Introduction to Software Development Week 8: Error Handling & Debugging

1. Learning Objectives

- Identify the three main types of programming errors.
- Use `try...except` blocks to gracefully handle runtime errors (exceptions).
- Implement common debugging strategies to find and fix logical errors.

2. Core Concepts

- **Types of Errors:**
  - **Syntax Errors:** "Grammar" mistakes in the code that prevent the program from running at all. Usually caught by the interpreter/compiler.
  - **Runtime Errors (Exceptions):** Errors that occur while the program is running, e.g., trying to divide by zero or access a file that doesn't exist.
  - **Logical Errors:** The program runs without crashing but produces incorrect results. These are often the hardest to find.
- **Exception Handling:**
  - The process of responding to runtime errors.
  - **`try` block:** Encloses the code that might raise an exception.
  - **`except` block:** Contains the code that will be executed if an exception occurs in the `try` block. This prevents the program from crashing.
- **Debugging Strategies:**
  - **Print Debugging:** Strategically placing `print()` statements to inspect the state of variables at different points in the program.
  - **Using a Debugger:** A tool (often built into an IDE) that allows you to pause program execution, step through code line-by-line, and inspect variables in real-time.

3. Code Examples

```
# Exception handling example

try:

    numerator = 10

    denominator = int(input("Enter a number to divide by: "))

    result = numerator / denominator

    print(f"The result is {result}")

except ZeroDivisionError:

    print("Error: You cannot divide by zero.")

except ValueError:

    print("Error: Please enter a valid number.")
```

```
# A logical error for debugging

# This function is supposed to calculate the average, but it's wrong.

# A debugger would show that total is not what we expect.

def calculate_average(numbers):

    total = 0

    # BUG: It should be total = total + number

    for number in numbers:

        total = number # This just keeps overwriting total!

    return total / len(numbers)
```

4. Summary

No programmer writes perfect code. Knowing how to anticipate errors with exception handling and how to hunt down bugs with effective debugging techniques is an essential and practical skill for any developer.