Introduction to Software Development Week 5: Functions & Modularity

1. Learning Objectives

- Define and call custom functions to encapsulate code for reuse.
- Pass data into functions using parameters (arguments).
- Return data from a function using the `return` statement.
- Understand the concept of variable scope (local vs. global).

2. Core Concepts

- **The Need for Functions:**
  - To follow the **DRY** principle: **D**on't **R**epeat **Y**ourself.
  - Break down complex problems into smaller, manageable pieces.
  - Improve code readability and maintainability.
- **Anatomy of a Function:**
  - **`def` keyword:** To define a function.
  - **Function Name:** A descriptive name.
  - **Parameters:** Variables listed inside the parentheses, acting as placeholders for data the function will receive.
  - **Function Body:** The indented block of code that runs when the function is called.
  - **`return` statement:** (Optional) Exits the function and sends a value back to the caller.
- **Scope:**
  - **Local Scope:** A variable created inside a function is only accessible within that function.
  - **Global Scope:** A variable created outside of any function can be accessed (but not modified without the `global` keyword) from anywhere.

3. Code Examples

```
# A simple function with a parameter and a return value

def calculate_area(width, height):

    area = width * height

    return area


# Calling the function and storing the result

room_width = 5.5

room_height = 3.0

room_area = calculate_area(room_width, room_height)
```

```
print(f"The area of the room is {room_area} square metres.")
```

4. Summary

Functions are the building blocks of modular, organized programs. They allow you to create logical, reusable, and testable units of code.