SOLVY

An AI-Powered Step-by-Step Math Solver for Algebra and Geometry

PUBLISHED BY:

ADHIRAJ SINGH

CLASS 12TH STUDENT

INDEPENDENT AI RESEARCHER

ABSTRACT

Solvy is an AI-powered math solver designed to provide clear, step-by-step solutions to a wide range of algebraic and geometric problems. Unlike traditional black-box solvers that deliver only final answers, Solvy emphasizes transparency and learning by breaking down the logic behind each step. Built using Python and SymPy, Solvy features both a Command-Line Interface (CLI) and a Graphical User Interface (GUI) for user-friendly interaction. It preprocesses user input by converting casual math language (like `4x` or `x^2`) into computational syntax, making it ideal for beginners. Geometry functionality includes formulas for area, perimeter, and volume, with units handled automatically. The system incorporates robust error handling, user-friendly formatting, and modular design for future expansion. Solvy bridges the gap between computation and comprehension, making it a practical tool for students and a research-worthy contribution to educational AI..

PREFACE

The idea for Solvy emerged from a personal frustration I experienced as a student: many math-solving tools give answers, but few explain the *why* or *how*. I've always believed that the true power of artificial intelligence lies not just in automation, but in augmenting human understanding. Solvy is a manifestation of that belief — a tool that teaches while it solves.

This project reflects a journey that began in my early childhood, when I built functional gadgets from broken appliances. Over time, my curiosity evolved into a passion for building useful tools using AI and programming. Solvy represents the intersection of my love for mathematics, coding, and a desire to make learning more accessible. It is more than just a solver — it's a step toward AI tools that **teach, explain, and assist**, rather than simply answer.

The development process involved designing a natural input parser, integrating symbolic computation, building both GUI and CLI interfaces, and handling varied user inputs gracefully. It has helped sharpen not only my programming skills, but also my understanding of user experience and educational psychology.

I hope this paper provides insight into the thought process, challenges, and innovations behind Solvy, and inspires other learners to create meaningful, problem-solving tools of their own.

INTRODUCTION

In the digital learning era, tools that simplify complex academic subjects are more essential than ever. Mathematics, in particular, remains a significant challenge for students due to its abstract nature and reliance on accurate step-by-step reasoning. While numerous online calculators provide instant answers, most fail to explain the logic behind the solution. This lack of explainability often leads to superficial understanding and dependency on black-box systems.

To address this gap, we developed **Solvy** — an AI-powered math solver designed to not only calculate, but also **explain each step** in the process. Solvy interprets user input written in natural math syntax (e.g., `4x + 5 = 13`) and converts it into a form that can be computationally solved using Python's symbolic algebra library, **SymPy**. The tool supports both **linear and quadratic algebraic equations**, as well as **geometry-based problems** involving area, perimeter, and volume.

Solvy emphasizes user accessibility by providing both a **Command-Line Interface (CLI)** and a **Graphical User Interface (GUI)**. Its smart input parser and unit-aware outputs make it suitable for beginners and advanced users alike. By focusing on explainability and error-handling, Solvy serves as both a homework assistant and an educational tool.

TECHNOLOGICAL FRAMEWORK

Solvy is built using the **Python programming language**, selected for its simplicity, readability, and extensive ecosystem of scientific libraries. At its core, Solvy uses **SymPy**, a symbolic mathematics library in Python that supports algebraic manipulation and step-by-step equation solving. This enables Solvy to deliver human-readable solutions rather than just numerical outputs.

The application is structured around two user modes:

- A **Command-Line Interface (CLI)** for quick usage by advanced users
- A **Graphical User Interface (GUI)** built with **Tkinter**, for more accessible and interactive use by beginners

To support natural mathematical input, Solvy includes a custom-built input preprocessor. It transforms casual mathematical syntax like `4x` or `x^2` into valid computational expressions like `4*x` and `x**2`. This ensures that even users unfamiliar with programming conventions can interact comfortably with the system.

Geometry support is implemented using modular Python functions for calculating area, perimeter, and volume of standard shapes (rectangle, circle, triangle, cube, cuboid, sphere, cone, etc.). Solvy also features **unit-aware formatting**, where answers are returned with appropriate units like cm^2 or cm^3.

Robust **error handling** ensures that invalid inputs or unresolvable equations return clear and user-friendly messages, maintaining the application’s stability and educational value.

METHODOLOGY

The architecture of Solvy is designed to be modular, beginner-friendly, and highly extendable. The system handles a user's query — whether algebraic or geometric — through a structured multi-step pipeline that transforms natural input into a precise, step-by-step solution. Below is an overview of the major functional components of Solvy:

System Workflow:

1. **User Input (CLI or GUI):**
   The user enters a mathematical query, such as `4x + 5 = 13` or `area of triangle: b=6, h=3`.
2. **Input Preprocessing Module:**
   Solvy parses the input string and converts casual math notation to valid Python syntax:
   - `4x` → `4*x`
   - `x^2` → `x**2`
     This step ensures compatibility with symbolic computation engines.
3. **Problem Classification:**
   The system detects whether the query is an **algebraic equation** or a **geometry calculation** based on keywords and structure.

4. **Mathematical Solver Engine:**
   - **For algebra**, Solvy uses the **SymPy** library to isolate variables and solve equations, generating a symbolic, step-by-step solution.
   - **For geometry**, Solvy applies shape-specific formulas for area, perimeter, or volume, tagging units accordingly.
5. **Error Handling Layer:**
   If invalid syntax or unsupported expressions are entered, Solvy displays helpful error messages (e.g., "Unable to solve the equation. Please check your input.")
6. **Formatted Output Module:**
   The final result is displayed either in the command-line interface or in a GUI window with proper formatting, explanations, and units.

Flow Diagram:

Here is an overview of the algorithm of this AI project:

```
[User Input]
       ↓
[Preprocessing]
       ↓
[Problem Type Detection]
       ↓
[Math Engine (Algebra / Geometry)]
       ↓
[Error Handling]
       ↓
[Output (CLI/GUI)]
```

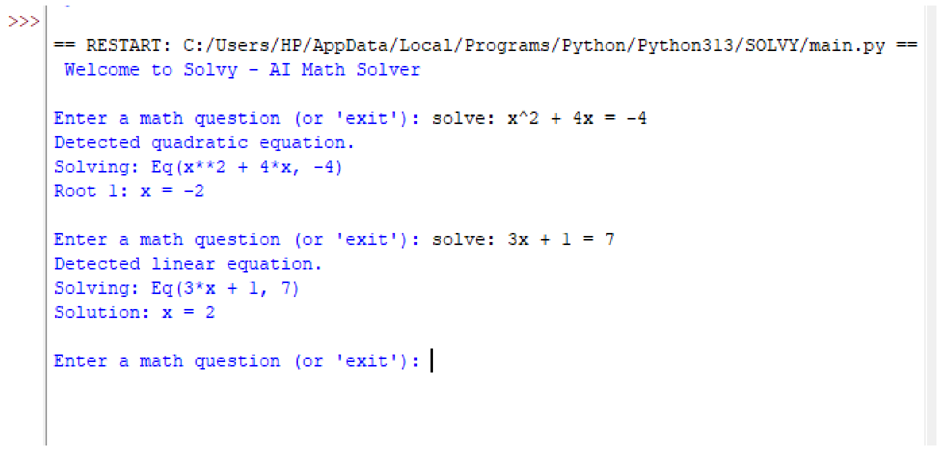
USE CASES & APPLICATIONS

Solvy is designed to support students in solving algebraic equations and basic geometry problems with a step-by-step approach. Its educational focus ensures that users understand the logic behind each calculation, rather than simply receiving the final result.

The following types of problems are currently supported:

- **Linear Equations**
  Solvy handles equations such as `3x + 5 = 14`, providing each algebraic transformation step until the variable is isolated.
- **Quadratic Equations**
  Equations like `x^2 + 4x + 4 = 0` are solved using the quadratic formula, and solutions are simplified to detect real, complex, or repeated roots.
- **Geometry Problems**
  Includes computation of **area**, **perimeter**, and **volume** for common shapes such as triangles, circles, cubes, and cones. All outputs are tagged with appropriate units (e.g., cm^2, cm^3).

Solvy is robust against invalid inputs. In such cases, the system returns a descriptive error message, guiding the user to correct the expression without crashing.



```
>>>
== RESTART: C:/Users/HP/AppData/Local/Programs/Python/Python313/SOLVY/main.py ==
 Welcome to Solvy - AI Math Solver

Enter a math question (or 'exit'): solve: x^2 + 4x = -4
Detected quadratic equation.
Solving: Eq(x**2 + 4*x, -4)
Root 1: x = -2

Enter a math question (or 'exit'): solve: 3x + 1 = 7
Detected linear equation.
Solving: Eq(3*x + 1, 7)
Solution: x = 2

Enter a math question (or 'exit'):
```

HERE IS SOLVY, SOLVING A QUADRATIC EQUATION.

CHALLENGES & LIMITATIONS

While developing Solvy, several challenges were encountered—both technical and conceptual. These shaped the design choices and exposed opportunities for future improvements.

1. Input Parsing Complexity

Translating natural mathematical expressions (like `4x^2 + 3x = 7`) into valid Python syntax (`4*x**2 + 3*x - 7`) proved to be non-trivial. While basic preprocessing rules were effective, the system still struggles with more ambiguous or incorrectly formatted inputs. Future versions could integrate NLP-based parsers for enhanced flexibility.

2. Step-by-Step Generation Limitations

Although SymPy provides structured solutions, fine-tuning the output into truly human-friendly, educational steps was a challenge. Some transformations appear too “mathy” or skip intuitive logic. A custom step-rendering engine could improve this for learners.

3. Geometry Constraints

The current version supports only basic 2D and 3D shapes with straightforward formulas. Complex shapes or dynamic figure rendering are not yet implemented. Integration with a visual geometry engine (like matplotlib or Pygame) could be explored.

4. No Word Problems Support

Solvy currently does not understand word problems or convert natural language questions into equations. This is a significant limitation for students who need help with real-world math modeling. NLP-based extensions could address this in future versions.

5. Offline Desktop App Only

Solvy is limited to a local desktop interface and does not yet offer a web-based or mobile version. Making the app cross-platform or browser-based could significantly increase accessibility and adoption.

Summary

Despite these challenges, Solvy achieves its core goal — delivering step-by-step solutions in algebra and geometry — with reliability and educational clarity. Each limitation opens the door to future research and iterations.

..

CONCLUSION

Solvy was developed to address a clear gap in the world of digital education tools: the lack of step-by-step mathematical guidance. Unlike black-box solvers that simply return final answers, Solvy is designed to **teach while solving**, providing students with clarity, logic, and confidence in tackling algebraic and geometric problems.

Built using Python, SymPy, and Tkinter, Solvy offers both a command-line and graphical interface, making it accessible to beginners and power users alike. The intelligent input parser, unit-tagged outputs, and clean UI make it an ideal supplement for classroom learning and independent study.

Throughout the development process, Solvy evolved from a simple equation solver into a **learning companion** — one that reflects the potential of combining artificial intelligence with education. This project also deepened my understanding of symbolic computation, user interaction design, and the real-world limitations of current AI in education.

FUTURE SCOPE

Looking forward, several improvements can make Solvy more powerful and scalable:

- **Natural Language Word Problem Solver:** Integrate NLP models (e.g., spaCy, BERT) to parse word-based math problems into solvable equations.
- **Web-Based Interface:** Deploy Solvy online using Streamlit or Flask to make it accessible on all platforms.
- **Mobile Version:** Build a lightweight app for Android/iOS using tools like Kivy or React Native.
- **Visual Geometry Engine:** Add figure rendering and dynamic diagrams for geometry problems using matplotlib or Pygame.
- **AI-Driven Step Optimization:** Train custom logic modules to explain steps in a more intuitive, human-like manner.

Solvy is not just a project — it is a foundation for a broader initiative: using AI to make learning **interactive, intelligent, and inclusive**.

References & Author Information

References

1. Meurer, Aaron, et al. *SymPy: Symbolic Computing in Python*. PeerJ Computer Science, 2017.
2. Python Software Foundation.
3. SymPy Documentation.
4. Tkinter GUI Library.
5. Geometry Formula References

Author Details

Name: Adhiraj Singh
Class: 12
School: Delhi Public School, Jankipuram
Academic Year: 2025–2026
Project Title: *Solvy – An AI-Powered Math Solver*
GitHub Repository:
https://github.com/ADHIRAJ191207/SOLVY---Math-Solver.git