COMP 141 - Computer Science I: Programming Fundamentals

Fall 2026

Instructor	Hong Xu
Meetings	MWF, 11:00-11:50 am, Briggs 019
Course website	See Canvas
Email	xuh@rhodes.edu
Office	Briggs 210
Office hours	See Canvas for scheduled office hours; I am also available by appointment.
CS Department Slack	See Canvas.

Official Course Description

An introduction to the fundamental concepts and practices of procedural programming. Topics include data types, control structures, functions, arrays/lists, files, and the mechanics of running, testing, and debugging. Emphasis is placed on program design and problem-solving techniques. The course also includes an introduction to the historical and social context of computing and an overview of computer science as a discipline.

Unofficial Course Description

COMP 141 is the first course in the computer science major sequence and should normally be taken during the first year. The course introduces the fundamental principles of programming, abstraction, algorithmic thinking, and design.

This course is aimed at helping students acquire the reasoning and abstraction skills needed to design algorithms and implement them as computer programs. We will practice breaking problems into smaller pieces, translating ideas into precise instructions, testing whether our solutions work, and improving our programs when they do not.

Computer science is more than programming, but programming is one of the central ways computer scientists express and test ideas. This course uses Python as the main programming language, but this is not simply a course about Python. It is a course about computational problem solving: how to think clearly, design carefully, and build programs that solve meaningful problems.

Prerequisites

None. This course does not assume previous programming or computer science experience. Students are expected to have a reasonable high-school mathematics background and a willingness to practice regularly.

Course Objectives

By the end of this course, students should be able to:

- create algorithms to solve simple computational problems;
- use Python to implement, test, and debug algorithms;
- apply abstraction and decomposition to break problems into smaller, understandable pieces;
- understand and modify algorithms and programs written by someone else;
- use variables, expressions, conditionals, loops, functions, strings, lists, and files effectively;
- write programs that are readable, well-organized, and appropriately documented;
- reason about the correctness of simple programs; and

- understand basic expectations of academic integrity and responsible computing practice.

Required Materials

Required and supplemental materials will be distributed through Canvas. These may include lecture notes, programming examples, short readings, practice problems, and links to free online Python references.

Students will also need regular access to a computer capable of running Python. We will use Jupyter notebooks for programming activities and projects. Setup instructions will be provided in class and on Canvas.

Coursework and Grading

The tentative grading breakdown is:

Component	Weight
Programming projects	20%
Programming labs and in-class activities	15%
Midterm exam 1	20%
Midterm exam 2	20%
Comprehensive final exam	25%

Grades of A-, B-, C-, and D- are guaranteed with final course grades of 90%, 80%, 70%, and 60%, respectively. If your final course grade falls near a letter-grade boundary, I may take into account participation, attendance, improvement during the semester, and demonstrated engagement with the course.

This grading scheme is tentative and may be adjusted if necessary. Any changes will be announced in class and on Canvas.

Workload

Learning to program takes regular practice. You should expect to spend at least 2-3 hours outside of class for each class meeting. Much of that time should be spent actively writing, testing, debugging, and revising code.

Programming is not learned by watching someone else program. It is learned by doing. You should not wait until the last minute to begin programming projects. A program that looks simple on the page can take substantial time to design, debug, and polish.

Attendance

Attendance is expected. Class meetings will include explanations, examples, practice problems, discussion, and lab-style work that may not be fully duplicated in posted materials.

If you miss class, it is your responsibility to find out what you missed. The best first step is usually to ask a classmate, then review Canvas materials, then come to office hours with specific questions.

Attendance, participation, and apparent improvement may be considered when assigning final grades, especially near grade boundaries. Repeated unexcused absences may affect your final grade and may result in referral to the appropriate academic support office.

Office Hours and Help

Office hours are for everyone, not only students who are struggling. You are encouraged to come to office hours to ask questions, review concepts, discuss projects, debug code, or talk about computer science more broadly.

When asking for help with code, be prepared to explain:

- what you are trying to make the program do;
- what you expected to happen;
- what actually happened;
- what you have already tried; and
- where you think the problem might be.

This helps you learn how to debug and helps me give you better guidance.

Tutoring

More senior students will hold tutoring sessions. Go to these!

Academic Integrity and Collaboration

Unless otherwise specified, all work submitted in this course must be your own and must represent your individual effort.

You may discuss general course concepts with classmates. For example, you may talk about what a loop does, how a conditional works, or how to interpret an error message in general terms. However, unless an assignment explicitly allows collaboration, you may not work together on solutions, share code, copy code, jointly write code, or look at another student's project code.

Do not submit code that you did not write. Do not allow another student to copy your work. Do not download or copy solutions from the internet. Do not use code from online sources unless I explicitly permit it for a particular project.

If you are unsure whether something is allowed, ask me before doing it. I would much rather help you clarify the rules than deal with an Honor Code violation after the fact.

AI Policy for COMP 141

The computer science department defines the following five classes of AI usage:

1. **No AI Use Permitted**: Students are strictly prohibited from using any AI tools or resources for any aspect of assignments, projects, exams, or other course-related work. This includes, but is not limited to, content creation,idea generation, analysis, problem solving, or information retrieval.
2. **AI Use for Learning and Exploration Only (No Submission)**: Students may only use AI tools as a tutor to ask for explanations of concepts, methods and terminology. AI may not be used to generate any content used for submission. (This prohibition also applies to the submission of paraphrased or altered AI generated content.) All submitted work must be produced entirely by the student without copying from AI.
3. **AI Use for Specific Approved Tasks (Limited Use)**: AI tools are permitted for specific, clearly defined tasks within assignments, provided that the AI's contribution is explicitly cited and the student demonstrates a clear understanding of the AI-generated content.
4. **AI Use with Citation (Comprehensive Use)**: Students may use AI tools as a substantial aid in completing assignments. All AI assistance must be clearly cited, specific details documented, and the student must demonstrate a clear understanding of the AI-generated content.
5. **Unrestricted AI Use (Exploratory/Research Contexts)**: Students are free to use AI tools as they see fit to complete assignments and projects, with minimal or no restrictions on their application. The focus is on the outcome and the student's ability to achieve objectives. However, citation is encouraged.

For this class, different policies apply to different assessements:

Midterm/Final Exam: 1. No AI

Programming projects and labs: 2. No Submission

Any submitted solutions to programming exercises, homeworks, labs, exams, and other graded items not wholly written by yourself or your group are considered plagiarism. All plagiarism will be reported to the Honor Council and will result in the grade of F for the course. This applies to improper AI use, copying, and any other form of plagiarism.

Programming Projects

All programs assigned in this course must be written in Python unless otherwise specified.

Programming projects will be submitted through Canvas or the programming environment specified for the projects. Emailed programs will not be accepted unless I explicitly instruct you to submit that way.

Late work policies will be described on Canvas and/or on individual projects. In general, you should assume that programming projects must be submitted by the stated deadline unless an extension has been granted.

Programming projects will be graded for both correctness and quality. A program that runs correctly but is poorly organized, difficult to read, or inadequately documented may lose points.

General programming grade guidelines

A-level work: The program is carefully designed, correctly implemented, well documented, readable, and produces clearly formatted, correct output.

B-level work: The program mostly works and is understandable, but may have minor issues such as incomplete documentation, inconsistent style, minor formatting problems, limited testing, or small design weaknesses.

C-level work: The program has more serious problems, such as incorrect output for important cases, poor organization, missing or inadequate documentation, failure to follow project instructions, or fragile design.

D-level work: The program runs but produces clearly incorrect output for typical cases, crashes frequently, or substantially deviates from the project requirements.

F-level work: The program does not run, produces little or no correct output, or does not meaningfully attempt the project.

Coding Style

Designing algorithms and writing code is not a purely mechanical process. Good code should be clear, organized, and understandable to another human reader. Programming projects will be graded for both correctness and style.

Unless otherwise specified, your programs should include:

- a comment at the top of the program with the author, date, and a brief description of what the program does;
- meaningful variable and function names;
- clear organization;
- appropriate use of functions to break problems into smaller parts;
- concise comments explaining major sections of code;
- comments for functions, unless the function is very short and its purpose is obvious;
- readable spacing and formatting; and
- avoidance of large blocks of copied-and-pasted code.

Exams and Quizzes

Exams and quizzes are individual work. Unless otherwise stated, no outside resources, collaboration, phones, AI tools, or internet use are allowed during exams or quizzes.

More specific exam policies will be announced before each exam.

Class Conduct

I encourage everyone to participate in class. Ask questions when you are confused. If something is unclear to you, it is likely unclear to someone else as well.

Teaching and learning are a partnership between the instructor and students. Your questions help you learn, help your classmates learn, and help me understand what needs more explanation.

Please observe the following expectations:

- Arrive on time and be prepared to participate.
- Keep phones silenced and put away unless they are being used for a specific class activity.
- Use laptops or tablets for course-related work only.
- Be respectful of classmates, the instructor, and differing levels of prior experience.
- If you must leave class early, let me know in advance when possible.
- If you miss class, take responsibility for catching up.

Students With Disabilities

If you have a documented disability and wish to receive academic accommodations, please contact the Office of Student Accessibility Services as soon as possible. I am happy to work with you and with Student Accessibility Services to support approved accommodations.

Diversity and Inclusion

A diverse learning community is a necessary part of a liberal arts education. We are committed to fostering a classroom environment in which all students are treated with dignity and respect.

Discrimination or harassment on the basis of race, gender, color, age, religion, disability, sexual orientation, gender identity or expression, genetic information, national origin, or ethnicity will not be tolerated.

Computer science benefits from many perspectives. Students enter this course with different backgrounds, levels of prior experience, and levels of confidence. Prior programming experience is not required, and no student should feel that they do not belong because the material is new or difficult. Struggle is a normal part of learning to program.

Sexual Misconduct Disclosure

Rhodes is committed to ensuring a safe learning environment that supports the dignity of all members of the Rhodes community. Rhodes prohibits sexual misconduct, including but not limited to dating/domestic violence, sexual assault, sexual exploitation, stalking, sexual harassment, and sex/gender discrimination.

Faculty and staff are Mandatory Reporters, with exceptions for confidential resources such as the Counseling Center, Chaplain, and Student Health Center. If you choose to share information related to sexual misconduct with me, I am required to report it to the Title IX Coordinator. You will control how your report is handled and are not required to pursue a formal claim.

For more information about Rhodes' sexual misconduct policy or to make a report, please see the Rhodes Title IX resources.

Final Exam

The final exam will be comprehensive. The date and time will follow the official Rhodes College final exam schedule.

Syllabus Changes

I reserve the right to alter this syllabus as necessary. Any substantive changes will be announced in class and on Canvas. Any changes are intended to improve fairness, class quality, rigor, and learning to cater to student interest and progress as the semester goes on.