

SciMaster: Towards General-Purpose Scientific AI Agents

Part I. X-Master as Foundation — Can We Lead on Humanity’s Last Exam?

Jingyi Chai^{1*} Shuo Tang^{1*} Rui Ye^{1*} Yuwen Du^{1*} Xinyu Zhu¹ Mengcheng Zhou¹
Yanfeng Wang¹ Weinan E^{1,2} Yuzhi Zhang² Linfeng Zhang² Siheng Chen¹

¹ School of Artificial Intelligence, Shanghai Jiao Tong University ² DP Technology

X-Master: <https://github.com/sjtu-sai-agents/X-Master>

SciMaster is a series of studies aimed at developing general-purpose scientific AI agents. In Part I, X-Master establishes the foundational architecture, laying the groundwork for enhancing the general capabilities of AI agents.

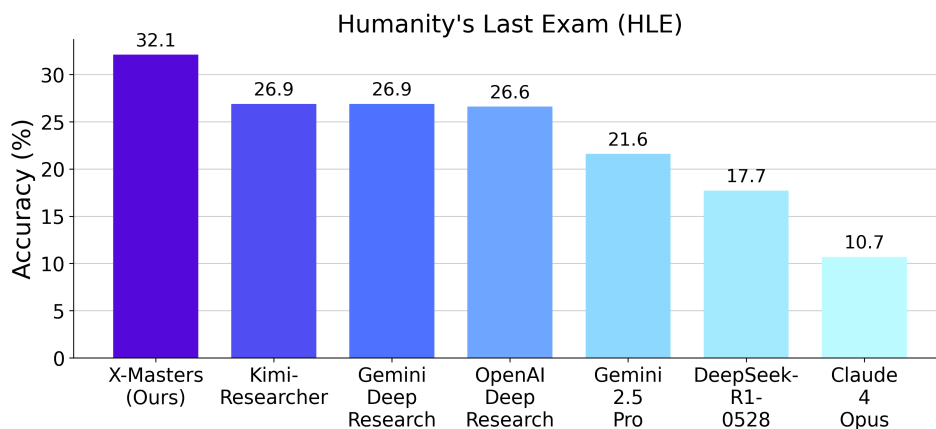


Figure 1: Comparisons on Humanity’s Last Exam. Our *X-Masters* achieves the state-of-the-art score of **32.1%**, surpassing deep research products from Kimi, Gemini, and OpenAI.

Abstract

The rapid advancements of AI agents have ignited the long-held ambition of leveraging them to accelerate scientific discovery. Achieving this goal requires a deep understanding of the frontiers of human knowledge. As such, Humanity’s Last Exam (HLE) provides an exceptionally challenging touchstone for evaluating scientific AI agents. In this work, we aim to construct the foundational architecture for general-purpose agents and validate the capabilities through leading performance on HLE. To achieve this, we introduce *X-Master*, a tool-augmented reasoning agent designed to emulate human researchers by interacting flexibly with external tools during its reasoning process. This agent, guided by the conceptualization of code as an interaction language, can flexibly leverage built-in Python libraries and our customized tools to augment the reasoning. We further scale its capabilities through *X-Masters*, a scattered-and-stacked agentic workflow that systematically enhances breadth and depth of reasoning. Our open-source solution, *X-Masters*, sets a new state-of-the-art record on HLE with a score of 32.1%, surpassing OpenAI’s and Google’s Deep Research (26.6% and 26.9%) and becoming the first to exceed the 30% threshold. This work allows us to gain a deeper understanding of complex task-solving and accumulates valuable experience that can inform future advancements, guiding subsequent model training.

*Equal Contributions. The ordering was randomized via a dice roll.

1 Introduction

Artificial intelligence (AI), particularly in the form of large language models (LLMs) (OpenAI, 2022; Dubey et al., 2024; Yang et al., 2024) is evolving at an unprecedented rate. In just two years, the landscape has shifted dramatically: from the conversational abilities of models like GPT-3.5/4 (Ouyang et al., 2022; OpenAI, 2023), to the emergence of powerful reasoners like DeepSeek R1 (Guo et al., 2025), and now towards AI agents like OpenAI’s o3 (OpenAI, 2025a) and Google’s Gemini 2.5 Pro (DeepMind, 2025) with flexible tool-use capabilities. This shift towards general-purpose agents has ignited the long-held ambition of leveraging AI agents to accelerate scientific discovery (Gottweis et al., 2025), where they can catalyze breakthroughs by reasoning over vast knowledge and exploring ideas beyond the constraints of human cognition. *In this series of studies, we aim to progressively build up SciMaster, open-source, general-purpose scientific AI agents.*

To enable AI-driven scientific discovery, an essential prerequisite is that an AI agent first demonstrate a deep grasp of the current frontiers of human scientific knowledge. To this end, Humanity’s Last Exam (HLE) (Phan et al., 2025) serves as a critical and exceptionally challenging touchstone. Developed through a global collaboration of nearly 1,000 subject experts from over 500 institutions, HLE comprises diverse, expert-level challenges at the frontier of human knowledge. Consequently, strong performance on HLE is highly indicative of an agent’s aptitude for navigating the complex problems inherent in advanced scientific research. While promising strides have been made on HLE by leading models from OpenAI (26.6%) (OpenAI, 2025c) and Google DeepMind (26.9%) (DeepMind, 2025), their closed-source nature severely limits community understanding and participation, hindering widespread exploration and innovation.

In light of this, in the first part of this series, we focus on constructing the foundational architecture for general-purpose agents and validating its general capabilities through leading performance on HLE. Our work explores a practical roadmap of inference-time computation that enables open-source models to lead. This approach, bypassing the need for extensive model training, allows us to gain a deeper understanding of diverse task-solving and accumulates valuable experience that can inform future advancements, guiding subsequent model training. By openly sharing the insights and methodologies developed throughout this process, we hope to foster greater participation and accelerate progress within the field.

To achieve this goal, we introduce *X-Master*, a general tool-augmented reasoning agent designed to flexibly interact with external tools during its reasoning process. As a general-purpose agent architecture, *X-Master* will provide the architectural foundation for the development of *SciMaster*. The design rationale of *X-Master* is to emulate the dynamic problem-solving process of human researchers who fluidly pivot between internal reasoning and external tool-use. This creates a symbiotic loop: tool outputs provide crucial feedback to sharpen the agent’s reasoning, while clearer reasoning leads to more intelligent and effective tool use. The core mechanism enabling this process is the conceptualization of code as an interaction language. When confronted with a problem it cannot solve internally, *X-Master* formulates a precise plan of action as a code block. This "plan" is then executed, interfacing with any required resource—from the numerical power of NumPy and SciPy to our custom-designed toolkit for live web searches and data extraction. The result is seamlessly absorbed back into the agent’s context, enriching its understanding and informing its subsequent reasoning. This architecture transforms the model from a static reasoner into an agile problem-solving agent, capable of actively seeking out and utilizing information just as a human would.

To unlock the full potential of our *X-Master*, we design a scattered-and-stacked agentic workflow designed to scale its intelligence at inference time. This workflow, called *X-Masters*, is engineered to systematically enhance both the breadth and depth of reasoning by orchestrating a multi-agent cognitive process, where instances of our *X-Master* adopt several specialized roles. The process begins with the ‘scattered’ phase to establish breadth: multiple Solver agents work in parallel to generate a diverse array of solutions, while Critic agents correct their potential flaws. Then, the ‘stacked’ phase enhances depth, where Rewriter agents synthesize all preceding outputs into superior solutions before a final Selector agent adjudicates the single best answer. Through such broad exploration and in-depth improvement, this workflow significantly enhances the capabilities of solving complex problems.

Our *X-Masters* sets a new record on Humanity’s Last Exam with a remarkable score of 32.1%. This score exceeds the previous records held by OpenAI and Google’s Deep Research by a substantial margin of 5.5 and 5.2 points respectively, being the first in the world to surpass

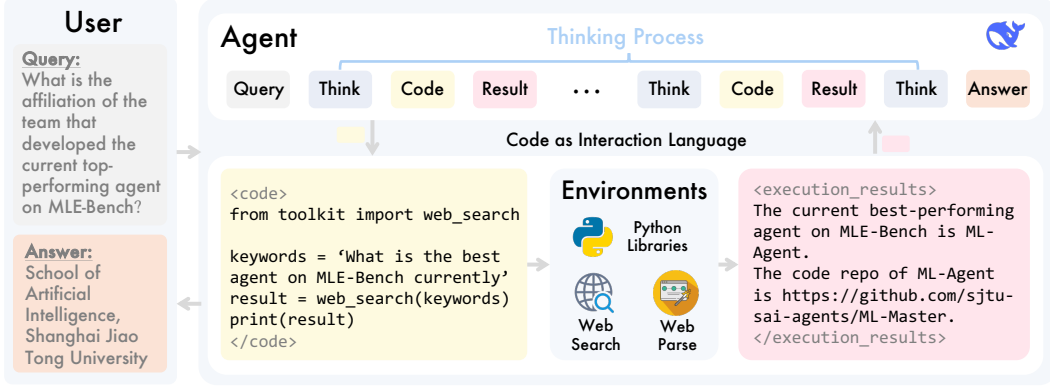


Figure 2: Overview of our *X-Master*, a tool-augmented reasoning agent. Given a user query, the agent starts the thinking process, where interaction with the environments such as tool calling would be invoked by generating a code snippet. The execution results will be appended to the agent’s context, enriching its understanding and informing its subsequent thinking. In this case, the agent invokes interactions for three times (search to get the GitHub link, parse to get the arXiv paper link, then parse to get the affiliation) until it arrives at the final answer.

the 30% threshold. Practically, when faced with a challenging problem and uncertain about which model or method to use, *X-Masters* offers one of the best—if not the best—solutions available. Critically, while these top-performing products remain closed-source, our solution is fully open-sourced, providing all the essential details for replications and further improvements.

In summary, the primary contribution of this paper is not a single, novel method, but rather the sharing of valuable "know-how" with the research community. We demonstrate the possibility of achieving leading performance on an important benchmark, using accessible open-source models. In the future, we plan to build upon this work by developing an end-to-end trained system that internalizes these capabilities. Related code will be publicly available to encourage further research and collaboration.

2 Tool-Augmented Reasoning Models as Agents

This section introduces our tool-augmented reasoning agent, starting from an overview, to our core design of code as interaction language, to tools for interaction with the internet.

2.1 Overview

X-Master is a tool-augmented reasoning agent powered by open-source models (e.g., DeepSeek-R1 (Guo et al., 2025)) designed for flexible interaction with external tools during its deep thinking process. To achieve this without requiring extensive model retraining, our approach primarily centers on the dynamic manipulation of the model’s context.

Given a query from the user that is potentially time-sensitive and knowledge-intensive, the reasoning agent initiates a thought process akin to that of non-agentic models. Distinctively, our method guides this agent to engage with external environments (e.g., the internet) as needed, for instance, when real-time information is required. Specifically, if the agent intends to interact with the environments, the agent is guided to generate Python code that accurately reflects its current requirements. This generated code is then extracted and dispatched to a code executor, which provides access to various Python libraries and our implemented tools. The execution results are subsequently appended to the model’s context, enabling the reasoning model to seamlessly continue its thought process. Notably, the agent possesses the capability to initiate such interactions iteratively, either to acquire novel information or to validate its existing conclusions. Please refer to Figure 2 for an overview.

This methodology effectively augments the reasoning capabilities of models by extending their access to external environments, thereby transcending the limitations of their inherent knowledge and

intelligence boundaries. In the following, we introduce how the agent interacts with environments via coding, how a non-agentic model is guided towards an agent, and our principal tools.

2.2 Code as Interaction Language

Our *X-Master* is designed to generate Python code as a language to interact with external environments. Specifically, during the thinking process of the agent (i.e., between `<think>` and `</think>`), the agent could either generate non-code texts for reasoning or codes within the special tokens of `<code>` and `</code>` for interacting with the environments. Once this pattern is detected via string matching, the code within would be extracted and executed in a sandbox environment, where various python libraries and tools are accessible. The executed result is then appended to the model’s context, enclosed by two special tokens `<execution_results>` and `</execution_results>`. Subsequently, the reasoning model continues the thinking process, interpreting the execution results and reasoning further, until the next interaction is invoked or the ending of thinking.

Representing interaction intention as Python code offers three advantages: (1) Universality: almost all functions can be implemented through writing code, facilitating the agent to address various tasks. (2) Accuracy: coding language can reflect the accurate needs of agents in a compact and logical manner, enabling efficient interaction. (3) Compatibility: using Python code indicates that the reasoning models can access any existing well-established libraries from the community.

2.3 Initial Reasoning Guidance

Since currently available strong reasoning models (e.g., DeepSeek-R1) are inherently non-agentic and often exhibit limited instruction-following capabilities, conventional prompt engineering alone proves insufficient to reliably guide these models towards expected agentic behavior. Instead, we introduce a simple yet effective mechanism: Initial Reasoning Guidance.

Specifically, rather than allowing the reasoning models to commence their unconstrained thinking process immediately upon receiving a user query, we embed a series of guiding texts directly after the model’s initial `<think>` token. These guiding texts are deliberately crafted from the perspective of the reasoning agent itself, speaking in the first person. For instance, such guidance includes statements like: (1) ‘I can answer this query effectively by leveraging access to external environments.’ (2) ‘Every time I determine the need for interaction with external tools, I will generate Python code enclosed between `<code>` and `</code>` tags.’

By concatenating these meticulously designed self-statements into the model’s context, we effectively lead the model to "believe" in its own enhanced capabilities. Consequently, the model is empowered to spontaneously generate and execute code, interact with its environment, and ultimately function as a capable agent, even without explicit finetuning for agentic behavior.

2.4 Tools

By expressing interactions as Python code, our *X-Master* gains strong flexibility, capable of integrating a wide spectrum of tools—from existing Python libraries to custom-built modules, and even generating new tools dynamically. Notably, when we equip *X-Master* with a suite of professional scientific computing tools, empowering it to solve domain-specific research problems and advance scientific discovery. Here, we mainly introduce our custom-built tools for information-seeking tasks.

To emulate human-like online information-seeking behavior, we design two core tools: web search and web parse. The web search tool facilitates the agent’s ability to identify relevant webpages based on the question. It provides concise summaries for each retrieved page, enabling the agent to strategically determine which links warrant deeper exploration. The web parse tool is employed when the agent requires a more thorough examination of a selected webpage to extract information directly related to the user query.

Web search. The web search tool leverages the Google search engine to identify the most relevant webpages for a given question. It furnishes three types of highly valuable information: (i) Entity-related facts: For questions containing identifiable entities (e.g., a company or application), the tool can detect these and retrieve structured facts from its knowledge graph, including name, description, and key attributes. Extracting these facts enables the agent to rapidly comprehend the core concept

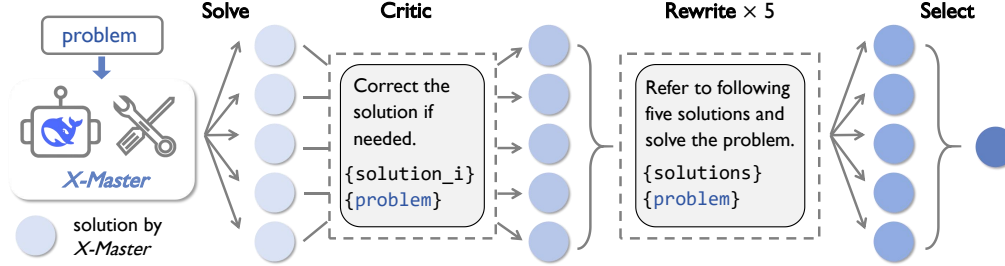


Figure 3: Overview of our *X-Masters*, a scattered-and-stacked agentic workflow. The workflow leverages *X-Master* as different roles to enhance solution quality at inference. It includes (1) Solvers generating five initial solutions, (2) Critics refining the initial solution, (3) Rewriters synthesizing all five solutions to generate new five, and (4) Selector choosing the best solution.

of the question, thereby providing crucial context for subsequent reasoning. (ii) Relevant webpage previews: For each pertinent page, the tool presents a preview comprising its title, URL, and a descriptive snippet. This functionality assists the agent in quickly discerning the page’s content and prioritizing those warranting deeper exploration. (iii) Related search queries: The tool also provides common related queries, which offer the agent avenues for follow-up searches and contribute to a broader understanding of the topic.

Web parse. The web parse tool offers support for two distinct parsing strategies, tailored for general webpages and scientific papers, respectively: (i) General webpage parsing: This strategy initiates by extracting the primary content from the target webpage. To ensure robust operation, a fallback mechanism is incorporated to manage instances where direct content extraction may encounter failures. Following content acquisition, the tool proceeds to identify segments highly relevant to the query. Furthermore, it automatically detects and returns links along with brief descriptions of pertinent subpages. This feature enables the agent to explore information more deeply, effectively emulating human behavior of navigating interconnected web content to gain comprehensive topic understanding. (ii) Scientific paper parsing: For scientific papers, the tool employs a two-step strategy to guarantee reliable content retrieval. Initially, it endeavors to retrieve an HTML version of the publication from ar5iv. In the event of an unsuccessful or incomplete HTML fetch, the system transparently reverts to downloading the authoritative PDF document. Upon successful acquisition of the paper’s full content, the tool automates the process of extracting information directly correlated with the question.

In this way, the web search and web parse tools not only help the agent find relevant information, but also encourage the agent to explore the web in a more human-like way—by searching, scanning, clicking, and digging deeper as needed.

3 Scattered-and-Stacked Agentic Workflows for Inference-Time Computation

To further scale the potential of our powerful tool-augmented reasoning agent at inference time, we propose *X-Masters*, an agentic workflow based on the principles of scattered and stacked process. Scattering enables broad problem-solving across multiple instances, generating diverse solutions, while stacking facilitates iterative improvements through rewriting and final selection. The agent, powered by *X-Master*, performs four specialized roles: Solver, Critic, Rewriter, and Selector in a tool-integrated manner, to execute this scattered-and-stacked process.

3.1 Workflow Overview

As depicted in Figure 3, this workflow alternates between expansive exploration (scattering) and sequential refinement (stacking), leveraging *X-Master* at every stage. Specifically, the workflow includes four stages: (1) broad initial solutions generation by Solver, (2) solution refinement by Critic, (3) broad solutions rewriting, and (4) final selection. In Stage 1, Solver generates five initial solutions using tool-augmented reasoning for exploring scattered potential solutions. In Stage 2, Critic diagnoses flaws and amends the solution. Stage 3 stacks these results with Rewriter referring

to all solutions and rewriting five. Stage 4 stacks once more: the Selector compares all five solutions from Rewriters, then selects the optimal one.

Overall, scattering aims to explore different diverse solutions, while stacking enhances the quality by integrating the intelligence of previous solutions. The scattered–stacked design combines breadth and depth: broadening solutions fosters diversity, whereas iterative improvement enhances precision.

3.2 Scattering for Broad Exploration

The scattering phase of the workflow involves two distinct broad processes to explore diverse solutions. For breadth, the agent answers the query five times concurrently, each with the same temperature, leveraging the random nature of the decoding process of large language models. In Stage 1, the Solver generates five initial solutions concurrently, each with tool-augmented reasoning. Afterward, the Critic concurrently evaluates each of these five solutions, identifying flaws and providing corrections or improvements, ensuring that all generated solutions are internally consistent and refined. The second part of the scattering phase (Stage 3) enhances the quality of solutions by the rewriting process. The Rewriter generates five refined versions of each initial solution. This ensures that multiple improvements are explored simultaneously, creating a diverse set of reworked answers. By combining breadth in both solution generation and rewriting, the system enhances its ability to explore and refine multiple solution paths, providing a richer set of potential solutions for further refinement.

3.3 Stacking for Sequential Refinement

The stacking phase enhances and combines solutions through rewriting and selection. In Stage 3, the Rewriter synthesizes the scattered solutions into a coherent draft, resolving redundancies and contradictions. Then, a second scattering occurs, where five Rewriters comprehensively analyze the previous five solutions and generate five. Finally in Stage 4, the Selector aggregates these refined solutions, selecting the most optimal one based on logical consistency and factual accuracy. By treating selection as a form of stacking, the workflow ensures that the final output combines diverse insights and iterative refinements, resulting in a high-quality solution.

3.4 Discussion: An Analogy to Rollouts in Reinforcement Learning

Our scattered-and-stacked architecture is fundamentally a strategy for structured exploration and exploitation, drawing a strong parallel to the concept of "rollouts" in reinforcement learning (RL) (Tesauro et al., 1995; Sutton et al., 1998; Auer et al., 2002; Guo et al., 2025). This underlying logic also resonates with recent advancements in RL, where enabling models to explore diverse solutions and leveraging self-generated rewards has proven effective in training stronger models (Zhao et al., 2025a,b; Prabhudesai et al., 2025).

(1) The "scattered" phase mirrors the exploratory principle of rollouts in RL that simulate several future trajectories to evaluate the potential of different actions. Leveraging the agent’s (model’s) stochastic decoding, each agent explores a different reasoning path, ensuring a diversity of initial strategies and preventing premature convergence on a single, potentially suboptimal idea. (2) The subsequent "stacked" phase is analogous to the aggregation and "exploitation" step that follows rollouts in RL. Agents aggregate insights from all parallel explorations to construct a superior, more robust solution, ensuring the final output synthesizes the best elements discovered.

In essence, the scattered-and-stacked workflow operationalizes the exploration-exploitation paradigm at inference time. Scattering broadly explores possibilities, while stacking deeply refines them. This structured thinking allows the system to tackle complex problems more effectively than a single reasoning process, thereby scaling the agent’s intelligence.

4 Experiments

4.1 Experimental Setups

Agent. We use DeepSeek-R1-0528 (DeepSeek-AI, 2025) as the reasoning model to drive the agent. The maximum completion of tokens is set to 64k with a temperature of 0.6.

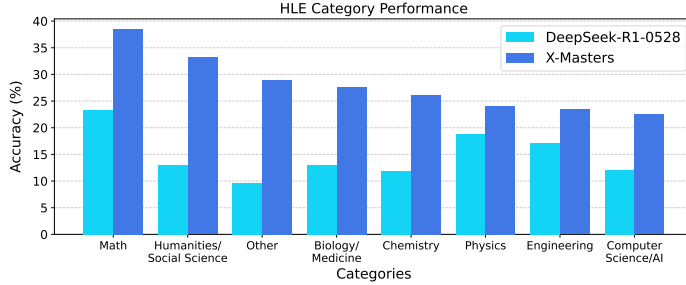


Figure 4: Performance across categories of DeepSeek-R1-0528 and X-Masters on HLE.

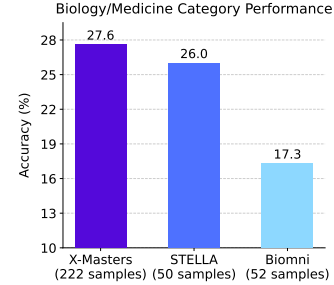


Figure 5: Performance of Biology/Medicine category of HLE.

Testing. Since the currently leveraged model is not multi-modal, we focus on the text-only subset from Humanity’s Last Exam (Phan et al., 2025), which comprises 2,518 samples following (Moonshot-AI, 2025). For evaluation, we run the workflow for three times and report the average score, and utilize o3-mini (OpenAI, 2025b) as the judge model following the official setup.

Baselines. We compare our workflow’s performance against systems spanning two categories: research agents (OpenAI’s Deep Research (OpenAI, 2025c), Google DeepMind’s Deep Research (DeepMind, 2025), Kimi-Researcher (Moonshot-AI, 2025)) and advanced models (Gemini 2.5 Pro (DeepMind, 2025), DeepSeek-R1-0528 (DeepSeek-AI, 2025), and Claude 4 Opus (Anthropic, 2025)). The results on HLE of these models are taken from existing leaderboards.

4.2 Main Results on Humanity’s Last Exam

The performance results of *X-Masters* against other systems in Humanity’s Last Exam (HLE) are shown in Figure 1. *X-Masters* achieves the highest score of **32.1%**, surpassing all existing agents and models, including Gemini Deep Research, and OpenAI Deep Research, which scored 26.9% and 26.6%. Excitingly, this is the first time that a system performs above 30% on HLE, setting a promising new record. The leading performance on extremely challenging HLE speaks volumes about the excellence of tool-augmented *X-Master* and the inference-time agentic workflow. In addition, we report the performance across different categories in Figure 4, where *X-Masters* shows a notable improvement across all categories compared to DeepSeek-R1-0528. This highlights the broad applicability and effectiveness of our tool-augmented reasoning and agentic workflow.

We also present the ablation results across workflow stages in Table 1. The results reveal progressive gains in the agentic workflow: tool-augmented reasoning (Solver) lifts baseline accuracy by 3.4%, iterative refinement (Critic and Rewriter) adds 9.5%, and final selection (Selector) achieves the record 32.1%. Detailed role-wise analysis follows in Section 4.4.

4.3 Comparisons with Scientific Agents on Biology

***X-Masters* outperforms existing systems on the Biology/Medicine category of HLE.** Recent works, such as Biomni (Huang et al., 2025) and STELLA (Jin et al., 2025b) have made progress in addressing biological challenges by leveraging LLM agents with an extensive collection of specialized tools. As illustrated in Figure 5, while Biomni gets 17.3% and STELLA gets approximately 26%, our *X-Masters* achieves 27.6% accuracy in Biology/Medicine category. Note that these two systems evaluate on their selected questions from the Biology/Medicine category; in contrast, we evaluate *X-Masters* on the complete 222 text-only questions. The domain-specific results illustrate *X-Masters*’ advanced capabilities in solving complex biomedical problems, emphasizing its significant potential for contributing to biological research.

***X-Masters* achieves the state-of-the-art performance on the biology benchmark TRQA.** Beyond general scientific challenges, we evaluate *X-Master* on TRQA-lit (choice) (Zhang et al., 2025), a specialized benchmark comprising 172 multiple-choice questions on biological research. The TRQA-lit dataset targets complex research tasks in the biological domain, including the identification of therapeutic targets, and biomedical mechanisms. These tasks represent high-level challenges faced by human experts in the field.

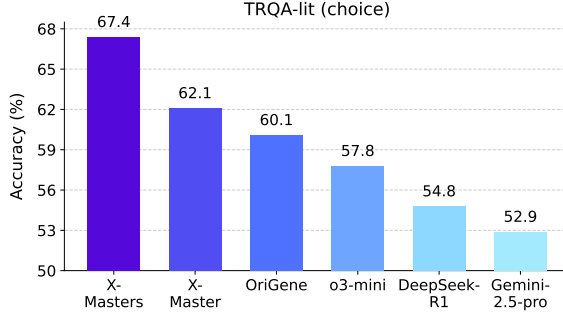


Figure 6: Performance of *X-Masters* with other models on a biology benchmark TRQA-lit (choice). Without any modification, *X-Masters* also achieves the state-of-the-art performance on this benchmark.

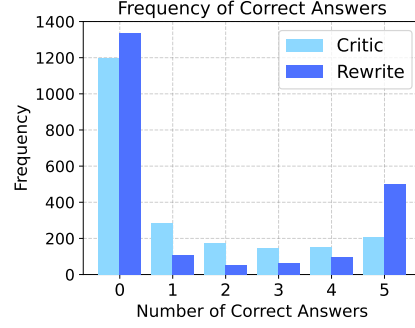


Figure 7: Frequency of correct answers before and after rewriting stage. Rewriting significantly enhances the likelihood of all 5 solutions being correct.

Table 1: Progressive accuracy improvement across stages of the *X-Masters* agentic workflow versus the DeepSeek-R1-0528. Tool augmentation and inference-time computation drive performance gains.

Method	R1	Solver	Critic	Rewriter	Selector	Accuracy (%)
Reasoning Model without Tool	✓	✗	✗	✗	✗	17.7
<i>X-Master (Ours)</i>	✓	✓	✗	✗	✗	21.1
	✓	✓	✓	✗	✗	25.0
	✓	✓	✓	✓	✗	30.6
	✓	✓	✓	✓	✓	32.1
	✓	✓	✓	✓	✓	32.1

Here, we report the results of *X-Master* and *X-Masters* in Figure 6, where the results of other models are referred from paper (Zhang et al., 2025). Results show that (1) standalone *X-Master* achieves 62.1%, already outperforming other models with tool-augmented reasoning. (2) With the simple agentic workflow, *X-Masters* achieves a state-of-the-art **67.4%**, demonstrating the effectiveness of scattered exploration and stacked selection. (3) Compared with OriGene (Zhang et al., 2025), a multi-agent system that integrates over 500 expert tools, our *X-Master*, utilizing only two web tools, obtains higher accuracy. This reinforces the demonstrated efficiency of *X-Master*’s tool-augmented reasoning process, where broad exploration and stacked selection enable it to effectively solve complex biological tasks.

4.4 Analysis

Tool-augmented reasoning enhances DeepSeek-R1. In our agentic workflow, Solver accesses external tools an average of 3 times per query while generating initial solutions. For simplicity, we use the average score of the parallel generated initial 5 solutions to estimate the pass@1. Shown in Table 1, when tool support is enabled, pass@1 rises from 17.7% to **21.1%**, indicating the effectiveness of inference-time tool augmentation in improving the accuracy of the first attempt.

Critic boosts refinement of solution. In the scattered solution exploration, each solution is first generated by the Solver and then refined by the Critic. We compare the average scores of the initial five solver solutions before and after applying the criticism in Table 1. We find that this stage delivers substantial improvement: the average accuracy rises from 21.1% to **25.0%**. This demonstrates the Critic’s crucial role in identifying errors and enhancing solution quality.

Rewriting significantly improves solution quality. From Table 1, the Rewriter consolidates insights from scattered solutions, achieving a 5.6 point leap to **30.6%** accuracy. To further understand the essence of the rewriting stage, we compare the frequency of the number of correct answers among 5 solutions before and after applying the rewriting. Shown in Figure 7, we can observe a noticeable increase in the frequency of higher correctness numbers (≥ 3), particularly for solutions with 5 fully correct answers. This shift suggests that the rewriting process effectively improves the quality of the solutions. In addition, the movement from mid-frequency correctness to higher correctness also

makes the subsequent Selector choose the optimal solution from 5 solutions. Therefore, the rewrite phase serves as a critical quality amplifier, transforming scattered candidate solutions into a refined, high-value solution pool.

Scattering and stacking matter in the agentic workflow.

We conduct an ablation study of our scattered-and-stacked agentic workflow, shown in Table 2. When no scattering, Solver only generates one solution, Critic refines the solution, and Rewriter analyzes the single solution to generate a new one. The absence of parallel exploration slashes accuracy to 25.5%. When no stacking, rewriting and final selection stages are removed, the workflow loses its depth-wise improvements and accuracy drops to 25.0%. Enabling both scattering and stacking boosts performance to 32.1%. The results confirm that scattering supplies diverse solutions, while stacking distills them into a higher-quality answer; only their synergy realizes both breadth and depth in reasoning.

Table 2: Ablation study of scattering and stacking feature in the agentic workflow.

scatter	stack	Accuracy (%)
✗	✓	25.5
✓	✗	25.0
✓	✓	32.1

Examples. We provide three examples of *X-Master*’s problem-solving trajectories in Figure 8, 9, 10.

5 Related Works

Tool-augmented LLMs. There are several recent works that focus on leveraging external tools to augment LLMs, which apply different strategies for tool invocation. (1) *Prompt engineering for structured outputs.* Some methods, like Agentic Reasoning (Wu et al., 2025b), leverage prompt engineering to guide LLMs in generating outputs in a specific format for subsequent processing by other LLM calls. While effective for fixed agentic workflows, these approaches often lack the adaptability needed for complex problems requiring iterative tool usage. (2) *Special token-guided tool invocation.* Methods such as Search-R1 (Jin et al., 2025a), Search-o1 (Li et al., 2025a), R1-Searcher (Song et al., 2025), WebThinker (Li et al., 2025b), and WebDancer (Wu et al., 2025a) guide the model to generate tool invocation content within specific special tokens (e.g., ‘search’). The results from these tools are then re-inserted into the model’s context. However, a significant limitation here is that integrating each new tool necessitates manual modifications to the tool invocation rules. This makes them less practical for diverse and complex real-world tasks, such as scientific research, which often demand a wide array of tools. (3) *Code generation for computation.* Methods such as ToRL (Li et al., 2025c) and ReTool (Feng et al., 2025) enable LLMs to generate code for computational tasks, like mathematical calculations. However, their scope is typically limited to computation, overlooking the broader need for custom tools tailored to real-world scenarios beyond mere calculations. Addressing the limitations of existing methods, our work offers the most general approach to tool usage currently available. We leverage Python code as an interaction language, which allows our agent to access: built-in Python libraries (e.g., for computation), our customized tools (e.g., web search), and even build its own tool during inference time. This unique extensibility empowers our agent to meet the demands of complex real-world tasks.

Agentic workflows. Agentic workflows empower LLMs to tackle more complex tasks by orchestrating multiple LLM calls and tool interactions. For instance, AI co-scientist (Gottweis et al., 2025) leverages multiple agents and tools for scientific research; ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2024) design agentic workflows for software development; MAS-GPT (Ye et al., 2025) generates query-specific workflows represented as Python code. However, a common limitation in these existing methods is that their agents operate in a fixed single-turn manner. This means they perform only one action—either generating a textual response or calling a tool—for each assigned (sub-)task. In contrast, our *X-Masters* framework features agents that work in a flexible multi-turn manner. This allows them to flexibly and iteratively interplay between internal reasoning and external tool usage, enabling a more dynamic and adaptive approach to problem-solving.

6 Conclusions and Future Works

This work presents *X-Master*, the foundational agent introduced as Part I of our *SciMaster* series, which aims to enhance the general capabilities of AI agents and accelerate the pace of future scientific discovery. *X-Master* is a general tool-augmented reasoning agent, designed to emulate human

problem-solving by flexibly interplaying between internal reasoning and external tool usage. Our conceptualization of code as an interaction language allows *X-Master* to flexibly interact with diverse resources. To further scale its intelligence at inference time, we develop a scattered-and-stacked agentic workflow *X-Masters*, systematically enhancing both the breadth of exploration and the depth of exploitation during task solving. Remarkably, our *X-Masters* achieves a score of 32.1% on Humanity’s Last Exam (HLE), setting a new world record that crosses the 30% threshold and surpassing the previous leading closed-source models from OpenAI (26.6%) and Google DeepMind (26.9%) by substantial margins.

Our primary contribution lies not in proposing yet another algorithm, but in openly sharing the practical "know-how" that enables open-source models to attain—and even surpass—state-of-the-art performance on challenging benchmarks, such as HLE. By providing a practical roadmap for enhancing LLM capabilities without requiring extensive retraining, we share insights into complex problem-solving and lay the groundwork for future advancements. With this work, we hope to demonstrate to the community that cutting-edge benchmarks are not the exclusive domain of resource-rich industry labs.

What’s next? Looking ahead, *SciMaster* will significantly expand on the architectural foundation of *X-Master*. Our roadmap includes the development of specialized scientific agents and tools to support literature analysis, scientific computing, and experimental workflows. In addition, we aim to build end-to-end trained agents that fully internalize the sophisticated reasoning and tool-use capabilities showcased by *X-Masters*.

References

- Anthropic. Introducing claude 4. <https://www.anthropic.com/news/claude-4>, 2025. Accessed: 2025-07-03.
- Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.
- Google DeepMind. Gemini 2.5: Our most intelligent ai model. <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking>, 2025. Accessed: 2025-06-24.
- DeepSeek-AI. Deepseek-r1-0528. <https://huggingface.co/deepseek-ai/DeepSeek-R1-0528>, 2025. Accessed: 2025-06-27.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.
- Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.
- Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, et al. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pages 2025–05, 2025.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025a.
- Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. Stella: Self-evolving llm agent for biomedical research. *arXiv preprint arXiv:2507.02004*, 2025b.

- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025a.
- Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025b.
- Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025c.
- Moonshot-AI. Kimi-researcher: End-to-end rl training for emerging agentic capabilities. <https://moonshotai.github.io/Kimi-Researcher/>, 2025. Accessed: 2025-06-26.
- OpenAI. Introducing chatgpt. <https://openai.com/index/chatgpt/>, 2022. Accessed: 2025-06-24.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- OpenAI. Introducing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2025a. Accessed: 2025-06-24.
- OpenAI. Openai o3-mini. <https://openai.com/index/openai-o3-mini/>, 2025b. Accessed: 2025-06-26.
- OpenAI. Introducing deep research. <https://openai.com/index/introducing-deep-research/>, 2025c. Accessed: 2025-06-26.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NIPS*, 35:27730–27744, 2022.
- Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity’s last exam. *arXiv preprint arXiv:2501.14249*, 2025.
- Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, 2024.
- Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.
- Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025a.
- Junde Wu, Jiayuan Zhu, and Yuyuan Liu. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv preprint arXiv:2502.04644*, 2025b.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. Mas-gpt: Training llms to build llm-based multi-agent systems. *arXiv preprint arXiv:2503.03686*, 2025.
- Zhongyue Zhang, Zijie Qiu, Yingcheng Wu, Shuya Li, Dingyan Wang, Zhuomin Zhou, Duo An, Yuhan Chen, Yu Li, Yongbo Wang, et al. Origene: A self-evolving virtual disease biologist automating therapeutic target discovery. *bioRxiv*, pages 2025–06, 2025.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025b.

7 Cases

Query

Fourteen circles of radius one are tightly packed into another circle. What is the radius of this circle up to 4 significant digits?

Response

Recognize that "tightly packed" means finding the smallest enclosing circle for 14 unit-radius circles; decide that known optimal values must exist online, so start by searching authoritative sources.

```
<code>
results = web_search("circle packing 14 equal circles inside a circle")
print(results)
</code>
```

```
<execution_results>
1. title: "Circle packing in a circle - Wikipedia", link: "https://en.wikipedia.org/wiki/Circle_packing_in_a_circle",
2. title: "The best known packings of equal circles in a circle", link: ...
</execution_results>
```

[omitted for illustration...]

Attempt to parse the Magdeburg page fails, **so pivot to Wikipedia.**

Find interesting links returned by `web_search` and try to get more details by `web_parse`

```
<code>
results = web_parse("https://en.wikipedia.org/wiki/Circle_packing_in_a_circle", "minimal radius for 14 unit circles")
print(results)
</code>
```

```
<execution_results>
"The minimal radius for 14 unit circles is  $\approx 4.328$ ," with a 2024 proof of optimality by Ekanayake & LaFountain.
</execution_results>
```

Checked whether higher precision was available. **Try Magdeburg site.**

```
<code>
results = web_parse("http://hydra.nat.uni-magdeburg.de/packing/cci/cci.html", "What is the radius of the smallest circle that can contain 14 equal circles of radius one?")
print(results)
</code>
```

```
<execution_results>None</execution_results>
```

Tried a calculator page (EngineeringToolbox) for **cross-checking.**

Try alternative ways to double check the answer

```
<code>
results = web_parse("https://www.engineeringtoolbox.com/smaller-circles-in-larger-circle-d_1849.html", "diameter for 14 circles")
print(results)
</code>
```

```
<execution_results>No numeric value.</execution_results>
```

Conclude 4.328 is the accepted value and already expressed to four significant digits.

```
<answer>4.328</answer>
```

Figure 8: Case 1. *X-Master* fluently handles the case where the tools do not work as expectation by adjusting the usage; and ensures the correctness of answer by cross-checking.

Query

What is the minimum number of diffraction gratings necessary to construct a full spectral volume from a single image using computed tomography?\n\nAnswer Choices:\nA. 1\nB. 2\nC. 3\nD. 4\nE. 5\nF. 6

Response

Decided the quickest clue would come from existing snapshot spectral-tomography systems.

```
<code>
results = web_search("minimum number of diffraction gratings spectral tomography")
print(results)
</code>
```

<execution_results>Generic optics pages, **nothing about** tomography or snapshot systems.</execution_results>

Re-phrased the query to tie gratings directly to tomography.

```
<code>
results = web_search("minimum gratings spectral tomography")
print(results)
</code>
```

The returned results are irrelevant to query, X-Master try different search words.

<execution_results>**Still mostly irrelevant**; no mention of CTIS or snapshot designs.</execution_results>

[omitted for illustration...]

Switched strategy: look for concrete CTIS (Computed Tomography Imaging Spectrometer) literature known to use gratings.

```
<code>
results = web_search("CTIS number of diffraction gratings")
print(results)
</code>
```

<execution_results>
1. "The original CTIS design employed three crossed cosine gratings."
2. Follow-up designs "employ a single two-dimensional reflective diffraction grating."
</execution_results>

Finally find some facts to point out the answer.

Interpreted the findings: early CTIS needed three separate gratings but improved 2-D grating technology condenses the function into one physical grating while still producing multiple diffraction orders (projections) necessary for tomographic reconstruction. Therefore, one grating is physically sufficient—and thus the minimum.

<answer>A. 1</answer>

Figure 9: Case 2. *X-Master* fluently handles the case where the tools do not return the expected content by trying different strategies.

Query

Pandora is a distant white draft star DB-class. I measured its spectral radiance at wavelength 400 nm and got a value of 1.2e15 watts per square meter per steradian per meter. I also estimated its surface temperature as 9000K. If this result looks ok answer 0. Otherwise answer in the form of Xv with X as the quantity most likely having a measurement error (L for wavelength, T for temperature, or B for spectral radiance) and v is its expected correct value (rounded to unit: nm for L, K for T, and watts per square meter per steradian per meter for B).

Response

Recall Planck's law to test if the given numbers are self-consistent; plan to compute the theoretical spectral radiance B_{calc} at $\lambda=400\text{nm}$, $T=9000\text{K}$.

```
<code> [omit constant inputs]
B_calc = 2 h c^2 / (lambda^5 (exp(hc/lambda*kT) - 1))
print(B_calc)
</code>
```

Get calculation results
by code execution

```
<execution_results>2.177e14</execution_results>
```

Calculated is about 2.177e14 W/m²/sr/m, measured is 1.2e15. That's about 5.5 times larger. **So not matching.** Because B and λ disagree with the assumed T, next solve for the temperature that would reproduce the measured radiance. **Rearrange** Planck's formula analytically to isolate T.

```
<code> [omit constant inputs]
y = 1 + (2 h c^2) / (B_meas lambda^5)
T_fit = (h c) / (lambda k_B ln y)
print(T_fit)
</code>
```

Find that two results are not
matching, try to get correct one

```
<execution_results>15178</execution_results>
```

Verify that this new temperature really yields the measured radiance.

```
<code> [omit complex codes]
# plug lambda = 400 nm, T = 15178 K back into Planck's law
print(B)
</code>
```

Check whether the new temperature
satisfies the condition in query.

```
<execution_results>1.20e15</execution_results>
```

[omitted more verification processes for illustration...]

Conclude the temperature is the erroneous quantity.

```
<answer>T15178</answer>
```

Figure 10: Case 3. *X-Master* conducts computations, adjusts strategies when encountering unmatched results, and verifies the final answer by writing python code.