University of Rhode Island

Department of Computer Science and Statistics

CSC402, Programming Language Implementation, Fall 2024

Instructor:	Lutz Hamel
Office Location:	Tyler Hall 251
Email:	lutzhamel@uri.edu
Class Days/Time:	MWF 11-11:50am
Classroom:	Engineering Building Room 040
Prerequisites:	CSC301
Webpage:	https://lutzhamel.github.io/CSC402 or BrightSpace

Course Description

- Have you ever wondered how the syntax highlighter in Eclipse works?
- Have you ever wondered how your favorite programming language is implemented?
- How about Python and JavaScript?
- What is the difference between interpreting a programming language and translating/compiling it?
- What is the difference between an interpreter and a virtual machine?

If any of these questions interest you, then CSC402 is for you. We will spend the semester looking at programming language implementations: from syntax highlighters to code analyzers, from interpreters and virtual machines to compilers.

As part of the course we will construct interpreters and translators for domain specific languages such as calculator languages and command line languages for steering your favorite game character. The course will also include one large semester project of a language implementation project of your choosing. This could be a graphics language, a new programming language (think Ruby), a domain specific language such as PHP or a new command line shell interpreter for Unix/DOS.

Course Goals

The goal of the course is to give you a solid foundation with respect to programming language implementation that includes:

- grammar construction
- parsing techniques
- intermediate representations
- abstract syntax tree construction
- tree pattern matching techniques

We will study a number of different programming language implementation techniques including,

- compilers
- interpreters, and
- virtual machines.

These tools will enable you to add domain specific and general programming language implementations to your tool chest to solve difficult engineering problems.

Upon successful completion of this course, each student will be able to:

- Understand the difference between compilers and interpreters.
- Use grammar specification tools effectively.
- Design and implement domain specific and general purpose programming languages.

Required Texts/Readings

Textbook

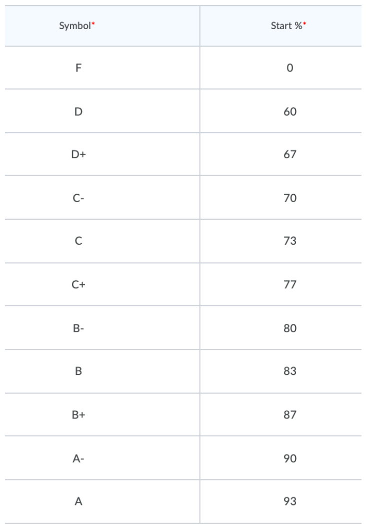
Programming Language Implementation: A Practical Introduction with Python. Lutz Hamel, Franklin & Beedle, 2024.

Exams, Assignments, and Grading Policy

Course Grade Composition:

Assignments, Quizzes	50%
Midterm	25%
Final	25%

Grading Key



Symbol*	Start %*
F	0
D	60
D+	67
C-	70
C	73
C+	77
B-	80
B	83
B+	87
A-	90
A	93

Homework consists of exercises to familiarize you with common tools and concepts in programming language implementation. Programming assignments are typically projects that can be completed within a couple of days. The midterm and the final comprise major projects and you should budget your time accordingly. Assignments are given on a weekly basis.

Classroom Protocol

- Check the website (often)! I will try to keep the website as up-to-date as possible.
- **Promptness, participation,** and **adequate preparation** for each class are expected. If you are absent, it is your responsibility to find out what you missed (e.g. handouts, announcements, assignments, new material, etc.)
- **Late assignments/project** will **not** be accepted without a valid excuse, such as illness etc. If you find that you are unable to submit an assignment/project please get in touch with me as soon as possible **before** the deadline expires.
- All work is to be the result of your own individual efforts unless explicitly stated otherwise. **Plagiarism, unauthorized cooperation or any form of cheating** will be handled according to the University Manual section 8.27.10 through 8.27.21 (see www.uri.edu/facsen/8.20-8.27.html). The penalty for cheating or plagiarism can range from a zero score on the assignment to a failing grade for the course. More details on programming assignments below.
- **Software piracy** will be dealt with exactly like stealing of university or departmental property. Any abuse of computer or software equipment will subject to disciplinary action.
- Any student with a documented disability should contact me early in the semester so that we can make reasonable accommodations to support your success in this course. You should also contact Disability Services for Students, Office of Student Life, 330 Memorial Union, 874-2098

Anti-Bias Statement: We respect the rights and dignity of each individual and group. We reject prejudice and intolerance, and we work to understand differences. We believe that equity and inclusion are critical components for campus community members to thrive. If you are a target or a witness of a bias incident, you are encouraged to submit a report to the URI Bias Response Team at www.uri.edu/brt. There you will also find people and resources to help.

***Academic Enhancement Center*:** Located in Roosevelt Hall, the AEC offers free face-to-face and web-based services to undergraduate students seeking academic support. Peer tutoring is available for STEM- related courses by appointment online and in-person. The Writing Center offers peer tutoring focused on supporting undergraduate writers at any stage of a writing assignment. The UCS160 course and academic skills consultations offer students strategies and activities aimed at improving their studying and test-taking skills. Complete details about each of these programs, up-to-date schedules, contact information and self-service study resources are all available on the AEC website, uri.edu/aec.

Programming Assignment Policy

Any programming assignment that is specified as **Individual Work** must be done on your own. While you may discuss general solutions, design ideas and algorithms with classmates, **YOU MAY NOT:**

- **share code with other students**
- **look at any other student's code**

If you use code that is not your own include in your comments where the code came from. If you ever have a question about what is acceptable when working on a programming assignment, please contact your instructor. **Your assignment may not consist entirely of third-party code, the majority of the code must be your own or based on templates provided in class.**

Generative AI tools like ChatGPT are very good at writing code. While these tools can be useful to programmers solving large problems, we want you to learn the fundamentals of programming. Therefore, you are expected to write all code on Individual Work by yourself or use templates provided by the instructor. You may use generative AI tools in a tutorial fashion where you ask general questions about coding and problem solving. But you **MAY NOT** use them to do your assignments. When doing work for this course, **follow these rules** if you are using generative AI tools:

- You may consult the AI tools on general questions
- You **MAY NOT** copy and paste your assignments or code into the tool
- You **MAY NOT** copy and paste any results from the tool into your assignments
- If you use an AI tool while working on an assignment, you **MUST** treat the results as code provided by a third party and document this in your comments providing the prompt that you used and the result of the prompt.
- If you use programming constructs that are not taught in class, you **MUST** cite where you found them and why you have chosen to use them instead of the constructs taught in class.

Any violation of these rules may result in a grade of 0 on the assignment. In addition, you may be reported to the Dean and the Office of Student Life. See the University Manual for more information about the potential consequences of cheating. https://web.uri.edu/manual/chapter-8/chapter-8-2/.

Tentative Course Schedule

1. Programming Languages and their Processors
2. Parsing and Lexing

3. Syntax-Directed Processing
4. Interpretation with Intermediate Representations
5. Tree Based Intermediate Representations
6. A Basic Compiler
7. Symbol Tables and Scope
8. Functions
9. Type Systems
10. Structured Data Types
11. Parser Generators