International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Python Based Code Debugger and Compiler

Prof. Babasaheb Waghmode, Parvathi Aaduparambil , Vasundhara Yande, Ananya Pandey

Department of Computer Engineering , MGM College of Engineering & Technology

---

Abstract –

In the evolving landscape of software development, efficient debugging and code optimization remain pivotal to enhancing programmer productivity and code quality. This project presents a web-based platform that functions as a line-by-line Python code debugger and compiler, designed specifically to assist students and developers in understanding, analyzing, and refining their Python programs. The platform provides an intuitive interface for real-time code execution, variable tracking, and step-wise debugging. Additionally, it integrates AI-powered tools that generate alternative solutions and suggest optimizations, fostering deeper learning and encouraging best coding practices. By combining traditional debugging features with modern AI capabilities, the platform not only simplifies the debugging process but also serves as an educational tool, making it an innovative contribution to the programming community.

Key Words: python, debugger, compiler, AI

1. INTRODUCTION

Debugging is a critical aspect of programming that directly influences code quality and development speed. While existing tools offer functional debugging capabilities, they often lack clarity and learning support, especially for beginners. To address this gap, we have developed a Python-based web platform that offers line-by-line code debugging and compilation, enabling users to observe program flow, variable states, and logic execution in a clear, stepwise manner.

What sets our platform apart is its integration of AI-powered tools that assist users by generating alternative code suggestions and improvements. These intelligent features not only help in fixing errors but also promote code optimization and better problem-solving techniques. Designed with both functionality and learning in mind, the platform serves as a valuable tool for students, educators, and developers seeking a more insightful debugging experience.

2. MOTIVATION

While learning to program, students often struggle not just with writing code, but with understanding how it runs and why it fails. Existing debuggers tend to be technical and lack step-by-step guidance, leaving beginners frustrated and reliant on guesswork. We wanted to create a tool that bridges this gap—one that not only helps track down bugs but also enhances conceptual clarity.

Additionally, we recognized the untapped potential of AI in code education. By integrating AI-driven suggestions into a debugging platform, we aimed to turn each error into a learning opportunity. Our motivation was to build a smarter, more approachable system that helps users grow as developers, not just fix their code.

3. SYSTEM ARCHITECTURE

For Users Panel

1. Login & Authentication Module

Provides secure user login and registration. Implements session handling and access control to ensure personalized usage.
Uses token-based authentication (e.g., JWT) for secure communication.

2. User Interface (UI Layer)

A clean and intuitive frontend built with modern web frameworks (React, HTML/CSS, JavaScript).
Offers navigation options such as: Dashboard/Home
Code Editor (for writing and editing Python code) Debug Option (to run code line by line and view variable states)
Compile Button (for full script execution)
AI Suggestions (to generate alternative or optimized code snippets)

3. Code Execution & Debug Interface Sends user code

to the backend via API calls.
Receives execution steps and variable tracking data to be displayed in the UI dynamically.
Highlights each line as it executes, improving traceability.

4. AI Suggestions Panel

Connected to an AI-powered engine that provides:
Alternative implementation
Optimized or cleaner code suggestions
Displays AI feedback next to the code editor for comparison and learning

5. User Preferences & Session Data (Optional)

Saves user preferences such as theme, editor layout, or recent files.
Maintains user history for recent debugging sessions (if enabled).

For Admin Panel

1. Admin Authentication & Access Control

Dedicated login system with role-based access
Ensures only authorized personnel can access administrative functions

2. Dashboard Interface (UI Layer)

Built with responsive web technologies (React/Vue, HTML/CSS).
Displays key metrics such a Number of active use Recent logins and activity Code execution
AI usage statistics

3. User Management Module Allows the admin to View user profiles and activity
Deactivate or remove accounts Reset user credentials if needed
Monitor frequent error patterns and usage behavior

4. Code Activity Monitor

Logs user-submitted code and execution data for auditing.
Flags suspicious or harmful code for review.
Helps analyze common errors and usage trends to improve system performance.

5. AI Usage & Feedback Tracking

Tracks frequency and nature of AI suggestion requests.
Collects feedback from users on AI-generated code for model improvement.
Allows the admin to update or retrain AI models (if applicable).

6. System Logs & Maintenance Panel

Provides real-time system health updates (CPU, memory usage, API latency).
Logs errors, crashes, or failed executions for debugging and system maintenance.
Offers tools for clearing logs, managing storage, and scheduling updates.

7. Content & Feature Management (Optional)

Admins can add announcements, update UI content, or manage help documentation.
Feature toggles for enabling/disabling AI suggestions or debug features during maintenance.

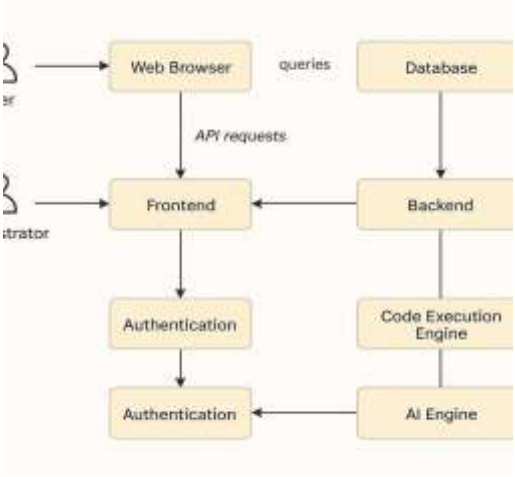


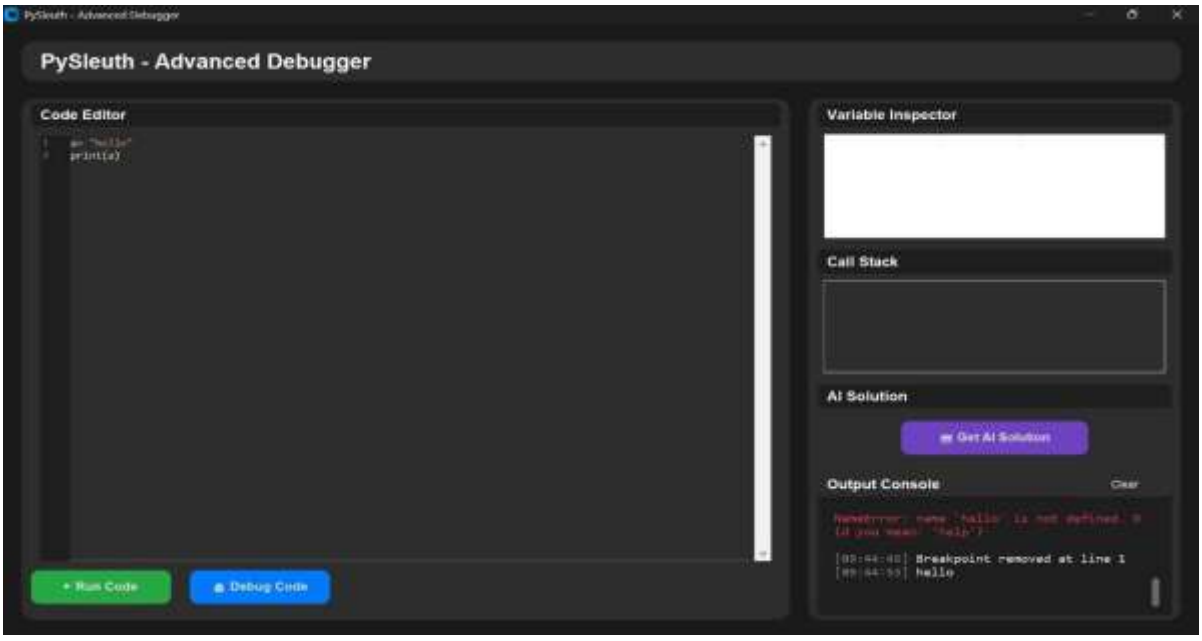
Fig 1 – System architecture

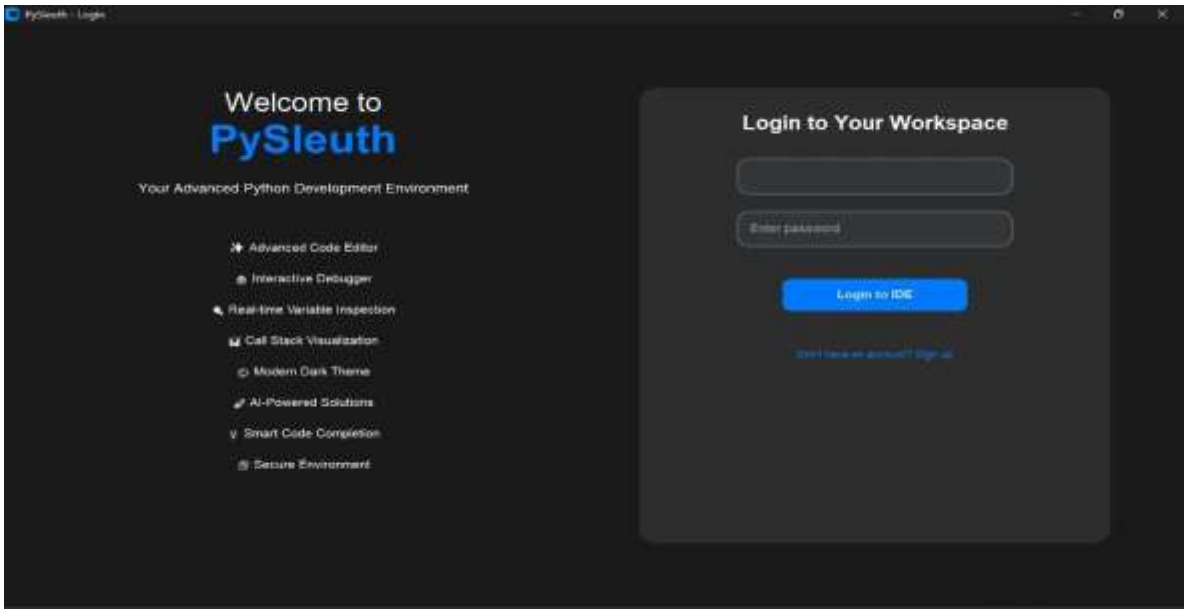
4. RESULTS & DISCUSSION

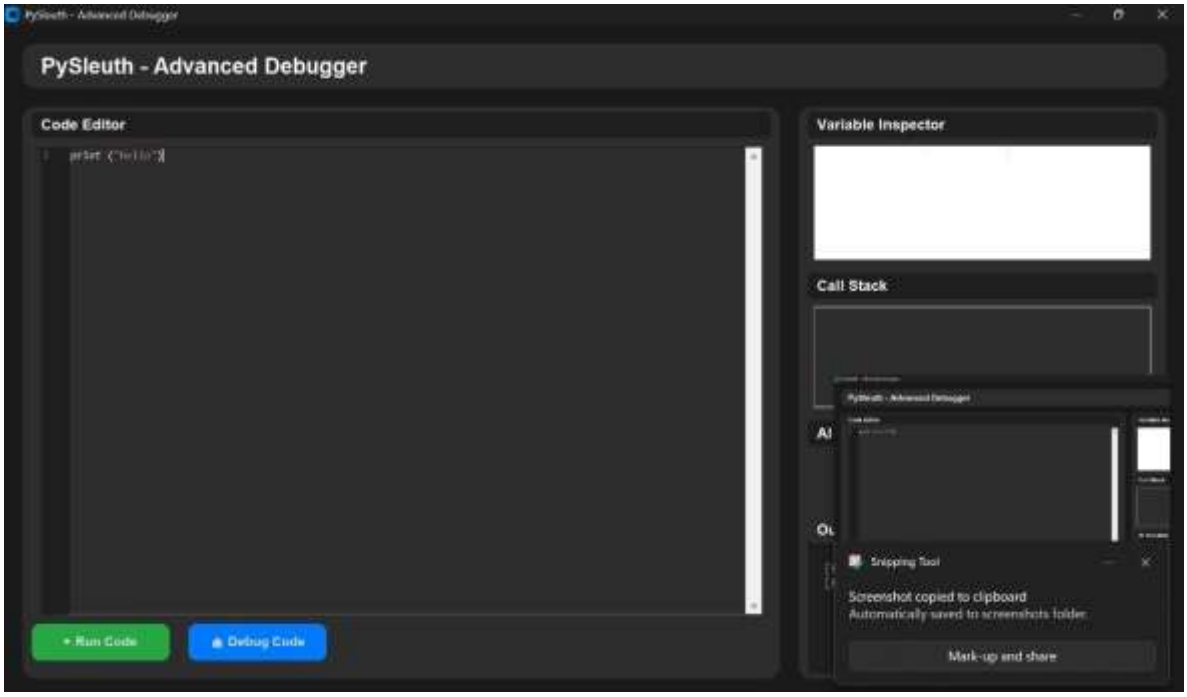
The developed Python-based debugger was tested on a range of Python scripts and successfully detected common runtime errors like IndexError, TypeError, and KeyError. It allowed step-by-step execution, real-time variable tracking, and provided clear error messages, making the debugging process smoother and more transparent.

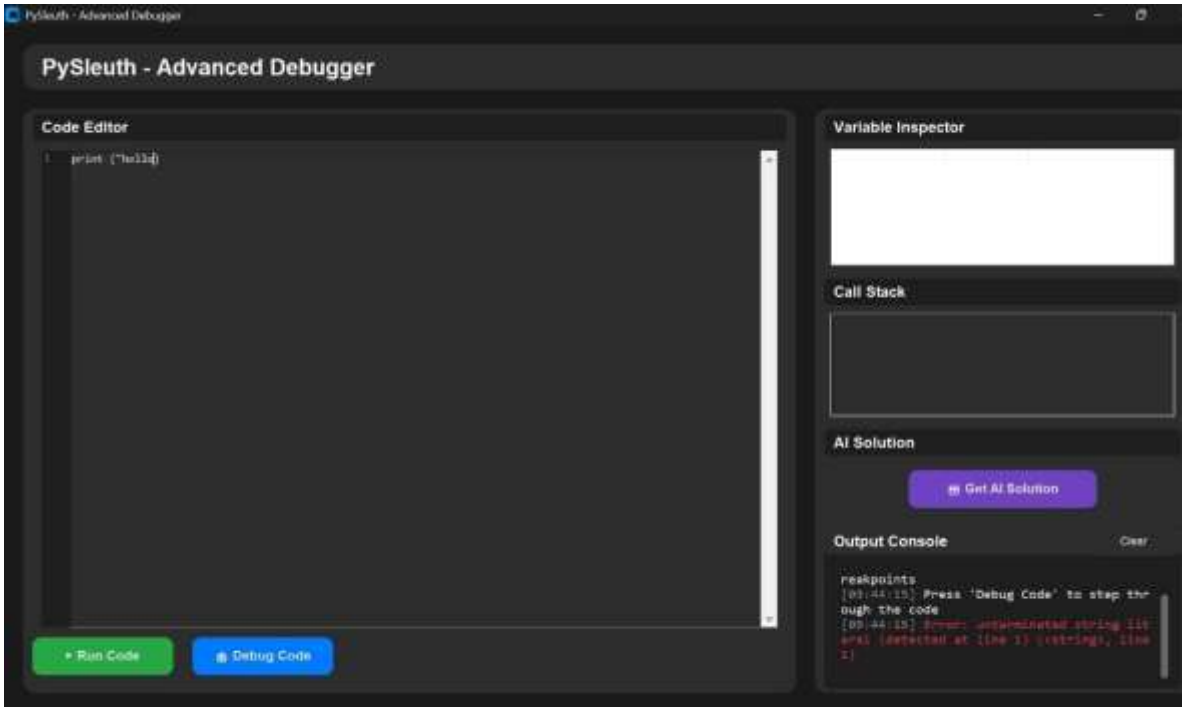
Users could easily monitor the flow of execution and inspect variable states at each step. The tool performed efficiently with minimal overhead, though slight slowdowns were noticed during intensive computations.

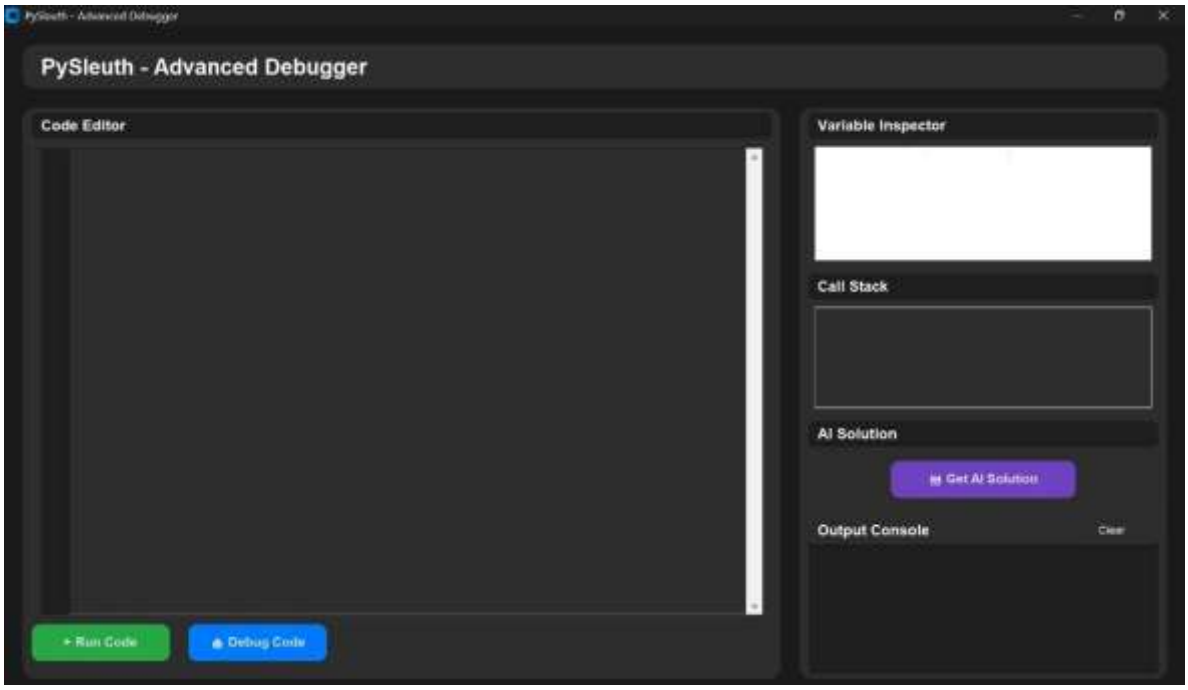
Overall, the debugger proved effective for educational and development purposes, especially for beginners. Future improvements could include a better UI, support for multithreading, and advanced features like memory tracking to enhance its usability and performance.

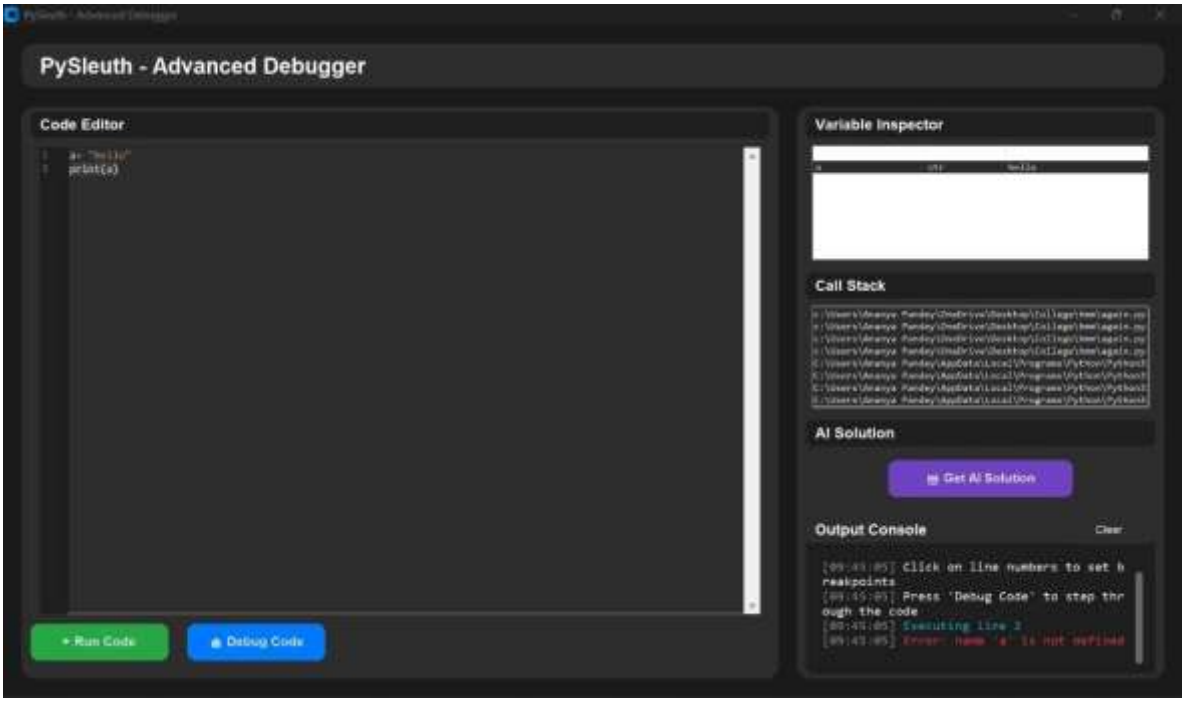

PySleuth - Advanced Debugger
Code Editor
Variable Inspector
Call Stack
AI Solution
Get AI Solution
Output Console
Clear
Run Code
Debug Code


Welcome to
PySleuth
Your Advanced Python Development Environment
Advanced Code Editor
Interactive Debugger
Real-time Variable Inspection
Call Stack Visualization
Modern Dark Theme
AI-Powered Solutions
Smart Code Completion
Secure Environment
Login to Your Workspace
Enter password
Login to IDE


PySleuth - Advanced Debugger
Code Editor
Variable Inspector
Call Stack
AI
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Mark-up and share
Run Code
Debug Code


PySleuth - Advanced Debugger
Code Editor
Variable Inspector
Call Stack
AI Solution
Get AI Solution
Output Console
Clear
Run Code
Debug Code


PySleuth - Advanced Debugger
Code Editor
Variable Inspector
Call Stack
AI Solution
Get AI Solution
Output Console
Clear
Run Code
Debug Code


PySleuth - Advanced Debugger
Code Editor
Variable Inspector
Call Stack
AI Solution
Get AI Solution
Output Console
Clear
Run Code
Debug Code

CONCLUSIONS

The Python-based debugger developed in this project offers a practical and user-friendly solution for identifying and resolving code errors. It simplifies the debugging process through step-by-step execution and real-time variable tracking, making it especially useful for learners and developers. With further enhancements in performance and interface design, the tool has the potential to evolve into a comprehensive debugging platform for Python applications.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to MGM College of Enginerring & Technology for providing the necessary infrastructure and support to carry out this project.

Special thanks to Prof. Babasaheb Waghmode, whose guidance and feedback were instrumental throughout the development and research process. We are also grateful to the users and testers who provided valuable insights during the evaluation phase.

REFERENCES

1. Van Rossum, G., & Drake, F. L. (2009). The Python Language Reference Manual. Network Theory Ltd.

2. Brett Cannon. (2020). Debugging Techniques in Python. Python Software Foundation. Retrieved from https://docs.python.org/3/library/pdb.html