Title: Evaluating Python Static Code Analysis Tools Using FAIR Principles
Author: O. N. Zane et al.
Publication: IEEE, 2024
Summary: This paper assesses various Python static analysis tools against FAIR principles (Findable, Accessible, Interoperable, Reusable). It benchmarks tools like Pylint, Flake8, and Bandit, highlighting their strengths and weaknesses in maintainability, transparency, and data availability.
Takeaways: Helps identify reliable static analysis tools for academic or industrial projects; emphasizes reproducibility and data accessibility in tool evaluation.

Title: Automated Assessment System for Programming Courses: A Case Study for Teaching Data Structures and Algorithms
Author: A. Ben Lakhdar et al.
Publication: Springer, 2023
Summary: The paper presents an automated assessment framework integrated into programming education to evaluate students' algorithmic problem-solving. It emphasizes automatic grading, code analysis, and instant feedback to enhance learning outcomes.
Takeaways: Demonstrates how automated feedback systems can improve students' understanding; relevant for feedback-based analysis in your project.

Title: Survey on Static Analysis Tools of Python Programs
Author: Gul et al.
Publication: CEUR Workshop Proceedings, 2020
Summary: Provides a comprehensive overview of static analysis tools for Python including Pylint, Pyflakes, and Mypy. Discusses their coverage, limitations, and evolving roles in code quality assurance.
Takeaways: Useful for identifying features and limitations of popular tools when designing a new static analysis framework.

Title: Franc: A Lightweight Framework for High-Quality Code Generation
Author: A. Ghosh et al.
Publication: IEEE, 2024
Summary: Franc introduces a modular framework for automated high-quality code generation using syntactic and semantic consistency checks. It applies static analysis concepts to ensure error-free output.
Takeaways: Highlights how lightweight frameworks can combine static analysis with intelligent generation; applicable to integrating rule-based checks.

Title: Unambiguity of Python Language Elements for Static Analysis
Author: S. Rytin et al.
Publication: IEEE, 2021
Summary: Investigates Python's syntax and semantics to identify ambiguities affecting static analysis. Suggests parsing strategies to handle dynamic features such as duck typing and runtime imports.
Takeaways: Improves understanding of Python's structural challenges, helping refine AST-based parsing in your tool.

Title: Towards More Sophisticated Static Analysis Methods of Python Programs
Author: T. Hall and M. Karlsen
Publication: IEEE, 2020
Summary: Explores novel static analysis techniques tailored for Python's flexibility. The paper introduces modular rule-based checking and hybrid static-dynamic methods.
Takeaways: Shows potential for combining multiple analysis dimensions (syntax, complexity, performance) in educational tools.

Title: ExcePy: A Python Benchmark for Bugs with Python Built-in Types
Author: P. Jimenez et al.
Publication: IEEE, 2022

Summary: ExcePy provides a benchmark dataset of Python bugs involving built-in types to test static analysis tools' accuracy and detection power.
Takeaways: Offers insights into designing datasets or benchmarks for evaluating static analysis performance.

Title: Large Language Models (GPT) for Automating Feedback on Programming Assignments
Author: R. Xu et al.
Publication: arXiv, 2023
Summary: Examines the use of GPT models for generating automated, human-like programming feedback. Evaluates the accuracy, tone, and relevance of AI-generated feedback.
Takeaways: Demonstrates AI's role in complementing rule-based analysis for context-aware feedback.

Title: Automated Assessment in Programming Courses: A Case Study during the COVID-19 Era
Author: L. Martinez et al.
Publication: MDPI, 2020
Summary: Discusses the implementation of automated coding assessment systems during remote learning, focusing on reliability, fairness, and usability.
Takeaways: Reinforces the importance of automated evaluation for scalable, accessible learning environments.

Title: Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation
Author: K. Wang et al.
Publication: arXiv, 2023
Summary: Proposes a dual-model framework for generating and validating feedback similar to human tutors using LLMs like GPT-3.5 and GPT-4.
Takeaways: Illustrates how dual AI systems can refine feedback quality; relevant for designing explainable feedback in your system.

Title: Enhancing Programming Education with ChatGPT: A Case Study on Student Perceptions and Interactions in a Python Course
Author: M. Ali et al.
Publication: arXiv, 2024
Summary: Analyzes how students perceive ChatGPT as an educational assistant for learning Python. Finds improvement in engagement but notes over-reliance risks.
Takeaways: Highlights the educational role of AI assistants and the need for balance between automation and user control.