**Week 1 Mini-Lecture: Introduction to AI-Assisted Coding & Structured Problem-Solving**

### **1. Overview**

This mini-lecture introduces students to the concept of AI-assisted programming, the role of structured problem-solving, and how to effectively use AI tools to generate code. Students will explore the benefits and limitations of AI-generated scripts and how structured problem-solving enhances AI-assisted development.

### **2. Learning Objectives**

By the end of this session, students will:

* Understand the role of AI in programming and structured problem-solving.
* Learn effective techniques for AI-assisted code generation.
* Recognize common AI-generated errors and how to critically evaluate AI outputs.
* Apply structured approaches to improving AI-generated scripts.

### **3. Key Concepts**

#### **What is AI-Assisted Programming?**

* AI-assisted coding refers to using tools like ChatGPT, Copilot, or other AI-powered generators to write, optimize, and debug code.
* AI acts as a co-pilot, providing suggestions, automating repetitive tasks, and accelerating development workflows.
* While AI can generate working code, human oversight is required to ensure correctness, efficiency, and security.

#### **The Importance of Structured Problem-Solving**

* AI can only assist effectively when given clear, structured instructions.
* A structured approach involves:
  + **Breaking down problems** into smaller tasks.
  + **Providing specific prompts** to AI tools.
  + **Evaluating AI-generated solutions** for logic, efficiency, and accuracy.
* Without structure, AI-generated outputs may be incorrect, inefficient, or irrelevant.

### **4. Step-by-Step Guide to AI-Assisted Code Generation**

#### **Step 1: Framing the Problem**

* Clearly define the problem before asking AI for code.
* Example: Instead of asking “Write a Python script for data analysis,” ask:
  + “Write a Python script that loads a CSV file, cleans missing values, and calculates summary statistics using Pandas.”
* The more specific the prompt, the better the AI-generated solution.

#### **Step 2: Generating Code with AI**

* Use AI tools to generate initial code based on structured prompts.
* Example Prompt:
  + “I have a dataset with air quality measurements (Date, PM2.5, PM10, CO). Generate a Python script that:
    - Loads the dataset,
    - Removes missing values,
    - Plots a time-series graph of PM2.5.”
* AI will return a structured script based on the provided details.

#### **Step 3: Evaluating AI-Generated Code**

* Check for **logic errors, inefficiencies, and missing details.**
* Does the code:
  + Use proper data handling techniques?
  + Include error handling for edge cases?
  + Follow best practices for readability and efficiency?
* Example: AI might drop all missing values instead of imputing them, leading to unintended data loss.

#### **Step 4: Refining & Debugging the Code**

* Iterate on AI-generated outputs by:
  + Improving prompts for better AI responses.
  + Debugging syntax and logic errors.
  + Optimizing performance through vectorized operations instead of loops.
* Example: If AI generates a slow for-loop, refine the request by asking for a vectorized Pandas approach.

### **5. Common AI-Generated Errors & Debugging Strategies**

#### **1. Logic Errors**

* AI sometimes generates incorrect logic, such as using > instead of < in conditions.
* **Debugging Tip:** Manually verify calculations and conditionals.

#### **2. Inefficient Loops Instead of Vectorized Operations**

* AI may generate inefficient for-loops for data transformations instead of using Pandas’ built-in functions.
* **Debugging Tip:** Request an optimized version of the code or rewrite using vectorized methods.

#### **3. Missing Error Handling**

* AI-generated code may not anticipate edge cases (e.g., missing values, incorrect data types).
* **Debugging Tip:** Add try-except blocks and manual error handling.

### **6. Wrap-Up & Takeaways**

* AI is a powerful assistant, but it requires **structured input and human oversight.**
* Structured problem-solving improves the accuracy and reliability of AI-generated code.
* Debugging and iterative refinement are essential when using AI-assisted coding.
* Next, students will **generate their own AI-assisted scripts** and evaluate them for accuracy and efficiency before debugging in Week 2.