

# COL106 - Data Structures and Algorithms

Sem II - 2024-25

Happy new year!

# Who are we ?

Srikanta Bedathur &



Nikhil Balaji



Along with a number of TAs

- TAs will be assigned to batches - the listing will be up by next week on the course website

# What is this course?

Introduces to the idea of building programmatic solutions to problems

- Use of **appropriate** data **abstraction** and **data structures**
- **Algorithms** to solve problems – correctness and efficiency
- Converting algorithms to programs, running them as processes, functional testing and debugging

We **do not** plan on teaching you

- a) Java language – besides the basics
- b) How to use Chat GPT to solve programming problems



Post by Evening\_Action217 on r/ChatGPT on reddit

# Details of the course

Please look at <https://col106iitd.github.io>

We will use Java language for all programs

- Ensure JDK version > 11

All submissions: [moodlenew.iitd.ac.in](https://moodlenew.iitd.ac.in)

All discussions: [piazza.com](https://piazza.com)

All evaluations: [gradescope.com](https://gradescope.com)

**Reference textbook:** Datastructures and Algorithms in Java (by Goodrich & Tamassia)

The screenshot shows a course page titled "Data Structures and Algorithms". At the top right, there are three links: "Gradescope", "Piazza", and "Moodle". Red arrows point from the text in the left column to these links: from "All evaluations" to "Gradescope", from "All discussions" to "Piazza", and from "All submissions" to "Moodle". The page content includes:

- Instructors:** Nikhil Balaji and Srikanth Bedathur
- Lectures:** Tue, Thu & Fri 11:00 - 11:50 am
- Labs:** Mon, Tues, Thurs, Fri 3:00 - 5:00 pm
- Location:** TBA.
- Programming Language:** Java
- Course Description:** Here is a tentative list of topics that will be covered in the course:
  - Abstract data types
  - Basic Data-structures: Arrays, Stacks, Queues, Linked-lists
  - Dynamic Arrays, Asymptotic Complexity
  - Sorting: merge, quick, radix, heap
  - Dictionaries: Skip lists, Hashing
  - Trees, Tree traversal, Binary Search Tree
  - Priority Queues, Binary Heaps
  - AVL Trees, 2-4 trees, B-trees, Multiway search tree, and applications
  - Introduction to Graphs, Adjacency matrix and List representation
  - Breadth first search and applications
  - Depth first search in directed and undirected graphs and applications
  - Dijkstra's algorithm for shortest path, Minimum Spanning Tree
- Prerequisites:** COL 100
- Setting up Java**

**Java set up:** It is advised that you install JDK 11.0.17. You must ensure that your submission code runs in JDK 11.0.17 environment.

**Note:** You are expected to learn Java (and use object oriented programming) on your own. For this you should refer to the Java module below and the exercises in it (thanks to Prof. Amitabha Bagchi for making these available).

This site uses Just the Docs, a documentation theme for Jekyll.

# Details of the course

Please look at <https://col106iitd.github.io>

## Office hours (instructors)

- By appointment over email

Office hours of TAs will be posted on website

Seek help from TAs for

- Coding issues, understanding of specific questions
- DO NOT CALL THEM or HARASS THEM — contact only through Piazza, during Lab sessions, and during their office hours

The screenshot shows the 'Data Structures and Algorithms' course page. The left sidebar contains links for Announcements, Calendar, Lectures, Logistics, Resources, and Staff. The main content area includes the following information:

- Instructors:** Nikhil Balaji and Srikanth Bedathur
- Lectures:** Tue, Thu & Fri 11:00 - 11:50 am
- Labs:** Mon, Tues, Thurs, Fri 3:00 - 5:00 pm
- Location:** TBA.
- Programming Language:** Java
- Course Description:** Here is a tentative list of topics that will be covered in the course:
  - Abstract data types
  - Basic Data-structures: Arrays, Stacks, Queues, Linked-lists
  - Dynamic Arrays, Asymptotic Complexity
  - Sorting: merge, quick, radix, heap
  - Dictionaries: Skip-lists, Hashing
  - Trees, Tree Traversal, Binary Search Tree
  - Priority Queues, Binary Heaps
  - AVL Trees, 2-4 trees, B-trees, Multiway search tree, and applications
  - Introduction to Graphs, Adjacency matrix and List representation
  - Breadth first search and applications
  - Depth first search in directed and undirected graphs and applications
  - Dijkstra's algorithm for shortest path, Minimum Spanning Tree
- Prerequisites:** COL 100
- Setting up Java**

**Java set up:** It is advised that you install JDK 11.0.17. You must ensure that your submission code runs in JDK 11.0.17 environment.
- Note:** You are expected to learn Java (and use object oriented programming) on your own. For this you should refer to the Java module below and the exercises in it (thanks to Prof. Amitabha Bagchi for making these available).

This site uses Just the Docs, a documentation theme for Jekyll.

# Logistics

- Lectures are Tue, Thu, Fri – 11:00-12:00
- Lab sessions require mandatory attendance
- **Attendance based on predefined, immutable seating**
  - Next class (tomorrow) you should pick your “favorite” seat and stick with it for the rest of the semester
  - Seating plan will be shared on the course website **early next week.**
  - We will take photos of the class at random instances during the class. You are expected to be in your seat during **all these instances** to be considered to have been present.

**You must maintain 75% attendance in Lectures as well as in Labs.  
Otherwise,**

**you will be given a grade less, and not allowed to sit in the major exam.**

**If due to medical reason, you can not maintain the required attendance, you must withdraw from the course.**

## Marks distribution

3 out of 4 lab quizzes	3 x 10 marks	<b>30 marks</b>
3 out of 4 written quizzes	3 x 10 marks	<b>30 marks</b>
Midterm exam	1 x 15 marks	<b>15 marks</b>
Major exam	1 x 20 marks	<b>20 marks</b>
Attendance	3 marks for lectures and 2 marks for labs (linearly scaled)	<b>5 marks</b>

Marks in individual quizzes/exams will be linearly scaled to the given weightage

# Plagiarism Policy

- **Every programming quiz submission** goes through plagiarism check
- Written quizzes and exams will be proctored
- All quizzes and exams are to be completed **individually**
  - DO NOT show your work, DO NOT seek other's work
  - DO NOT borrow/copy/steal work from other students
  - Legitimate submissions are thought out, typed and tested by you alone
  - All submissions will be done through Moodle - VPL setup in the lab
- Note that impersonating someone else is considered cheating

## **Use of unfair means in quizzes and exams will be penalized**

- 1. First offense, you will be awarded the negative of the marks allocated to that component.**
- 2. Second offense, you will be awarded 'D' grade with no remajor/reminor options.**
- 3. Next offense, you will be asked to withdraw from the course, and your name will be reported.**



# Audit Criteria

- You will be awarded NP if all the following criteria are met:
  - At least “B-” grade equivalent of total marks.
  - At least 50% in lab quizzes OR in written quizzes.
  - At least 75% attendance in lectures and labs.
- In all other cases, you will earn NF grade

**No exceptions will be made**

# Programming Evaluations

- For each unit, we will release a set of programming practice questions
- These are **only for practice**. None of them need to be submitted.
- Model solutions for most questions will be released after one week.
- Some of these questions will be discussed during the lab session, and you are required to implement them, debug, and submit them (but they will not be graded)
- **4 Lab tests**
  - Based on the questions released
  - Will be conducted in labs in the LHC
  - Schedule is already available on the course website
  - No rescheduling due to any reason
  - We will consider the best 3 out of 4 lab tests

# Written Quizzes

- There are **4 written in-classroom quizzes**
- To be completed individually (plagiarism rules apply)
- The schedule of written quizzes is on the course website (click on “Weekly Schedule” on the left)
- How to prepare for written quizzes?
  - Please read the textbook, and solve the textbook questions
  - Attend the lectures and make sure you clear your doubts as lectures progress

Programs = Data Structures + Algorithms  
Processes are “Programs in Motion”

# Representations Matter!

Consider Numeral System

⋮   ·   :   ′′   ′   .   ′′   ′′   ′′   ·

0 1 2 3 4 5 6 7 8 9

○ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨

· Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ

○ 一 二 三 四 五 六 七 八 九

→ 零 壹 貳 参 肆 伍 陆 柒 捌 玖

— I II III IV V VI VII VIII IX

L M C

Revolutionary concepts in number representation

1. Introduction of zero  
(well before 7th century in India)

2. Positional notation of numbers  
(possibly 2000 BC in Babylonia)

Number and its position together determine the value

# From Individual to **Collections**

Programs rarely work with individual numbers, instead on collections of them.

A collection could be a

- **Set** : holds unique values, with no specific order.

*membership  
union / ...  
add/delete*

- **List** : maintains some form of linear ordering of values it holds.

# Set of Numbers

- **Set** : holds unique values, with no specific order.

# Representing related data: Representing Students in COL106

Each student is unique and inimitable

- Each student consists of nearly infinite “characteristics”

Thank God - we do not need all characteristics for every task!

In a class roster, we just need the following for each student:

- Official name - a string
- Entry number - a string
- Batch number - a number between 1-4

How do we represent this?



# Representing a Collection of Students of COL106

- **Approach 1:**

Names = ["Veeru", "Basanti", "Jai"]

Batch= [1, 4, 3]

Entrynumber = [001, 004, 002]

## **Typical operations:**

Adding a new student, deregistering Basanti from the course, ...

# Representing a Collection of Students of COL106

- **Approach 2:**

Student1 = { Name : "Veeru", Batch : 1, Entrynumber : "001" }

Student2 = { Name : "Basanti", Batch : 4, Entrynumber : "004" }

Student3 = { Name : "Jai", Batch : 3, Entrynumber : "002" }

**Related values are together.**

Course roster = {Student1, Student2, Student3}

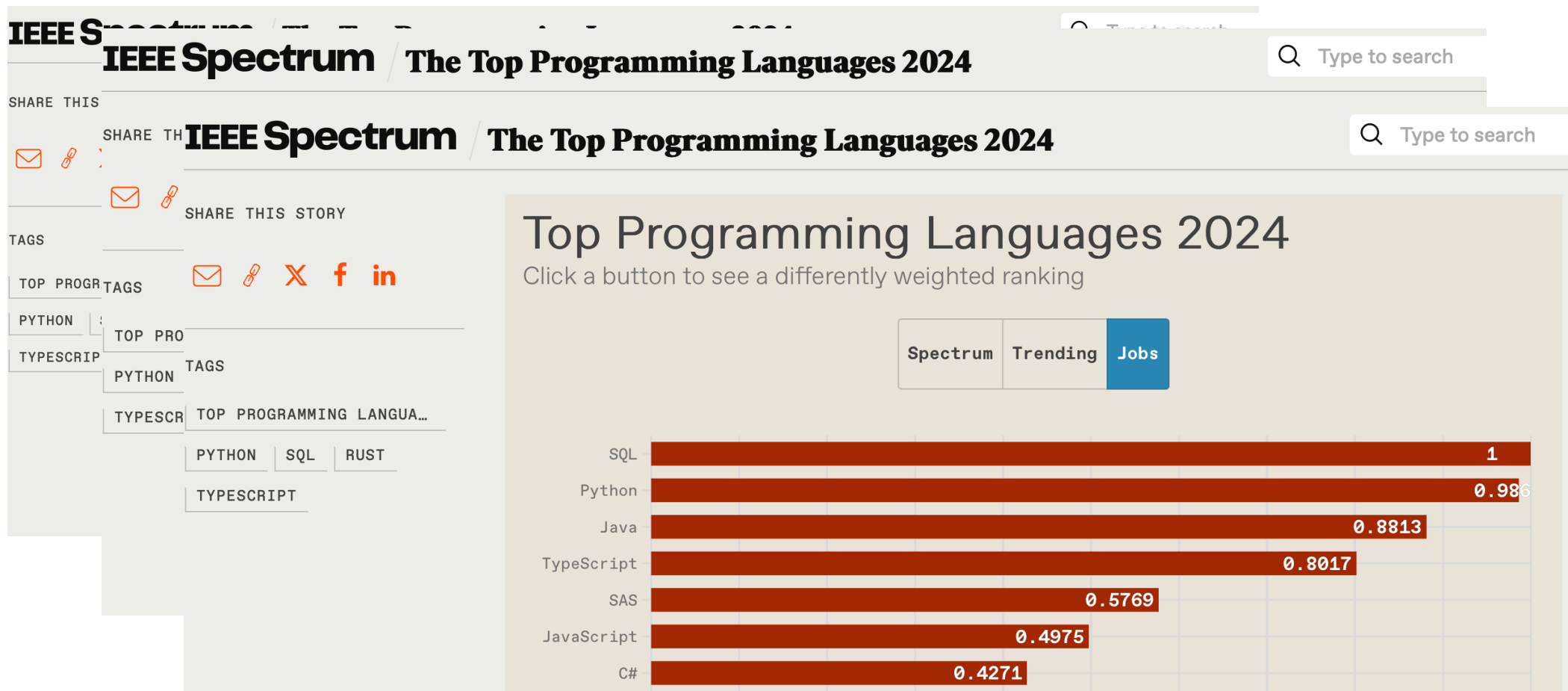
**Typical operations:**

Adding a new student, deregistering Basanti from the course, ...

# Object Orientation

- Every Object contains
  - A set of value(s) related to the object
  - A set of behaviors that can be applied to the object's value (e.g., correct the name, update the batch)
- Class
  - A “template” for objects – values identifiers, methods that define the behavior of the object
- Instantiate Objects from Class

# Java Programming Language



# Java Programming Language

- An object-oriented, high-level language with automatic garbage collection that follows “Write-Once Run Anywhere” (WORA) paradigm
  - It is not an interpreted language, but is compiled into an intermediate byte-code – **javac**
  - Executed over a “virtual machine” that mediates with the underlying hardware and the byte-code – **java**

