Phase 2: Intermediate Python (Building Blocks)

his phase focuses on making your code more modular, efficient, and robust, preparing you for larger projects.

- **Functions:**
  - Functions are named blocks of reusable code designed to perform a specific task. They help break down complex problems into smaller, manageable pieces.
  - You can pass information into functions (called **arguments** or **parameters**) and functions can send information back out (using a **`return`** statement).
  - Understanding **scope** (where variables are accessible – local to a function or global) is key.
- **Data Structures (Part 2: Advanced - Dictionaries & Sets):**
  - **Dictionaries (`{key: value}`):** Unordered collections of key-value pairs. Each `key` is unique and maps to a `value` . They are incredibly fast for looking up values when you know the key (like looking up a word in a dictionary). Dictionaries are mutable.
  - **Sets (`{item1, item2}`):** Unordered collections of *unique* elements. Sets are useful for quickly checking if an item exists, removing duplicates from a list, or performing mathematical set operations (union, intersection, difference). Sets are mutable.
- **Error Handling (Exceptions):**
  - This mechanism allows your program to gracefully respond to errors (called "exceptions") that occur during execution, instead of crashing.
  - **`try` block:** Contains the code that might raise an exception.
  - **`except` block:** Contains the code that executes if a specific type of exception (or any exception) occurs in the `try` block. You can catch different types of errors (e.g., `ValueError` , `ZeroDivisionError` , `FileNotFoundError`).
- **Modules and Packages:**
  - **Modules:** Simply Python files (`.py`) containing functions, classes, and variables. You can reuse code from one module in another using the `import` statement.
  - **Packages:** Directories that contain multiple modules and a special `__init__.py` file. They help organize related modules into a hierarchical structure, making large projects manageable.
- **File I/O (Input/Output):**
  - The process of reading data from files and writing data to files on your computer's storage.
  - You use the `open()` function to get a file object, specify the mode (`"r"` for read, `"w"` for write, `"a"` for append), and then use methods like `read()` , `readline()` , `readlines()` , `write()` , `writelines()` .
  - The `with` statement is highly recommended for file operations because it automatically handles closing the file, even if errors occur, preventing resource leaks.