**Week 2 Mini-Lecture: Debugging & Iterative Thinking**

### **1. Overview**

This mini-lecture introduces students to the fundamentals of debugging AI-generated code, identifying common errors, and applying structured debugging techniques. Students will also learn how iterative thinking helps refine code through multiple cycles of improvement.

### **2. Learning Objectives**

By the end of this session, students will:

* Understand the importance of debugging and iterative thinking in programming.
* Identify and classify common AI-generated coding errors.
* Apply structured debugging techniques to improve AI-generated scripts.
* Use iterative approaches to refine and optimize code.

### **3. Key Concepts**

#### **What is Debugging?**

* Debugging is the process of identifying and fixing errors (bugs) in code.
* Common types of errors include:
  + **Syntax Errors** – Incorrect code structure (e.g., missing colons, incorrect indentation).
  + **Logical Errors** – Code runs but produces incorrect results.
  + **Runtime Errors** – Errors that occur during execution (e.g., division by zero, missing variables).
* AI-generated code often contains logical inefficiencies that require human intervention.

#### **What is Iterative Thinking?**

* Iterative thinking involves refining code through multiple cycles of testing and improvement.
* Rather than seeking a perfect solution immediately, programmers:
  1. Generate an initial solution.
  2. Test and debug the code.
  3. Optimize for efficiency and accuracy.
  4. Repeat the process until the solution meets desired performance.

### **4. Step-by-Step Debugging Process**

#### **Step 1: Identify the Problem**

* Carefully read error messages and warnings.
* Compare expected vs. actual output.
* Use print statements or logging to track variable states.

#### **Step 2: Isolate the Issue**

* Test code in small sections to pinpoint the source of the error.
* Use debugging tools (e.g., Python’s pdb or R’s debug() function).
* Check for incorrect assumptions in AI-generated logic.

#### **Step 3: Fix the Bug & Verify the Solution**

* Modify the faulty section of the code.
* Re-run tests to ensure the fix works without introducing new errors.
* Check if AI-generated logic needs structural improvements.

#### **Step 4: Optimize & Iterate**

* Look for inefficiencies (e.g., redundant loops, unnecessary computations).
* Refactor code for readability and maintainability.
* Use test cases to validate improvements.

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

#### **1. Incorrect Data Processing**

* AI-generated scripts may drop essential data instead of handling missing values correctly.
* **Debugging Tip:** Ensure AI suggestions align with best data-cleaning practices.

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

* AI often writes inefficient for-loops where vectorized functions (e.g., Pandas, NumPy, dplyr) are faster.
* **Debugging Tip:** Rewrite loops using optimized built-in functions.

#### **3. Poor Error Handling**

* AI-generated code may lack exception handling for user inputs and missing files.
* **Debugging Tip:** Add try-except (Python) or tryCatch() (R) blocks to handle edge cases.

### **6. Hands-On Example: Debugging AI-Generated Code**

#### **Faulty AI-Generated Code (Python)**

import pandas as pd

def load\_and\_clean\_data(file\_path):

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

df.dropna() # AI mistakenly drops all missing values instead of filling them

df['date'] = pd.to\_datetime(df['date']) # AI may not account for different date formats

return df

#### **Debugging & Optimized Solution**

import pandas as pd

def load\_and\_clean\_data(file\_path):

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

df.fillna(method='ffill', inplace=True) # Forward fill instead of dropping all missing values

df['date'] = pd.to\_datetime(df['date'], errors='coerce') # Handle invalid date formats

return df

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

* Debugging is an essential skill for improving AI-generated code.
* Iterative thinking leads to refined, more efficient solutions.
* AI-generated code is a starting point, but human oversight is required for quality assurance.
* Next, students will apply these debugging techniques to real AI-generated scripts.