

# Computer Science 101

## Core Concepts

June 09, 2025

**John Doe**

johndoe@example.com

## Contents

1. Code Blocks .....	2
2. Admonitions .....	2
3. Proof Trees .....	3
4. Finite State Machines .....	3
5. Data Visualisation .....	3
6. Pseudocode .....	4

## 1. Code Blocks


This section demonstrates how to include code blocks in your lecture notes using the `codly` package. Code blocks are useful for showing programming examples, syntax demonstrations, or algorithm implementations in various languages.

Below are a couple of examples:

```
1 pub fn main() {  
2     println!("Hello, world!");  
3 }
```

 Rust

```
1 def fib(n):  
2     if n <= 1:  
3         return n  
4     else:  
5         return fib(n - 1) + fib(n - 2)  
6 print(fib(25))
```

 Python

## 2. Admonitions

Admonitions allow you to include visually distinct callouts to highlight important information. These are great for emphasizing tips, warnings, examples, and extra info during lectures.

These clues are created using the `gentle-clues` package:

### Info

Use info boxes to highlight background knowledge or context.

### Warning

Warnings are perfect for common mistakes or critical caveats.

### Tip

Tips are great for best practices, shortcuts, or helpful advice.

### Example

Examples are used to clarify concepts with concrete scenarios.

### 3. Proof Trees

Proof trees are a visual way to represent logical derivations or inference steps. They are commonly used in logic, type theory, and formal systems.

This section uses the `curryst` package to typeset formal proofs. You can define premises, apply rules, and construct derivations with clarity.

Here's a generic proof tree layout:

$$\text{Label} \frac{\text{Premise 1} \quad \text{Premise 2} \quad \text{Premise 3}}{\text{Conclusion}} \text{Rule name}$$

And here's a natural deduction proof showing implication introduction:

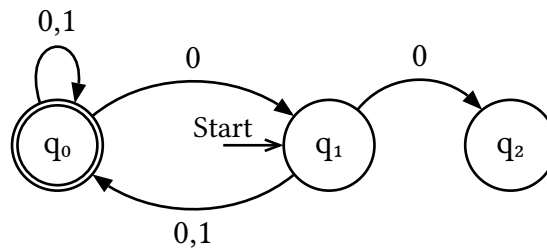
$$\frac{\frac{\frac{}{\Gamma \vdash p \rightarrow q} \text{ax}}{\Gamma \vdash q} \text{ax} \quad \frac{\frac{\frac{}{\Gamma \vdash p \wedge \neg q} \text{ax}}{\Gamma \vdash p} \wedge_e^\ell \quad \frac{\frac{}{\Gamma \vdash p \wedge \neg q} \text{ax}}{\Gamma \vdash \neg q} \wedge_e^r}{\frac{p \rightarrow q, p \wedge \neg q \vdash \perp}{\Gamma} \neg_e}{\frac{p \rightarrow q \vdash \neg(p \wedge \neg q)}{\vdash (p \rightarrow q) \rightarrow \neg(p \wedge \neg q)} \neg_i \rightarrow_i}$$

### 4. Finite State Machines

Finite state machines (FSMs) are used to model systems with discrete states and transitions. This is common in automata theory, language processing, and hardware design.

With the `finite` package, you can visually define FSMs using a clear syntax.

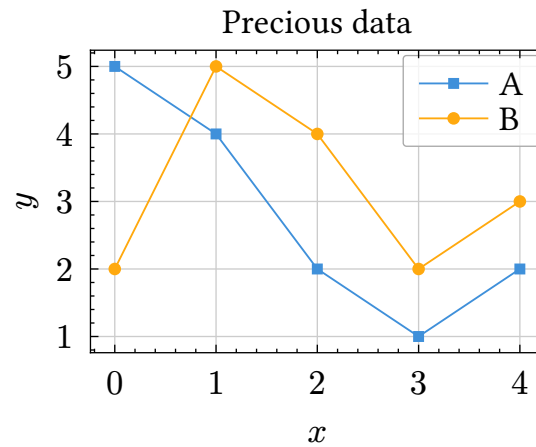
Here is an example automaton:



### 5. Data Visualisation

Data visualisation helps present trends, comparisons, or distributions in a clear, visual format. It's especially useful when discussing performance metrics, simulation results, or algorithmic behavior.

This example uses the `lilaq` package, which provides plotting utilities for line charts and other diagram types.



## 6. Pseudocode

Pseudocode is a great way to illustrate algorithms without being tied to a specific programming language. It emphasizes logic and structure rather than syntax.

The `lovelace` package provides tools for typesetting pseudocode lists in a structured, readable format.

Here's an example representing a generic control flow:

```
1 do something
2 do something else
3 while still something to do
4   do even more
5   if not done yet then
6     wait a bit
7     resume working
8   else
9     go home
10  end
11 end
```