CS0001

Discrete Structures 1

Module 1: Introduction and Foundations

Technology Driven by Innovation

Discrete Structures 1 (CS0001)

This course introduces the formal language and foundational structures of computer science. Students will learn to construct sound **logical arguments**, write **mathematical proofs**, and model **complex systems** using the core concepts of **logic**, **set theory**, **relations**, **functions**, and **mathematical induction**.

Discrete Structures 1 (CS0001)

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

1. Analyze and translate statements from informal language into formal **propositional and predicate logic**, and use the **rules of inference** to construct and validate logical arguments;
2. Construct valid **mathematical proofs** using techniques such as **direct proof, proof by contradiction, and mathematical induction** to formally verify the correctness of algorithms and mathematical statements;
3. Apply the operations and properties of **sets**, **relations**, and **functions** to model computational problems, data structures, and the relationships between them; and
4. Model simple real-world scenarios, such as social networks or file systems, using the **basic structures of graphs and trees**, and explain the properties of the model using the fundamental terminology of graph theory.

Discrete Structures 1 (CS0001)

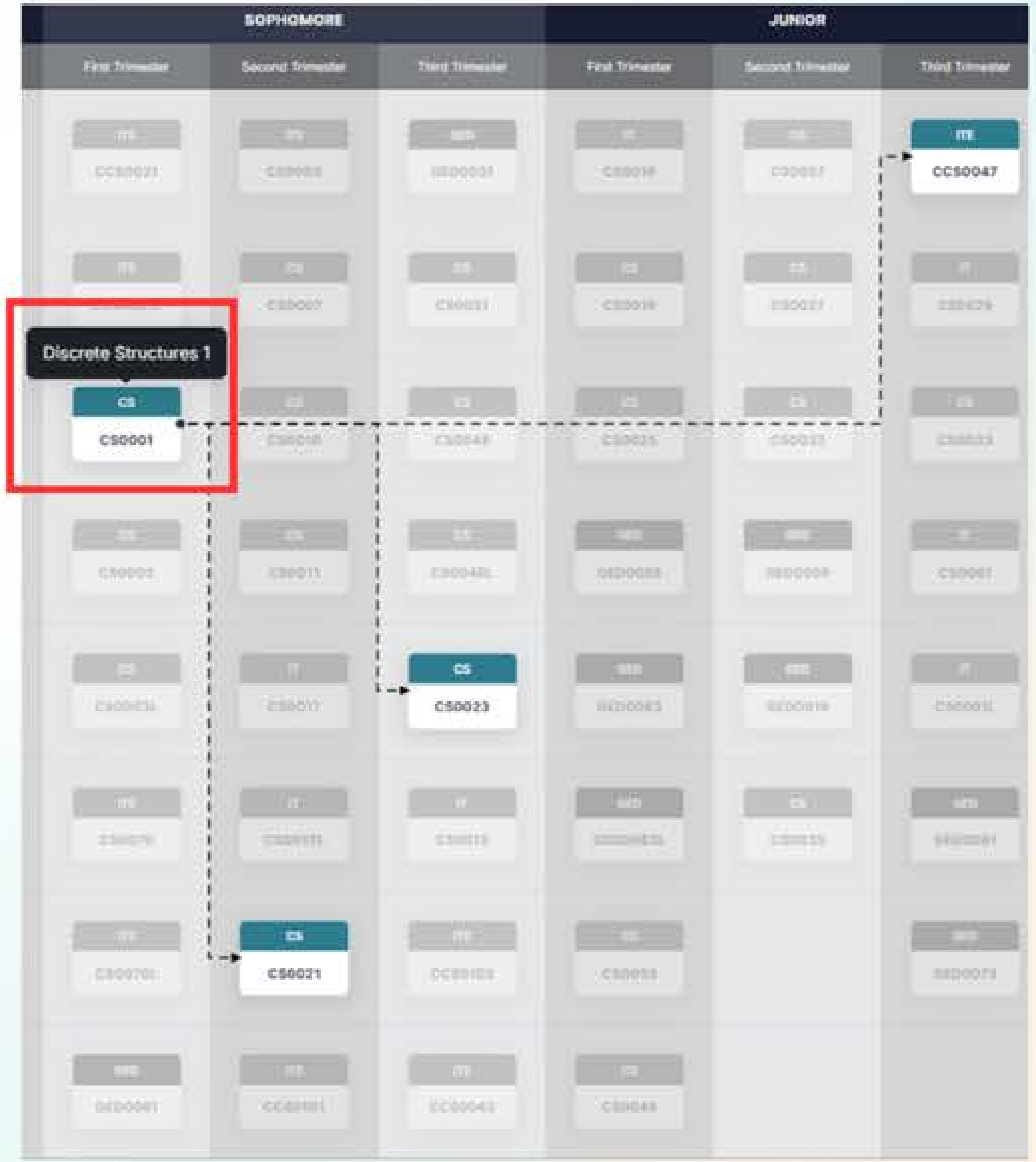
COURSE SYLLABUS	
01	Introduction and Foundations
02	Propositional Logic and Its Applications
03	Predicate Logic and Euler Circles
04	Formal Proof Techniques
05	Set Theory
06	Relations
07	Functions
08	Introduction to Graphs and Trees

Discrete Structures 1 (CS0001)

This course, Discrete Structures 1, is the first and most important step in a sequence that will build your expertise as a computer scientist.

[This Trimester] -> Discrete Structures 1: The Fundamentals

Here, you will learn the fundamental "language and grammar" of computer science: logic, proofs, and basic structures.



Technology Driven by Innovation

Discrete Structures 1 (CS0001)

Building upon this foundation, **Discrete Structures 2** will allow you to apply these skills to create a powerful toolkit for modeling and analyzing algorithms, with a focus on graphs, trees, and combinatorics.

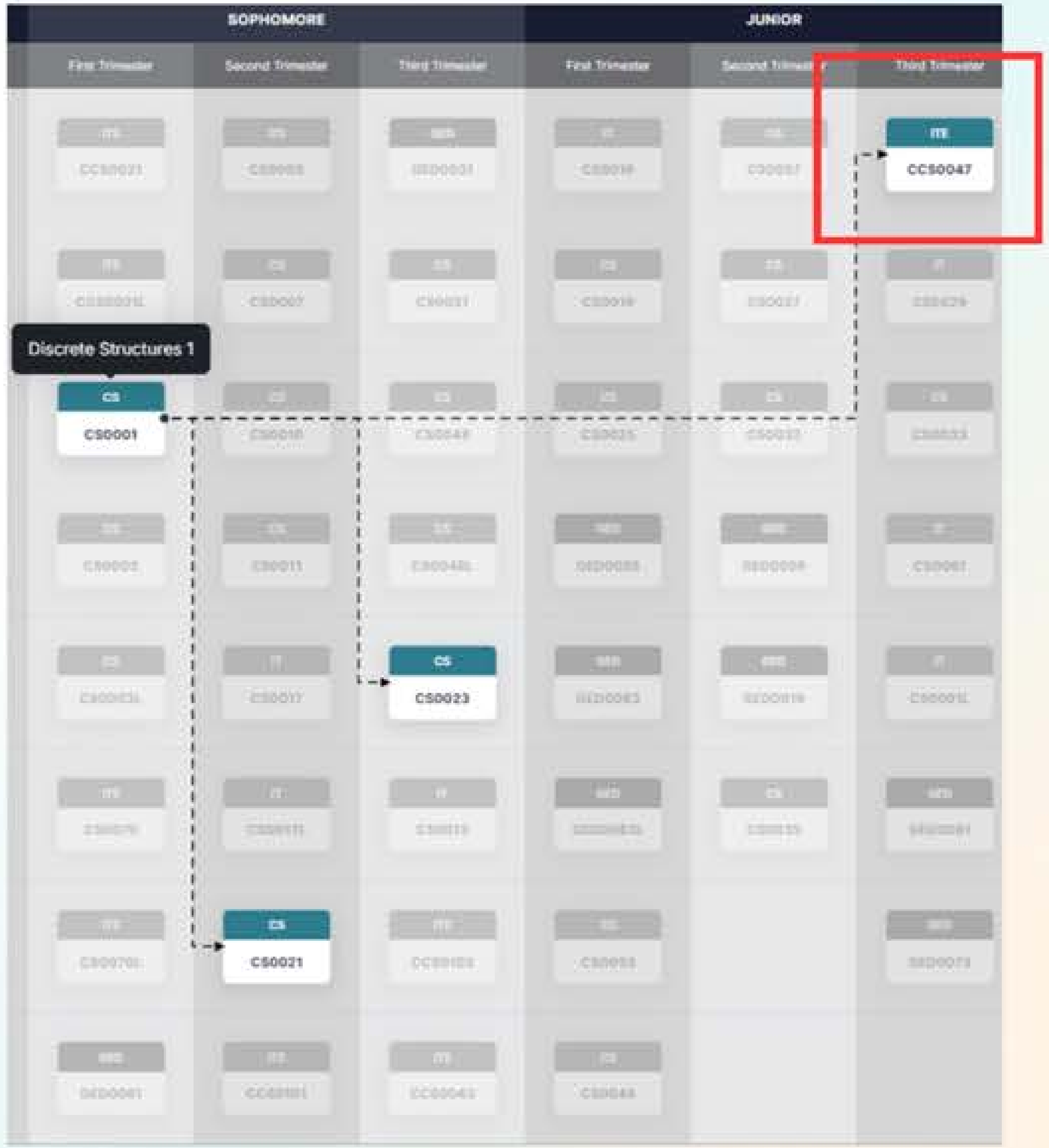
Technology Driven by Innovation

Discrete Structures 1 (CS0001)

This theoretical groundwork then enables you to explore the fundamental question of "What can be computed?" in **Automata Theory & Formal Languages** by studying the formal models of computers and languages.

Discrete Structures 1 (CS0001)

The sequence culminates in **Number Theory**, where you will take a deep dive into the properties of integers—the mathematical bedrock for modern cryptography and computer security.



CS0001

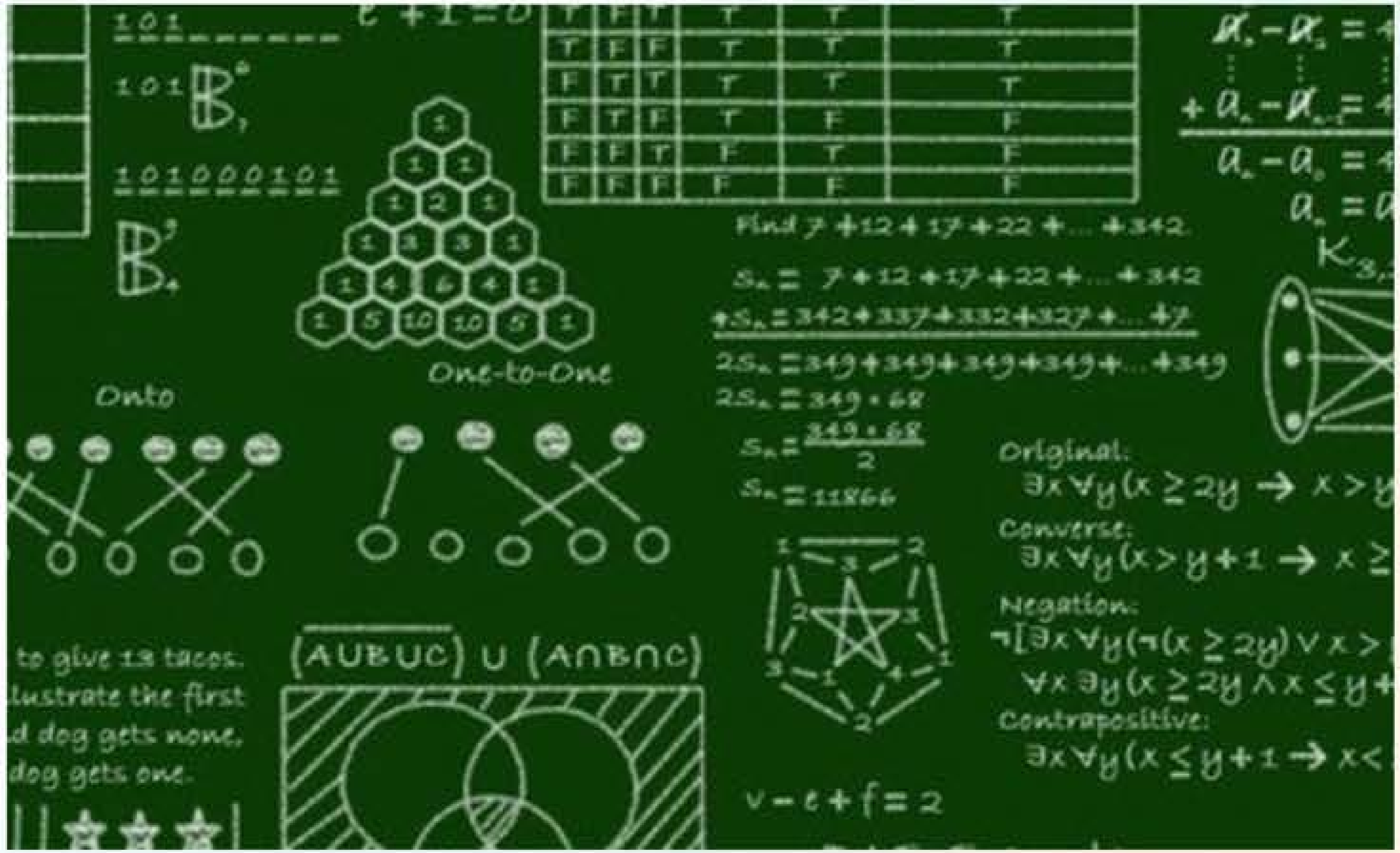
Discrete Structures 1

Subtopic 1: Preliminaries - The What, Why, and How of Discrete Structures

The Mathematics of Computer Science

Discrete Structures is the branch of mathematics that deals with objects that can only assume distinct and separated values.

It is opposite to continuous mathematics (like calculus), which deals with objects that can vary smoothly.



The Mathematics of Computer Science

It provides the **foundational language** and **logical framework** for computer science.

It focuses on studying **fundamental**, **non-continuous structures** such as:

- Logical Statements (True or False)
- Sets (Collections of distinct objects)
- Relations (Relationships between objects)
- Graphs and Trees (Networks of nodes and connections)

Discrete vs. Continuous

Discrete - Describes values that are **distinct**, **separate**, and **countable**. There are clear gaps between one value and the next.

Examples:

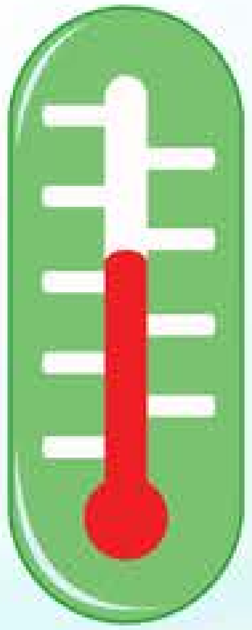
- The **number** of students in this class
- The **steps** in an algorithm or recipe
- The **score** in a basketball game

Discrete vs. Continuous

Continuous - Describes values that change smoothly and can be broken down **infinitely**. There is always another possible value **between any two points**.

Examples:

- A person's exact **height**
- The **temperature** of a room
- The **speed** of a car



Think About This

For each item below, determine if it is best described as Discrete (countable, distinct units) or Continuous (measurable on a spectrum).

1. The number of lines of code in a program.
2. The amount of time it takes for an algorithm to complete.
3. The number of CPU cores in a processor.
4. The signal strength of a Wi-Fi connection.
5. The size of a file in bytes.
6. The download speed of your internet connection.
7. The screen resolution of a monitor (e.g., 1920x1080 pixels).
8. The exact voltage being supplied to a computer component.
9. The available refresh rates for a gaming monitor (e.g., 60Hz, 120Hz, 144Hz).
10. The battery percentage displayed on your phone.

Think About This

The highlighted items in the list are examples of Discrete variables, while the remaining items are Continuous.

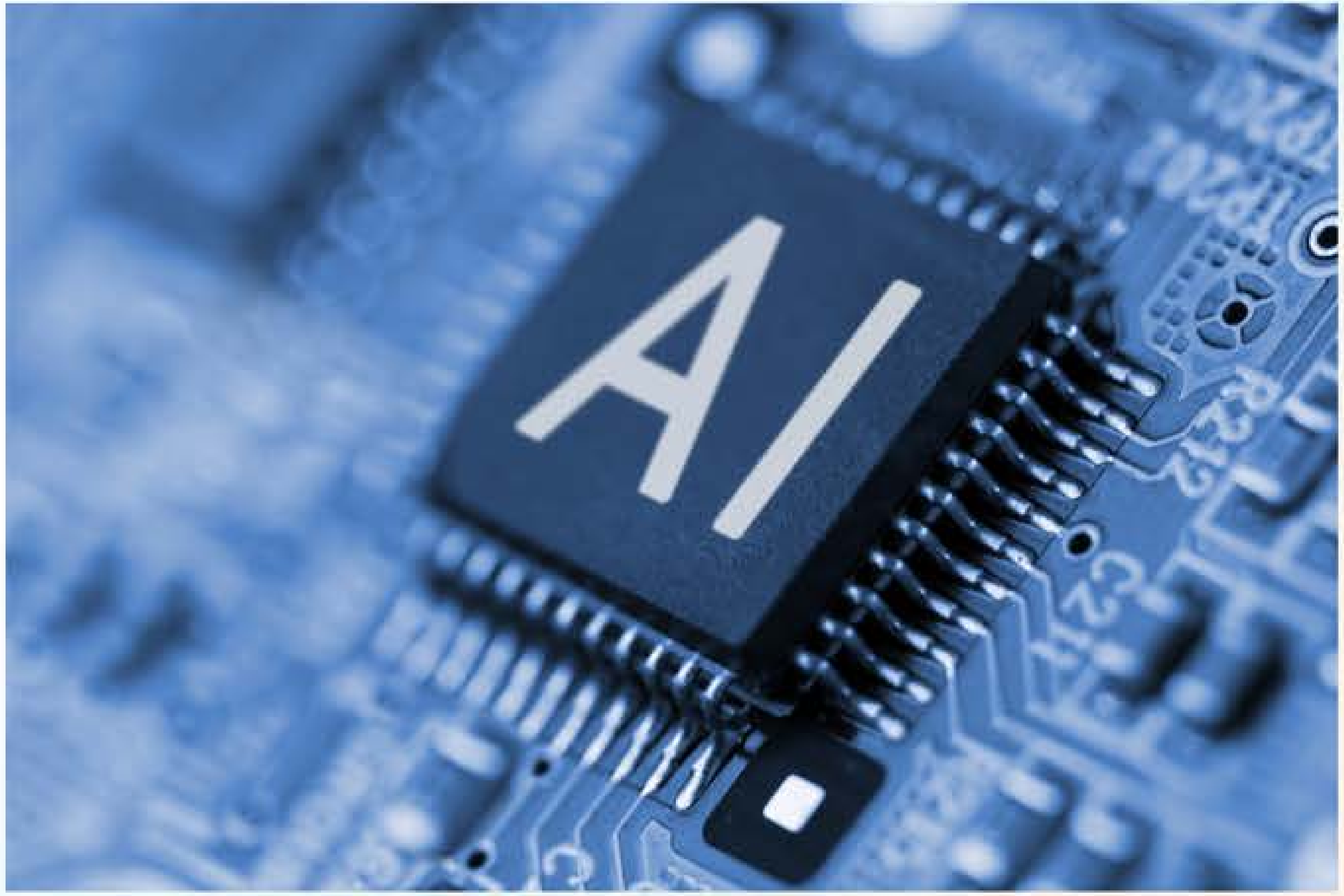
1. **The number of lines of code in a program.**
2. The amount of time it takes for an algorithm to complete.
3. **The number of CPU cores in a processor.**
4. The signal strength of a Wi-Fi connection.
5. **The size of a file in bytes.**
6. The download speed of your internet connection.
7. **The screen resolution of a monitor (e.g., 1920x1080 pixels).**
8. The exact voltage being supplied to a computer component.
9. **The available refresh rates for a gaming monitor (e.g., 60Hz, 120Hz, 144Hz).**
10. The battery percentage displayed on your phone.

Why Study Discrete Structures?

Logic → Digital Circuits, AI, & Databases

The logic we study is directly implemented in the **physical hardware of computer chips**.

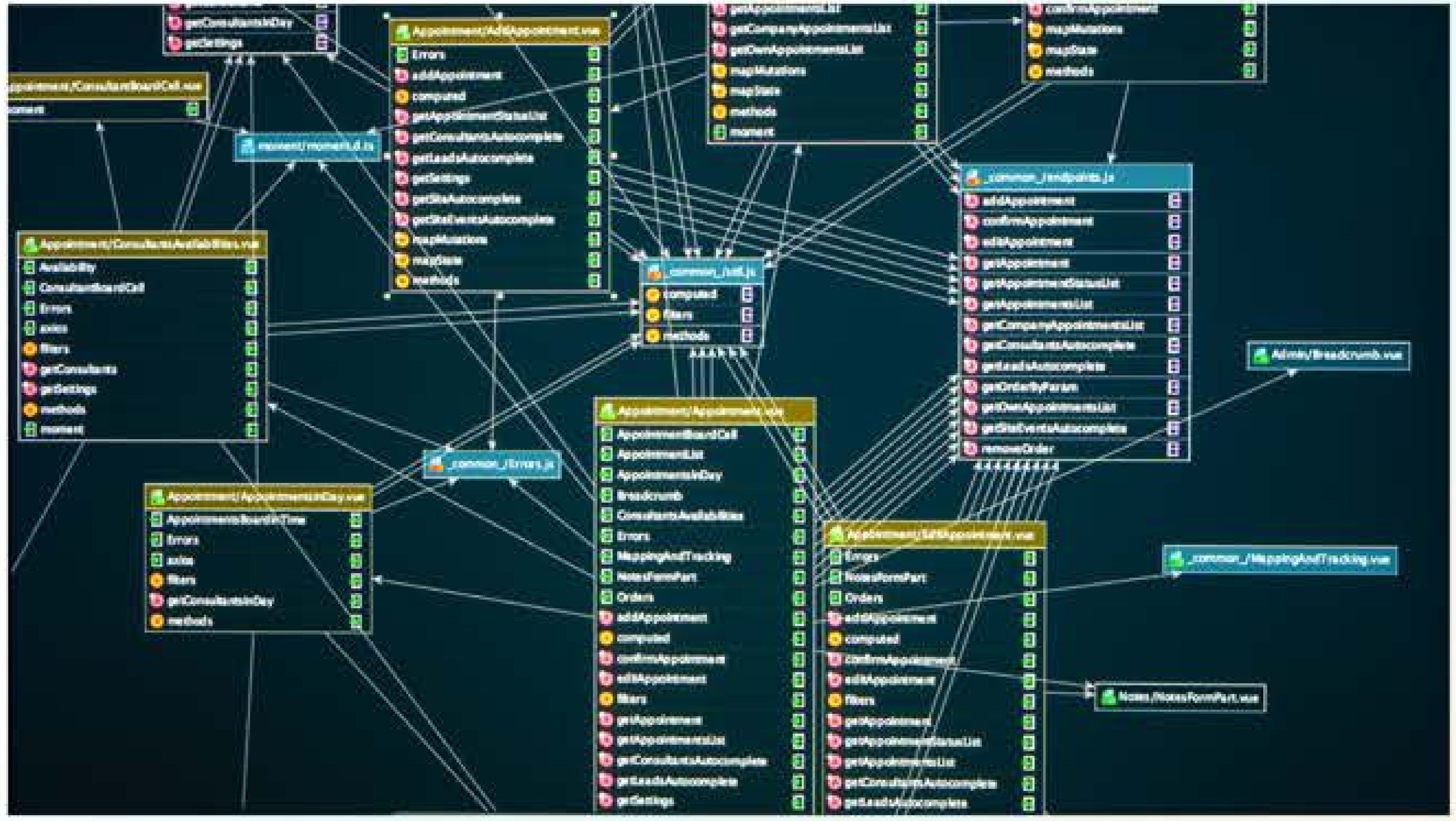
It's also the foundation for **database queries** and **artificial intelligence**.



Why Study Discrete Structures?

Sets & Relations → Modern Databases

The entire theory of relational databases (like SQL) is built on the mathematical principles of sets and relations.



Why Study Discrete Structures?

Graphs & Trees → Networks, Social Media, & Navigation

Graphs are the single most important structure for modeling networks — from Google Maps finding the shortest path to analyzing the connections on LinkedIn.

Roadmap in Studying Discrete Structures

Our Journey This Term:

- First, we will learn the language of computer science (Logic & Proofs).
- Then, we will use that language to explore the fundamental structures of the digital world (Sets, Relations, Functions, Graphs).

A Shift in Focus: From Calculating to Reasoning

- In many math classes, the goal is to calculate a final numerical answer.
- In this course, the goal is to construct a valid, logical argument. The "why" your solution is correct is more important than the solution itself.

CS0001

Discrete Structures 1

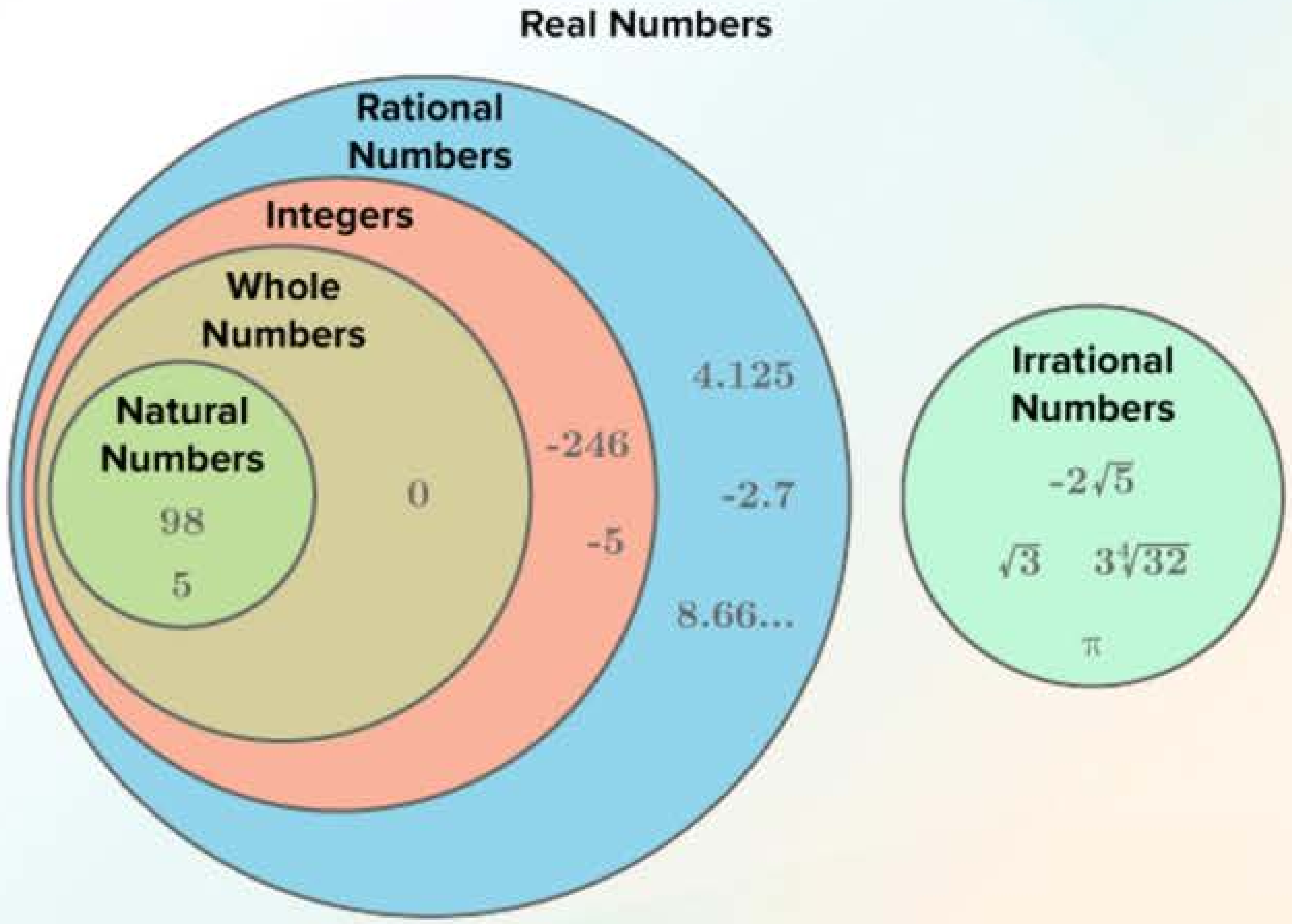
Subtopic 2: Review of Essential Mathematical Concepts & Notation

Real Number System

This diagram shows how the numbers we use are organized. For this course, it's helpful to remember the main categories:

Real Numbers: The set of all numbers on the number line. They are divided into two main groups:

- **Rational Numbers**: Any number that can be written as a simple fraction, like 4.125 or −5.
- **Irrational Numbers**: Numbers that cannot be written as a simple fraction and have non-repeating decimals, like π or 3.

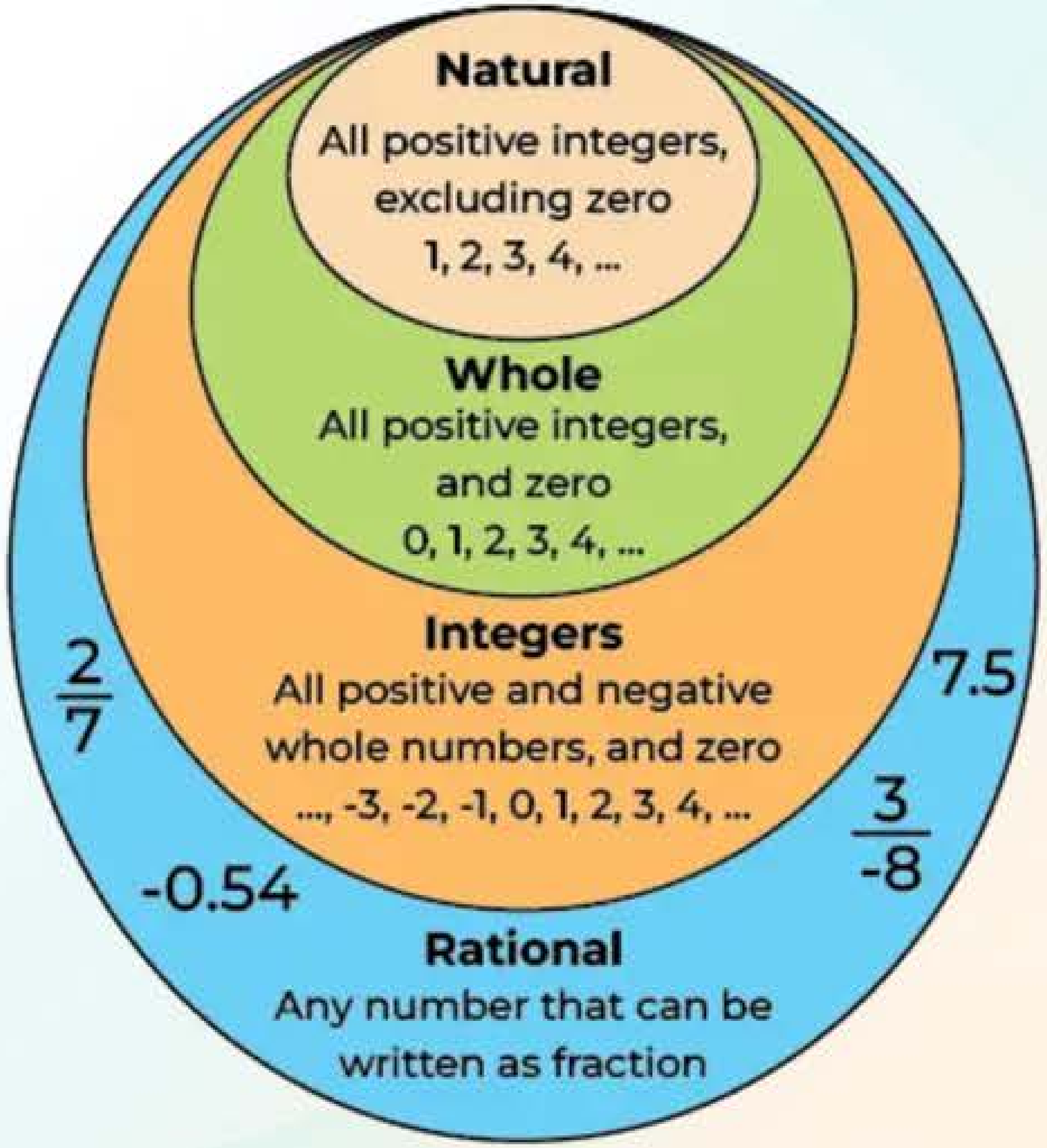


Real Number System

Inside the Rational Numbers, we find nested groups:

- **Integers (Z):** All positive and negative whole numbers, including zero (e.g., -246, -5, 0, 98).
- **Whole Numbers:** The non-negative integers (0, 1, 2, ...).
- **Natural Numbers (N):** The positive "counting" numbers (1, 2, 3, ...).

NOTE: While all these numbers exist, our work in this course will primarily take place in the world of Integers (Z) and Natural Numbers (N).



Important Notations for Sequences and Series

Summation notation (or sigma notation) allows us to write a long sum in a single expression.

Stop at $n = 3$ (inclusive)

$$\sum_{n=1}^{3} 2n - 1$$

Start at $n = 1$

Expression for each term in the sum

Important Notations for Sequences and Series

This is a summation of the expression 2n−1 for integer values of n from 1 to 3:

$$\sum_{n=1}^{3} 2n - 1$$

$= [2(1) - 1] + [2(2) - 1] + [2(3) - 1]$

$= 1 + 3 + 5$

$\boxed{= 9}$

n is our summation index. When we evaluate a summation expression, we keep substituting different values for our index.

Important Notations for Sequences and Series

Product notation, or pi notation, is a mathematical tool that indicates repeated multiplication.

Stop at n = 4

$$\prod_{k=1}^{4}(2k)$$

Expression for each term in the product

Start at k = 1

Important Notations for Sequences and Series

Product notation, or pi notation, is a mathematical tool that indicates repeated multiplication.

$$\prod_{k=1}^{4}(2k) = (2 \cdot 1) \cdot (2 \cdot 2) \cdot (2 \cdot 3) \cdot (2 \cdot 4) = 2 \cdot 4 \cdot 6 \cdot 8 = \boxed{384}$$

Similar to how **∑ (Sigma)** is used for adding terms, the **Product Notation (∏)** is a shorthand for multiplying a sequence of terms together.

Important Notations for Sequences and Series

Factorial notation is a mathematical symbol that represents the product of all positive integers from 1 up to a given number. It is written as an exclamation mark (!) after a positive integer n and is read as "n factorial".

Examples of factorial notation:

- $4! = 4 \times 3 \times 2 \times 1 = \boxed{24}$
- $5! = 5 \times 4 \times 3 \times 2 \times 1 = \boxed{120}$
- $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = \boxed{720}$

Recursive Property: A factorial can also be defined recursively as **n! = n x (n - 1)!**

Module 1 Summary

What We Learned:

- Discrete mathematics is the foundational language of computer science, dealing with distinct, countable objects.
- This course is a journey from the language of logic and proofs to the core structures of CS (sets, graphs, etc.).
- Success in this course requires a shift from calculation to formal, logical reasoning.

What's Next:

- We will begin our journey into this formal language with our first core topic:

Propositional Logic and Its Applications.

CS0001

Discrete Structures 1

End of Module

REFERENCES

Haggard, G., Schlipf, J., & Whitesides, S. (2006). Discrete mathematics for computer science. Thomson Brooks/Cole.

Kolman, B., Busby, R. C., & Ross, S. C. (2009). Discrete mathematical structures (6th ed.). Pearson Prentice Hall.

Rosen, K. H. (2024). Discrete mathematics and its applications (9th ed.). McGraw-Hill Education.

Rosen, K. H., Michaels, J. G., Gross, J. L., Grossman, J. W., & Shier, D. R. (Eds.). (2018). Handbook of discrete and combinatorial mathematics (2nd ed.). CRC Press.

Ross, K. A., & Wright, C. R. B. (2003). Discrete mathematics (5th ed.). Prentice Hall.