CS3360: Design/Implementation Programming Languages

Tuesday and Thursday from 3:00 pm to 4:20 pm (MST)

Spring 2023

Quinn Hall 206

Instructor: Dr. Saeid Tizpaz Niari (saeid@utep.edu)

IA: Alan Ochoa (aeochoa2@miners.utep.edu)

COURSE DESCRIPTION

CS 3360 is a required credit for a BS in computer science. The primary goal of CS 3360 is to study the design and implementation that underlay the programming languages. In previous courses, students have examined how to write programs in individual languages such as C, Java, or Python. In this class, we will take a broader view of programming languages, and study the key concepts and techniques that allow developers to implement languages such as Java or C.

Certainly, there are social factors and personal preferences to choose and use programming languages. But there is also a body of principles and mathematical theories that allow us to discuss and think about languages in a rigorous manner. We study these underpinnings because a language affects the way one approaches problems working in that language and affects the way one implements that language.

We will dissect programming languages by constructing interpreters. The semester project is to construct an interpreter for a toy language called Lettuce (a language based on a family of programming languages called ML). We will see that interpreters are the basis for realizing computation, and we will study the programming language theory that enable us to reason carefully about a language's design and implementation.

The course covers many aspects of using, understanding, and reasoning about programming languages (e.g., syntax, scoping, induction, data types, and typing). We will build up a set of mathematical tools for careful discourse. A significant part is devoted to abstraction, that is, how languages help programming in the large.

COURSE OBJECTIVES

Upon the completion of course:

- Students will be able to learn new languages quickly and select a suitable one for your task.
- Students will gain new ways of viewing computation and approaching algorithmic problems.
- Students will gain new ways of viewing programs.
- Students will gain insight into avoiding mistakes for when you design languages.

PREREQUISITES

The prerequisite of this course is CS 2302 with a grade of C or better. It is also recommended to take CS 3331 (Advanced Object-Oriented Programming) and CS 3432 (Computer Architecture I) before you take this course.

Course Outlines

- Basic Introduction to Scala and Functional Programming
- Recursion and Tail Recursive Programming
- Pattern Matching and Inductive Definition
- Map, Reduce (Fold), Filter
- Grammar, BNF form, and Abstract Syntax Tree
- Parser and EBNF form
- Grammar and Abstract Syntax for Arithmetic Expressions
- Big step semantics for Arithmetic Expression
- Intro to Lettuce, a functional language with let bindings and Scoping.
- Describe/Implement the syntax of Lettuce: Arithmetic Expression, Let binding, and Conditions
- Define and implement semantics rule for basic Lettuce
- Lettuce with functions and function calls (Closure)
- Lettuce with recursive functions (Inductive Environment)
- Calling Conventions, Currying, and Continuation Passing Style (CPS)
- Lettuce with side effects (imperative programming paradigm)
- Lettuce with Explicit and Implicit References (Store and reference variables)
- Lettuce with Type: mechanisms for type checking in Lettuce
- Lettuce with Type inferences
- Introduction to Probabilistic Programming Languages (PyMC3)
- Introduction to Logic Programming Languages (Prolog)

REQUIRED MATERIALS

The primary reading for the course is the Jupyter notebooks that will be distributed each week. Students are required to install Jupyter notebook with Scala kernel (First, install Jupyter and then install Almond that provides Scala kernel supports for Jupyter and available for Mac, Linux, and Windows). Please find more information on installation in the ``install`` folder in Blackboard. The course follows many ideas from supplemental texts such as Robert W. Sebesta, *Concepts of Programming Languages*, 12th edition, Addison-Wesley, 2018 and *Essentials of Programming Languages (Third Edition)* by Friedman & Wand. **These books are recommended but are not required.** We also strongly recommend (but not required) that you get access to Programming in Scala, third edition, by Martin Odersky, Lex Spoon, and Bill Venners. This book is an extended tutorial for learning Scala by those directly involved in the language's development.

COURSE ASSIGNMENTS AND GRADING

This class will have pre-recorded lectures (~ 10 mins) sent to students at the beginning of week. Students are required to go through the videos and corresponding readings. The class time will devote for discussions, clarifying issues, answering questions, and practicing problem sets. Please bring your laptop to the class to participate in different activities and write code! The course work is divided into four parts: (a) weekly assignments (online quiz and problem set); (b) regularly scheduled spot exams (c) mini projects that consist of a programming assignment using principles learned in this class; and (d) the final exam.

Category	Percentage
Weekly Assignments	40%
Online Quiz	10%
Mini Projects	10%
Spot Exams	20%
Final Exam	20%
Class/group activity	5% (extra points)

Weekly Online Quiz (10% of the grade)

You will have two attempts for each quiz. You can also check your answers for each question, but a wrong answer will involve a small penalty to your grade. The online quizzes are due to 3:00 pm on Tuesday (right before the class). ***The lowest grade for the weekly online quiz will be dropped. Therefore, no late excuses will be granted.***

Weekly Problem Sets (40% of the grade)

Each week you will receive a problem set that will include concept-based problems and programming assignments in Scala. We project that there will be **8 weekly assignment**. They are due to 5:00 pm on Friday. These assignments will be posted as jupyter notebooks that need to be filled out by the students. ***The lowest grade for the assignments will be dropped. Therefore, no late excuses will be granted.***

Mini Projects (10% of the grade)

We will have **two mini projects** that will involve building interesting applications using the skills you have learned in class. One of the mini-projects will implement a language from scratch. Another mini project will explore building a domain specific language. ***No late submissions will be accepted (pay attention to the deadlines for mini-projects).***

Spot Exams (20 % of the grade)

We will have **three spot exams** that will be conducted during your recitations. These exams will last roughly 40 minutes and consist of material covered in class or through your assignments. ***The grade for the lowest spot exam will be dropped. Therefore, no excuses will be considered for missing these exams.***

Final Exam (20 % of the grade)

A final exam will account for 20% of the grade. This will be at a time announced by the University registrar. *The final is compulsory for all students desiring a passing grade in this class.*

Extra Bonus (5 % of the grade)

A bonus of up to 5% is available for engaging and participating in class and online board discussions. Especially, students who are not able to participate in a class discussion are encouraged to post their questions and participate in the class discussions offline. A full bonus requires exceptional participation.

Your grade is independent of anyone else's grade in this class; that is, we do not grade on a curve. Everyone can get an *A* in this class. Our purpose in grading is to uphold a standard of quality and to give you feedback, it is not to rank students.

Percentage	Grade
P >= 90%	A
80% <= P < 90%	B
70% <= P < 80%	C
60% <= P < 70%	D
P < 60%	F

The instructor reserves the right to adjust these criteria downward, e.g., so that 88% or higher represents an A, based on overall class performance. The criteria will not be adjusted the other way around.

ATTENDANCE POLICY

Based on the health situations, the instructor encourages students to **wear masks** and practice in **social distance** as far as possible in class. The class attendance and discussion participations have up to 5% bonus in the final grade. The instructor is open to adjusting all things related to the class, if there is a good reason and justification.

COVID PRECAUTIONS

Please stay home if you have been diagnosed with COVID-19, are experiencing COVID-19 symptoms, or feel comfortable to stay home. If you are feeling sick, please let me know as soon as possible, so that we can work on appropriate accommodations. If you have tested positive for COVID-19, you are encouraged to report your results to covidaction@utep.edu, so that the Dean of Students Office can provide you with support and help with communication with your professors. The Student Health Center is equipped to provide COVID 19 testing.

The Center for Disease Control and Prevention recommends that people in areas of substantial or high COVID-19 transmission wear face masks when indoors in groups of people. The best way that Miners can take care of Miners is to get the vaccine. If you still need the vaccine, it is widely available in the El Paso area, and will be available at no charge on campus during the first week of classes. For more information about the current rates, testing, and vaccinations, please visit epstrong.org

EXCUSED ABSENCES AND/OR COURSE DROP POLICY

I will not drop you from the course. However, if you feel that you are unable to complete the course successfully, please let me know and then contact the Registrar's Office to initiate the drop process. Otherwise, you are at risk of receiving an "F" for the course.

EXPECTED LEARNING OUTCOMES

Level 1: Knowledge and Comprehension:

Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions.

Upon successful completion of this course, students will:

a. Describe broad trends in the history of development of programming languages.
b. Explain the stages of programming language interpretation and compilation.
c. Understand data and control abstractions of programming languages.
d. Understand how attribute grammars describe static semantics.
e. Describe ways to formally specify the dynamic semantics of small subsets of programming languages, such as expressions and control structures.
f. Understand code snippets written in a paradigm beyond imperative, object-oriented, and functional, e.g., algebraic, aspect-oriented, logic, or probabilistic languages.

Level 2: Application and Analysis:

Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.

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

a. Define syntax of a small context-free grammar in BNF.
b. Define the syntax of a small subset of a programming language using BNF.
c. Compare different approaches to naming, storage bindings, typing, scope, and data types.
d. Analyze design dimensions of subprograms, including parameter passing methods, subprograms as parameters, and overload subprograms.
e. Be able to write programs to solve simple problems in a purely functional language.
f. Be able to write programs to solve simple problems in a scripting language.

Level 3: Synthesis and Evaluation

Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery.

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

a. Evaluate modern, representative programming languages critically considering design concepts, design alternatives, and implementation issues for variables, types, expressions, control structures, and program modules.
b. Choose a suitable programming paradigm and language for a given problem or domain.

TECHNOLOGY REQUIREMENTS

Ensure your UTEP e-mail account is working and that you have access to the Web and a stable web browser. Mozilla Firefox and Google Chrome are the most supported browsers for Blackboard; other browsers may cause complications with the LMS. If you encounter technical difficulties beyond your scope of troubleshooting, please contact the Help Desk as they are trained specifically in assisting with technological needs of students.

STANDARDS of CONDUCT

You are expected to conduct yourself in a professional and courteous manner, as prescribed by the Handbook of Operating Procedures: Student Conduct and Discipline. All graded work (homework, projects, exams) is to be completed independently and should be unmistakably your own work, although you may discuss your work with others in a general way. You may not represent as your own work material that is transcribed or copied from another source, including persons, books, or Web pages. **Plagiarism is a serious violation of university policy and will not be tolerated**. All cases of suspected plagiarism will be reported to the Dean of Students for further review. You are welcome and encouraged to work together in learning the material. However, whatever you submit must be your own. In other words, cutting and pasting or copying verbatim from another source be it a classmate, an online source or even something that the TA/instructor showed you is strictly forbidden.

- *Cite Your Sources:* If you worked with someone on an assignment, or if your submission includes quotes from a book, a paper, or a web site, you should clearly acknowledge the source. **Bottom line**: feel free to use resources that are available to you as long as the use is reasonable, and you cite them in your submission. However, copying answers directly or indirectly from solution manuals, web pages, or your peers is certainly forbidden.
- *Inspiration is free:* you may discuss homework assignments with anyone. You are especially encouraged to discuss in black board with your instructor and your classmates.
- *Plagiarism is forbidden:* the assignments and code that you turn in should be written entirely on your own. You should not need to consult sources beyond your textbook, class notes, posted lecture slides and notebooks, programming language documentation, and online sources for basic techniques. Copying/soliciting a solution to a problem from the internet or another classmate constitutes a violation of the course's collaboration policy and the honor code and will result in an F in the course and a trip to the honor council.
- *Do not search for a solution online:* You may not actively search for a solution to the problem from the internet. This includes posting to sources like StackExchange, Reddit, Chegg, etc.
- *StackExchange Clarification:* Searching for basic techniques in Python/Pandas/Numpy is totally fine. If you want to post and ask "How do I group by two columns, then do something, then group by a third column" that's fine. What you cannot do is post "Here's the problem my professor gave me. I need to convert Age in Earth years to Martian years and then predict the person's favorite color. Give me code!" That's cheating.
- *When in doubt, ask:* We have tried to lay down some rules and the spirit of the collaboration policy above. However, we cannot be comprehensive. If you have doubts about this policy or would like to discuss specific cases, please ask the instructor. *If **it** has not been described above, you should discuss **it** with us first*

Please also pay attentions to the following netiquettes:

- Always consider audience. Remember that members of the class and the instructor will be reading any postings.
- Respect and courtesy must be provided to classmates and to instructor at all times. No harassment or inappropriate postings will be tolerated.
- Blackboard is not a public internet venue; all postings to it should be considered private and confidential. Whatever is posted on in these online spaces is intended for classmates and professor only. Please do not copy documents and paste them to a publicly accessible website, blog, or other space. If students wish to do so, they have the ethical obligation to first request the permission of the writer(s).

ACCOMMODATIONS POLICY

The University is committed to providing reasonable accommodations and auxiliary services to students, staff, faculty, job applicants, applicants for admissions, and other beneficiaries of University programs, services and activities with documented disabilities in order to provide them with equal opportunities to participate in programs, services, and activities in compliance with sections 503 and 504 of the Rehabilitation Act of 1973, as amended, and the Americans with Disabilities Act (ADA) of 1990 and the Americans with Disabilities Act Amendments Act (ADAAA) of 2008. Reasonable accommodations will be made unless it is determined that doing so would cause undue hardship on the University. Students requesting an accommodation based on a disability must register with the UTEP Center for Accommodations and Support Services.

ACKNOWLEDGEMENTS

This course is adapted from Prof. Sriram Sankaranarayanan's course on Principles of Programming Languages at University of Colorado Boulder. This course is also inspired from Prof. Bor-Yuh Evan Chang fantastic Programming Languages course. Special thanks to Sriram and Evan for their advice and assistance in the preparation of this course.