# Run the interactive UI

## Theory of Everything Documentation

## Introduction

y and # Theory of Everything - Modular API This repository contains a modular API for interacting with the Theory of Everything, providing both human-friendly interfaces and advanced tools for AI agents.

## Overview

The Theory of Everything API is designed with a modular architecture, consisting of small, focused components that can be used independently or combined through the unified interface. This approach provides flexibility, maintainability, and extensibility.

## Modules

The API consists of the following modules: ### Core Module (`toe_core.py`) The core module provides essential functionality used by all other modules, including: - Math module conflict handling - File operations - Script execution utilities - JSON handling ### Formula Module (`toe_formulas.py`) The formula module provides tools for working with mathematical formulas, including: - Formula retrieval - Formula exploration - Formula comparison - Formula search - Formula dependency analysis - LaTeX export ### Visualization Module (`toe_vis.py`) The visualization module provides tools for generating visualizations, including: - Visualization generation - Parameter validation - Parameter suggestions - Batch processing ### Agent Tools Module (`toe_agent.py`) The agent tools module provides specialized tools for AI agents, including: - Session management - Theory exploration - Visualization sequence generation - Insight extraction ### UI Module (`toe_ui.py`) The UI module provides a command-line interface for human users, including: - Interactive menus - Formula exploration - Visualization generation - Formula comparison ### Unified API (`toe_unified.py`) The unified API combines all modules into a single interface, providing: - Access to all functionality through a consistent interface - Agent mode for advanced features - Command-line interface for scripting

## Usage

### For Human Users Human users can interact with the API through the command-line interface: ```bash python toe_ui.py python toe_unified.py formula get unified_action python toe_unified.py visualization generate 4d_spacetime_curvature --show ``` ### For AI Agents AI agents can interact with the API programmatically: ```python from toe_unified import ToEUnified api = ToEUnified(agent_mode=True) exploration = api.explore_theory() result = api.generate_visualization_for_formula("gravity_action") insights = api.extract_insights(formula_name="gravity_action") ``` See `agent_advanced_example.py` for more examples of how AI agents can use the API.

## Advanced Features for Agents

The API provides several advanced features specifically designed for AI agents: ### Session Management Agents can create and manage sessions to track their interactions with the API: ```python session_info = api.get_session_info() ``` ### Theory Exploration Agents can explore the Theory of Everything at a high level: ```python exploration = api.explore_theory() results = api.explore_theory(query="gravity") ``` ### Visualization Sequences Agents can generate sequences of visualizations by varying parameters: ```python param_ranges = { "mass": [0.5, 1.0, 2.0, 5.0] } sequence = api.generate_visualization_sequence("4d_spacetime_curvature", param_ranges) ``` ### Insight Extraction Agents can extract insights from formulas and visualizations: ```python insights = api.extract_insights(formula_name="gravity_action") insights = api.extract_insights(visualization_name="4d_spacetime_curvature") ``` ### Batch Processing Agents can generate multiple visualizations in batch: ```python configs = [ {"name": "4d_spacetime_curvature", "params": {"mass": 1.0}}, {"name": "quantum_foam_3d", "params": {"amplitude": 0.7}} ] results = api.batch_generate_visualizations(configs) ``` ### Formula Exploration Agents can explore formulas and their relationships: ```python exploration = api.explore_formula("unified_action") dependencies = api.get_formula_dependencies("unified_action") comparison = api.compare_formulas(["gravity_action", "matter_action"]) ``` ### LaTeX and PDF Generation Agents can generate LaTeX and PDF documentation: ```python from latexagent import LaTeXAgent from pdfagent import PDFAgent latex_agent = LaTeXAgent() latex_content = latex_agent.generate_latex(formula="unified_action", include_components=True) latex_agent.save_latex(latex_content, "unified_action.tex") pdf_agent = PDFAgent() pdf_path = pdf_agent.generate_pdf(formula="unified_action", output="unified_action.pdf") ``` The LaTeX and PDF agents can also be used from the command line: ```bash python latexagent.py --formula unified_action --output unified_action.tex python pdfagent.py --formula unified_action --output unified_action.pdf python latexagent.py --list-formulas python latexagent.py --help python pdfagent.py --help ```

## Examples

The repository includes several example scripts: - `agent_advanced_example.py`: Demonstrates how AI agents can use the advanced features of the API - `toe_ui.py`: Provides an interactive command-line interface for human users - `toe_unified.py`: Provides a command-line interface for scripting

## Tests

The repository includes a comprehensive test suite in the `tests` directory. These tests are designed to verify the functionality of each module and can be run individually or as a complete suite. ```bash python tests/run_tests.py python tests/run_tests.py imports python tests/run_tests.py core python tests/run_tests.py modules python tests/run_tests.py formula python tests/run_tests.py visualization python tests/run_tests.py agent ``` The test suite includes: - **Import Tests**: Test importing all modules - **Core Tests**: Test the core functionality like math module conflict handling and file operations - **Module Tests**: Test importing and creating instances of all modules - **Formula Tests**: Test the formula functionality including retrieval, exploration, and comparison - **Visualization Tests**: Test the visualization functionality including parameter validation and generation - **Agent Tests**: Test the agent-specific features like session management and insight extraction - **LaTeX Tests**: Test the LaTeX agent functionality for generating LaTeX documentation - **PDF Tests**: Test the PDF agent functionality for generating PDF documentation **Note**: Some tests may require additional setup or dependencies. If you encounter issues running the tests, please check the individual test files for specific requirements.

## Installation

No installation is required. Simply clone the repository and run the scripts.

## Requirements

- Python 3.6 or higher - NumPy - Matplotlib

## License

This project is licensed under the MIT License - see the LICENSE file for details.