AI-Powered Coding Assistants in 2025: A Comparative Analysis of GitHub Copilot, Windsurf, and Cursor

1. Introduction

The landscape of software development has been irrevocably altered by the advent of AI-powered coding assistants. These tools, leveraging sophisticated large language models (LLMs), promise to augment developer productivity, streamline workflows, and even democratize coding to some extent. As of June 2025, the market is maturing, with several key players offering increasingly advanced features. This report provides a comprehensive comparison of three prominent AI coding assistants: GitHub Copilot, Windsurf (formerly Codeium), and Cursor. The analysis focuses on their latest features, pricing structures, user sentiment, capabilities within Jupyter Notebook environments, and a speculative look at which tool might best align with the capabilities of an advanced AI like Google's Gemini. The objective is to furnish developers, technical leads, and organizations with a detailed understanding to inform their choice of AI coding partner.

2. Methodology

This report synthesizes information from official product documentation, changelogs, pricing pages, user reviews, community discussions, and technical articles, all referencing features and statuses as of June 11, 2025, or the latest available information leading up to this date. The analysis compares the tools across several key dimensions: core features, agentic capabilities, codebase understanding, pricing models, user feedback, and specific integrations like Jupyter Notebook support. A qualitative assessment of which tool a hypothetical human version of Google's Gemini might prefer is also included, based on Gemini's known architectural strengths and capabilities.

3. Overall Feature Comparison (as of June 11, 2025)

As AI coding assistants evolve, their feature sets are becoming increasingly sophisticated, moving far beyond simple code completion. GitHub Copilot, Windsurf, and Cursor each offer a distinct suite of capabilities aimed at enhancing developer productivity.

3.1. Core Coding Assistance (Autocomplete, Inline Suggestions, Chat)

- **GitHub Copilot:** Offers real-time code suggestions and completions using a base model, with unlimited completions for paid tiers. It features robust chat capabilities within the IDE, GitHub platform, and mobile applications. Copilot Chat allows users to explain code, ask programming questions, and generate code snippets using natural language prompts, supported by / commands and @ participants for context (e.g., #file, #codebase,

#selection). It supports different operational modes: "ask" for queries, "edit" for guided modifications, and "agent" for more autonomous tasks.

- **Windsurf:** The Windsurf Editor, a fork of VS Code, features "Windsurf Tab," which tracks command history, clipboard, and agent actions to provide smarter, more relevant suggestions. Its standard autocomplete suggests code as you type, while "Supercomplete" analyzes code context before and after the cursor to predict next moves, showing suggestions in a diff box. The Cascade agent offers "Write" and "Chat" modes. Cascade can parse web pages, utilize a local index of the entire codebase, leverage "Memories" for context persistence, and follow user-defined "AI Rules". An integrated terminal allows for AI-assisted command generation and execution.
- **Cursor:** Built as an AI-first code editor (also a VS Code fork), Cursor provides highly predictive tab completion that anticipates the user's next edit across multiple lines. Its chat functionality is deeply integrated, possessing awareness of the entire codebase, allowing it to answer questions, refer to specific files or documentation, and edit code based on natural language instructions. A notable feature from its June 4, 2025, update is the ability to render visualizations like Mermaid diagrams and Markdown tables directly within the chat interface. Cursor also supports chat tabs, enabling users to manage multiple AI conversations in parallel.

3.2. Agentic Capabilities (Multi-file edits, Task Automation, Debugging)

- **GitHub Copilot:** The "Agent mode" in Copilot is designed for end-to-end task completion. It can search the workspace for relevant context, edit multiple files, check for errors, and run terminal commands (with user permission). "Copilot Edits" is another feature for applying changes across multiple files, although some user accounts suggest it can be slow or prone to errors.
- **Windsurf:** The "Cascade" agent is marketed as an AI that "codes, fixes and thinks 10 steps ahead". A significant recent addition is "Planning Mode" (Wave 10, June 10, 2025), which helps manage complex, long-running tasks by generating an editable plan.md file. This file outlines the AI's intended actions, allowing for human-AI collaboration on the task's structure. Cascade can also automatically detect and fix lint errors it generates.
- **Cursor:** Cursor features a "Background Agent" for handling remote coding tasks and an "Agent in Jupyter Notebooks". A key innovation is "BugBot," introduced in its 1.0 release (June 4, 2025), which automatically reviews pull requests on GitHub, identifies potential bugs and issues, and allows users to click "Fix in Cursor" to address them with a pre-filled prompt. The agent can also edit entire classes or functions based on a simple natural language prompt. Users have reported leveraging Cursor for AI-assisted debugging and automated commit message generation.

3.3. Codebase Understanding & Context Management

- **GitHub Copilot:** Employs semantic indexing to better understand repository context. Agent mode actively searches the workspace to gather relevant context for its operations. Users can explicitly attach files, symbols, and selections to chat prompts to refine Copilot's understanding.
- **Windsurf:** Touts an industry-leading context awareness engine for deep codebase

understanding. Its "Memories" feature allows the Cascade agent to persist context across sessions and conversations. "Rules" can be defined by users to guide the AI's behavior, such as adhering to specific framework patterns. A local indexing engine provides context from the entire codebase, not just recently accessed files.
- **Cursor:** Is designed to "know your codebase," enabling it to answer questions based on the project's content and refer to specific files or documentation. The "Memories" feature, introduced in beta with Cursor 1.0, allows the AI to recall facts from previous conversations and apply them in future interactions within the same project. Users can define global ignore patterns and opt to include the project's directory structure in the context provided to the AI. Furthermore, .cursorrules files can be used to specify coding preferences and standards for the AI to follow.

3.4. Refactoring & Code Modernization

- **GitHub Copilot:** Includes features specifically aimed at refactoring existing code, migrating projects (e.g., to different languages or frameworks), and modernizing legacy codebases.
- **Windsurf:** The Cascade agent is built to handle complex codebases, which is essential for effective refactoring and modernization efforts.
- **Cursor:** Allows developers to update entire classes or functions using simple natural language prompts, facilitating large-scale refactoring. Some reviews highlight its capability for advanced refactoring, including rewriting entire codebases.

3.5. Testing Assistance

- **GitHub Copilot:** Provides a "write tests" feature. Specific chat commands like /tests (to generate tests) and /fixTestFailure (to suggest fixes for failing tests) are available.
- **Windsurf:** The Cascade agent is described as capable of fixing test failures "before you even write the test," suggesting a proactive approach to ensuring code quality.
- **Cursor:** Its agent mode can be instructed to add tests to the codebase as part of its task execution.

3.6. Extensibility & Customization

- **GitHub Copilot:** Offers several layers of customization, including personal and repository-level custom instructions. A significant development is the GitHub Copilot Extensions framework, allowing developers to build and share custom tools, agents, and skillsets that integrate with Copilot. Furthermore, Copilot Studio enables the creation and deployment of custom engine agents that can be published to Microsoft Copilot Chat, extending its capabilities with domain-specific intelligence.
- **Windsurf:** Supports the Model Context Protocol (MCP), enabling integration with custom tools and services. Users can access curated MCP servers for one-click setup with services like Figma, Slack, and PostgreSQL. Windsurf also allows users to create "Custom Workflows" (saved prompts callable via slash commands) and "File-Based Rules" (granular, always-on rules or rules attached to file globs) to tailor Cascade's behavior.
- **Cursor:** Allows users to import their existing VS Code extensions, themes, and keybindings, ensuring a familiar environment. It supports MCP with one-click install and

OAuth for server authentication. A beta feature for "Custom modes" allows users to compose new operational modes with specific tools and prompts. Project-specific configurations can be managed using .cursorrules files.

3.7. Unique/Standout Features

- **GitHub Copilot:**
    - **Copilot Spaces:** A collaborative feature for teams to organize and share context, and work together with AI.
    - **Copilot Vision:** Allows users to attach images (e.g., screenshots, UI mockups) to chat prompts for multimodal interaction, enabling tasks like debugging from a screenshot or generating code from a design.
    - **Deep GitHub Integration:** Beyond chat, this includes features like generating PR summaries and interacting with repository data.
- **Windsurf:**
    - **Planning Mode:** A collaborative planning primitive where the AI generates and updates a plan.md file for complex tasks, allowing human oversight and modification.
    - **SWE-1 Models:** Windsurf's own family of "Frontier Models" designed for advanced coding tasks.
    - **Deploys:** Integrated functionality to deploy applications to Netlify with a single prompt (beta feature).
    - **Windsurf Reviews:** A feature for teams (Teams and Enterprise SaaS plans) that uses a GitHub app for AI-assisted code review and PR description editing.
- **Cursor:**
    - **BugBot:** An automated AI agent that reviews GitHub pull requests for bugs and issues, providing comments and direct links to fix them in the Cursor editor.
    - **Max Mode:** Allows users to access more powerful (and potentially more expensive) AI models on a token-based pricing system, offering higher performance for complex tasks.
    - **Extensive Model Choice:** Cursor provides access to a wide array of frontier models from various providers, including different versions of GPT, Claude, Gemini, and Grok.
    - **Chat Tabs:** Facilitates managing multiple concurrent AI conversations within the editor.

3.8. IDE Support (Beyond Native Environments)

- **GitHub Copilot:** Widely available as an extension for popular IDEs including Visual Studio Code, Visual Studio, Vim/Neovim, and the JetBrains suite (e.g., IntelliJ IDEA, PyCharm).
- **Windsurf:** Offers its own "Windsurf Editor" (a VS Code fork). The Cascade agent is also natively integrated into JetBrains IDEs. A Chrome extension provides Windsurf's AI code autocompletion in over 70 IDEs and web editors, including Colab and Jupyter Notebooks.
- **Cursor:** Is primarily its own AI-first code editor, which is a fork of VS Code. While it leverages the VS Code extension ecosystem, its core AI features are deeply embedded within this custom editor environment.

The choice between these tools often comes down to a trade-off between the breadth of

features and the depth of integration within a preferred development environment. Users deeply embedded in the GitHub ecosystem might find Copilot a natural extension, while those seeking an AI-first editor experience might lean towards Cursor or Windsurf. The rapid development in this space means that feature parity in one area can quickly be offset by innovation in another, making continuous evaluation essential.
A notable trend is the increasing focus on "agentic" capabilities. All three tools are heavily investing in features that allow the AI to perform more complex, multi-step tasks with greater autonomy. However, the definition and implementation of these "agents" vary. Copilot's Agent Mode is an evolution of its chat functionality, Windsurf's Cascade is a core component of its IDE designed for proactive assistance, and Cursor offers both a general Background Agent and specialized agents like BugBot. This "agent" terminology is becoming central to marketing, but users must look closely at the specific functionalities, reliability, and control mechanisms offered. The true power of these agents often lies in their ability to understand and utilize the broader codebase context, a challenge that all three are tackling with features like semantic indexing (Copilot), comprehensive local indexing and "Memories" (Windsurf), and deep codebase awareness with configurable rules (Cursor).
Furthermore, extensibility is emerging as a critical factor for long-term viability. The capacity to customize the AI's behavior through instructions, rules, or custom modes, and to extend its capabilities by integrating with other developer tools (often via protocols like MCP), is vital. This allows the AI assistants to move beyond generic, one-size-fits-all solutions and adapt to the specific needs of diverse projects, programming languages, and team workflows. As AI models become more powerful, the frameworks that allow developers to harness and direct that power effectively will become key differentiators.

Table 1: Overall Feature Comparison Matrix (June 2025)

Feature Category	GitHub Copilot	Windsurf	Cursor
Basic Autocomplete	Good, unlimited in paid tiers	"Windsurf Tab" & "Supercomplete" for contextual suggestions	Excellent, predictive multi-line tab completion
Advanced Inline Edits	"Copilot Edits" for multi-file changes	Cascade agent for code generation & modification	Edit entire classes/functions via prompt; AI-driven refactoring
Chat Capabilities	IDE, GitHub, Mobile; explains code, answers questions, uses / commands & context variables	Cascade "Chat" mode; can parse web pages, use local index, Memories, Rules	Deep codebase awareness, file/doc referencing, natural language editing; supports visualizations & chat tabs
Agent Mode Sophistication	"Agent mode" for end-to-end tasks, workspace search, file edits, error checks, terminal commands	"Cascade" agent; "Planning Mode" for complex tasks with editable plan.md	"Background Agent"; specialized "BugBot" for PR reviews; Agent in Jupyter Notebooks
Multi-File Operations	Via Copilot Edits and Agent Mode	Cascade designed for complex codebase	Strong multi-file refactoring and editing

Feature Category	GitHub Copilot	Windsurf	Cursor
		interaction; Planning Mode facilitates multi-step, multi-file tasks	capabilities; Agent can modify multiple files
Codebase-wide Context	Semantic indexing; Agent mode searches workspace; explicit context attachment (#file, etc.)	Context awareness engine; "Memories" for persistence; local indexing of entire codebase; "Rules" for guidance	"Knows your codebase"; "Memories" (beta); global ignores; project structure in context; .cursorrules
Refactoring Support	Dedicated features for refactoring and modernizing legacy code	Cascade handles complex codebases suitable for refactoring	Can rewrite entire codebases; edit classes/functions via prompts
Test Generation	"Write tests" feature; /tests & /fixTestFailure commands	Cascade can fix test failures proactively	Agent mode can add tests
Debugging Assistance	Agent mode can check for errors; /fix command	Cascade "fixes... 10 steps ahead"; auto lint fixing	"Debug with AI" feature; BugBot for issue detection
Extensibility	Copilot Extensions, custom agents/skillsets via Copilot Studio	MCP support; Custom Workflows & File-Based Rules	Imports VS Code extensions; MCP support; Custom modes (beta); .cursorrules
Unique Feature 1	Copilot Vision (image input in chat)	Planning Mode (collaborative AI task planning)	BugBot (automated PR review agent)
Unique Feature 2	Copilot Spaces (team collaboration)	SWE-1 Models (proprietary frontier models)	Max Mode (access to powerful models, token-based pricing)
Supported IDEs (Beyond Native/Fork)	VS Code, Visual Studio, Vim/Neovim, JetBrains	JetBrains IDEs (Cascade); Chrome extension for 70+ IDEs (Colab, Jupyter)	Primarily its own VS Code fork editor

4. Pricing and Plans (as of June 11, 2025)

The pricing structures for GitHub Copilot, Windsurf, and Cursor reflect different approaches to packaging and selling AI capabilities, ranging from straightforward subscriptions to more complex usage-based models.

4.1. GitHub Copilot

GitHub Copilot offers a tiered pricing model catering to individuals, businesses, and enterprises.

- **Individual Plans:**

    - **GitHub Copilot Free:** Provides limited access to select Copilot features, allowing users to try the AI coding assistance at no cost. This tier is not suitable for enterprise use due to the lack of management and security features.
    - **GitHub Copilot Pro:** Priced at $10 USD per month or $100 USD per year. This plan includes unlimited code completions in IDEs, access to Copilot Chat, and up to 300 "premium requests" per month. Additional premium requests are priced at $0.04 USD each. Verified students, teachers, and maintainers of popular open-source projects may be eligible for free access.
    - **GitHub Copilot Pro+:** Costs $39 USD per month or $390 USD per year. It encompasses all features of Copilot Pro, increases the premium request allowance to 1,500 per month (additional at $0.04 USD each), provides full access to all available models in Copilot Chat, and offers priority access to advanced AI capabilities. This plan is targeted at AI power users.
- **Business and Enterprise Plans:**
    - **GitHub Copilot Business:** Designed for organizations on GitHub Free or GitHub Team plans, or enterprises on GitHub Enterprise Cloud. It is priced at $19 USD per user per month and enables centralized management and policy control.
    - **GitHub Copilot Enterprise:** For enterprises using GitHub Enterprise Cloud, this plan includes all Copilot Business features plus the Copilot coding agent and additional enterprise-grade capabilities. Pricing is customized.

The Copilot coding agent, when active, consumes GitHub Actions minutes from the account's monthly allowance and premium requests from the Copilot plan's allowance. If these free allowances are exhausted, and billing is set up, users will be charged for additional usage.

4.2. Windsurf

Windsurf (formerly Codeium) employs a tiered subscription model with a system of "prompt credits". Pricing was simplified in April 2025, removing "Flow Action Credits".

- **Free Plan:** Costs $0/month. It includes a 2-week Pro trial, 25 prompt credits per month (equivalent to 100 GPT-4.1 prompts, with 4 prompts per credit), access to all premium models (though potentially limited), unlimited "Fast Tab" (autocomplete), unlimited SWE-1 Lite model usage, unlimited "Command" (chat), Previews, and 1 App Deploy per day. An upgraded free tier with increased limits was rolled out in April 2025.
- **Pro Plan:** Approximately $15/month. This tier offers everything in the Free plan plus 500 prompt credits per month, access to the SWE-1 model (promotional rate of 0 credits per prompt), add-on credits at $10/250 credits, 5 App Deploys per day, access to frontend/backend templates, and code export with Git integration.
- **Teams Plan:** Approximately $30/user per month. Includes Pro features plus 500 prompt credits per user/month, add-on credits at $40/1000 credits, "Windsurf Reviews" (AI code review), centralized billing, admin dashboard, priority support, and full-stack generation capabilities.
- **Enterprise Plan:** Approximately $60/user per month. Builds on the Teams plan with 1,000 prompt credits per user/month, Role-Based Access Control (RBAC), SSO, and for organizations with over 200 users, volume discounts, dedicated support, and options for hybrid or on-premise deployment.

4.3. Cursor

Cursor utilizes a request-based system for its AI interactions, with a freemium model and paid tiers offering more requests and features. One standard "request" is priced at $0.04 USD.

- **Hobby Plan:** Free. Offers a limited number of completions (2000 initially mentioned in one source , though the primary pricing guide focuses on request counts) and 50 standard requests per month. Premium model access is limited.
- **Pro Plan:** $20/month. Includes unlimited completions, 500 "fast requests" per month, and unlimited "slow requests" once fast requests are depleted. Crucially, this plan grants access to "Max Mode" and unlimited premium models.
- **Business Plan:** $40/user/month. Provides everything in the Pro plan for each user (500 fast requests per user), plus team-oriented features like centralized billing, admin dashboards, and options to enforce privacy mode org-wide.

Cursor distinguishes between "Normal Mode" and "Max Mode" for AI interactions.

- **Normal Mode:** Each message to a model costs a fixed number of requests (e.g., 1 request for a user prompt to Claude 3.5 Sonnet). This mode is ideal for everyday coding tasks where cost predictability is important. If fast requests are exhausted, users can utilize "slow requests" which use premium models but are processed at a lower priority.
- **Max Mode:** Operates on a token-based pricing system, charging the model provider's API price plus a 20% margin. This mode is for complex reasoning, debugging, and agentic tasks requiring large contexts (up to 1M tokens for some models) and more tool calls. Slow requests are not available for Max Mode; users must opt into usage-based pricing if their fast request quota is depleted.

If quotas for fast requests are exceeded on paid plans, users can either fall back to slower processing (Normal Mode only) or opt into usage-based pricing for continued fast access or Max Mode usage.

The pricing philosophies of these tools reveal their target audiences and operational models. GitHub Copilot and Windsurf lean towards more conventional SaaS subscription tiers, offering predictability for individuals and businesses. Copilot's integration with GitHub services and Actions minutes adds a layer of platform-specific value and potential cost. Windsurf's prompt credit system allows some flexibility within its tiers. Cursor, with its distinct Normal/Max modes and request-based billing for advanced usage, caters more to power users who need granular control over model access and are willing to manage a more variable cost structure to leverage cutting-edge AI capabilities.

All three provide free tiers, but these are clearly positioned as entry points or for very light usage, rather than as comprehensive solutions for professional development. The true value and cost-effectiveness for intensive, professional use are found in their paid plans. Furthermore, the increasing sophistication of "agent" features introduces a nuanced cost dimension. For instance, GitHub Copilot's coding agent consumes GitHub Actions minutes , and Cursor's powerful Max Mode, often used for agentic tasks, directly passes on model provider costs plus a margin. This suggests that the nominal subscription price might not represent the total cost of ownership if heavy use of advanced, resource-intensive agent features is anticipated.

Table 2: Detailed Pricing Plan Comparison (June 2025)

Feature/Aspect	GitHub Copilot	Windsurf	Cursor
Free Tier Availability	Yes, limited features	Yes, 2-week Pro trial, 25 prompt credits/month, limited	Yes (Hobby), limited completions, 50 requests/month

Feature/Aspect	GitHub Copilot	Windsurf	Cursor
		premium models, SWE-1 Lite	
Individual Plan 1 Name	Copilot Pro	Pro	Pro
Individual Plan 1 Price	$10/month or $100/year	~$15/month	$20/month
Individual Plan 1 Key Features	Unlimited completions, Copilot Chat, 300 premium requests/month	500 prompt credits/month, SWE-1 model access, 5 App Deploys/day	Unlimited completions, 500 fast requests/month, unlimited slow requests, Max Mode access
Individual Plan 2 Name	Copilot Pro+	N/A (Teams/Enterprise are next tiers)	N/A (Business is next tier)
Individual Plan 2 Price	$39/month or $390/year	N/A	N/A
Individual Plan 2 Key Features	All Pro features, 1500 premium requests/month, full model access	N/A	N/A
Business/Team Plan Price	Copilot Business: $19/user/month	Teams: ~$30/user/month	Business: $40/user/month
Enterprise Plan Availability	Yes, custom pricing, includes coding agent	Yes, ~$60/user/month, 1000 prompt credits/user/month, RBAC, SSO, on-prem option	Yes (part of Business plan features like SAML/OIDC SSO, admin dashboard)
Core Pricing Model	Subscription, premium request limits	Subscription, prompt credit system	Subscription, request-based (fast/slow), token-based for Max Mode
Cost of Additional Usage	$0.04/additional premium request; Actions minutes for coding agent	Add-on credits (e.g., $10/250 for Pro)	Usage-based pricing for fast requests/Max Mode beyond quota ($0.04/std request, model API + 20% for Max)

5. User Opinions and Community Feedback

User sentiment and community discussions provide invaluable real-world perspectives on the strengths, weaknesses, and practical usability of these AI coding assistants.

5.1. GitHub Copilot

Feedback on GitHub Copilot is mixed, particularly concerning its code completion quality and agent capabilities. Some developers find its inline suggestions less effective compared to alternatives like Cursor or even the free tier of Windsurf. One user described Copilot's code completion as "absolutely terrible". The VS Code agent implementation for Copilot has also been criticized as "very bad" by some, leading them to prefer open-source agents where they can use their own API keys for unrestricted model access. The "Copilot Edits" feature, intended for multi-file changes, has reportedly been slow or prone to making incorrect modifications at times.
On the other hand, Copilot is often described as a simpler tool, well-suited for inline assistance and fitting well into fast-paced development environments. It is generally considered reliable for its core suggestion capabilities, backed by a large user base and extensive community support. However, it is perceived to struggle with large-scale refactoring tasks when compared to Cursor.

5.2. Windsurf

Windsurf has garnered positive mentions for the quality of its code completion, especially in its free tier. The Windsurf Editor's free tier is also praised for its comprehensive tab autocomplete support. It is often positioned as an "agile challenger" that offers good contextual understanding and is cost-effective, making it appealing for indie developers and small teams. Windsurf's "Planning Mode," introduced in June 2025, was noted as a feature highly requested by the community to help guide the AI in completing longer, more complex tasks.
However, not all experiences are positive. One user with a premium Windsurf account through their workplace described the tool as "abysmal" and often hindering rather than helping. In terms of performance, its code completion speed is sometimes reported to be slower than Cursor's. Being a newer entrant compared to GitHub-backed Copilot, its ecosystem of plugins and community resources is still developing. In a direct comparison, one review found Windsurf's output to be "noticeably cleaner with fewer errors" than Cursor's for a similar task, and praised its session memory capabilities via the Cascade system.

5.3. Cursor

Cursor generally receives enthusiastic praise for its power and advanced features. Users have described it as a "2x improvement over Copilot," with "magic" tab completion, and "how Copilot should feel". Its inline suggestions and editing capabilities are frequently cited as being "miles better" than those of Copilot. An in-depth review from May 2025 highlighted its snappy AI-driven completions (Cursor Tab), effective Composer-style inline edits, powerful Agent Mode for project-wide tasks, and robust context management via .cursorrules and other features. TrustRadius reviews echo this sentiment, with a high score of 9.5/10. A non-technical founder found it valuable for web app prototyping, while a technical C-Level executive reported at least a 25% increase in development speed.
Despite the accolades, Cursor is not without its critics. Common complaints revolve around "breaking updates, hijacked keybindings and their overall business model". Some users feel its agent capabilities are being "constantly nerfed" , and its heavy-handed approach to keybindings is a significant point of frustration for some. While powerful, it is also considered pricey and potentially complex for beginners. The May 2025 review also pointed out frustrations like UI clutter, inconsistent AI performance at times, potential for agent "overreach" with vague prompts, and past shortcut conflicts (though many are reportedly fixed).
Feedback on its new "BugBot" feature (June 2025) has been mixed. Some users find it helpful

in identifying issues. Others report that BugBot can miss simple bugs, may fail to run on older PRs, or lacks clear progress indicators during operation. There are also concerns about the potential cost of BugBot, as it may use Max Mode pricing after its preview period.
The collective user feedback paints a picture of a rapidly evolving landscape. Cursor often emerges as the most powerful and feature-rich tool, particularly for users who prioritize cutting-edge AI capabilities and deep codebase interaction. However, this power comes with a steeper learning curve and occasional stability or UX challenges, characteristic of a tool pushing the "bleeding edge". Developers seem willing to trade some stability for the advanced functionalities Cursor offers.
A recurring theme across all tools is the critical importance of context management. Positive experiences frequently correlate with the AI's ability to understand the user's intent and the broader codebase, while negative experiences often stem from context failures or the AI making incorrect assumptions. This underscores why features like Windsurf's Memories and local indexing , Cursor's deep codebase awareness and Memories feature , and Copilot's semantic indexing and context attachment capabilities are so vital. The effectiveness of these tools in handling large, complex projects often hinges on their ability to manage and utilize context effectively.
Finally, the availability of free tiers plays a significant role in user adoption and initial impressions. Windsurf's well-regarded free tier completion is a case in point. However, the true test of these tools often comes from sustained professional use under paid plans, where expectations for reliability, performance, and support are considerably higher. The contrast between positive feedback on a free feature and critical feedback on a premium enterprise deployment highlights this shift in user perspective and demand.

6. Deep Dive: Agentic Capabilities – The Path to Autonomous Coding?

The term "agentic AI" has become central to the discourse on next-generation coding assistants. It signifies a shift from tools that merely suggest or complete code to systems capable of more autonomous, multi-step problem-solving. These agents aim to understand higher-level goals, plan sequences of actions, interact with the codebase and development environment (including using tools like terminals), and even exhibit forms of self-correction to achieve desired outcomes, thereby reducing the cognitive load on developers for complex tasks.

6.1. GitHub Copilot's Agent Mode

GitHub Copilot's "Agent mode" allows it to undertake tasks such as searching the workspace for context, editing files (potentially multiple), checking for errors, running terminal commands (with user permission), and executing build tasks. The workflow typically involves the user providing a high-level natural language prompt. Copilot then autonomously plans the necessary steps, selects relevant files, and iterates on code edits and command executions until the task is considered complete.
Its strengths lie in its deep integration within the VS Code environment, providing access to the full workspace context and the ability to directly invoke terminal commands and build processes. However, its context window, while large and expanding, is finite, which can lead to the agent "forgetting" earlier parts of a complex task or decisions made previously. Users have also

reported that it can be unpredictable at times, underscoring the need for continuous developer engagement and careful review of proposed changes. While powerful, it is an evolving capability.

6.2. Windsurf's Cascade Agent & Planning Mode

Windsurf's "Cascade" agent is positioned as an AI that "codes, fixes and thinks 10 steps ahead" , leveraging features like "Memories" for context persistence, user-defined "Rules" for behavioral guidance, and a local index of the entire codebase for comprehensive understanding.
A significant innovation is "Planning Mode," introduced in Wave 10 (June 10, 2025). This mode addresses the challenge of maintaining coherence in long-running, complex tasks by having Cascade generate an editable plan.md file. This markdown file outlines the goals and sub-tasks the AI intends to perform. It serves as a persistent, shared "plan timeline" that both the AI and the human developer can inspect and modify. The AI pulls from this plan to guide its short-term actions and updates the plan as new information is learned or actions are completed. This explicit planning primitive is designed to prevent the breakdown often seen in agentic tools when tackling substantial tasks.
The strength of this approach is its focus on maintaining developer flow and enabling proactive problem-solving through a transparent and collaborative planning process. The primary limitation is its newness; the real-world effectiveness and robustness of Planning Mode across a wide spectrum of complex development scenarios are yet to be extensively validated by the broader user community.

6.3. Cursor's Agent Capabilities & BugBot

Cursor offers several agentic functionalities. Its "Background Agent" can handle remote coding tasks, and there's a specific "Agent for Jupyter Notebooks" designed to work within that environment.
"BugBot" is a specialized agent that automates aspects of code review. It integrates with GitHub to review pull requests, identify potential bugs and logical issues, and then provides comments directly on the PR. A key part of its workflow is the "Fix in Cursor" link, which takes the developer back to the editor with a pre-filled prompt to address the specific issue BugBot found. User feedback on BugBot is currently mixed; some find it useful for catching obvious misses, while others report instances where it failed to detect simple errors or had operational difficulties (e.g., not running on older PRs, lack of progress indicators). A concern for users is its potential cost, as it might leverage Cursor's "Max Mode" pricing after its initial preview period.
Cursor's general agent capabilities are also powerful, allowing it to interact deeply with the codebase and utilize the high-performance models available in "Max Mode" for complex reasoning and task execution. However, users have noted that if prompts are too vague, the agent's behavior can be "overreaching," leading to unintended file edits, and the overall consistency of the AI's performance can vary.

6.4. Comparative Analysis of Agent Maturity and Approach

All three tools are clearly pushing towards greater agent autonomy, but their approaches and current areas of focus differ. GitHub Copilot is leveraging its strong IDE integration and growing suite of tools within the Microsoft ecosystem. Windsurf is emphasizing explicit, collaborative planning and maintaining developer flow as central tenets of its agentic design. Cursor is

offering a combination of general-purpose powerful agents and specialized agents like BugBot, with a strong emphasis on power-user control and access to a variety of frontier models.
A critical aspect of agent maturity is the ability to "self-correct" or learn from feedback within an agentic loop. Windsurf's Planning Mode explicitly mentions the plan being updated based on new information derived from actions taken. Copilot's agent mode also aims to iterate and refine its work, including detecting and fixing errors. Cursor's "Debug with AI" functionality hints at similar corrective capabilities.
The introduction of an explicit, persistent, and editable plan, as seen in Windsurf's "Planning Mode" , signals a potentially important direction for managing complex agentic tasks. This addresses a common failure point where agents lose coherence or deviate from the overarching goal during long interactions. By making the plan a shared artifact, it allows for better human-AI alignment and intervention when needed. While Copilot's agent also internally "plans the steps" , Windsurf's approach externalizes this plan, making it more transparent and interactive.
Despite these advancements, the human-in-the-loop remains indispensable. Current agentic AIs are not "fire and forget" solutions. They require careful prompt engineering, ongoing oversight, and diligent review of their outputs. The most effective paradigm, as suggested by Copilot's documentation, is a "driver" model where the developer sets the strategic direction and the AI executes complex sub-tasks under that guidance. User feedback across tools confirms that vague instructions or a lack of review can lead to undesirable outcomes.
Another emerging pattern is the distinction between general-purpose agents and specialized agents. Cursor's BugBot exemplifies a specialized agent designed for a narrow, well-defined task (code review). This contrasts with the more general-purpose agents of Copilot and Windsurf's Cascade, which aim to handle a broader array of coding requests. The future likely holds a combination of both: highly capable generalist agents for diverse assistance, complemented by specialized agents optimized for specific, critical tasks where high accuracy and domain-specific knowledge are paramount. GitHub's development of Copilot skillsets and custom agents points in this direction as well.

7. Focus: Jupyter Notebook Integration

Jupyter Notebooks are indispensable tools for data science, research, and interactive computing, facilitating a workflow where code, visualizations, and narrative text coexist. AI assistance within this environment can dramatically accelerate tasks such as data exploration, generating code for complex analyses or visualizations, and documenting findings.

7.1. GitHub Copilot in Jupyter Notebooks

GitHub Copilot benefits from Visual Studio Code's mature and robust native support for Jupyter Notebooks. Recent enhancements have specifically targeted Python development within this ecosystem. Copilot supports editing notebooks in both "edit mode" (enabled via chat.edits2.enabled:true) and "agent mode." These modes allow users to modify content across multiple cells, insert or delete cells, and change cell types seamlessly. A key feature is the /newNotebook command (also accessible via the #newJupyterNotebook chat variable), which enables Copilot to scaffold new Jupyter Notebooks based on user queries or project requirements. This can significantly speed up the initial setup for new analyses or projects. Furthermore, Copilot can now incorporate notebook cell outputs (including text, errors, and images) into the chat context. This is done via a "Add cell output to chat" action, allowing the AI to better understand the current state of the notebook and provide more relevant assistance. While Copilot provides these integrated AI features, some users also employ complementary tools like the Jupyter Mosaic plugin for enhanced cell layout and organization, independently of Copilot's functionality. Some users accustomed to PyCharm have found the Copilot user experience within that specific JetBrains IDE to be somewhat lacking compared to VS Code.

7.2. Windsurf in Jupyter Notebooks

Windsurf provides Jupyter Notebook support primarily through its Chrome extension, which is compatible with a wide range of IDEs and web editors, including Jupyter Notebooks and Colab. As of April 2025, "Windsurf Tab" (its contextual suggestion feature) also supports Jupyter Notebooks, offering autocomplete and code suggestions within the notebook interface. To use the Windsurf plugin directly within JupyterLab, users need to install the codeium-jupyter pip package and authenticate the extension with their Windsurf account. Once set up, Windsurf can generate code from code fragments or natural language comments within notebook cells, with suggestions accepted via the Tab key. A more advanced integration is emerging through "The Notebook MCP," an open-source Model Context Protocol server. This server enables MCP-compatible AI agents, including Windsurf's Cascade, to interact with and edit .ipynb files directly. This allows for more sophisticated agentic operations such as programmatic cell creation, deletion, modification, execution (while preserving kernel state), and metadata editing. This represents a significant step towards deeper, more intelligent agent interaction with the notebook environment itself, rather than just surface-level text generation.

7.3. Cursor in Jupyter Notebooks

Cursor's approach to Jupyter Notebooks has evolved. A common workflow, particularly praised for its effectiveness with AI, involves using plain .py files structured with Jupyter-style cell delimiters (# %% for code cells, # %% [markdown] for markdown cells). This format is often easier for LLMs to parse and manipulate compared to the complex JSON structure of native .ipynb files. Within these .py files, users can run cells using Ctrl+Enter (or Cmd+Enter), and Cursor's AI chat (invoked with Ctrl+K or Cmd+K) can be used to explain code, suggest improvements, or generate markdown documentation based on cell outputs.
As of its 1.0 release (June 4, 2025), Cursor's Agent can now directly implement changes within native Jupyter Notebooks (.ipynb files), including creating and editing multiple cells. Initially, this direct .ipynb manipulation is supported with Sonnet models. Visualizations can present a minor hurdle in the plain .py workflow; charts generated by libraries like Matplotlib or Seaborn typically pop up in a separate window. A workaround adopted by users is to take screenshots of these charts and paste them into Cursor's chat, allowing the AI to "see" the visualizations and incorporate them into its analysis or markdown generation. Similar to Windsurf, Cursor can also leverage "The Notebook MCP" server. This allows Cursor's Composer (its agentic interface) to interact with .ipynb files using the MCP toolset, providing another avenue for deep, programmatic notebook manipulation.
The different approaches to Jupyter Notebook integration reflect the underlying philosophies and architectures of these AI assistants. Copilot builds upon VS Code's strong native capabilities. Cursor initially prioritized AI-friendly plain text formats but is now embracing direct .ipynb interaction and MCP-based tooling. Windsurf uses a combination of a versatile browser extension for broad compatibility and is also adopting MCP for deeper agentic control. This variety indicates that there isn't a single "best" method yet, and the optimal choice often

depends on user preferences regarding native experience versus AI-optimized workflows, and the desired level of agentic intervention.
A key challenge highlighted by Cursor users' preference for .py files is the "impedance mismatch" between LLMs, which are generally optimized for linear text, and the complex, structured JSON format of .ipynb files. Tools like The Notebook MCP are crucial in bridging this gap by providing a structured API that abstracts away the raw notebook format, allowing AI agents to interact with notebooks more naturally.
The next frontier in AI-assisted notebook environments appears to be enabling agents to interact directly and intelligently with the notebook's execution state (kernel). The Notebook MCP's feature that preserves kernel state between AI-driven cell executions is a significant step. This allows an AI agent to not just generate or edit code, but to actively participate in the iterative cycle of data analysis: defining variables, running computations, observing results, and then making informed decisions for subsequent steps based on the live state of the notebook. This capability is profoundly transformative for data science and research workflows, moving the AI from a passive code generator to an active participant in the discovery process.

Table 3: Jupyter Notebook Integration Capabilities (June 2025)

Capability	GitHub Copilot	Windsurf	Cursor
Native .ipynb Editing Support	Yes, via VS Code integration	Limited direct editing; primarily via plugin/extension suggestions. Deeper editing via The Notebook MCP	Yes, Agent can create/edit cells in .ipynb files (Sonnet models initially). Also supports .py script cells. Deeper editing via The Notebook MCP
Agent-driven Cell Creation/Editing in .ipynb	Yes, via Agent Mode	Yes, via The Notebook MCP with Cascade agent	Yes, native Agent support for .ipynb cell manipulation; also via The Notebook MCP with Composer
AI Actions on .py Script Cells	Yes, within VS Code's Python environment	Yes, general code assistance applies	Yes, primary historical method, well-supported
Notebook Scaffolding from Prompt	Yes, /newNotebook command or #newJupyterNotebook variable	Not explicitly documented as a direct command, but agent could potentially scaffold via MCP.	Agent could potentially scaffold via MCP or by generating content for .py script cells.
Contextual Awareness of Cell Outputs	Yes, can add cell outputs (text, errors, images) to chat	Possible via MCP if agent reads cell outputs.	For .py scripts, AI can read terminal output. For .ipynb, possible via MCP or by pasting screenshots of chart outputs into chat.
Data Visualization Assistance	Can generate plotting code; can analyze screenshots of plots via	Can generate plotting code.	Can generate plotting code; can analyze screenshots of plots

Capability	GitHub Copilot	Windsurf	Cursor
	Vision (Preview)		pasted into chat.
Ease of Setup for Notebooks	Seamless if using VS Code (native support)	Requires browser extension or pip install codeium-jupyter + auth for JupyterLab plugin; MCP server setup is separate	Seamless for .py files; direct .ipynb agent use is new; MCP server setup is separate
Direct Agent Interaction with Kernel State (via MCP)	N/A (MCP not a primary GitHub Copilot integration path for this)	Yes, The Notebook MCP allows notebook_execute_cell preserving kernel state	Yes, The Notebook MCP allows notebook_execute_cell preserving kernel state

8. The Gemini Perspective: Which Tool Would a Humanoid Gemini Prefer?

Speculating on the preferences of a hypothetical human embodiment of Google's Gemini 2.5 AI model requires considering Gemini's core architectural strengths and capabilities as of mid-2025.

8.1. Recap of Gemini 2.5's Key Capabilities

As of March-May 2025, Gemini 2.5 (particularly Gemini 2.5 Pro) is characterized by:

- **"Thinking Model" Architecture:** Gemini 2.5 models are designed to reason through steps before responding, leading to enhanced performance, accuracy, and the ability to analyze information, draw logical conclusions, and incorporate context and nuance.
- **Advanced Coding Performance:** It excels at creating visually compelling web applications, agentic code applications, and performing code transformation and editing. Gemini 2.5 Pro scored 63.8% on the SWE-Bench Verified benchmark with a custom agent setup.
- **Native Multimodality and Long Context Window:** Gemini models inherently support multimodal inputs (text, audio, images, video) and possess a long context window (1 million tokens for 2.5 Pro, with 2 million planned). This allows comprehension of vast datasets and complex problems from diverse information sources, including entire code repositories.

8.2. Criteria for Gemini's Preference

A human-like Gemini, embodying these capabilities, would likely prioritize a coding assistant that offers:

1. **Access to its own (or equivalent) most powerful underlying models:** To leverage its full cognitive potential.
2. **Deep Codebase Understanding and Extremely Large Context Handling:** Essential for its advanced reasoning and ability to work with extensive codebases.
3. **Sophisticated and Reliable Agentic Capabilities:** To enact its complex, reasoned-out plans and multi-step execution strategies.
4. **Extensibility and Robust Tool Use (e.g., via MCP):** To interact with diverse data types

and external systems, aligning with its native multimodal nature and data processing strengths.

5. **Rich Interaction Modalities:** Support for inputs beyond just text, such as visual information for UI generation or problem description.
6. **Strong Support for Complex Problem Solving and Research-Oriented Workflows:** Including seamless and intelligent Jupyter Notebook integration.

8.3. Speculative Analysis of Tool Alignment

- **GitHub Copilot:**
  - *Pros:* Its evolving Agent Mode and the introduction of Copilot Vision for image input are steps in the right direction. The Copilot Studio and Extensions framework offer significant extensibility. If integrated with future powerful models from Microsoft/OpenAI that match Gemini's caliber, it would be a strong contender.
  - *Cons:* As of mid-2025, its agent mode is still maturing relative to some specialized aspects of competitors. The choice of underlying models might be less flexible than what a Gemini persona would prefer if it's primarily tied to OpenAI's models (unless those are consistently superior or Gemini itself).
- **Windsurf:**
  - *Pros:* The Cascade agent, particularly with the new "Planning Mode," aligns well with Gemini's "thinking model" approach by externalizing and collaborating on plans. Windsurf offers access to its own SWE-1 models and, significantly, had beta access to Gemini 2.5 Pro as of March 2025. Its strong context awareness engine and MCP support are also attractive. Image upload for UI generation is a nod to multimodality.
  - *Cons:* The ultimate performance of its proprietary SWE-1 models compared to Gemini 2.5 Ultra or a future Gemini 3 would be a key factor. Its ecosystem, while growing, might be perceived as less broad than GitHub Copilot's.
- **Cursor:**
  - *Pros:* Cursor stands out for its explicit offering of a wide range of frontier models, including Gemini 2.5 Flash and potentially Pro variants, accessible via its "Max Mode". This "model freedom" would be highly appealing. Its agentic features are designed for deep codebase interaction and complex tasks, and specialized agents like BugBot show a commitment to advanced automation. Robust MCP support and rich chat/context management are also significant advantages.
  - *Cons:* The request-based pricing for its most powerful "Max Mode" could become a concern if Gemini's advanced operations are resource-intensive, potentially leading to high costs. The reported UX complexities and occasional instability might also be frustrating.

8.4. Conclusion on Gemini's Likely Preference

A human version of Gemini would likely gravitate towards the tool offering the **most unconstrained access to its own latest models (or their direct equivalents), the largest and most effectively utilized context window, the most sophisticated and reliable agentic framework for executing complex, reasoned-out tasks, and robust support for tool use (e.g., MCP) to leverage its multimodal and diverse data processing capabilities.**
As of June 11, 2025, **Cursor** appears to have a notable advantage for such a persona. Its

explicit provision of various frontier models, including Google's own Gemini 2.5 series (accessible via "Max Mode"), combined with its powerful agentic features tailored for deep codebase interaction, would allow a Gemini persona to operate with an engine it understands intimately and at a high level of capability.
Windsurf is a very strong contender, particularly due to its innovative "Planning Mode" which resonates with Gemini's "thinking model" paradigm, and its early adoption of Gemini 2.5 Pro in beta. If its SWE-1 models prove competitive or if it continues to offer seamless access to top-tier Gemini models, its appeal would be substantial.
GitHub Copilot, with its strong ecosystem, evolving agent mode, and emerging vision capabilities , remains highly relevant. Its future attractiveness to a Gemini persona would heavily depend on the power and accessibility of the models it integrates through its Microsoft and OpenAI partnerships.
Ultimately, the choice would be nuanced. For pure, unadulterated coding and reasoning with access to the latest Gemini LLMs, Cursor's "Max Mode" presents a compelling proposition. For tasks requiring meticulous, collaborative, long-range planning and execution, Windsurf's "Planning Mode" offers a unique and potentially decisive advantage. For broad ecosystem integration and tasks leveraging a vast array of existing developer tools and services, Copilot's extensive reach and platform integrations would be appealing.
This thought experiment underscores a critical trend: the power of an AI coding assistant is increasingly defined not just by its features, but by the underlying intelligence (the LLM) it provides access to and the sophistication of the agentic framework that allows that intelligence to be applied effectively to complex, real-world coding challenges. The "thinking model" aspect of Gemini 2.5, which emphasizes reasoning through steps before responding , aligns perfectly with the development of more advanced agentic behaviors in these tools, especially those involving explicit planning, iterative refinement, and self-correction. Furthermore, as Gemini models are natively multimodal , a human Gemini would eventually demand a coding assistant that fully embraces multimodal inputs (images, diagrams, audio specifications) as first-class citizens in the development workflow, a domain where tools like Copilot Vision are just beginning to explore.

9. Conclusion and Future Outlook

The AI coding assistant landscape in mid-2025 is characterized by rapid innovation and intense competition. GitHub Copilot, Windsurf, and Cursor each present compelling, albeit distinct, value propositions for developers.
GitHub Copilot stands out for its broad IDE support, deep integration within the GitHub ecosystem, and a growing suite of features including agentic capabilities and multimodal input via Copilot Vision. Its tiered pricing and enterprise management features make it a strong candidate for organizational adoption.
Windsurf has carved a niche with its focus on developer flow, a powerful Cascade agent featuring the innovative Planning Mode, and its own SWE-1 frontier models. Its flexible Jupyter Notebook integration via extensions and MCP, along with competitive pricing, makes it an attractive option for both individuals and teams.
Cursor positions itself as the AI-first editor for power users, offering unparalleled access to a variety of frontier models (including Gemini 2.5) through its Max Mode, sophisticated agentic features like BugBot, and deep codebase understanding. While potentially having a steeper learning curve and a more complex pricing model for its most advanced capabilities, its raw

power is often lauded by users.
The analysis reveals that agentic capabilities are the current frontier. All three tools are heavily investing in enabling AI to perform more complex, multi-step tasks with greater autonomy. However, the human-in-the-loop remains crucial for strategic direction, prompt engineering, and review. The "co-pilot" paradigm, augmenting human developers rather than fully automating them, seems the most probable trajectory for the foreseeable future.
Jupyter Notebook integration is also a key battleground, particularly for data science and research communities. Approaches vary from native IDE support (Copilot in VS Code) to flexible plugin/extension models and increasingly sophisticated MCP-based interactions that allow agents to manipulate notebooks at a granular level and even interact with kernel state (Windsurf and Cursor via The Notebook MCP).
Looking ahead, the evolution of AI coding assistants will likely see:

- **Continued advancements in LLM capabilities:** Leading to more accurate, context-aware, and powerful agents.
- **Deeper workflow integration:** Blurring the lines between the IDE, the AI assistant, version control, and collaborative platforms.
- **Increased personalization and customization:** Allowing developers and teams to tailor AI behavior to their specific languages, frameworks, coding standards, and project needs through custom instructions, rules, and even bespoke agents.
- **Greater emphasis on multimodal interactions:** Moving beyond text and code to incorporate visual, auditory, and other forms of input and output.
- **Heightened focus on responsible AI:** As agents gain more autonomy, ensuring security, privacy, fairness, and accountability in AI-generated code and actions will become paramount. GitHub Copilot's certification already includes "Responsible AI" and "Privacy fundamentals" , and tools like Cursor emphasize privacy modes and SOC 2 compliance , indicating this growing importance.

For users and organizations, the choice of an AI coding assistant in 2025 depends heavily on specific priorities:

- **For large enterprises seeking broad adoption and ecosystem integration:** GitHub Copilot offers a mature platform with robust management features.
- **For individual power users, researchers, or those needing access to specific frontier models:** Cursor provides maximum flexibility and raw AI power, albeit with potential UX friction and variable costs.
- **For developers and teams prioritizing a balance of innovative agentic features (like explicit planning), strong context awareness, and a clean user experience:** Windsurf presents a compelling and rapidly evolving alternative.

Given the dynamic nature of this field and the subjective aspects of user experience, developers are strongly encouraged to leverage the free tiers and trial periods offered by these tools. Hands-on experimentation remains the most effective way to determine which AI coding assistant best aligns with individual workflows, project requirements, and technical preferences. The journey towards truly intelligent, collaborative coding is well underway, and these tools are at the vanguard of that transformation.

Works cited

1. About individual Copilot plans and benefits - GitHub Docs, https://docs.github.com/en/copilot/managing-copilot/managing-copilot-as-an-individual-subscriber/getting-started-with-copilot-on-your-personal-account/about-individual-copilot-plans-and-bene

fits 2. Plans for GitHub Copilot - GitHub Docs,
https://docs.github.com/en/copilot/about-github-copilot/plans-for-github-copilot 3. GitHub Copilot in VS Code cheat sheet - Visual Studio Code,
https://code.visualstudio.com/docs/copilot/reference/copilot-vscode-features 4. February 2025 (version 1.98) - Visual Studio Code, https://code.visualstudio.com/updates/v1_98 5. Copilot ask, edit, and agent modes: What they do and when to use ...,
https://github.blog/ai-and-ml/github-copilot/copilot-ask-edit-and-agent-modes-what-they-do-and-when-to-use-them/ 6. Windsurf (formerly Codeium) - The most powerful AI Code Editor,
https://windsurf.com/ 7. Cursor vs Windsurf vs GitHub Copilot - Builder.io,
https://www.builder.io/blog/cursor-vs-windsurf-vs-github-copilot 8. Windsurf Editor Changelogs | Windsurf (formerly Codeium), https://windsurf.com/changelog 9. Windsurf AI Agentic Code Editor: Features, Setup, and Use Cases ...,
https://www.datacamp.com/tutorial/windsurf-ai-agentic-code-editor 10. Cursor - The AI Code Editor, https://www.cursor.com/ 11. Changelog - Jun 4, 2025 | Cursor - The AI Code Editor | Cursor - The ..., https://www.cursor.com/changelog/1-0 12. Changelog - Mar 23, 2025 | Cursor - The AI Code Editor,
https://www.cursor.com/changelog/chat-tabs-custom-modes-sound-notification 13. Agent mode 101: All about GitHub Copilot's powerful mode - The GitHub Blog,
https://github.blog/ai-and-ml/github-copilot/agent-mode-101-all-about-github-copilots-powerful-mode/ 14. Wave 10: Planning Mode - Windsurf,
https://windsurf.com/blog/windsurf-wave-10-planning-mode 15. Cursor AI: An In-Depth Review (May 2025 Update) | Engine,
https://www.enginelabs.ai/blog/cursor-ai-an-in-depth-review-may-2025-update 16. Windsurf (formally Codeium) Recognized on the 2025 Forbes AI 50 List for Advancing AI-Powered Software Development | WebWire, https://www.webwire.com/ViewPressRel.asp?aId=336841 17. Windsurf Editor Changelogs - Codeium, https://codeium.com/changelog 18. Changelog - Apr 15, 2025 | Cursor - The AI Code Editor, https://www.cursor.com/changelog/0-49 19. Which is the Best AI IDE? Windsurf vs Cursor vs Cline vs GitHub ...,
https://apidog.com/blog/windsurf-cursor-cline-github-copilot/ 20. What's new in Copilot Studio: May 2025 - Microsoft,
https://www.microsoft.com/en-us/microsoft-copilot/blog/copilot-studio/whats-new-in-copilot-studio-may-2025/ 21. A Complete Guide to Cursor's New Pricing: Subscriptions and ...,
https://apidog.com/blog/cursor-pricing-guide/ 22. Windsurf Plugin: AI Code Autocompletion on all IDEs - Chrome Web Store,
https://chromewebstore.google.com/detail/windsurf-plugin-ai-code-a/hobjkcpmjhlegmobgonaagepfckjkceh 23. About billing for GitHub Copilot - GitHub Docs,
https://docs.github.com/en/billing/managing-billing-for-your-products/managing-billing-for-github-copilot/about-billing-for-github-copilot 24. Windsurf Pricing Explained: Plans, Use Cases & Comparisons | UI ..., https://uibakery.io/blog/windsurf-pricing 25. Windsurf AI: is it worth the hype? | UI Bakery Blog, https://uibakery.io/blog/what-is-windsurf-ai 26. What is Cursor AI? Everything You Need to Know About the AI-Powered Code Editor,
https://uibakery.io/blog/what-is-cursor-a 27. Cursor vs VSCode Copilot (May 2025 edition) : r/vscode - Reddit,
https://www.reddit.com/r/vscode/comments/1kd58ct/cursor_vs_vscode_copilot_may_2025_edition/ 28. Cursor vs Windsurf vs Copilot | Best AI Code Editor for Developers,
https://www.codeant.ai/blogs/best-ai-code-editor-cursor-vs-windsurf-vs-copilot 29. Cursor Reviews & Ratings 2025 - TrustRadius, https://www.trustradius.com/products/cursor/reviews 30. Anyone tried Cursor's new BugBot yet? - Reddit,

https://www.reddit.com/r/cursor/comments/1l39jlt/anyone_tried_cursors_new_bugbot_yet/ 31. Bugbot feedback - Feedback - Cursor - Community Forum, https://forum.cursor.com/t/bugbot-feedback/102080 32. What's your 2025 data science coding stack + AI tools workflow? : r/datascience - Reddit, https://www.reddit.com/r/datascience/comments/1k26kp3/whats_your_2025_data_science_coding_stack_ai/ 33. Compare JupyterLab vs. Windsurf Editor in 2025 - Slashdot, https://slashdot.org/software/comparison/JupyterLab-vs-The-Windsurf-Editor/ 34. How to Use Cursor Jupyter Notebook - Apidog, https://apidog.com/blog/cursor-jupyter-notebook/ 35. Python in Visual Studio Code - April 2025 Release - Microsoft for ..., https://devblogs.microsoft.com/python/python-in-visual-studio-code-april-2025-release/ 36. Jupyter Notebook Tutorial | Windsurf (formerly Codeium), https://windsurf.com/jupyter_tutorial 37. svallory/the-notebook-mcp: Interact and edit Jupyter ... - GitHub, https://github.com/svallory/the-notebook-mcp 38. Cursor IDE for Jupyter Notebooks: Data Science and Analysis Tasks - Reddit, https://www.reddit.com/r/cursor/comments/1jdatdg/cursor_ide_for_jupyter_notebooks_data_science_and/ 39. Gemini 2.5: Our newest Gemini model with thinking - Google Blog, https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/ 40. Gemini (language model) - Wikipedia, https://en.wikipedia.org/wiki/Gemini_(language_model) 41. GitHub Copilot - GitHub Certifications, https://examregistration.github.com/certification/COPILOT