# Syllabus

## Course Description

This is the honors add on to CS125. This course is designed for students of any skill level. The goal of the course is to help students become self-sufficient programmers. To do this, we will teach some of the necessary skills to be a programmer today, which is to be proficient in bash, git, and "googling". Furthermore, a good programmer is one who knows what tools and languages are best fit for a certain problem. To do this, students will do an intermediate dive into Haskell and Rust to learn programming paradigms that are different from traditional languages such as Java, C++, or Python. Most importantly, students will work with a project manager and a group of peers throughout the semester on a project. This project will require students to learn how to manage tasks via scrum board, communicate with their team, and use git as a team properly.

## Information

### Instructors

| Name | Email |
| --- | --- |
| Joshua Dunigan | joshuad2@illinois.edu |
| Prithvi Ramanathan | pramana2@illinois.edu |

### Lectures

Tuesdays and Thursdays at 7:00PM in 1404 Siebel Center.

## Structure

### Overview

The course has two parts. There will be two 1 hour lectures each week. Some lectures will be more of a lab while others willbe more of a discussion. Before each lecture, there will be a short prelecture reading with an activity. The lectures will be divided into three sections, for developing, Rust, and Haskell. The first section is to teach students how to use the tools at their disposal as a programmer. Rust and Haskell are meant to teach students ways of looking at and solving problems that are different from traditional general purpose languages.

### Prerequisites

None

### Time Commitment

The lower and upper bounds of time spent in this course will be 4-7 hours for students, varying on experience

beforehand. The goal of this course is to teach students not only the concepts mentioned, but to give students with no experience a way to catch up, and students with a lot of experience to keep going higher. The course is designed to give students a minimum amount of coursework (4-7 hours), but is also designed such that students can sink many hours a week if they want to.

## Projects

Students will meet with their team and PM for a standup once a week, and will be required to do their assigned weekly task(s). By week 4, students should be committing code to github and will be graded accorrdingly.

Projects will go as follows - Meet with team and PM. - Come up with and submit a plan for what the project's MVP(minimum viable product) would look like - The plan will be accepted, request changes, or denied. The reason a project needs changes or is denied is to make sure projects are going to be educational, worthwhile, and a long enough project for 13 weeks of coding. - Start working on project

## Lecture Schedule

| Lecture | Date | Topic | Homework |
|---|---|---|---|
| 1 | August 27 | Intro lecture | NA |
| 2 | August 29 | Bash | Bash 1 |
| 3 | September 3 | Bash | Bash 2 |
| 4 | September 5 | Project Kickoff | NA |
| 5 | September 10 | Git | Git 1 |
| 6 | September 12 | Git | Git 2 |
| 7 | September 17 | Git | Git 3 |
| 8 | September 19 | How to Google | Google Scavenger Hunt |
| 9 | September 24 | Rust - I | Rust 1a |
| 10 | September 26 | Rust - I | Rust 1b |
| 11 | October 1 | Rust - I | Rust 1c |
| 12 | October 3 | Rust - I | Rust 1d |
| 13 | October 8 | Rust - II | Rust 2a |
| 14 | October 10 | Rust - II | Rust 2b |
| 15 | October 15 | Rust - II | Rust 2c |
| 16 | October 17 | Rust - II | Rust 2d |

| Lecture | Date | Topic | Homework |
|---|---|---|---|
| 17 | October 22 | Rust - II | Rust 2e |
| 18 | October 24 | Midterm Presentation | NA |
| 19 | October 29 | Open Source | Req: Open Issue, EC : Merge a PR |
| 20 | October 31 | Haskell - I | Haskell 1a |
| 21 | November 5 | Haskell - I | Haskell 1b |
| 22 | November 7 | Haskell - I | Haskell 1c |
| 23 | November 12 | Haskell - I | Haskell 1d |
| 24 | November 14 | Haskell - II | Haskell 2a |
| 25 | November 19 | Haskell - II | Haskell 2b |
| 26 | November 21 | Haskell - II | Haskell 2c |
| 27 | December 3 | Haskell - II | Haskell 2d |
| 28 | December 5 | Haskell - II | Haskell 2e |
| 29 | December 10 | Deploying Code | Deploy Your Code |
| 30 | December 12 | Final Presentations | NA |

### Homework

THe Rust 1/2 and Haskell 1/2 are multi-week MPs. We will release the solutions to each part (1a, 1b, ..., etc) so that if a student does not complete a part, they are not penalized disproportionately. The open source homework has an extra credit part, if students show that they kept working on the repo they selected (or another, but make clear of this) till the end of the semester, 1% extra credit. If they make a pull request to any large enough repo, they will get 3% extra credit in the course.

## Grades

### Projects

Projects are worth 65% of students overall grade. Students will get a grade every two weeks after project kickoff to show them if they are using github, their scrumb board, and communicating effectively. There will be a midterm grade which will grade on how close students are to halfway with their MVP, and a final presentation for how close overall. The reason we enforce proper github usage, scrum board usage, and team communication is so that we have a history of individual contribution and a paper trail to show that if a group does not get the MVP done, there was good reasons why. Students must use pull requests to get points for github usage.

### Lectures and Homework

There are 3 parts to students grades for lecture. There will be a short prelecture activity that shows comprehension of the prelecture notes. There will be attendance for lecture. There will be a grade for completing the in-lecture activities.

For the homework listed above, there will be regular due dates for when to complete them. Students will have an autograder that gives instant feedback on most assignements with unlimited runs.

### Drops and Curve

The goal of this course is not to stress students about grades, but give students an environment to learn as much as possible. For lectures, pre lecture, and attendance, there will be 4 drops each. However, each drop can only be for one day, meaning if a student is dropping a prelecture, we will drop the corresponding lecture activity, attendance. We will not drop any homeworks, but we may curve the homeworks. Similarly, we may curve project scores as well.

### Breakdown

| Grade | Points |
|---|---|
| Prelecture Activity | 5 |
| Lecture Attendance | 5 |
| Lecture Activity | 5 |
| Homework | 30 |
| Project Checkmarks | 25 |
| Midterm Presentation | 10 |
| Final Presentation | 30 |
| Total | 100 |

## Academic Integrity

We reserve the right to use code plagiarizer detection tools on any code. The first offense will be a -20% on the assignment, any subsequent offences will increase by another 20%, and so one. For example, if a student was to get caught cheating 5 times, they will get a -100% on an assignment, meaning it would cancel out an entire homework they did do as well. This is to deter students from copying and instead just not do the code at all. If students use online sources, cite the sources. We encourage googling (re: How to Google lecture), but not just giving up and having someone do their homework for them. We also reserve the right to change the punishment, either to increase or decrease it as we see fit.