Pearson

Claude API for Python Developers

https://github.com/DataForScience/CodeAI

Bruno Gonçalves

Data For Science, Inc

www.data4sci.com/newsletter

data4sci.substack.com

Contacts

Bruno Gonçalves

https://data4sci.com

info@data4sci.com

https://data4sci.com/call

https://www.linkedin.com/in/bmtgoncalves/

Bruno Gonçalves is an author, public speaker, corporate trainer, and consultant specializing in Generative AI, Blockchain Analytics, and Machine Learning. He founded Data For Science, Inc in 2009 to help individuals and companies solve their data driven problems.

Coming Up:

- **CrewAI for Production-Ready Multi-Agent Systems**
  Feb 19, 2026 10am-2pm (PST)
- **Gemini API with VertexAI for Developers**
  Mar 18, 2026 10am-2pm (PST)
- **LangChain for Generative AI Pipelines**
  Apr 22, 2026 10am-2pm (PST)
- **Claude API for Python Developers**
  Apr 29, 2026 10am-2pm (PST)

Question

What’s your job title?

- Data Scientist
- Statistician
- Data Engineer
- Researcher
- Business Analyst
- Software Engineer
- Other

Question

How experienced are you in Python?

- Beginner (<1 year)
- Intermediate (1-5 years)
- Expert (5+ years)

Question

How did you hear about this webinar?

- O’Reilly Platform
- Newsletter
- data4sci.com Website
- Previous event
- Other?

Pearson

Table of Contents:

Pearson

1. AI Assistants Fundamentals

What Are AI Code Assistants?

- AI-powered tools that help developers write, understand, and optimize code

- **Core Components**:
  - Large Language Models (LLMs)
  - Code context understanding
  - Natural language processing
  - Pattern recognition from billions of lines of code

- **Timeline:**
  - 1980s-1990s: Syntax highlighting, auto-indentation
  - 2000s: IntelliSense, code completion
  - 2010s: Linters, static analysis
  - 2020s: AI-powered code generation
  - 2024+: Agentic AI assistants

Three Categories of AI Assistants

- **Code Completion** (GitHub Copilot, Tabnine)
  - Inline suggestions
  - Real-time completions
- **Chat-Based** (ChatGPT, Claude)
  - Conversational interface
  - Explanations and refactoring
- **Agentic** (Cursor, Windsurf, Claude Code)
  - Multi-step reasoning
  - File editing and execution
  - Autonomous problem-solving

How AI Assistants Work

Multistep Process:

1. **Input**: Natural language prompt or code context
2. **Processing**: LLM analyzes intent and context
3. **Generation**: Creates code suggestions
4. **Refinement**: Filters for syntax and best practices
5. **Output**: Presents options to developer

Large Language Models for Code

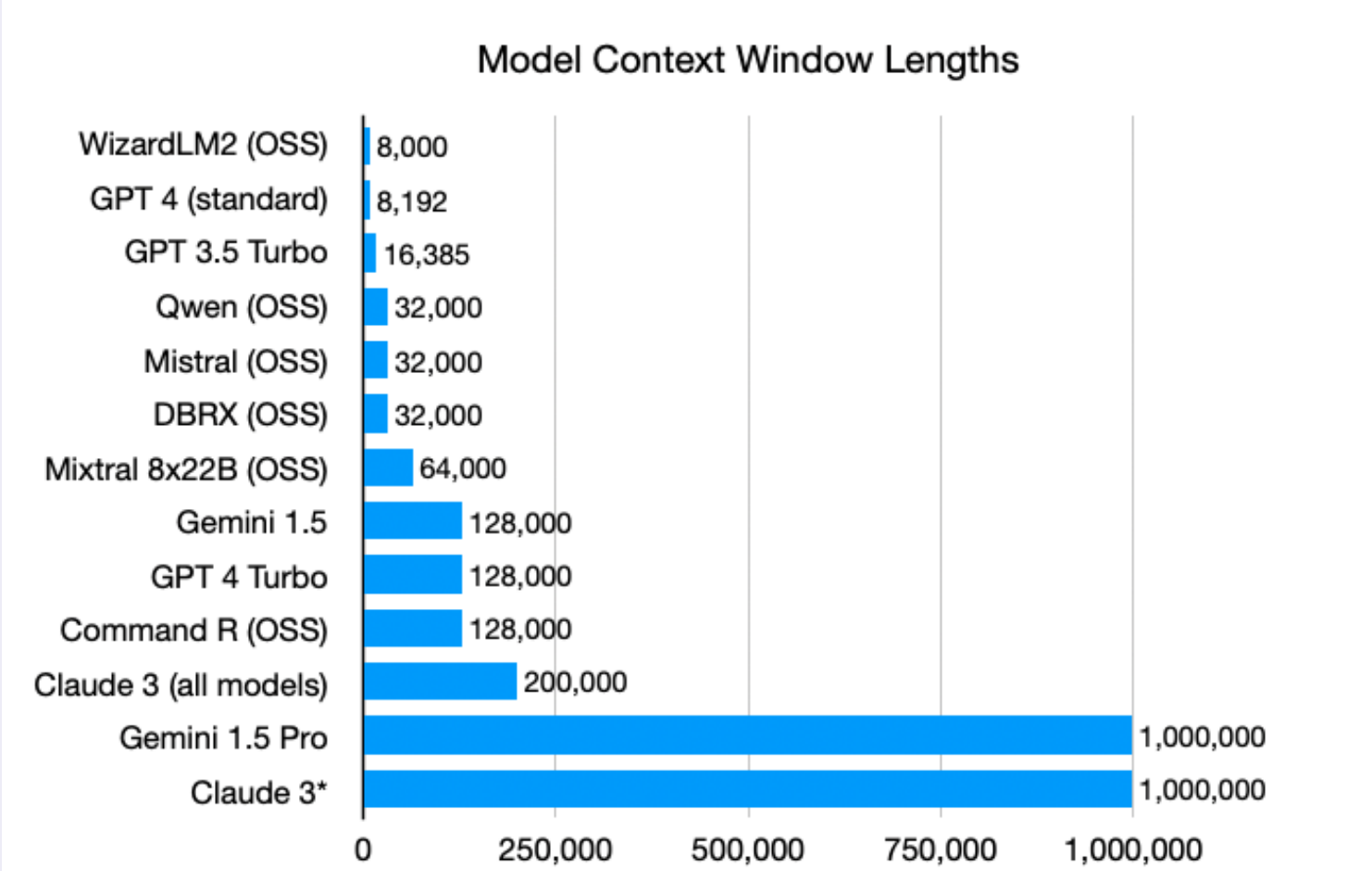
- Neural network trained on massive text/code datasets
- Learns patterns, syntax, and semantics
- Predicts next tokens based on context

- **Examples**:
  - GPT-5 (OpenAI)
  - Claude 4 (Anthropic)
  - Codex (OpenAI)
  - Code Llama (Meta)

- **Sources:**
  - **Public Repositories**: GitHub, GitLab, Bitbucket
  - **Documentation**: API docs, tutorials
  - **Stack Overflow**: Q&A pairs
  - **Technical Books**: Programming guides

Context Windows and Code

- **Context Window**: Amount of text an LLM can process at once
- **Why It Matters**:
  - Larger context = Better understanding
  - Can reference multiple files
  - Maintains conversation history



https://www.zenml.io/blog/bigger-isnt-always-better-the-case-for-rag-in-the-age-of-infinite-context

Prompt Engineering Techniques

1. **Be Specific**

   Vague: “Create a web server”

   Specific: “Create a Flask REST API with endpoints for CRUD operations on a User model with email, name, and age fields”

2. **Provide Context**

   I'm building a Django app for an e-commerce site. I need a model for Products that: (…)

3. **Specify Constraints**

   Create a Python function to sort a list of dictionaries. Requirements:

   Use built-in functions only (no external libraries)

   Handle empty lists gracefully (…)

4. **Request Explanations**

   Explain this regex pattern step by step:

   ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$

   Then suggest improvements for edge cases.

Two Key Capabilities:

- **Understanding** (Analysis):
  - Explain complex code
  - Identify bugs
  - Suggest optimizations
  - Review for security
- **Generation** (Synthesis):
  - Write new functions
  - Create boilerplate
  - Implement algorithms
  - Generate tests

Productivity Gains

Pearson

Reduced Cognitive Load

- **Less Time On**:
  - Remembering syntax
  - Looking up documentation
  - Typing repetitive code
  - Context switching

- **More Time For**:
  - Architecture decisions
  - Problem solving
  - Creative solutions
  - Code review

Democratizing Development

- Junior developers learn faster
- Non-native speakers get syntax help
- Career changers accelerate learning
- Domain experts prototype quickly

Democratizing Development

- Junior developers learn faster
- Non-native speakers get syntax help
- Career changers accelerate learning
- Domain experts prototype quickly

More diverse developer community!

Not a Replacement

AI Cannot Replace

❌ **Critical Thinking**: Architecture decisions ❌ **Domain Knowledge**: Business logic ❌ **Creativity**: Novel solutions ❌ **Judgment**: Trade-off decisions ❌ **Context**: Project-specific needs

AI is a tool, not a teammate

Security Concerns

Risks to Consider

1. Code Vulnerabilities

- SQL injection patterns
- XSS vulnerabilities
- Insecure configurations

2. Privacy Issues

- Code sent to external servers
- Potential data leakage
- IP considerations

3. Dependency Risks

- Outdated packages

Debugging AI-Generated Code

- **Difficult to Debug**:
    - Complex patterns
    - Unfamiliar idioms
    - Indirect approaches
    - Hidden assumptions

- **Best Practice**:
    - Understand before accepting
    - Prefer simpler solutions
    - Add your own comments
    - Test thoroughly

Beyond Autocomplete

Traditional AI: Suggests code **Agentic AI**: Completes tasks

Capabilities: - Multi-step problem solving - File system navigation - Command execution - Autonomous debugging - Iterative refinement

Examples: Cursor Agent, Windsurf Cascade, Claude Code

Golden Rules

- **Verify Everything**: Test all generated code
- **Understand Output**: Don’t use code you don’t understand
- **Maintain Skills**: Practice without AI
- **Review for Security**: Check for vulnerabilities
- **Document Changes**: Track AI contributions
- **Iterate**: Refine prompts for better results
- **Learn**: Use AI as educational tool
- **Stay Current**: Tools evolve rapidly
- **Be Skeptical**: Question suggestions
- **Keep Human Oversight**: Final decisions are yours

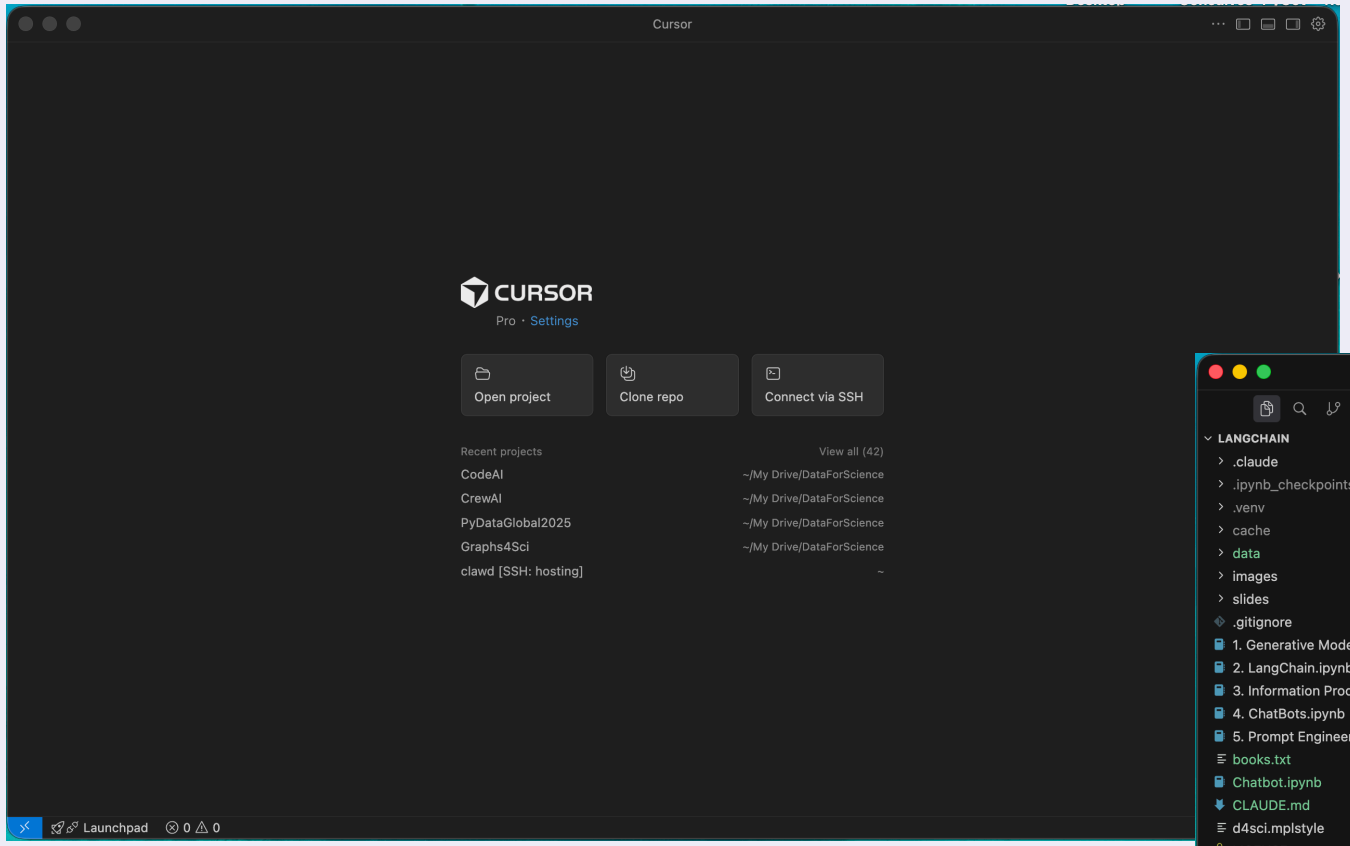
AI Assistants Fundamentals

https://github.com/DataForScience/CodeAI

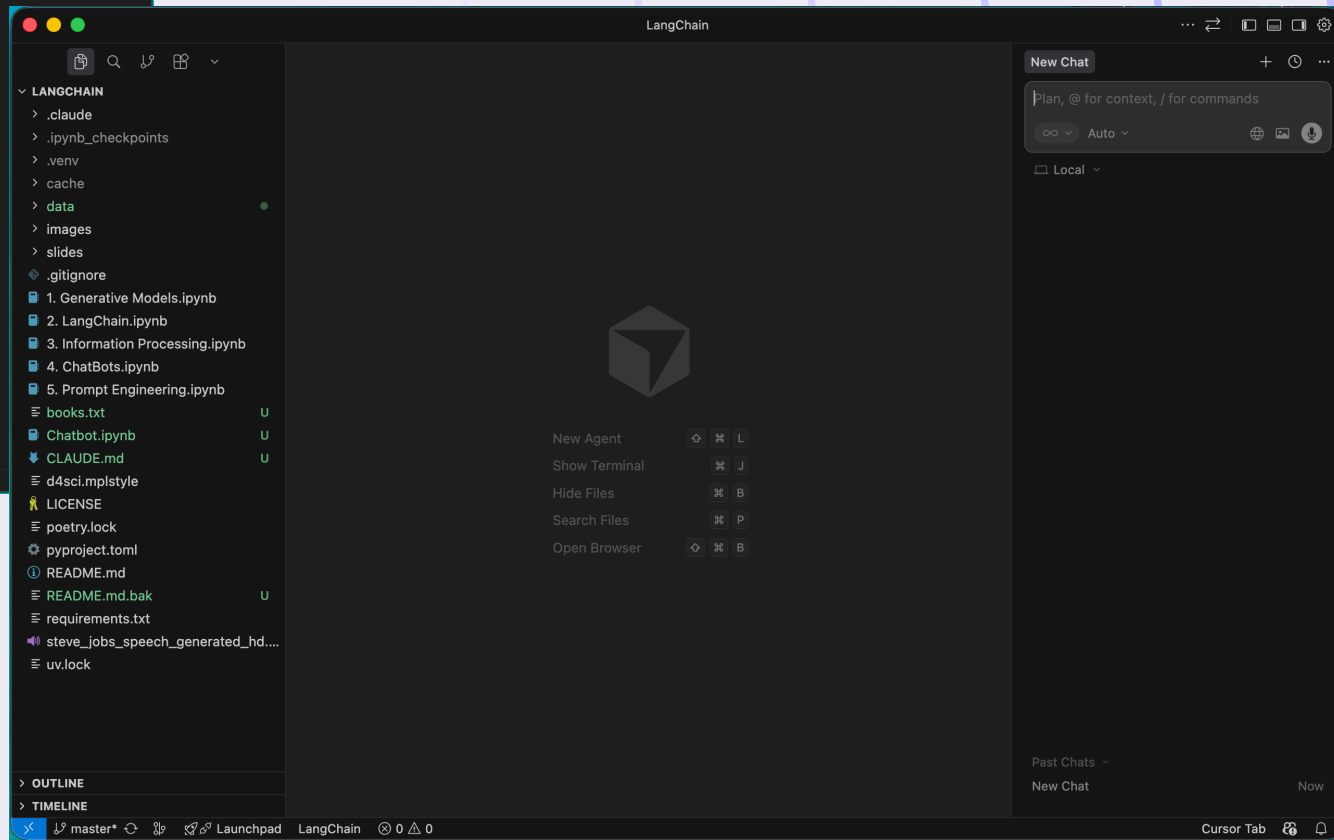
Pearson

2. Cursor

What is Cursor?

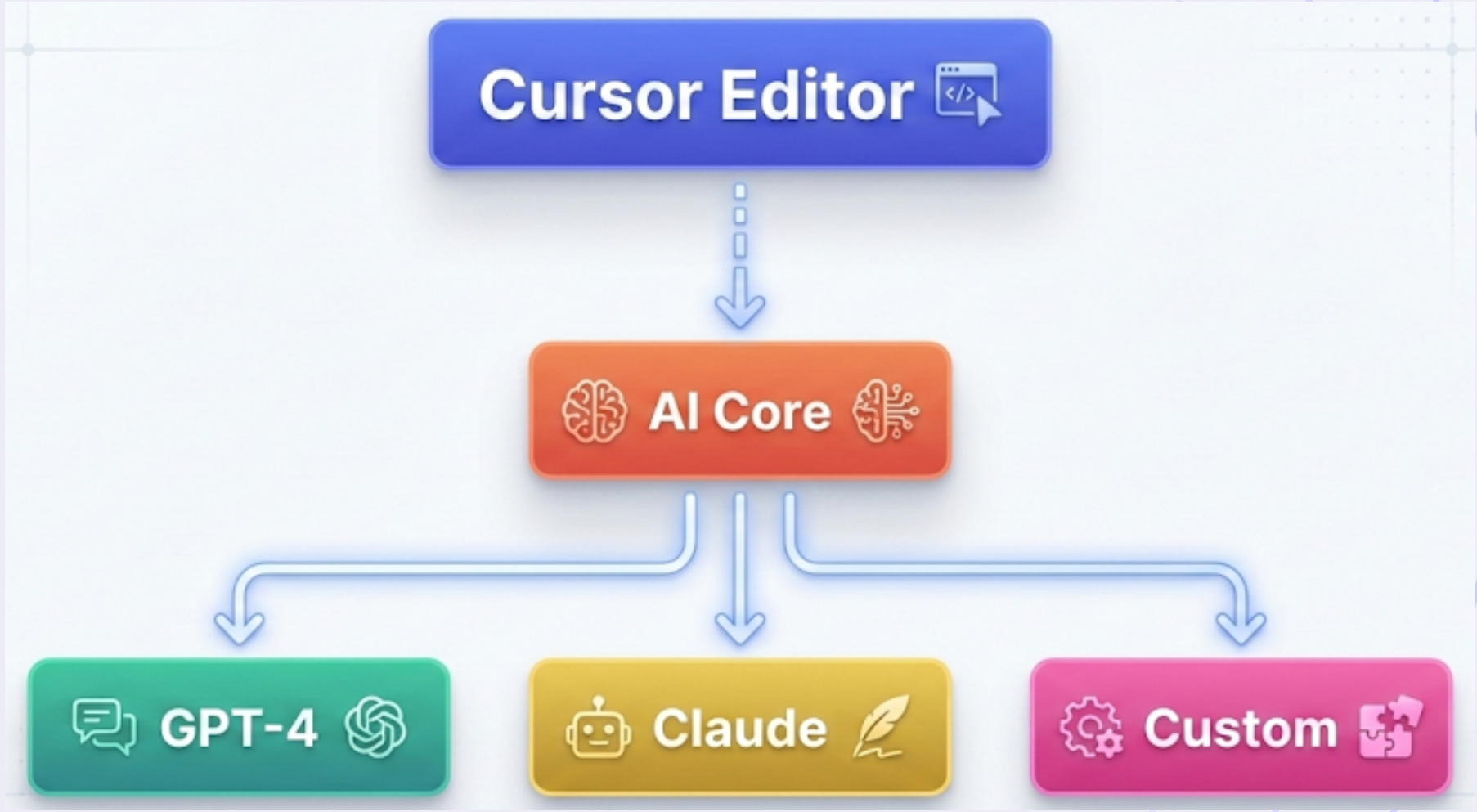


- A Fork of VS Code built from the ground up for AI-assisted development
- Native AI integrationChat with your codebase
- Agent mode for complex tasks
- Multi-file editing

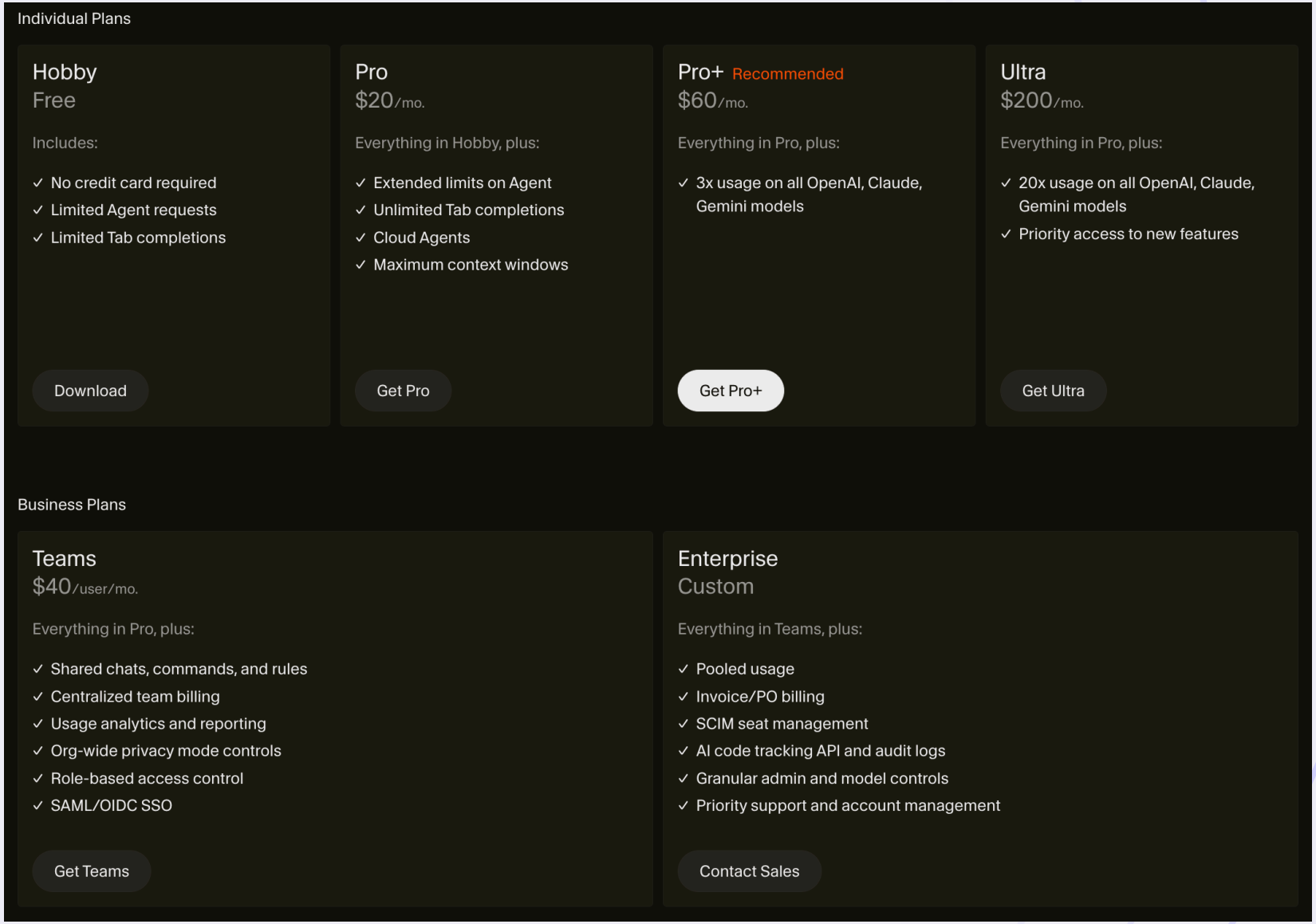


cursor.sh

Cursor Architecture



Pricing Model

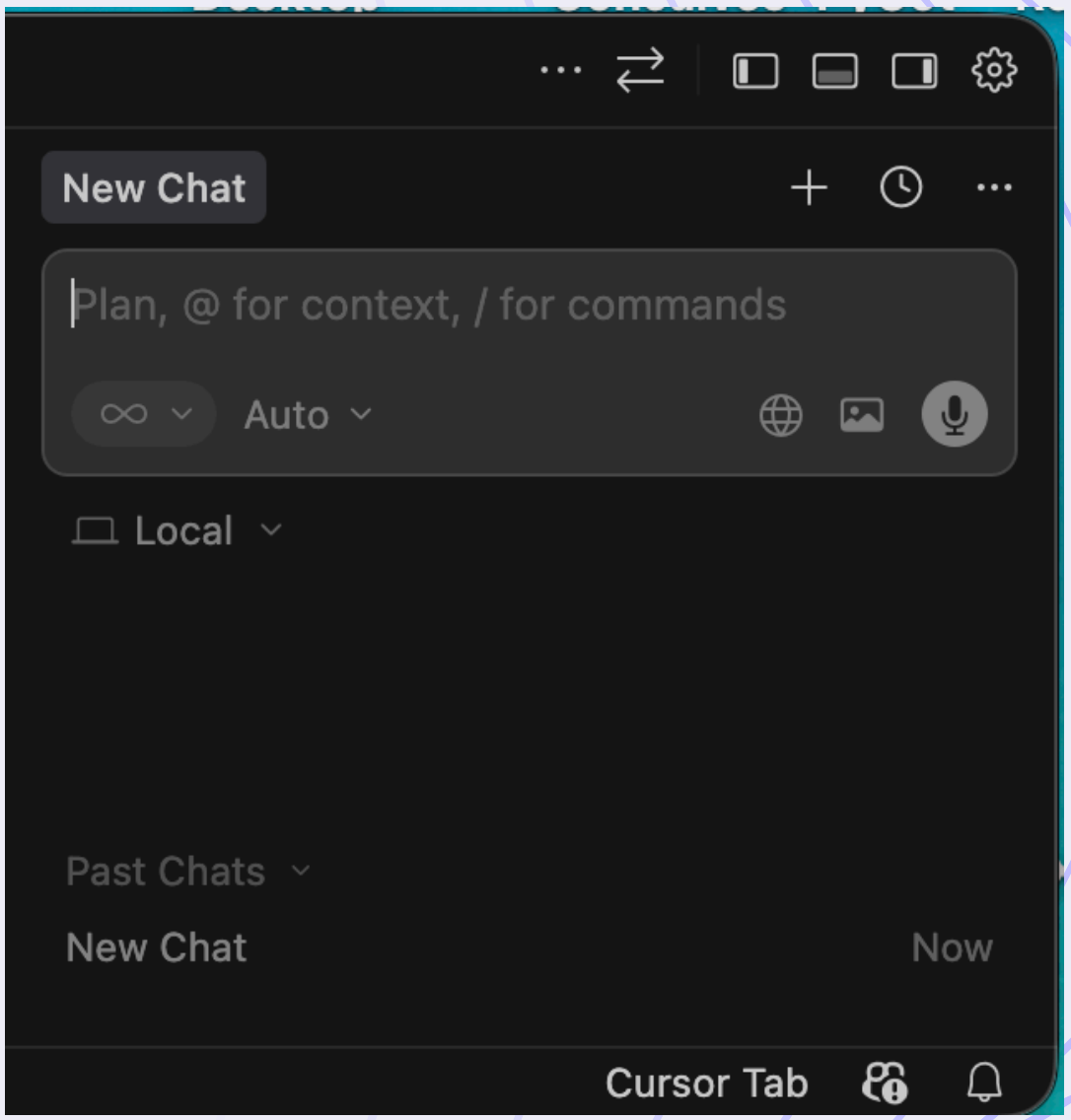


Pearson

What Cursor Sees

- **Automatic Context**:
  - Current file
  - Recently edited files
  - Imported modules
  - Project structure

- **Manual Context**:
  - `@filename`: Include specific file
  - `@folder`: Include directory
  - `@docs`: Include documentation
  - `@web`: Search web



Three Ways to Generate

- **Inline Generation** (Cmd/Ctrl + K)
  - Generate at cursor position
  - Quick snippets
- **Chat Generation** (Cmd/Ctrl + L)
  - Discuss and generate
  - Complex logic
- **Autocomplete**
  - As you type
  - Tab to accept

Inline Generation

Power Users

Use Case: Generate function body

```
def calculate_fibonacci(n: int) -> int:
    # Cmd+K: "recursive implementation with memoization"
```

Result: Complete function implementation

Demo Time! 🎬 *[Live demonstration of inline generation]*

What is Agent Mode?

Autonomous AI Coding

Traditional AI: Suggest code **Agent Mode**: Complete tasks

Capabilities: - Read multiple files - Edit multiple files - Run commands - Install dependencies - Debug errors - Iterative refinement

When to Use Agent Mode

Best Use Cases

Good For: - Multi-file refactoring - Adding features across codebase - Debugging complex issues - Setting up new projects - Migrating dependencies

❌ **Not Good For**: - Simple single-line edits - Learning new concepts - Exploring code

Activating Agent Mode

How to Start

Method 1: Chat panel

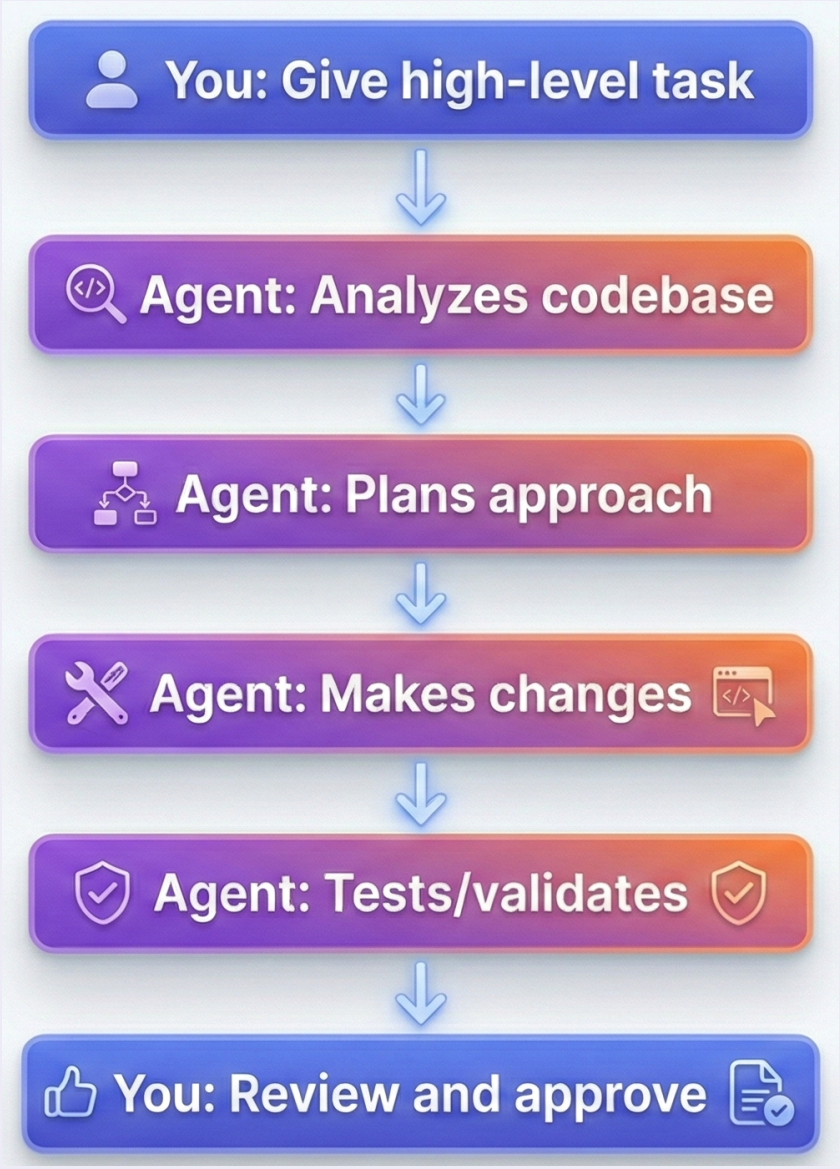
```
   Enable Agent Mode toggle
Then: "Add authentication to the app"
```

Method 2: Command Palette

```
Cmd/Ctrl + Shift + P
  Type: "Cursor: Enable Agent Mode"
```

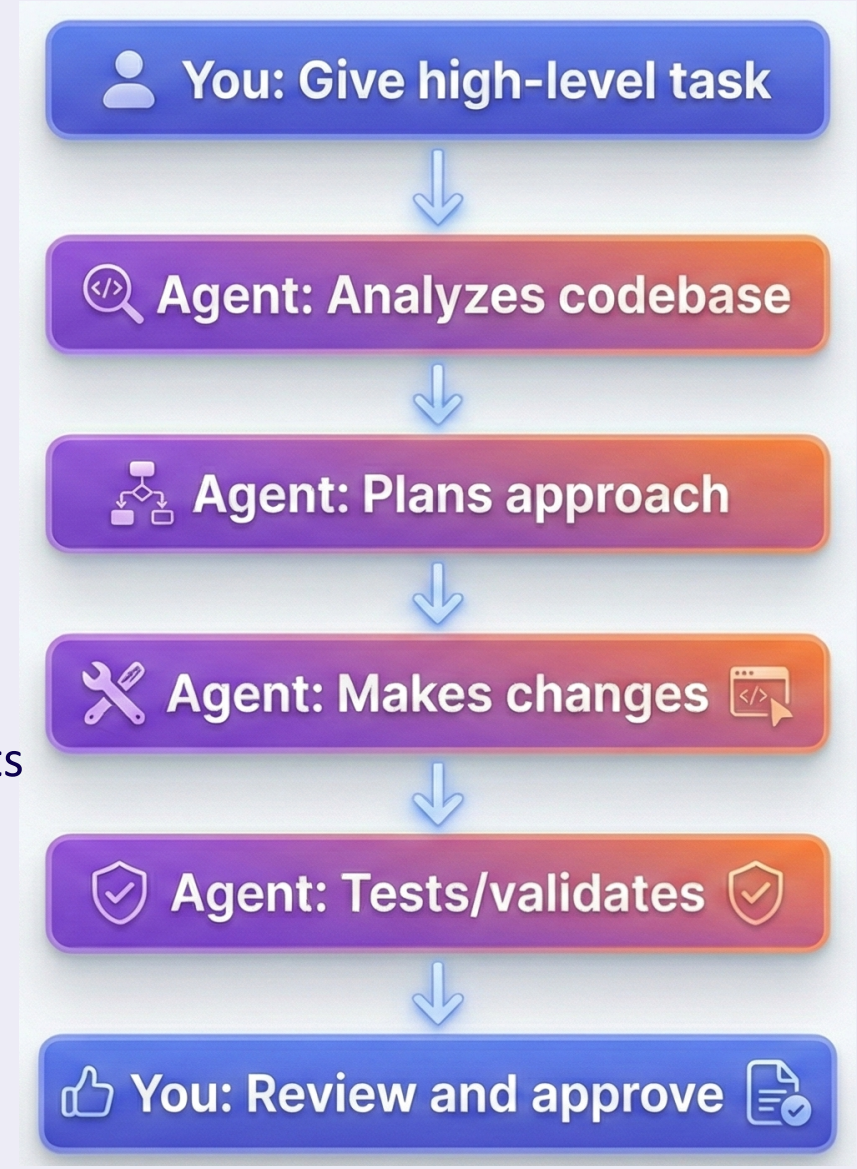
Visual Indicator: Agent icon in chat

Agent Mode Workflow



Agent Mode Workflow

- **Do**:
  - Give clear, specific tasks
  - Review all changes
  - Test before committing
  - Use for complex, well-defined tasks
- **Don’t**:
  - Accept changes blindly
  - Use for vague requirements
  - Run without supervision
  - Expect perfection



Prompt Pattern 1: Task-Based

Structure

```
[Action] [Target] [Requirements]

Example:
Refactor the login function to:
- Use async/await
- Add input validation
- Improve error messages
- Add logging
```

Prompt Pattern 2: Example-Based

Show, Don’t Just Tell

```
 Create a function similar to this:


@app.route('/users', methods=['GET'])
def get_users():
    return jsonify(users)


But for products with:
- Pagination
- Filtering by category
- Sorting by price
```

Prompt Pattern 3: Constraint-Based

Define Boundaries

```
Add caching to the API with these constraints:
- Use Redis
- Cache for 5 minutes
- Invalidate on POST/PUT/DELETE
- Only cache GET requests
- Add cache hit/miss metrics
```

Prompt Pattern 4: Error-Driven

Fix Issues

```
This code throws "Connection refused" error:

[paste code]

Fix it by:
1. Adding connection retry logic
2. Better error messages
3. Graceful degradation
```

.cursorrules File

Custom Instructions

Create: `.cursorrules` in project root

Example:

```
# .cursorrules

Always follow these conventions:
- Use type hints for all functions
- Docstrings in Google style
- Maximum line length: 88 (Black)
- Use pytest for tests
- Prefer composition over inheritance
```

Effect: Cursor follows these rules in all suggestions

Workflow 1: New Feature

Step-by-Step

1. Plan (Chat):

```
I need to add email notifications.
1.What's the best approach?
```

Generate (Agent):

```
2.Add email notifications using SendGrid:
  - Create email templates
3.- Add notification service
  - Trigger on user events
```

Test (Chat):

```
@notification.py Generate comprehensive tests
```

Workflow 2: Debugging

Systematic Approach

1. **Describe Problem**:

```
@app.py This endpoint returns 500 error
1.when user email is missing
```

Get Analysis:

Cursor identifies issue

2. Suggests fix

Apply Fix:

Review suggestion

3. Accept or refine

Add Test:

Workflow 3: Refactoring

Large-Scale Changes

Scenario: Code cleanup

Approach:

```
  Agent Mode:
Refactor this codebase to:
  - Extract repeated logic into utilities
  - Add type hints everywhere
  - Improve error handling
  - Add docstrings
  - Follow PEP 8
```

Result: Comprehensive refactoring

Workflow 4: Learning

Using Cursor to Learn

1. Understand Code:

```
@complex_algorithm.py
1.Explain this code step by step
```

Ask Questions:

```
2.Why use functools.lru_cache here?
What are alternatives?
```

Try Alternatives:

```
Show me 3 different ways to implement this
```

Golden Rules

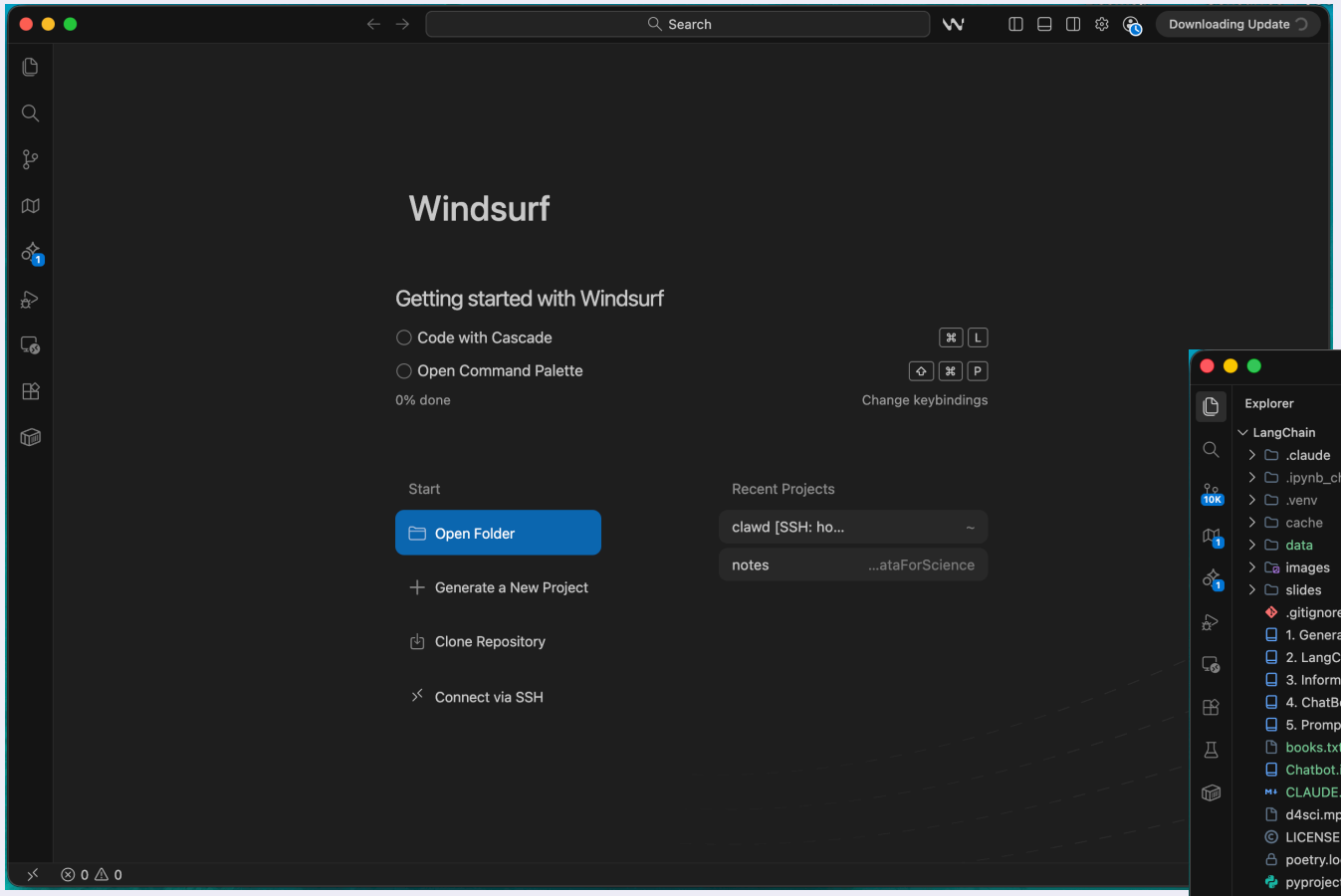
- Use .cursorrules for consistency
- Be specific in prompts
- Use @-mentions for context
- Test all generated code
- Use Agent Mode for complex tasks
- Review security implications
- Keep learning the fundamentals

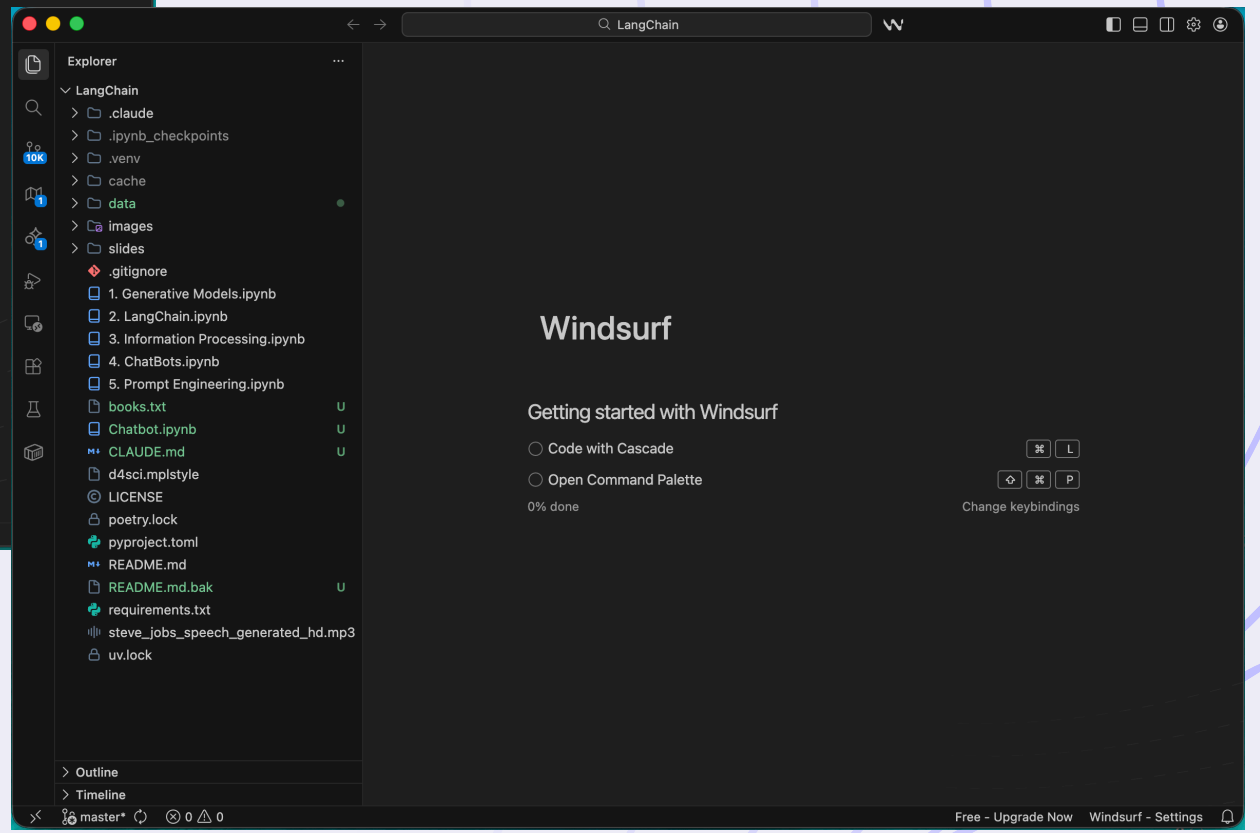
Cursor

https://github.com/DataForScience/CodeAI

3. Windsurf

What is Windsurf?





- AI-powered IDE by Codeium focused on developer flow state
- “Stay in flow, let AI handle the rest”

codeium.com/windsurf

What is Flow State?

- **Flow State**: Complete immersion in coding
  Deep focus - Peak Productivity - “Time disappears”
- **Traditional Interruptions**:
  - Switching to browser for docs
  - Copy-pasting from ChatGPT, Claude, Gemini, etc
  - Context switching
  - Manual file navigation
- **Windsurf Solution**: AI that doesn’t break your flow
- **Design Principles**
  - Anticipatory: Predicts your next move
  - Contextual: Understands your project
  - Non-Intrusive: Suggestions don’t break flow
  - Fast: Optimized for speed
  - Transparent: Shows reasoning

What Makes Windsurf Different

- **Cascade**:
  - Multi-step agentic AI
  - Searches entire codebase
  - Multi-file awareness
  - Executes commands

- **Flow**:
  - Ultra-fast autocomplete
  - Context-aware suggestions
  - Inline editing
  - Minimal latency

- **Command Palette**:
  - Natural language commands
  - Quick actions
  - Workflow automation

Pricing

FREE CURRENT

$0

Download

- 25 prompt credits/month
  Across leading models (OpenAI, Claude, Gemini, xAI, and more)
- Basic model access
- Fast Context trial access
- Unlimited Tab completions
- Unlimited inline edits
- Windsurf app previews

PRO POPULAR

$15 per month

Select plan

Everything in Free, plus:

- 500 prompt credits/month
- All Premium Models
- SWE-1.5 Model
- Full Fast Context access
- Add-on credits at $10/250 credits

Free trial available for first-time users only.

TEAMS

$30 per user / month

Select plan

Everything in Pro, plus:

- 500 prompt credits/user/month
- Add-on credits available for purchase
- Windsurf Reviews
- Centralized billing
- Admin dashboard with analytics
- Priority support
- Automated zero data retention
- SSO available for +$10/user/month

And more...

ENTERPRISE

Let's talk

View plan options

Everything in Teams, plus:

- 1,000 prompt credits/user/month
- Add-on credits available for purchase
- Role-Based Access Control (RBAC)
- SSO + Access control features

If the org has more than 200 users:

- Highest priority support
- Dedicated account management
- Hybrid deployment option

Pearson

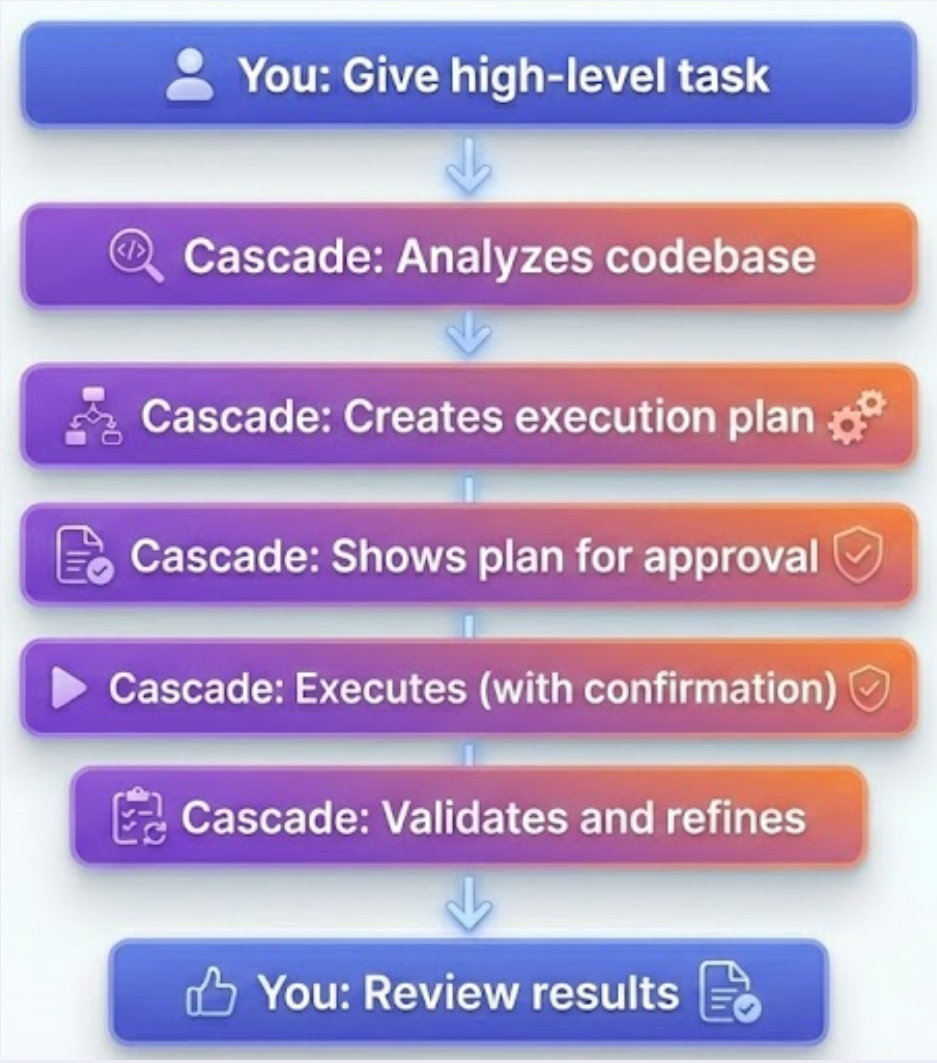
.windsurfrules

```
# Project conventions for Windsurf

Style:
- Python: Black formatting, 88 char line length
- Use type hints
- Docstrings in Google style

Architecture:
- MVC pattern
- Repository pattern for data access
- Service layer for business logic

Testing:
- pytest for Python
- 80% coverage minimum
```

What is Cascade?

- A Multi-step autonomous AI agent for Windsurf
- **Philosophy**: “Cascading” through your codebase
- **Capabilities**:
    - Analyze entire codebase
    - Make multi-file changes
    - Run terminal commands
    - Install dependencies
    - Debug iteratively
    - Learn from errors

How Cascade Works

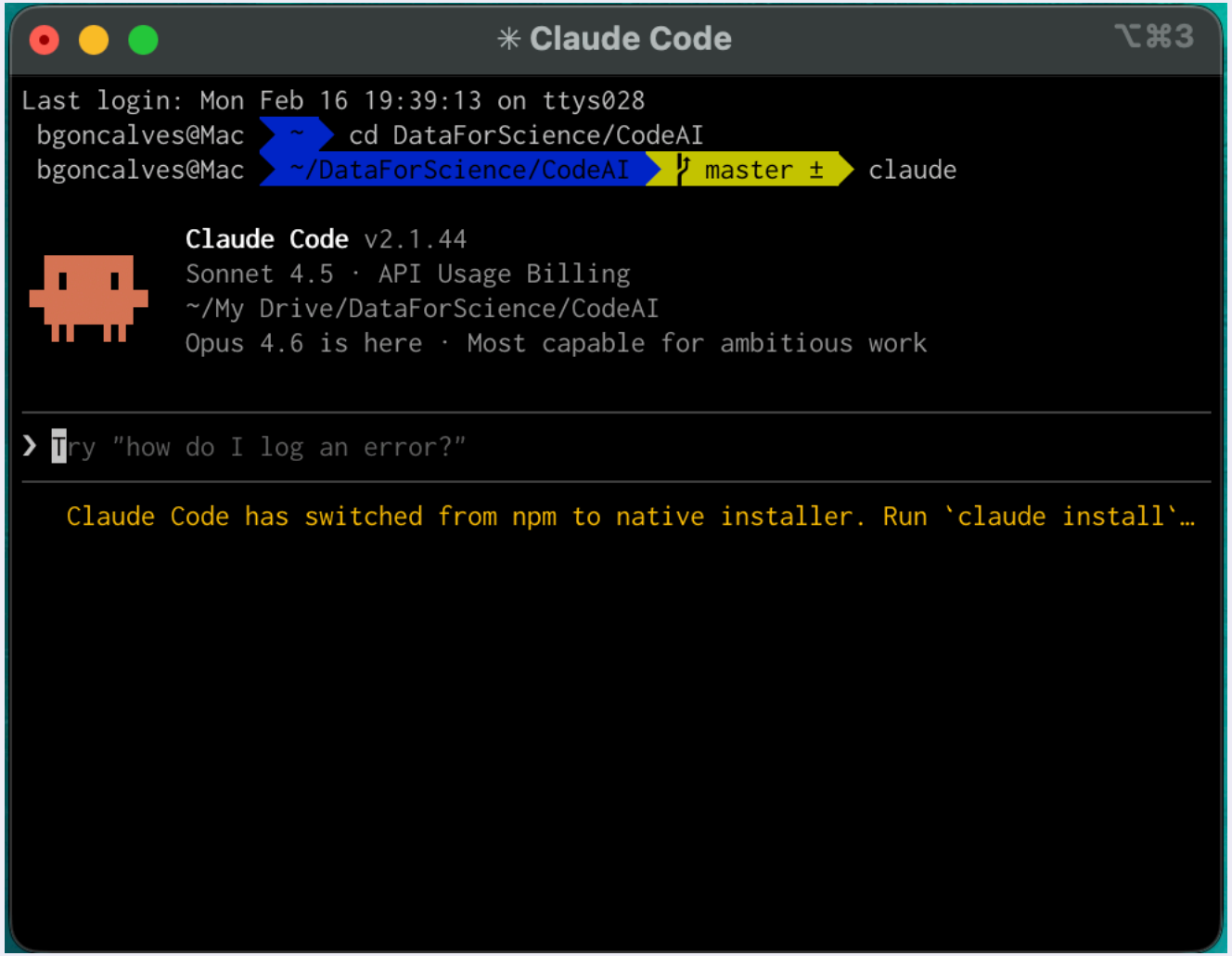


Windsurf

https://github.com/DataForScience/CodeAI

4. Claude Code

What is Claude Code?



- Command-line interface for Claude AI focused on software development
- “Powerful AI assistance without leaving your terminal”

claude.ai/code

The Terminal Advantage

- **For Developers Who**:
    - Live in the terminal
    - Use vim/neovim/emacs
    - Prefer keyboard over mouse
    - Work on remote servers
    - Script their workflows
    - Value minimalism

- **Benefits**:
    - Lightweight (no IDE overhead)
    - Fast startup
    - Scriptable
    - SSH-friendly
    - Integrates with existing tools

Why Claude?

- **Capabilities**:
  - Long context (200K tokens)
  - Advanced reasoning
  - Code understanding
  - Architectural thinking
  - Nuanced responses
  - Tool use

- **Best At**:
  - Complex problem solving
  - System design
  - Code review
  - Explaining code
  - Multi-step reasoning

Claude Code Features

- **Code Generation**:
  - Create files
  - Edit existing code
  - Multi-file projects
  - Follow best practices
- **Code Understanding**:
  - Explain complex code
  - Architectural analysis
  - Identify patterns
  - Security review
- **Assistant Features**:
  - Terminal integration
  - Git integration
  - File system access
  - Command execution

Pricing

Free

Try Claude

$0

Free for everyone

Try Claude

- Chat on web, iOS, Android, and on your desktop
- Generate code and visualize data
- Write, edit, and create content
- Analyze text and images
- Ability to search the web
- Create files and execute code
- Unlock more from Claude with desktop extensions
- Connect Slack and Google Workspace services
- Integrate any context or tool through connectors with remote MCP
- Extended thinking for complex work

Pro

For everyday productivity

$17

Per month with annual subscription discount ($200 billed up front). $20 if billed monthly.

Try Claude

Everything in Free, plus:

- More usage*
- Includes Claude Code and Cowork
- Access to unlimited projects to organize chats and documents
- Access to Research
- Memory across conversations
- Ability to use more Claude models
- Claude in Excel

Max

Get the most out of Claude

From $100

Per person billed monthly

Try Claude

Everything in Pro, plus:

- Choose 5x or 20x more usage than Pro*
- Higher output limits for all tasks
- Early access to advanced Claude features
- Priority access at high traffic times
- Claude in PowerPoint (research preview)

Configuration

Key Settings

Config File: `~/.claude/config.json`

```
{
  "model": "claude-sonnet-4-5",
  "editor": "vim",
  "auto_commit": false,
  "memory": true,
  "style": "concise"
}
```

CLAUDE.md

```
# CLAUDE.md

## Project Context

This is a Flask API for task management.

## Architecture

- MVC pattern
- SQLAlchemy ORM
- Blueprint-based routes

## Conventions

- Type hints required
- Pytest for tests
- Black formatting
- 88 character lines

## Commands

- Run: `python app.py`
- Test: `pytest tests/`
- Format: `black .`
```

Special Commands

- **File Operations**:
  - `/read file.py` - Read file
  - `/edit file.py` - Edit file
  - `/write file.py` - Create file

- **Project Context**:
  - `/ls` - List files
  - `/tree` - Show structure
  - `/git status` - Git status

- **Session**:
  - `/clear` - Clear conversation
  - `/save session.md` - Save chat
  - `/help` - Show help

Programmatic Usage

Example Script: `review.sh`

```
#!/bin/bash

echo "Reviewing changes..."

claude --non-interactive \
  --format json \
  "/git diff" \
  "List any issues as JSON"

# Parse JSON output
# Fail CI if critical issues found
```

Memory and Context

- **Persistent Context**

- **Claude Remembers**:
    - Project conventions
    - Previous discussions
    - Architecture decisions
    - Your preferences

- **Managed Automatically**:
    - Stored in `.claude/memory`
    - Loaded each session
    - Pruned intelligently

Aliases and Shortcuts

Config: `~/.claude/config.json`

```
{
  "aliases": {
    "review": "Review this for bugs and best practices",
    "test": "Generate pytest tests with fixtures",
    "doc": "Add comprehensive docstrings",
    "opt": "Optimize for performance"
  }
}
```

Usage: `/review @file.py`

Project Templates

Create: `.claude/templates/`

Example: `api_endpoint.md`

```
Create a Flask API endpoint:
- Route: {{ route }}
- Method: {{ method }}
- Parameters: {{ params }}
- Validation: Use marshmallow
- Error handling: Custom exceptions
- Tests: pytest with fixtures
- Docs: OpenAPI spec
```

Use: `/template api_endpoint route=/users method=GET`

Claude Code

https://github.com/DataForScience/CodeAI

Pearson

5. Comparative Analysis & Future Trends

Quick Recap

- **Cursor**:
  - AI-first code editor
  - VS Code fork
  - Agent mode
  - Multiple models
- **Windsurf**:
  - Flow-state IDE
  - Cascade mode
  - Speed optimized
  - Codeium-powered
- **Claude Code**:
  - Terminal AI assistant
  - CLI interface
    Architecture focus
  - Scriptable

Quick Recap

- **Cursor**:
  - AI-first code editor
  - VS Code fork
  - Agent mode
  - Multiple models
- **Windsurf**:
  - Flow-state IDE
  - Cascade mode
  - Speed optimized
  - Codeium-powered
- **Claude Code**:
  - Terminal AI assistant
  - CLI interface
    Architecture focus
  - Scriptable

Feature	Cursor	Windsurf	Claude Code
Interface	GUI (IDE)	GUI (IDE)	CLI (Terminal)
Base	VS Code	Custom	Terminal
AI Agent	Agent Mode	Cascade	Conversational
Autocomplete	Yes	Yes (fastest)	No
Chat	In-editor	In-editor	Terminal
Multi-file	Yes	Yes	Yes (manual)
Git Integration	GUI	GUI	CLI
Remote Work	Possible	Possible	Excellent
Scripting	Limited	Limited	Native

Model Comparison

AI Models Available

Cursor: - GPT-4 / GPT-4 Turbo - Claude 3.5 Sonnet - GPT-3.5 - User choice per request

Windsurf: - Codeium models (proprietary) - Claude 3.5 Sonnet - GPT-4 (Pro) - Optimized for speed

Claude Code: - Claude Haiku (Free) - Claude Sonnet (Pro) - Claude Opus (Team) - Single model per session

Performance Metrics

Speed and Responsiveness

Autocomplete Latency: - Windsurf: ~50ms (fastest) - Cursor: ~100ms - Claude Code: N/A (no autocomplete)

Agent Response Time: - Windsurf Cascade: Fast - Cursor Agent: Medium - Claude Code: Medium-Slow (thorough)

Startup Time: - Claude Code: Instant - Cursor: 3-5 seconds - Windsurf: 3-5 seconds

Pricing Comparison

Cost Analysis

Tier	Cursor	Windsurf	Claude Code
Free	Limited	Generous	Basic
Pro	$20/mo	$15/mo	$20/mo
Team	$40/user	Custom	$30/user
Features	500 fast	Unlimited*	5x usage

Context Handling

How Each Manages Context

Cursor: - Automatic file tracking - @-mentions for specific files - Recent files included - Good for exploration

Windsurf: - Intelligent codebase search - Automatic dependency tracking - Fast indexing - Best for large projects

Claude Code: - Manual @-mentions required - 200K token context window - CLAUDE.md for project info - Best for focused tasks

Code Generation Quality

Comparison of Output

All Three: - High quality code generation - Follow best practices - Handle complex logic

Differences: - **Cursor**: Most flexible (model choice) - **Windsurf**: Fastest generation - **Claude Code**: Best for architecture

Subjective: Similar quality overall

Strategy 1: Single Tool Approach

Commit to One

Philosophy: Master one tool deeply

Pros: - Become expert - Muscle memory - Consistent workflow - Lower cost

Cons: - Miss other strengths - Tool lock-in - Limited flexibility

Best For: Beginners, tight budgets

Strategy 2: Phase-Based Usage

1. Planning → Claude Code
   - Architecture discussions
   - Design decisions

2. Implementation → Windsurf
   - Fast coding
   - Flow state

3. Review → Claude Code
   - Code review
   - Security check

4. Documentation → Cursor
   - Visual editing
   - Screenshots

Strategy 3: Task-Based Selection

Choose by Task Type

Quick Tasks → Windsurf - Single features - Bug fixes - Refactoring

Complex Tasks → Cursor - Multi-step features - Learning - Exploration

Reviews & Planning → Claude Code - Code review - Architecture - Documentation

Strategy 4: Environment-Based

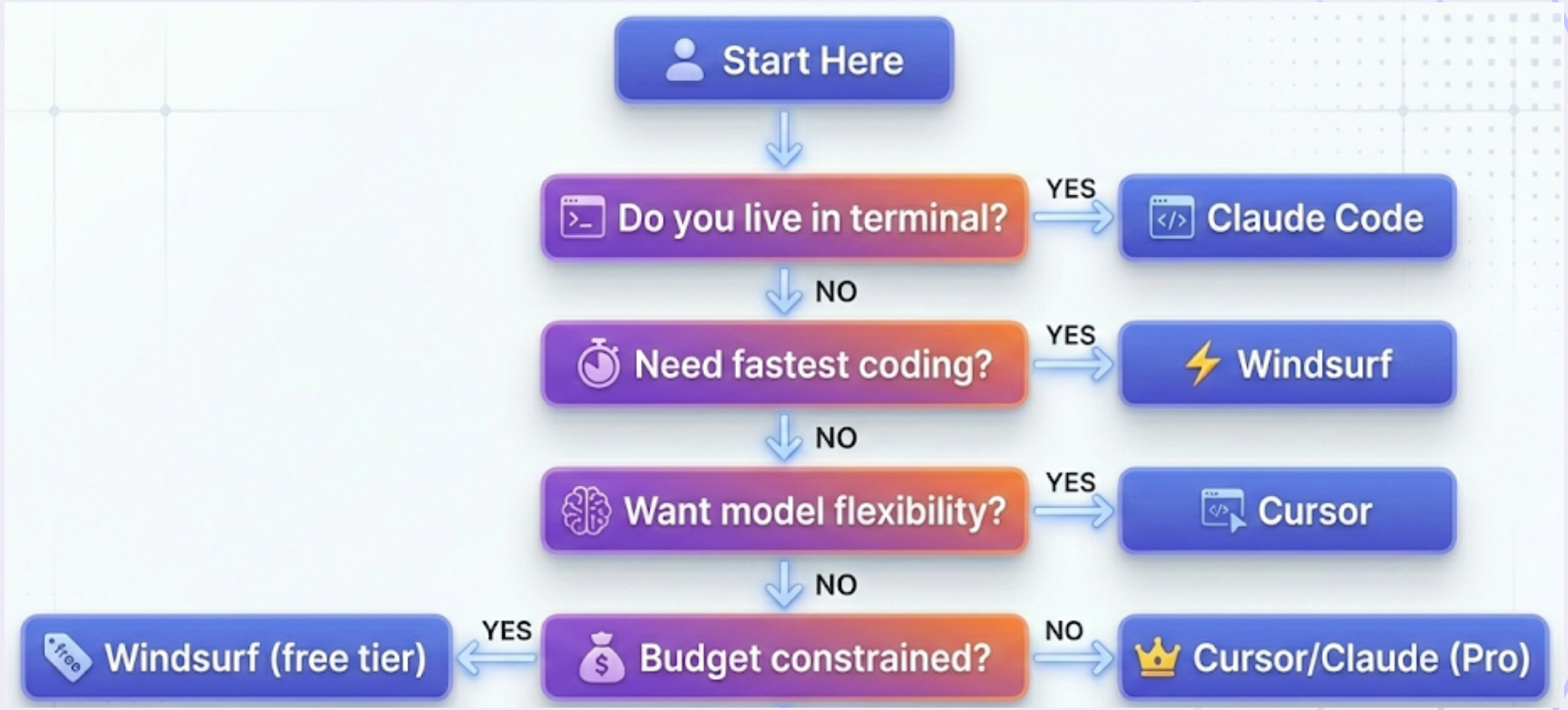
Choose by Where You Work

Local Machine → Cursor or Windsurf - Full IDE features - Visual interface - Comfortable setup

Remote Servers → Claude Code - Lightweight - SSH-friendly - No GUI needed

CI/CD → Claude Code - Scriptable - Automated - Fast

Decision Tree



Decision Tree

Universal Principles

1. **Be Specific**: Clear prompts = better results
2. **Review Everything**: AI makes mistakes
3. **Test Thoroughly**: Generated code needs validation
4. **Iterate**: Refine prompts and code
5. **Context Matters**: Provide relevant information
6. **Stay Current**: Tools evolve rapidly
7. **Security First**: Check for vulnerabilities
8. **Document**: AI can help, but verify
9. **Version Control**: Commit frequently
10. **Keep Learning**: Maintain fundamentals

Universal Principles

1. **Be Specific**: Clear prompts = better results
2. **Review Everything**: AI makes mistakes
3. **Test Thoroughly**: Generated code needs validation
4. **Iterate**: Refine prompts and code
5. **Context Matters**: Provide relevant information
6. **Stay Current**: Tools evolve rapidly
7. **Security First**: Check for vulnerabilities
8. **Document**: AI can help, but verify
9. **Version Control**: Commit frequently
10. **Keep Learning**: Maintain fundamentals

Comparative Analysis

https://github.com/DataForScience/CodeAI

Pearson

Question

- How was the technical level?
  - 1 - Too Low (too many details)
  - 2 - Low
  - 3 - Just Right
  - 4 - High
  - 5 - Too High (not enough details)

Question

- How was the level of the Python code and explanations?
  - 1 - Too Low (too many details)
  - 2 - Low
  - 3 - Just Right
  - 4 - High
  - 5 - Too High (not enough details)

Contacts

Bruno Gonçalves

https://data4sci.com

info@data4sci.com

https://data4sci.com/call

https://www.linkedin.com/in/bmtgoncalves/

Bruno Gonçalves is an author, public speaker, corporate trainer, and consultant specializing in Generative AI, Blockchain Analytics, and Machine Learning. He founded Data For Science, Inc in 2009 to help individuals and companies solve their data driven problems.

Coming Up:

- **CrewAI for Production-Ready Multi-Agent Systems**
  Feb 19, 2026 10am-2pm (PST)
- **Gemini API with VertexAI for Developers**
  Mar 18, 2026 10am-2pm (PST)
- **LangChain for Generative AI Pipelines**
  Apr 22, 2026 10am-2pm (PST)
- **Claude API for Python Developers**
  Apr 29, 2026 10am-2pm (PST)

Natural Language Processing (NLP) Fundamentals, 3rd Edition

https://bit.ly/NLP_LL_v3

Begin

Complete this course and earn a badge!

6h 14m • 11 sections

Course Outline

Natural Language Processing LiveLessons covers the fundamentals and some of the more advanced aspects of Natural Language Processing in a simple and intuitive way, empowering you to add NLP to your toolkit. Using the powerful NLTK package, it gradually moves from the basics of text representation, cleaning, topic detection, regular

Python Data Visualization: Create impactful visuals, animations and dashboards

https://bit.ly/DataViz_LL

Begin

Complete this course and earn a badge!

6h 36m • 11 sections

Sneak Peek

The Sneak Peek program provides early access to Pearson video products and is exclusively available to subscribers. Content for titles in this program is made available throughout the development cycle, so products may not be complete, edited, or finalized, including video post-production editing.

Information visualization, as David McCandless aptly puts it, is a form of "knowledge compression." Our highly evolved visual processing system enables us to efficiently handle vast amounts of information. Visualization's

Course Outline

Times Series Analysis for Everyone

https://bit.ly/Timeseries_LL

Begin

Complete this course and earn a badge!

6h • 14 sections

Times Series Analysis for Everyone LiveLessons covers the fundamental tools and techniques for the analysis of time series data. These lessons introduce you to the basic concepts, ideas, and algorithms necessary to develop your own time series applications in a step-by-step, intuitive fashion. The lessons follow a gradual progression, from the more specific to the more abstract, taking you from the very basics to some of the most recent and sophisticated algorithms by leveraging the statsmodels, arch, and Keras state-of-the-art models.

Course Outline

END