BST236: Computing I

Junwei Lu

Spring 2025

Chapter 0: Course Introduction and Syllabus

Course Logistics

- **Schedule**: Tuesday and Thursday 08:00 AM - 09:30 AM
- **Location**: Kresge 502
- **Instructor**: Junwei Lu
  - Email: junweilu@hsph.harvard.edu
  - Office: Building 2 Room 409
  - Office Hours: Tuesday 9:30 AM - 10:30 AM
- **Teaching Assistant**: Phillip Nicol
  - Email: phillipnicol@g.harvard.edu
  - Office Hours: Building 2 Room 434, Thursday 2:00 PM - 3:00 PM
  - Lab: FXB G03, Friday 9:45 AM - 11:15 AM

Course Overview

- A beginner-friendly course on (statistical) computing
- Emphasis on breadth over depth
- Emphasis on AI assisted coding
- Prerequisites:
  - Basic Python and R programming
  - Probability, Statistics, Linear Algebra
- Optional skills:
  - Markdown
  - LaTeX
- Coding is a life-long learning process

Course Structure

- Meta-Coding
  - Good coding principles: best practices
  - Workflow for reproducibility and collaboration
  - Code with AI
- Data Structures
  - How to manipulate data in Python
  - How to manipulate really big data
- Algorithms
  - Tree traversal, Dynamic programming
- Linear Algebra: Linear equations and eigenvectors
- Optimization
- AI Models: Neural networks and Language models
  - PyTorch

Python vs R

- Most of the course will be in Python
- You can do homework in R if you prefer
- English is the only best coding language in 21st century

Happy Lunar New Year of Snake!

Learning Tips

- Practice implementation
- Customize workflows
- Seek community help
- Explore beyond course content
- Skip adaptively
- Abstract blackboxes: modular programming
- Tip: Problem decomposition is more important than knowing how to implement a function

Course Evaluation

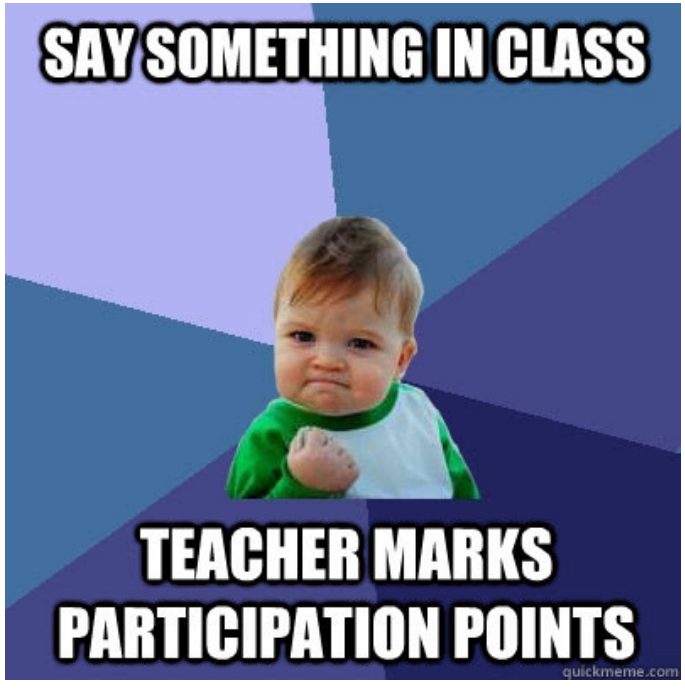
- Participation: 20%
- Assignments: 40%
- Midterm Project: 20%
- Final Project: 20%

Class Participation

No attendance needed. Sleep is more important.

Class Participation

- Classroom discussions and questions
- Contributing to course website repository
- Participating in GitHub Discussions
- One-minute summaries after each class
  - Due at midnight
  - Submit via Canvas
  - Brief content and confusion summary



Homework

- Weekly homework assignments
- Due at 11:59 PM every Thursday
- Submit via GitHub Classroom

Late Assignment Policy

- 3 free late days per squad for the semester
- Late days measured in hours
- Can buy additional late days with `byte` currency
- After late days are used:
  - 25% penalty per day
  - Maximum 5 days late each assignment
  - Zero credit after 5 days

Code Squad

- Teams of 3-4 students
- Why squad?
  - Learning to code alone is frustrating.
  - Learning from each other is more efficient.
  - Learning how to work in a team is important.
- Individual for first two assignments, squad for later assignments

Bit Battle

- Olympics for coding
- Prize: Course currency `byte`
- Buy late days with `byte`

Bit Battle

- Rank code by running time, accuracy, etc, which will be specified each time
- Course Currency: `byte`
  - Initial deposit: B100 bytes per squad
  - Can buy late days (B5 = 1 hour)
- Battle Rewards:
  - 1st Place: B100 bytes
  - 2nd Place: B80 bytes
  - 3rd Place: B60 bytes
  - 4th Place: B40 bytes
  - 5th Place: B20 bytes
  - 6th Place: B10 bytes
  - 7th Place: B5 bytes
- Auction for the squad of 4 students.

AI Tools Policy

- ChatGPT and other AI tools are encouraged
- Use AI as a learning tool
- Understanding is essential
- No detection/checking of AI usage
- Focus on learning and comprehension

Checklist

- Course website: Lecture notes
- Get a GitHub account
- Course GitHub repository: Discussions, Homework issues, Course code
- Installation
- Canvas: One-minute summary, Zoom recordings

Chapter 1: Coding Principles

Coding Principles

- Don't Repeat Yourself (DRY)
  - Make your workflow automated, transferable, and reproducible
  - Write reusable code to handle different cases/hyperparameters
- Documentation
  - Document functions
  - Document different implementations

Documentation Best Practices

- Document before coding
- Include:
  - Project goals
  - Function decomposition
  - Input/output specifications
  - Implementation approaches
- Use documentation for:
  - Clear thinking
  - AI prompt engineering
  - Team communication

Premature optimization is the root of all evil

1. Make It Work
   - Handle common cases first
   - Get basic functionality running
2. Make It Right
   - Fix special cases
   - Add error handling
3. Make It Fast
   - Optimize critical sections