**Week 3 Lesson Plan: Scaling Complexity & Algorithmic Thinking**

**1. Overview:**

This session builds on debugging skills from Week 2, introducing structured problem-solving approaches for more complex programming tasks. Students will work with AI-assisted code generation and debugging to implement algorithmic patterns that improve efficiency and scalability.

**2. Learning Goals & Key Principles:**

By the end of this session, students will be able to:

* Apply AI-assisted code generation to **structured problem-solving.**
* Understand and implement **loops, conditionals, and recursion.**
* Optimize AI-generated code for **efficiency and scalability.**
* Develop iterative solutions for **increasingly complex problems.**

Key principles include:

* **Breaking Down Complex Problems** – Structuring solutions into smaller steps.
* **Algorithmic Efficiency** – Reducing redundant code and improving runtime.
* **AI as a Co-Pilot** – Refining AI-generated code for complex logic.

**3. Mini-Lecture: Algorithmic Thinking & AI Assistance (Step-by-Step)**

**(20 min total)**

1. **From Debugging to Structured Problem-Solving** (5 min)
   * The transition from fixing AI-generated code to designing optimized solutions.
   * Why structured approaches improve AI-assisted programming.
2. **Fundamentals of Algorithmic Thinking** (10 min)
   * Loops & conditionals: Automating repetitive tasks.
   * Recursion & efficiency: Reducing manual logic with structured functions.
3. **Optimizing AI Code for Scalability** (5 min)
   * How to refine AI-generated scripts for larger datasets and dynamic problems.

**4. Hands-on Coding Activities: Scaling Complexity**

**(60 min total, divided into three core tasks)**

**Task 1: Automating Repetitive Processes (20 min)**

* Students prompt AI to generate code that automates data transformations.
* **Focus:** Using loops and conditionals to handle large datasets.
* **Discussion:** How well does AI handle automation? What optimizations are needed?

**Task 2: Recursion & Efficiency (20 min)**

* Students generate recursive functions using AI prompts.
* **Focus:** Understanding recursion in problem-solving (e.g., factorial, Fibonacci, tree traversal).
* **Discussion:** When is recursion better than loops? What inefficiencies arise?

**Task 3: Optimizing AI-Generated Code (20 min)**

* Students analyze AI-generated scripts for inefficiencies.
* **Focus:** Refactoring and reducing redundant code.
* **Discussion:** What improvements make the code scalable?

**5. Wrap-Up & Reflection (10 min)**

* **Group Discussion:** Lessons learned about AI-generated algorithms.
* **Key Takeaways:** How structured problem-solving improves efficiency.
* **Looking Ahead to Week 4:** Applying AI-assisted techniques to real-world problems.

**Appendix: Additional Exercises**

**Analyzing Algorithmic Efficiency**

1. **Big O Complexity Estimation:**
   * Students estimate the time complexity of AI-generated solutions.
   * Compare iterative vs. recursive implementations.
2. **Comparing AI-Generated vs. Handwritten Solutions:**
   * Students write an algorithm without AI assistance.
   * Compare efficiency and structure to an AI-generated version.

**Advanced Problem-Solving Challenges**

1. **Multi-Step Data Processing Pipelines:**
   * AI generates a step-by-step data processing script.
   * Students optimize and modularize the code.
2. **AI-Generated Algorithm Debugging:**
   * Instructor provides faulty AI-generated algorithms.
   * Students debug and refactor for efficiency.

**Outcome:**

By the end of this session, students will:

* Develop structured approaches for handling complex programming challenges.
* Improve AI-generated code by applying loops, conditionals, and recursion.
* Recognize and optimize inefficient AI-assisted solutions.
* Prepare for Week 4’s applied problem-solving and final projects.