Week 1 – Fundamentals, Testing, and GitHub Basics (Language-Agnostic)

Review core programming concepts, introduce basic testing practices, and ensure interns can confidently use Git and GitHub.

---

1. Learning Objectives

By the end of Week 1, interns should be able to:

- Write clean, well-structured code using basic constructs (loops, conditionals, functions).
- Understand file and project organization on GitHub.
- Use version control: commit changes, create branches, and submit pull requests.
- Write and run simple tests for their code (unit testing principles).
- Practice documenting and reflecting on their learning.

---

2. Core Programming Review Topics

Focus on **concepts**, not syntax. Each intern can implement exercises in **their preferred language**.

Topic	Concept	Example Exercise
Variables & Data Types	Declare, initialize, and manipulate variables	Swap the values of two variables
Conditional Logic	Use if-else or switch/case	Check if a number is even, odd, or prime
Loops & Iteration	Repeat actions with for, while, etc.	Sum all numbers in a list or array
Functions / Methods	Accept parameters, return results	Write a function to reverse a string/list
Arrays / Lists & Objects / Maps	Store and access collections	Find the largest number in a list or collection

Topic	Concept	Example Exercise
Clean Code	Naming, indentation, modular design	Refactor messy code into readable functions

---

3. Git & GitHub Basics

All interns must learn how to **manage code professionally** using Git and GitHub.

Task	Description
Repository Creation	Learning-Github
Cloning	Clone the repository locally
Branching	Create a branch for Week 1: week-1
Committing	Make frequent, meaningful commits
Pushing	Push changes to GitHub
Pull Requests	Open a PR to the main branch for review
README	Document your progress and lessons learned

Optional Git commands to practice:
git status, git add ., git commit -m "message", git log, git diff, git pull

---

4. Basic Testing Principles

Testing Concepts:

- A test validates that a function or method produces the correct output.
- Tests should include:
    - Input values
    - Expected output
    - A way to compare actual output to expected output

Exercise Examples

1. Test a function that adds two numbers.
2. Test a function that reverses a string/list.
3. Test a function that finds the largest number in a collection.

Each intern implements the tests in their chosen language