

Ministry of Education of the Russian Federation

Penza State University

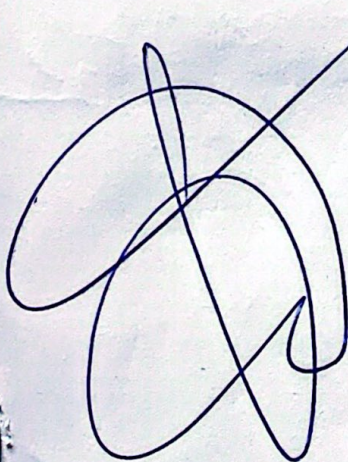
Department of Computer Engineering

Explanatory note:

To the course work:

In the course: "Programming"

On the topic: "Scientific Calculator"



Handwritten signature of the student, OSAMA MOHAMED, in blue ink.

Performed by a student of the 24VVA1 group:

OSAMA MOHAMED

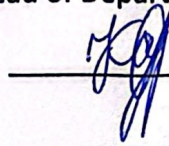
Candidate of Technical Sciences, Professor, Deputy Dean of the Faculty

Mr. Роман Федюнин

✓

PENZA STATE UNIVERSITY
FACULTY OF COMPUTER ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

Head of Department of Technical Education



Mrs : Karamsheva

"14" 10 2026

Task

Course project assignment for the course

Programming

Project topic: development of a complex structure program using top-down programming.

Initial data (technical requirements) for the design

Scientific calculator is a desktop application built using Python, aimed at executing a variety of complex functions and automating other simple calculations.

Mandatory requirements for the program:

- 1-Use of built-in Python data types: string, integer, Boolean and list.
- 2-Screensaver.
- 3-Graphical operating mode of the video system.
- 4-Menu and toolbar.
- 5-MS Windows as the target operating system.
- 6-PyCharm as the development environment.



Scope of work for the course

Alghorithm part
Program development.

The grafical part

Data diagram, system resource diagram,
system operation diagram, hierarchical
structure of the program, program
interaction diagram.

Experimental part

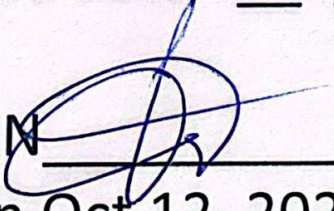
Debugging the program.

Project completion time by section

According to schedule.

The date of issue of the task is "October
12", 2026.

project defense date * 14 * 10.2026

Teacher Fediunin R.M. 

Received the task on Oct 12, 2026.

Student Ahmed Osama 



Table of Contents

1. Introduction	3
1.1 Purpose & Scope	3
1.2 Target Users	3
1.3 Key Features at a Glance	3
2. System Overview	3
2.1 High-Level Architecture (GUI, Core, Math Engine)	3
2.2 Data Flow: Input, Tokeniser, AST, Evaluator, Display	4
2.3 Precision & Rounding Strategy	4
3. Technology Stack	4
3.1 Language & Version: Python 3.11+	4
3.2 GUI Toolkit: Tkinter (or PyQt6)	4
3.3 Math Libraries: decimal, cmath, math (optional mpmath)	4
3.4 Persistence: JSON flat file (no RDBMS)	5
3.5 Packaging: PyInstaller single executable	5
4. Code Structure	5
4.1 Package Layout / Folder Map	5
4.2 Core Classes & Responsibilities	5
4.3 Utility Modules: constants, exceptions, helpers	5
5. Functionalities	6
5.1 Expression Entry (algebraic mode)	6
5.2 Basic Arithmetic with Operator Precedence	6
5.3 Scientific Functions	6
5.4 Complex-Number Mode	6
5.5 Base Conversions (DEC/HEX/BIN/OCT)	6
5.6 Unit Conversions (length, mass, temperature)	6
5.7 Last-Answer (ANS)	6
5.8 History Tape & Session Persistence	6
5.9 Keyboard-Only Workflow & Global Shortcuts	7
6. Data Design	7
6.1 In-Memory Models (Decimal, complex, deque)	7
6.2 JSON Schema for History & Settings	7
7. User-Interface Design	7
7.1 Layout Sketches (resizable button grid, display line, side panels)	7
7.2 Themes: Light / Dark / High-Contrast	7
7.3 Accessibility: font scaling, colour-blind palette, screen-reader labels	7
8. Error Handling & Validation	8
8.1 Syntax Errors (friendly cursor hint)	8
8.2 Math Errors (division by zero, overflow, domain)	8
8.3 Undo/Redo Failure Paths	8
9. Security & Reliability	8
9.1 Input Sanitisation (no eval(), no code injection)	8
9.2 Safe Serialization (JSON schema validation)	8
9.3 Crash Reporting (optional opt-in telemetry)	8
10. Testing Strategy	8
10.1 Unit Tests: tokeniser, AST, evaluator	8
10.2 GUI Tests: QtTest or pytest-qt	9
10.3 Precision Tests: reference values from IEEE-754 & WolframAlpha	9
11. Performance & Portability	9
11.1 Startup Time Budget (< 2 s on 4 GB RAM)	9
11.2 Memory Footprint (< 150 MB peak)	9
11.3 Cross-Platform CI: Windows, macOS, Ubuntu	9



12. Future Enhancements.....	10
12.1 Mini-CAS (symbolic simplification via sympy)	10
12.2 2-D Plotter (embedded matplotlib canvas).....	10
12.3 Plugin API (user-defined functions & buttons).....	10
13. Conclusion.....	10
References.....	10
Appendix A. Complete Source Code.....	11
Appendix B. Results of the Code	19

List of Figures

Figure 1 – Interaction between the user, the GUI and the math engine.....	4
Figure 2 – Main window of PySciCalc ($7 + 6 = 13$)	19
Figure 3 – Main window with the History window open.....	19

12. Future Enhancements

12.1 Mini-CAS (symbolic simplification via sympy)

The sympy library could add simple symbolic simplification and equation solving.

12.2 2-D Plotter (embedded matplotlib canvas)

An embedded matplotlib canvas could plot functions of one variable.

12.3 Plugin API (user-defined functions & buttons)

A plugin interface would let users register their own functions and buttons. RPN mode, memory registers and rectangular-to-polar conversion are other candidate improvements.

13. Conclusion

PySciCalc shows that a complete scientific calculator can be built with the Python standard library alone. Its central design decision, a hand-written tokeniser, precedence-climbing parser and syntax-tree evaluator, gives full control over precision and error handling and removes the need for the unsafe `eval()` function.

The program meets the mandatory requirements of the task: it uses complex Python data types (strings, integers, Booleans, lists), a screensaver, a graphical interface with buttons and an input line, and it runs on Microsoft Windows. During the review of the source code several defects were corrected, including a crash in the screensaver, a temperature conversion that mixed Decimal and float values, missing undo and redo methods, and the use of `eval()` in the base-conversion routine.

Future work will concentrate on RPN mode, memory registers, a plotter and a plugin interface.

References

- [1] W3Schools. Python Tutorial. <https://www.w3schools.com/python/>
- [2] Python Software Foundation. decimal: Decimal fixed point and floating point arithmetic. <https://docs.python.org/3/library/decimal.html>
- [3] Python Software Foundation. cmath: Mathematical functions for complex numbers. <https://docs.python.org/3/library/cmath.html>
- [4] Python Software Foundation. tkinter: Python interface to Tcl/Tk. <https://docs.python.org/3/library/tkinter.html>