**Week 4 Mini-Lecture: Applied Problem Solving & AI-Assisted Development**

### **1. Overview**

This mini-lecture focuses on applying AI-assisted programming techniques to real-world problem-solving. Students will integrate everything they’ve learned—structured problem-solving, debugging, and optimization—to develop robust, scalable solutions using AI-generated code.

### **2. Learning Objectives**

By the end of this session, students will:

* Apply AI-assisted coding to real-world data science and software development tasks.
* Develop structured problem-solving workflows for AI-generated projects.
* Optimize AI-generated code for readability, efficiency, and maintainability.
* Reflect on AI’s strengths and limitations in software development.

### **3. Key Concepts**

#### **What is Applied Problem Solving?**

* Applied problem-solving involves using programming to solve **real-world challenges** such as data analysis, automation, and decision-making.
* AI-assisted coding helps automate repetitive tasks, suggest solutions, and optimize workflows.
* Students must critically evaluate AI-generated code to ensure accuracy, efficiency, and ethical considerations.

#### **Developing Structured Workflows for AI-Assisted Projects**

* **Define the problem:** Clearly outline the project’s goals and expected outcomes.
* **Break down the problem into steps:** Identify core tasks (data loading, transformation, visualization, etc.).
* **Use AI as an assistant, not a replacement:** Guide AI with structured prompts and evaluate its output.
* **Refine & optimize:** Debug, optimize, and document AI-generated solutions.

### **4. Best Practices for AI-Assisted Development**

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

* Example: Instead of asking AI, "Write a Python script to analyze financial data," refine the prompt:
  + "Generate a Python script that:
    - Loads financial transaction data from a CSV file.
    - Filters transactions based on user-defined conditions.
    - Aggregates spending by category and visualizes trends."

#### **Step 2: Evaluating AI-Generated Code**

* **Check for logical errors:** AI can generate incorrect logic that needs human validation.
* **Assess efficiency:** Replace inefficient loops with optimized methods (e.g., vectorized Pandas operations).
* **Ensure proper documentation:** AI-generated code may lack meaningful comments and structure.

#### **Step 3: Debugging & Iterating on AI Solutions**

* **Run small tests first:** Validate individual functions before executing the entire script.
* **Refine AI prompts based on output quality:** If the script is inefficient or incorrect, rephrase the prompt.
* **Use version control:** Track changes and iterate using Git or another versioning system.

### **5. Hands-On Example: AI-Generated vs. Optimized Code**

#### **AI-Generated Code (Suboptimal)**

import pandas as pd

def analyze\_sales(file\_path):

df = pd.read\_csv(file\_path)

total\_sales = 0

for sale in df['amount']:

total\_sales += sale # Inefficient looping

return total\_sales

#### **Optimized Code (Vectorized Approach)**

import pandas as pd

def analyze\_sales(file\_path):

df = pd.read\_csv(file\_path)

return df['amount'].sum() # More efficient vectorized approach

### **6. Ethical Considerations in AI-Assisted Development**

* **Bias in AI-Generated Code:** AI may introduce biases based on training data. Validate outputs before deployment.
* **Security Risks:** AI-generated scripts may lack proper security practices (e.g., input validation).
* **Dependency Awareness:** Over-reliance on AI can hinder deeper understanding of programming concepts.

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

* AI-assisted coding is a powerful tool, but human oversight is essential.
* Structured problem-solving enhances AI-generated solutions.
* Debugging and iterative refinement ensure robust, optimized code.
* Next, students will complete their final projects and present AI-assisted solutions for review.