Tutorials in Computer Science

Computer Networks

Covers how computers communicate over networks, from physical connections to modern Internet protocols. Includes practical packet analysis and configuration tasks.

Prerequisites: Basic programming and computer systems knowledge.

Topics covered:
1. Network architecture and models.
2. Data link layer and Ethernet.
3. IP addressing and routing basics.
4. Transport protocols (TCP/UDP).
5. Application layer protocols (HTTP, DNS).
6. Network security basics.
7. Wireless and mobile networking.
8. Modern trends (SDN, QUIC).

Computer Security

An introduction to protecting systems from cyber threats. Students will learn about authentication, software vulnerabilities, secure networking, and emerging attacks.

Prerequisites: Basic programming, computer systems knowledge.

Topics covered:
1. Security principles and threat models.
2. Authentication and access control.
3. Software vulnerabilities.
4. Web application security.
5. Operating system security.
6. Network security.
7. Malware and intrusion detection.
8. Privacy and advanced topics.

Computer Vision

Learn how computers process and interpret images and video, from classical image processing to modern deep learning applications. Students will gain hands-on experience with vision libraries.

Prerequisites: Basic Python, linear algebra, probability.

Topics covered:
1. Digital images and pixel operations.
2. Edge detection and filtering.
3. Feature extraction and matching.
4. Camera models and geometry basics.

5. Classical recognition methods.
6. Convolutional neural networks for vision.
7. Object detection and segmentation basics.
8. Mini-project: simple object detector.

Cryptography

Introduces the mathematics and algorithms behind secure communication. Covers symmetric and public-key encryption, digital signatures, and secure protocols.

Prerequisites: Discrete mathematics, modular arithmetic.

Topics covered:
1. Math foundations and security goals.
2. Symmetric encryption.
3. Hash functions and MACs.
4. Public-key cryptography.
5. Digital signatures.
6. Secure protocols (TLS).
7. Post-quantum and advanced crypto.
8. Applied cryptography in practice.

Digital Systems

Explores the design of hardware systems from logic gates to simple processors. Students will work with basic hardware description languages.

Prerequisites: Basic mathematics; programming helpful.

Topics covered:
1. Number systems and logic gates.
2. Simplifying circuits (K-maps).
3. Arithmetic circuits (adders, ALU).
4. Sequential logic basics.
5. Memory systems.
6. Microarchitecture basics.
7. Pipelining concepts.
8. Building and testing a small CPU model.

Distributed Systems

Learn how to design computer systems that work across multiple machines but appear unified to users. Topics include replication, coordination, and handling failures, with examples from modern cloud platforms. Students will gain the ability to reason about and troubleshoot distributed applications.

Prerequisites: Familiarity with computer networks and basic programming.

Topics covered:
1. What makes a system distributed? Models and failures.

2. Naming and service discovery.
3. Replication basics and consistency models.
4. Consensus algorithms (Raft basics).
5. Time, ordering, and coordination.
6. Distributed storage examples.
7. Streaming and messaging systems.
8. Case study: cloud-based service design.

Logic Circuits

This course introduces the principles of digital logic design, from Boolean algebra to practical circuit implementation. Students will learn to analyse and design efficient combinational and sequential logic systems, with an emphasis on reliability and simplification techniques. By the end, they will be able to design and verify small-scale digital systems.

Prerequisites: Basic mathematics; no prior electronics experience required.

Topics covered :
1. Boolean algebra and propositional logic basics.
2. Logic gates and canonical forms.
3. Circuit simplification (K-maps).
4. Common combinational blocks (mux, decoder, encoder).
5. Arithmetic logic (adders, comparators).
6. Sequential elements (flip-flops, counters, shift registers).
7. Finite state machines.
8. Mini-project: design & verify a traffic-light controller.

Machine Learning

An introduction to algorithms that allow computers to learn from data. Students will learn model training, evaluation, and applications in prediction and classification.

Prerequisites: Basic Python, linear algebra, and probability.

Topics covered:
1. What is ML? Supervised vs unsupervised learning.
2. Linear regression and gradient descent.
3. Classification and logistic regression.
4. Decision trees and ensembles.
5. Feature engineering and preprocessing.
6. Clustering and dimensionality reduction.
7. Introduction to neural networks.
8. Responsible ML and project wrap-up.

Object-Oriented Programming

Explore how to design software using the object-oriented paradigm, focusing on classes, objects, and design patterns. Students will learn to build maintainable and reusable code for larger systems.

Prerequisites: Basic programming knowledge.

Topics covered:

1. OOP concepts: encapsulation, inheritance, polymorphism.
2. Classes and objects in practice.
3. Interfaces and abstract classes.
4. Error handling and immutability.
5. Collections and iterators.
6. Common design patterns (Strategy, Observer).
7. More patterns (Factory, Adapter).
8. Testing and refactoring OOP code.

Principles of Programming

This course builds a foundation in core programming concepts, problem-solving strategies, and code organisation. Students will learn imperative and functional approaches, basic data structures, and good coding practices.

Prerequisites: No prior programming required, but logical thinking is essential.

Topics covered:

1. Programs, syntax, and semantics.
2. Variables, control flow, and functions.
3. Recursion and problem decomposition.
4. Data types and structures (lists, maps).
5. Higher-order functions and modularity.
6. Memory basics and references.
7. Version control and collaborative coding.
8. Mini-project: build a small application.