ExecExam: A Tool to Facilitate Effective Executable Examinations in Python

Pallas-Athena Cain, Hemani Alaparthi, and Gregory Kapfhammer

May 15, 2025

What is an Executable Examination?

⚙️ **Goal: Assess a student's ability to program with real tools**

- A student writes, modifies, and runs code to solve a real problem
- Graded via automated tests that use Pytest tests and assertions
- Unlike static examinations an executable examination assesses:
    - Programming logic
    - Debugging ability
    - Tool use (e.g., text editor, terminal, IDE, and Git)

🎯 **Like a take-home project — but precise, consistent, and scalable!**

Reference: Chris Bourke, Yael Erez, and Orit Hazzan. 2023. "Executable Exams: Taxonomy, Implementation and Prospects". In Proceedings of 54th SIGCSE.

Problems with Computing Assessments

👩‍🏫 Why do we need better assessments?

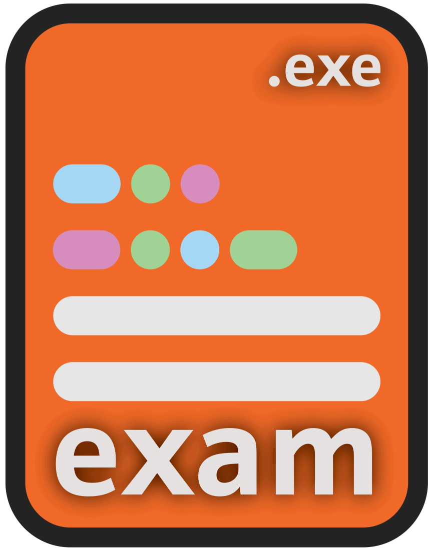
- Manual grading is slow and inconsistent
- Students often don’t know why their code fails
- Feedback is shallow or missing altogether
- Limited assessment of effective tool use
- Pytest not a good fit for assessment

🚫 **Test assertion failure is not enough**! ExecExam is a compelling alternative to either manual assessment or running only Pytest.

What is ExecExam?

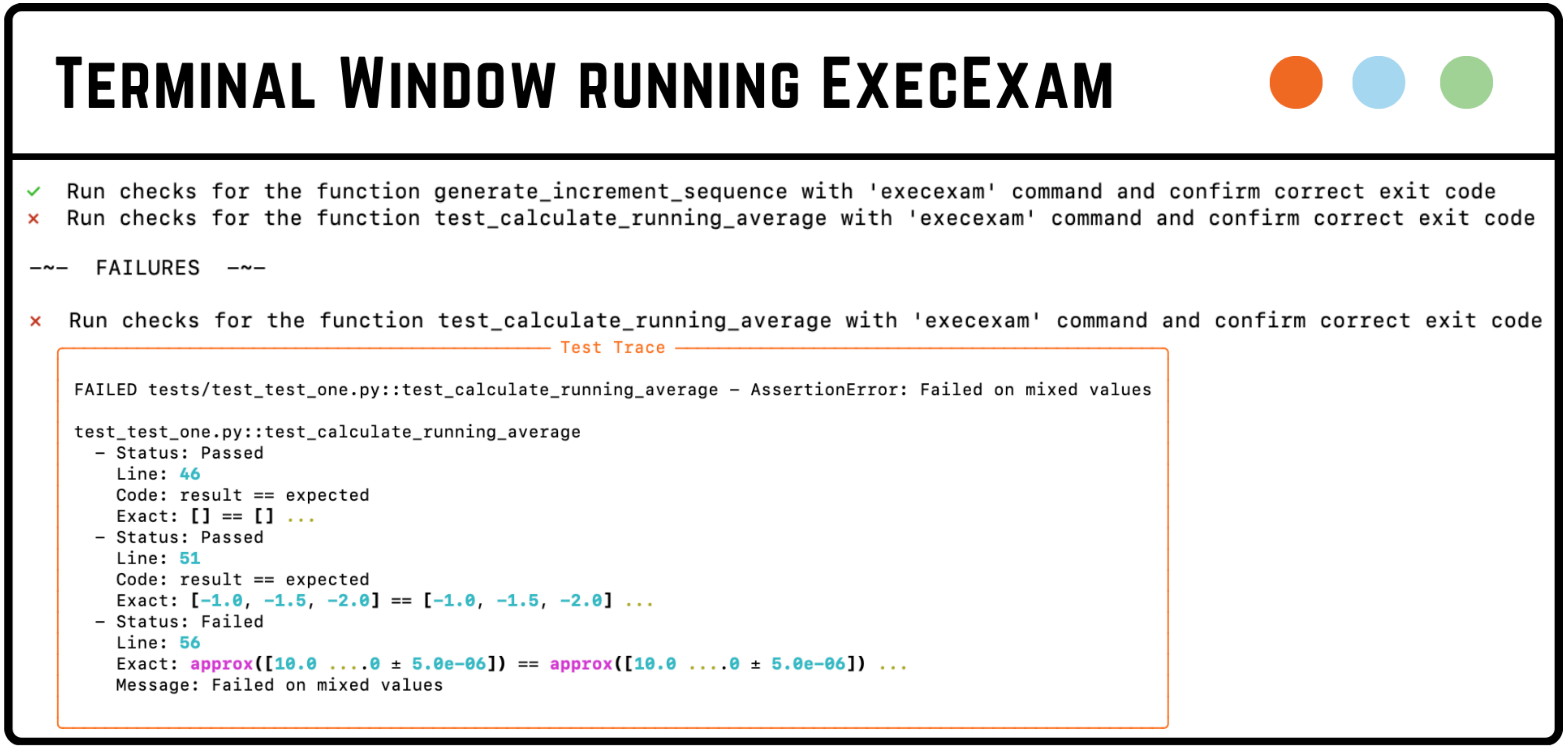
Scalable, feedback-rich assessment tool built in Python

- Runs Pytest tests on student code
- Reports all test failures and context
- Clearly explains why a test failed
- Suggests how to fix tested function
- Uses LLMs for enhanced feedback



Next Step: Explore ExecExam's features and how teachers can integrate them into the assessments for their programming courses!

Understanding ExecExam's Output



```
✓  Run checks for the function generate_increment_sequence with 'execexam' command and confirm correct exit code
✗  Run checks for the function test_calculate_running_average with 'execexam' command and confirm correct exit code

-~-  FAILURES  -~-

✗  Run checks for the function test_calculate_running_average with 'execexam' command and confirm correct exit code
  ---------------------------------------- Test Trace ----------------------------------------

  FAILED tests/test_test_one.py::test_calculate_running_average - AssertionError: Failed on mixed values

  test_test_one.py::test_calculate_running_average
    - Status: Passed
      Line: 46
      Code: result == expected
      Exact: [] == [] ...
    - Status: Passed
      Line: 51
      Code: result == expected
      Exact: [-1.0, -1.5, -2.0] == [-1.0, -1.5, -2.0] ...
    - Status: Failed
      Line: 56
      Exact: approx([10.0 ....0 ± 5.0e-06]) == approx([10.0 ....0 ± 5.0e-06]) ...
      Message: Failed on mixed values
```

Key Features of ExecExam

💡 **Why use ExecExam for your next assessment?**

- 🧪 Configured Pytest runs for streamlined assessment
- 💻 Runs on student laptop through assessment process
- 📜 Provides contextualized, detailed test failure reports
- ⚙️ Integrates with GitHub and GitHub Actions for CI/CD
- 🧠 Features flexible, democratized LLM-powered debugging
- 🔁 Offers actionable insights to instructors and students!
- 🛠️ Open-source tool collaboratively developed on GitHub

Getting Started with ExecExam

⚙ **How instructors can adopt automated assessments**

- Create a **solution repository**
  - Design scaffolded coding tasks
  - Write test cases using Pytest
  - Add ExecExam as a dependency
  - Use GatorGrader to run all checks
- Using **solution ablation** to create a **starter repository**
- GitHub Classroom **distributes** and **receives** examinations

Conclusions and Future Work

- 📊 **Analytics and Instructor Features**
  - Store test outcomes and feedback over time
  - Visualize student debugging and improvement paths
  - Log LLM interactions to evaluate effectiveness
  - Hold out hidden test cases for instructor-only grading
- 🧠 **Adaptive Feedback Loops**
  - Tailor feedback complexity to student performance
  - Allow students to rate different types of LLM feedback

- 🔗 GitHub Repository: https://github.com/GatorEducator/execexam
- 💻 PyPI: https://pypi.org/project/execexam/