ARTIFICIAL INTELLIGENCE | BEGINNER | 2 DAYS

Cursor for Developers

A practical grounding in Cursor for daily development. Covers migration from VS Code, tab and inline editing, Composer for multi-file work, context management, model selection, and the refactoring, testing and review habits that keep AI-assisted code trustworthy.

Why Cursor for Developers?

Context is the whole game

Most poor output traces back to context, not the model. Learn to control it.

Model choice matters

Practical guidance on picking a model per task rather than defaulting to one.

Works in your stack

Labs run against your team's real languages and repositories.

Honest about failure modes

Clear on where Cursor produces confident, plausible, wrong code.

What you'll be able to do

- Install Cursor and migrate settings, extensions and keybindings from VS Code
- Use tab completion and inline editing effectively
- Apply Composer to scoped multi-file changes with proper diff review
- Control context through file, folder, documentation and web references
- Select models appropriately for different kinds of task
- Prompt for code with clear intent, scope and constraints
- Refactor, test and debug existing code with AI assistance
- Review AI-generated output critically before it reaches a commit

Who this is for

- Software developers new to Cursor
- Engineers migrating from VS Code or another AI coding tool
- Technical leads evaluating Cursor for team adoption
- Graduate and early-career engineers

Curriculum

01 Getting Started and Migration

- Installing Cursor and bringing across VS Code settings and extensions
- Interface tour: editor, chat, composer and agents
- Plan tiers and what is available at each
- Configuring the editor for how you actually work
- Where Cursor differs from a VS Code plus extension setup

02 Tab and Inline Editing

- Tab completion and predicting the next edit
- Multi-cursor edits and following a change through a file
- Inline edit for scoped, targeted changes
- Accept and reject discipline: not taking suggestions on autopilot
- **Hands-on:** implement a small feature using inline tools only

03 Composer and Multi-File Changes

- Scoping a change across several files
- The diff view and reviewing before anything is applied
- When Composer is the right tool and when agent mode is better
- Recovering when a multi-file change goes wrong
- **Hands-on:** perform a cross-cutting refactor with Composer

04 Context Management

- Referencing files, folders, symbols, documentation and web sources
- Codebase indexing and what the model can actually see
- Why too much context degrades output as badly as too little
- Diagnosing bad answers as context problems rather than model problems
- **Hands-on:** fix a failing request purely by changing context

05 Model Selection and Prompting for Code

- Available models and their practical differences
- Choosing per task: fast edits versus complex reasoning
- Intent, scope and constraints in a code-specific request
- Asking for alternatives and trade-off explanations
- Cost awareness: what your choices consume

06 Refactoring, Testing and Debugging

- Understanding unfamiliar or legacy code quickly
- Establishing test coverage before refactoring, not after
- Generating and extending meaningful tests
- Debugging, error explanation and documentation generation
- **Hands-on:** refactor and test an existing module in your own stack

07 Reviewing Output and Responsible Use

- Failure modes: code that looks right and is not
- What always needs verification before it reaches a commit
- Secrets, dependencies and security risks in generated code
- Licensing and organisational policy considerations
- Building a personal review checklist you will actually follow