4143 Programming Language Concepts

General Course Info

- **Days:** TTh 9:30 q.m. - 10:50 a.m.
- **Location:** BO 320
- **Semester:** Monday August 28nd - Friday December 8th
- **Holidays:**
  - **Labor Day** Monday September 4th
  - **Thanksgiving** Wednesday November 22nd - Sunday November 26th
- **Last Day for "W":** Monday October 30th
- **Last Day of Class:** Friday December 8th
- **Final Exam:** Tuesday December 12th from 8:00 pm - 10:30 pm

Broad Topics

- □ **Names, Binding, and Scope (Declarations)**
  - How do we give names to entities? And when we encounter a name, how do we know which entity it refers to?
- □ **Evaluation (Expressions)**
  - How do we express computations, using values and operators?
- □ **Execution (Control Flow)**
  - How to we organize computation in time? What actions or effects can we produce?
- □ **Types**
  - How do we classify values so that they may behave in certain, predictable ways?
- □ **Functional Abstraction (Subroutines and Coroutines)**
  - How can we abstract computations into chunks so that we can invoke them whenever we need them?
- □ **Data Abstraction (Objects and Modules)**
  - How do we make little bundles of data together with behavior?
- □ **Concurrency**
  - How do we arrange to do different computations at the "same" time (safely)?
- □ **Metaprogramming**
  - How can our programs know about themselves? How can we answer questions about the code itself?

Semester Schedule

Weeks	Description
1-2	**Introduction and Names, Binding, and Scope**
□	Introduction to programming languages and their importance
□	Syntax vs semantics

Weeks	Description
□	Compilation vs interpretation
□	Names, identifiers, and keywords
□	Binding time: static, dynamic, early, and late binding
□	Scope: lexical vs dynamic scoping
□	Nested scopes and scope rules
□	Static and dynamic scoping examples
3-4	**Evaluation (Expressions) and Execution (Control Flow)**
□	Expressions and their evaluation
□	Precedence and associativity
□	Order of evaluation
□	Short-circuiting and its effects
□	Control structures: sequencing, selection, iteration
□	Conditionals: if, if-else, switch
□	Loops: while, for, foreach
□	Control flow pitfalls and examples
5-6	**Types**
□	Data types and their significance
□	Static typing vs dynamic typing
□	Strong typing vs weak typing
□	Type checking and type inference
□	Primitive types: integers, floating-point, booleans, characters
□	Composite types: arrays, records, tuples
□	Type compatibility and type coercion
7-8	**Functional Abstraction (Subroutines and Coroutines)**
□	Introduction to subroutines and functions
□	Function declaration, parameters, and return values
□	Call stack and activation records
□	Recursion and tail recursion
□	Higher-order functions and function composition
□	Introduction to coroutines and cooperative multitasking

Weeks	Description
□	Coroutines vs threads
□	Coroutine synchronization and communication
9-10	**Data Abstraction (Objects and Modules)**
□	Introduction to data abstraction
□	Object-oriented programming principles
□	Classes, objects, methods, and attributes
□	Inheritance and polymorphism
□	Encapsulation and information hiding
□	Introduction to modules and modularity
□	Module interfaces and implementations
□	Packaging, namespaces, and access control
11-12	**Concurrency**
□	Introduction to concurrency and parallelism
□	Threads vs processes
□	Thread synchronization and coordination
□	Race conditions and critical sections
□	Mutual exclusion and semaphores
□	Deadlocks and livelocks
□	Parallel programming models
□	Concurrent programming pitfalls
13-14	**Metaprogramming**
□	Introduction to metaprogramming
□	Macros and code generation
□	Reflection and introspection
□	Compile-time vs runtime metaprogramming
□	Template-based metaprogramming
□	Aspect-oriented programming
□	Language-integrated query
□	Examples of metaprogramming in various languages
15	**Review and Future Trend**

Weeks	Description
□	Recap of key concepts from the course
□	Discuss emerging programming language trends
□	Domain-specific languages (DSLs)
□	Metaprogramming and code generation advancements
□	Language support for parallelism and distributed systems
□	Language design challenges and opportunities

Adventures in GoLang

- □ Introduction to Go
    - □ Introduction to Go programming language
    - □ Setting up the Go development environment
    - □ Go syntax and basic program structure
    - □ Variables, data types, and basic operations
    - □ Control flow statements (if/else, loops)
    - □ Functions and packages in Go
- □ Data Types and Structures
    - □ Complex data types (arrays, slices, maps, structs)
    - □ Working with strings and characters
    - □ Pointers and memory management
    - □ Error handling in Go
    - □ File I/O operations
- □ Concurrency and Goroutines
    - □ Understanding concurrency in Go
    - □ Goroutines and channels
    - □ Synchronization and data sharing
    - □ Patterns for concurrent programming
    - □ Error handling in concurrent programs
- □ Object-Oriented Programming in Go
    - □ Structs and methods in Go
    - □ Encapsulation and data hiding
    - □ Inheritance and composition
    - □ Polymorphism and interfaces
    - □ Object-oriented design principles in Go
- □ Error Handling and Testing
    - □ Error handling best practices in Go
    - □ Panic and recover mechanisms

  - □ Unit testing in Go (using the testing package)
  - □ Writing testable code in Go
  - □ Code coverage and test automation

- □ Advanced Topics in Go
  - □ Reflection and runtime type information
  - □ Concurrency patterns (e.g., worker pools, fan-out/fan-in)
  - □ Memory optimization and profiling
  - □ Benchmarking and performance tuning
  - □ Working with external libraries and APIs

- □ Web Development with Go
  - □ Overview of web development in Go
  - □ HTTP server programming in Go
  - □ Routing and handling requests
  - □ Middleware and authentication
  - □ Introduction to web frameworks (e.g., Gin, Echo)

- □ Final Projects and Wrap-up
  - □ Student final projects: Implement a significant Go application
  - □ Project presentations and feedback
  - □ Recap and review of course concepts
  - □ Q&A session and open discussion

Grading

Will be based on number of problems solved. Full credit for on time solutions. Half credit for late submissions. Some credit for accepted solutions with issues (e.g. presentation errors),

Categories	Portion of Course	:::	Letter Grade	Grade Range
Exams	45%	:::	A	90-100
Github	10%	:::	B	80-89
Participation	5%	:::	C	70-79
Presentation	10%	:::	D	60-69
Project	10%	:::	F	below 60
Final	20%	:::		

:::

> **Participation**: Obviously this has to do with going to class, asking questions, and generally being a physical part of class. But even more importantly it has to do with interacting with our class on Slack. Responding to queries on Slack either with text response or an emoticon reaction to a post. Asking me questions with direct message is a huge help, as I can turn that into (what the military calls) an "overhead correction". Most questions help bring to my attention things that need

> clarification for everyone. I will be gauging everyone's participation and the best guarantee is on slack as that creates a digital record.

Miscellaneous

- All students need a Github account
- All programs need to be turned in to pass the course
- General Assignment Rules:
  - Due dates and times are as listed on assignment and can change with prior notice to class.
  - Formatting of programs is important, and will be graded accordingly.
  - You name is required on ALL documents uploaded or turned in. Handwritten name is not acceptable.
  - All files / programs created by you will end up in your assignments folder within your Github repository.
- Attending class is one of the primary keys to doing well in this class. Students may be dropped for excessive absences. There is no distinction made between excused and unexcused.
- Make-up exams are not given. If I see fit, then I will replace a missed exam with your final exam test grade (but this is optional to instructor based on circumstances, attendance, participation, etc.).
- Late work will be accepted on a case by case basis. Late penalty is 15 points (out of 100) for initial lateness and 1 half a letter grade (5 points) for every class period until the total reduced is 50 (half credit). Extremely late work is totally at the instructors discretion on whether it will be accepted or not.
- Programs containing syntax errors are unacceptable and will be returned without grading (your programs must work).
- Periodically homework assignments will be taken up and graded. It is the student's responsibility to keep up with assignments and to ask questions over the assigned work, even if absent. All homework assignments are due at the specified time that may or may not be in conjunction with a class day. All assignments / homeworks will be uploaded via Github.

My View on Cheating / Plagiarism

- Most plagiarizing, when it comes to programming, happens for two reasons:

  - 1. You don't have a clue how to solve the problem, so you get a friend or the internet to help.
  - 2. You didn't start early enough, and you're desperate to get something working the night before it's due, so you get a friend or the internet to help.

- Both are easy to fix.

  - 1. Come ask me to explain. I promise you're not the only one who is confused.
  - 2. Start early. Then when you get stuck, you can ask for help the right way!

- Please read this article as it pertains to our field of computer science: How To Code Without Plagiarizing

- Also, this might help: plagiarize and get an "F".

  - https://msutexas.edu/student-life/conduct/

    - https://cs.msutexas.edu/documents/CMPS_Cheating_Policy.pdf

- Ultimately it's not cool. It's an insult to those that actually do the work.

- Lastly: Let me reiterate that I'm available for help ... **a lot**. No excuses. I've helped students at 1am via Slack. I'm online consistently afternoons and most nights, just shoot me a message.

Official Policy on Academic Honesty

The Department of Computer Science had adopted the following policy related to cheating (academic misconduct). The policy will be applied to all instances of cheating on assignments and exams as determined by the instructor of the course. (See below for link to MSU definitions.)

- 1st instance of cheating in a course: The student will be assigned a non-replaceable grade of zero for the assignment, project or exam. If the resulting grade does not result in a letter grade reduction, the student will receive a one letter grade reduction in course.
- 2nd instance of cheating in a course: The student will receive a grade of F in course & immediately be removed from course.
- All instances of cheating will be reported to the Department Chair and, in the case of graduate students, to the Department Graduate Coordinator.

Official Policy on Testing Process

The Department of Computer Science has adopted the following policy related to testing.

- All bags, purses, electronics (turned off), books, etc. will be placed in the front of the room during exams, or in an area designated by the instructor.
- Unless otherwise announced by the instructor, nothing is allowed on the desk but pen/pencil/eraser and test papers.
- No student is allowed to leave the room during an exam and return.

> See Also: MSU Student Handbook: Appendix E: Academic Misconduct Policy & Procedures.

Major Points

- All students need a Github account.
- All course communication will be done via Slack or Discord.
- All programs need to be turned in to pass the course.
- Programs containing syntax errors are unacceptable and will be returned without grading (your programs must work).
- Your name is required on ALL documents uploaded or turned in. Handwriting on any document is not acceptable (unless specified).
- All files / programs created by you will end up in your assignments folder within your Github repository.
- Attending class is one of the primary keys to doing well in this class. Students may be dropped for excessive absences. There is no distinction made between excused and unexcused.
- Make-up exams are not given. If I see fit, then I will replace a missed exam with your final exam test grade (but this is optional to instructor based on circumstances, attendance, participation, etc.).
- Cheating or plagiarism on any assignment will not be tolerated.