignments as we assign; doing another assignment (e.g., one from the other 101 section) will result in a 0.

Final grades will be computed from a weighted sum of points as follows. There will be no rounding. The default weighting is:

55%: homework (including programming and written answers submitted)
15%: midterm exam
20% final exam
10%: class participation / engagement

Course percentages will be translated into letter grades as follows: A: 95% and up; A-: 92-95-%; B+: 88-92-%; B: 85-88-%; B-: 82-85-%; C+: 78-82-%; C: 75-78-%; C-: 72-75-%; D: 65-72-%; D-: 62-65; F: 0-62-%.

Absences will only be excused in accordance with university policy.

Class Participation

3.0 pts – Participate in a Code Jam during a class period (dates on Canvas)
1 pt – Visit office hours at least once during the semester (max 1 pt)
1.5 pts – Participate in in-class exercises. Not all will be pre-announced.

Most of the points will be earned by coding on your laptops in class, sometimes in pairs, along with occasional group work that reinforces a more abstract concept (e.g., algorithms, arrays, etc.)

Your participation grade = points earned / fewer points than possible (extra credit possible for perfect attendance).

Communication & Help

We will use **Discord** for class communication and help (link on Canvas).

- Ask public questions in the main course channel.
- Post private code-related questions in the ticket channel.
- Do not email code to the instructor/TAs — to ensure proper review, code must be submitted via Discord or Canvas (for final grading).

We will set up Discord together in the first week for those of you unfamiliar.

Attendance & Time Management

Programming is best done steadily throughout the week, not the night before. Expect to spend at least twice the number of credit hours outside of class, especially if you have little or no prior experience.

If you're falling behind, contact us early — we can help you catch up.

Academic Integrity

All work must be your own. Discussing concepts and algorithms is encouraged, but **do not share code**. If your code is used by someone else, both parties face referral to **Student Conduct** under university policy.

Always cite collaborators and sources in comments. If in doubt, ask us before submitting.

AI Policy: Generative AI tools (e.g., ChatGPT, GitHub Copilot) are not permitted for graded work. We will discuss acceptable use cases as long as its disclosed. As an example, I used chatGPT to (1) blend my typical syllabus format with the previous one and (2) soften by scoring each section from 1 to 10, since I have never taught freshman. This was useful since I have been teaching for a long time… you are just getting started.

ADA Statement
If you need an accommodation based on a disability, contact Dr. Emrich privately. Full accommodation will be made once approved by **Student Disability Services** (https://sds.utk.edu/).

Tips to Succeed in COSC 101

1. **Start assignments early —** waiting until the last-minute limits your ability to get help. You get 0 bonus points for difficulty/completing it the day it is due.
2. **Use multiple resources —** slides, textbook, notes, office hours, and online references.
3. **Make mistakes —**there is a very high chance the TAs and I know the answer because we got it wrong the first time also.
4. **Practice regularly —** programming is a skill learned by doing.
5. **Get help early —** don't wait until you're stuck for hours.
6. **Collaborate responsibly —** talk through ideas, not code.
7. **Review feedback —** learn from your mistakes before the next assignment.