GCIS-123: Software Development & Problem Solving I

Course Description

This is the first course introducing students to the fundamentals of computational problem solving with a software engineering approach. Students will come into the course with widely varying skill levels. Some students may have completed a formal computer science course in high school or even at the college level. Others may be self-taught avid programmers with lots of personal projects. Others will come into the course with little or no prior experience at all! The good news is that this course is designed with all skill levels in mind. Experienced students should be introduced to new topics and new ways of thinking as they solve problems. Beginning students will learn the fundamentals of computer literacy and programming as they steadily build and hone their skills.

Yes, this is an introductory programming course, during the course of which students will learn one of the most popular and widely used contemporary programming languages: Python. However, much more important than learning any one programming language will be the development of problem-solving skills; we will use an algorithmic approach to solve computing problems of varying scale; you will learn how to solve large computing problems incrementally by breaking them into small pieces that can be solved individually with the tools that you have learned in class. Problem-solving and programming are difficult things to do and require persistence! Your first attempt won't always work and that is OK. The important thing is that you take the time and put in the effort to learn from the experience so that you can increase your knowledge, understanding, and skills.

There will also be an emphasis on good software engineering practices including software design principles, unit testing, design patterns, and version control. Students that complete this and the second course in the sequence will be well prepared to tackle the most challenging computing courses throughout the rest of their college education.

Course Topics

The following topics will be covered primarily during lectures:

1. Statements, expressions, variables, standard output/input
2. Types, variables, functions, parameters, arguments
3. Arrays, Boolean expressions, conditionals, iteration
4. File I/O, raising exceptions, exception handling
5. Basic string parsing, regular expressions
6. Arrays, recursion, searching, sorting
7. Classes, objects, constructors, fields, methods

The following topics are introduced and reinforced in every assignment through practice.

1. A systematic approach to computational problem solving using software engineering including informal and formal approaches to problem solving
2. Algorithmic thinking, computational problem solving

Course Objectives

- Learn to program in a selected, contemporary, high-level programming language (Python).
- Describe and apply problem-solving skills, algorithms, and data structures that are appropriate to solve a variety of computing problems of varying degrees of complexity.
- Describe and apply fundamental concepts of software engineering including understanding needs, software design, solution testing, and incremental development.
- This course is the first in a two-course introductory sequence that is a prerequisite for many later courses. The prerequisite for enrolling in the second course in the sequence is a minimum grade of C-.

Office Hours

Every instructor holds office hours on specific days and times each week for the sole purpose of supporting their students. No appointment is necessary to visit your instructor during these scheduled office hours. There is no set agenda or prerequisite of any kind for visiting! Students are free to stop by and ask for help on coursework, but also just to chat about anything at all including personal projects, resume writing, applying for jobs, hobbies, and so on. Visiting your instructor's office hours often is a great way to get to know them, and every instructor enjoys it when their students stop by. You can find your instructor's office hours in their Content section on MyCourses. If your instructor's office hours do not fit your schedule, you should email and request an appointment.

While it's OK to visit your instructor just to chat, if you are looking for help with your coursework, it is always best to come prepared.

- At a minimum, you should prepare a list of questions that you would like to ask; consider writing or typing these questions in advance.
- If you would like help solving a problem in a program that you have written, make sure that your laptop is charged, and that your program is open and ready to review.
- Be prepared to answer the following questions:
    - What did you expect to happen?
    - What actually happened?
    - What have you tried to solve the problem yourself?

Other Resources

In addition to your instructor, there are many resources available to help you. These resources are provided because we understand that programming can be challenging and sometimes requires significant effort. It is important that you utilize these resources to support your work in learning and understanding the course material.

- The lecture slides and class activities are excellent resources because they contain all of the information and examples you need to build on to complete your out of class assignments. Try spending 30 minutes after each lecture reviewing the content and practicing the coding activities. If you are unable to solve one of the activities without looking at the solution, try asking your instructor or course assistant for help.
- Many of the Course Assistants hold *virtual mentoring hours* on the course Discord server. These mentoring hours are open to all students in the course and can be used to seek help on your coursework. You can voice chat with the Course Assistant and share your screen. See the `#waiting-room` channel for the posted

schedule of hours.
- A discussion topic will be created on the course Discord server for each assignment. Ask questions! Everyone is here to learn and chances are you aren't the only one with that question! The course staff (and maybe even other students) will provide you with answers and advice.
- The Society of Software Engineers (SSE) holds open mentoring hours every weekday. You can find the schedule on the SSE site.
- Women in Computing (WiC) offers tutoring for many classes in GCCIS. You can find the schedule on the WiC site.
- Tutoring support for GCIS 123/124 will be provided in the GCCIS Tutoring Center (GOL-2410, 2nd floor overlooking the atrium). The schedule can be found on the GCCIS Tutoring Center site. Tutoring for your other GCCIS classes may also be available. The Academic Success Center provides Math and Physics Tutoring, as well as other academic-related support.

Grading

Component	Percentage of Grade
Quizzes	5%
Class-Activities	10%
Problem Solving	10%
Assignments	25%
Practica	30%
Final Exam	20%

Each student's grade is based on a series of artifacts and deliverables that are evenly distributed throughout the semester. All grades are expected to be returned to students within 1-2 weeks of the submission date, and so all students should be provided with an accurate representation of their current standing in the course at all times in the MyCourses grade book. This grade book also includes written feedback from graders that should give you a good idea of the strengths in your submission and specific areas for improvement. It is expected that you will review this feedback, so that you may learn from it to improve your knowledge, understanding, and skills.

Every student can succeed in this course, but you will need to be proactive in seeking help and avail yourself of the resources provided to help you learn. Students that feel that they are lagging behind mid-semester will have ample opportunities to recover if they increase participation during lecture by making sure to arrive early to take quizzes, complete the in-class coding activities, and actively participate in the team problem solving. Of course, it is also important to seek help on the out-of-class assignments!

This course is the first in a year-long introductory sequence. The prerequisite for enrolling in the next course in the sequence is a minimum grade of C-.

Daily Quizzes

At the start of each lecture, your instructor will provide you with the password necessary to access that day's quiz. Each quiz includes a series of questions based on topics covered in the previous lecture. The goal of the daily quizzes is twofold: first, to encourage you to arrive on time to class each and every day, and second to refresh your memory about the topics that were covered previously. If there are any questions that you are unsure about, be sure to make a note and ask your instructor during lecture!

Quizzes are graded based on participation. Provided that the student completes the quiz in the first 5 minutes of class, they will receive full credit for that quiz. At the same time, the questions on the quizzes will be very similar to the questions on the written portion of the midterm and final exams. You would take the quizzes seriously, and ask questions if there is something on a quiz that you do not understand.

Class Activities

Each lecture includes 10-15 hands-on active learning activities designed to provide students with an opportunity to practice as they learn new topics in the course. Students are expected to spend about 2-5 minutes working independently on the activity. The goal is not necessarily to completely solve every activity within this short period of time but to think about how to apply what you have just learned to a potential solution. One technique is to try to write your solution as a "recipe" in the form of comments in your code. Course Assistants are available in the classroom to assist students during these activities. After enough time has passed to give everyone an opportunity to think about the problem, your instructor will go over the solution before moving on.

Class activities are graded based on participation. Graders will be looking for signs of obvious effort, including at least an attempted solution to each activity and frequent commits to source control throughout the lecture. Students that demonstrate such effort will receive full credit for the class activities.

Problem Solving

The last 60 minutes of class on some days will be devted to team problem solving, during which students are placed into teams of 3-4 by their instructor and presented with a series of problems to solve together. The problems are typically directly related to that day's homework assignment. Every student on the team is expected to contribute to the team's solution. Course Assistants and your instructor will be available to assist.

Problem solving is graded based on participation. Students should contribute to their team's solution by actively engaging in the discussions and taking turns writing the team's solution on the whiteboard or paper. Provided that a student is present and contributing to their team's solution, they will receive full credit.

Homework Assignments

Homework will be assigned at the end of most classes and is due before the start of the next class. The homework assignments provide students with an opportunity to practice what they learned during that day's lecture to solve problems that are significantly larger and more complex than the activities practiced during class. The lecture and class activities will have included everything that you need to solve your homework assignments - it is a good idea to look for problems that are similar and try to adapt the solutions to your homework. Students will have a minimum of two days to complete each assignment.

Homework assignments are graded based on the following rubric, which evaluates effort and adherence to instructions.

Exceptional Performance	Competent Performance	Acceptable Performance	Developing Performance	Beginning Performance	Needs Improvements	Unacc Perfo
5 (100%)	4 (95%)	3 (88%)	2 (75%)	1 (50%)	1 (25%)	0 (
The solution is complete and	The solution is functionally	The solution includes at least	The solution includes at least	The student began the	The solution shows some evidence of	Little or Almost

Exceptional Performance	Competent Performance	Acceptable Performance	Developing Performance	Beginning Performance	Needs Improvements	Unacc Perfo
5 (100%)	4 (95%)	3 (88%)	2 (75%)	1 (50%)	1 (25%)	0 (
correct. Instructions were followed completely. or The student made a small number of minor errors *for the first time*, e.g. wrongly named files, output that does not match specifications, etc. A warning should be issued regarding any errors.	complete other than a small number of functional errors. Instructions were followed completely. or The solution is complete and correct but the student made a small number of errors *that they heve made at least once before*, e.g. wrongly named files, output that does not match specifications, etc. An explanation should be issued regarding any of these errors. Instructions were followed completely.	80% of the required functionality and all instructions have been followed. or The student's effort is obvious based on the volume of work and commit history, but there are a modest number of errors that range from relatively minor to significant that can be easily fixed by a member of the course staff with minimal effort.	50% of the required functionality and all instructions have been followed. The student's effort based on volume of work and commit history is obvious. There is at least some attempt to implement the missing functionality. Errors can be easily fixed by a member of the course staff in a short amount of time.	solution, and there is obvious effort, but more than 50% of the required functionality is missing or broken. This may include code that exists but does not compile and/OR run, but could be fixed with significant effort.	effort but does not demonstrate the desired learning outcomes. Less than 50% of the required functionality is working. There are serious errors that require significant effort to fix including code that does not compile or that crashes when executed.	req functio missi commi shows activity of the
What should I do if I get this grade?						
You got full credit! But you should still read the feedback from your grader! You may have made one or more small errors that will lower your score on future assignments if they are not corrected.	So close! But you may have made one or more small errors and not for the first time, e.g. forgetting comments, not following style guidelines, etc. Be sure to read the feedback from your grader on this and previous assignments!	The solution is very close, but some minor functionality is missing and/or you may not have followed the instructions. Be sure to carefully read the feedback from your grader to understand why you didn't get a higher score this time. Also, always make sure to read the assignment instructions carefully!	An extremely solid start, but significant required functionality is missing. Read the feedback from your grader carefully! Make sure to give yourself enough time (2-4 hours) to work on the assignment. Plan to work before or during a time when you can seek help from your instructor's office hours and/or virtual mentoring hours on the course Discord server.	A solid start, but significantly more work is needed. Try to get started earlier. Make sure to allocate plenty of time. Review lecture materials and practice in-class activities without looking at the answers. Plan to work before or during a time when you can seek help from your instructor's office hours and/or virtual mentoring hours on the course Discord server.	You appear to be struggling with the fundamental concepts that the assignment is meant to demonstrate. Seek help from your instructor or course assistants during office hours. You should consider reviewing the material from this and previous units and practicing the class activities.	You submit to this as or the so you di was incor Try to g ea Make allocate ti Reviev materi practic activ Plan t before o time w can se from instructo hours virtual r hours course se

Midterm Exams

There will be three midterm exams throughout the semester, each of which is worth 10% of the final grade. Each exam will comprise two parts: a practicum (worth 70% of the grade) and a written exam (worth 30%). In general, these exams are not cumulative and will instead focus on the material covered since the previous exam. Students will be given the entire class period to complete both parts of the exam and may divide the time as they see fit.

The practicum is designed to assess whether or not you have sufficiently developed the expected skills up to that point in the semester. Each practicum will comprise several questions, each of which will be graded using the standard assignment grading rubric. Your grade will be the average of the grades on the individual problems.

Several of your homework assignments will be "mini-practica" that will help you to assess your readiness for each practicum. You should practice solving the problems on each mini-practicum within a set time, e.g. 15-30 minutes per problem, without consulting your notes.

The written exam is designed to assess your comprehension of conceptual topics that are difficult to test in a practicum. The questions on these exams will be similar to those on the daily quizzes. The written component is graded based on correct answers to the questions. The questions on the written portion of each exam will be similar to those on the daily quizzes, and so you can use the daily quizzes to assess your readiness for this part of the exam.

Final Exam

The final exam comprises two parts: a practicum and a written exam, each of which is worth 50% of the grade. The final exam will be cumulative but will emphasize more recently covered topics that were not included in previous exams. Students will be given the entire exam period to complete both parts and may divide the time as they see fit.

The practicum is designed to assess whether or not you have sufficiently developed the expected skills up to that point in the semester. Each practicum will comprise several questions, each of which will be graded using the standard assignment grading rubric. Your grade will be the average of the grades on the individual problems.

The written exam is designed to assess your comprehension of conceptual topics that are difficult to test in a practicum. The questions on these exams will be similar to those on the daily quizzes. The written component is graded based on correct answers to the questions.

10% Final Grade Rule

One of the key learning outcomes for Software Development & Problem Solving is that each student develops a minimum baseline of programming and software engineering skills. A student that has not demonstrated a baseline level of skill in these areas should not receive a passing grade the course.

The midterm and final exams are your opportunity to demonstrate that you have mastered the learning objectives of the course. You will need to demonstrate a practical application of your skills with minimal outside assistance.

Because of this, your final grade in the course will be limited to no more than 10% higher than your exam average.

For example, a student with an exam average of 64.5% cannot receive a grade higher than 74.5% (even if their overall average including assignments and activities is higher).

Resources

1. Provided Materials (lecture notes, assignments, code examples, etc.).
2. Recommended Text Books:
    a. The Pragmatic Programmer
    b. Learning Python, 5th Edition
3. Online resources.
    a. Python Web Site
    b. Python Online Documentation
    c. Project Euler
    d. Git Documentation

Grading Scheme

Letter Grade	Percentage Range
A	>=92%
A-	>=89%
B+	>=86%
B	>=82%
B-	>=79%
C+	>=76%
C	>=72%
C-	>=69%
D	>=59%
F	<59%

Please note that, given the number of students taking the course in any given semester, a fair and equitable rounding policy has been built into the grading scheme. For example, in many courses a student earning a 69% would receive an D, but in this course the student would receive an C-. For this reason, the course policy states that the above grading scheme is strictly applied and no additional rounding of grades will be performed.

This course is the first in a year-long introductory sequence. The prerequisite for enrolling in the next course in the sequence is a minimum grade of C- in this course.

Tips for Success

- Spend ~30 minutes reviewing the lecture materials before starting the homework assignments. Practice the coding activities to see if you can solve them without looking at the answers. If you cannot solve one or more problems, ask for help!
- Read the assignment fully before asking for help! Make a note of instructions that raise questions for you. Visit your instructor during their office hours, or ask questions in the assignment channels or during virtual mentoring hours on the course Discord server.
- Do not try to solve a large problem all at once. Divide the problem into smaller parts and solve each one individually, building a solution incrementally.
- Refer to your team's problem solving solution! Take a picture at the end of class!
- If you're not sure how to do something, look for a similar example in the class activities.
- Understand that you have the tools that you need to solve the problem; the skills you are developing will help you recognize which tools can be used to solve which problems.

Academic Honesty

RIT's Academic Integrity Policy defines the basic forms of academic dishonesty (cheating, duplicate submission, and plagiarism) and explains the official RIT policy regarding academic dishonesty.

Apart from the problem solving activity during lecture, all work in this course is individual. All other quizzes, in-class activities, assignments, and practica are expected to be the result of individual effort, not teamwork or collaboration. Students are permitted to discuss class activities and assignments with other students provided that it is not at a level of detail that results in more than one student writing identical solutions. In other words, high level discussion of topics, approaches, or algorithms with other students are acceptable, but low level implementation details (such as line-by-line descriptions of a solution) are prohibited.

Submitting work that was authored all-or-in-part by another individual is considered a violation of academic honesty. Conversely, providing all-or-part of a solution to another student is also a violation of academic honesty. If a student is suspected of cheating, the instructor shall notify the students involved and conduct an investigation in accordance with RIT's Academic Integrity Policy.

If the investigation conducted by the instructor determines that the student violated RIT's Academic Integrity Policy, the minimum penalty is a 0 for the assignment on which the student cheated. Multiple and/or egregious violations will result in a failing grade in the course and a note in the student's permanent record. Subsequent offenses (e.g. in later courses) will be treated as additional instances of academic honesty and may result in a failing grade in the course (even for a single offense) or expulsion from the program or university.

AI Code Generation

Using software tools to automatically generate all or part of the solution to an assignment in this course is a violation of the academic integrity policy. As stated above, the minimum penalty for the first violation is a 0 on the assignment. Any subsequent violations will result in an F in the course.

Using such a tool to generate all or part of the solution to an exam question is an egregious violation of the academic integrity policy. The minimum penalty for the first offense is an F in the course.