Chapter 1: Up and Running - Lesson Plan

Course: C++ for Python/Java Programmers

Estimated Time: 2.5 hours maximum (self-paced)

Learning Objectives

By the end of this lesson, students will be able to:

1. **Set up** a C++ development environment on their operating system
2. **Write, compile, and run** a basic C++ program
3. **Explain** the C++ compilation process (preprocessor, compiler, linker)
4. **Use** basic C++ syntax including variables, functions, and control flow
5. **Apply** debugging techniques using IDE debuggers
6. **Compare** C++ syntax and concepts with Python/Java equivalents

---

Key Concepts with Python/Java Equivalencies

1. Program Structure & Entry Point

Language	Entry Point	Notes
C++	`int main() { return 0;}`	Required; returns exit code
Python	No explicit main	Convention: `if __name__ == "__main__":`
Java	`public static void main(String[] args) {}`	Must be in a class

2. Compilation Model

Language	Type	Process	Output
C++	Compiled	Source → Object → Executable	Machine code
Python	Interpreted	Source → Bytecode (automatic)	Runs directly
Java	Hybrid	Source → Bytecode → JVM	Platform independent

3. Including External Code

Language	Syntax	When Processed	Example
C++	`#include <header>`	Preprocessor (before compilation)	`#include <cstdio>`
Python	`import module`	Runtime	`import math`
Java	`import package.Class;`	Compile time	`import java.util.Scanner;`

4. Variable Declaration

Aspect	C++	Python	Java
Declaration	`int x;`	`x = None` or just use	`int x;`
Initialization	`int x = 42;`	`x = 42`	`int x = 42;`
Type Required	Yes (static)	No (dynamic)	Yes (static)
Type Change	Not allowed	Allowed	Not allowed

5. Console Output

Language	Basic Output	Formatted Output
C++	`printf("Hello\n");`	`printf("Value: %d\n", x);`
Python	`print("Hello")`	`print(f"Value: {x}")`
Java	`System.out.println("Hello");`	`System.out.printf("Value: %d\n", x);`

6. Function Definition

Language	Syntax	Return Type
C++	`int add(int a, int b) { return a + b;}`	Must specify
Python	`def add(a, b): return a + b`	Dynamic
Java	`public static int add(int a, int b) { return a + b;}`	Must specify

Lesson Structure

Pre-Work: Environment Setup (Not counted in course time)

Complete before starting Chapter 1

Required Pre-Work

1. Install IDE/compiler for your OS using provided setup guide
2. Verify installation by compiling a test program
3. Submit screenshot showing successful "Hello, World!" compilation

Support Available

- Detailed setup guide in resources section
- Office hours for troubleshooting
- Online compiler backup option if local setup fails

Module 1: First C++ Program (30 minutes)

Content Delivery

- **Quick Comparison**: Hello World in C++, Python, and Java
- **Essential Syntax**: Basic program structure
- **Compilation Basics**: Why C++ needs compilation (5 minute overview)

Activities

1. Type the Hello World program
2. Compile and run successfully
3. Modify to print personal information

Check for Understanding

- Program compiles and runs correctly
- Can explain basic syntax differences

Module 2: Variables and Types (45 minutes)

Content Delivery

- **Static Typing**: Why types matter in C++
- **Basic Types**: int, double, char
- **Variable Declaration**: Required vs optional in Python/Java

Activities

1. Temperature conversion program
2. Practice with different data types
3. Simple arithmetic operations

Key Concepts

- Type declarations are mandatory
- Semicolons end every statement
- Format specifiers for printf

Module 3: Functions and Control Flow (60 minutes)

Content Delivery

- **Function Basics**: Declaration, definition, calling

- **Control Flow**: if/else statements
- **Function Order**: Declare before use

Activities

1. Write simple functions (square, add)
2. Create decision-making programs
3. Combine functions with control flow

Practice Focus

- Function syntax and calling
- Basic if/else logic
- Parameter passing

Module 4: Integration Practice (35 minutes)

Content Delivery

- **Putting It Together**: Review all concepts
- **Simple Debugging**: Basic error checking
- **Testing Strategies**: Verify your programs work

Activities

1. Mini calculator project
2. Test with different inputs
3. Handle simple errors (division by zero)

Final Check

- All concepts work together
- Can write, compile, run, and test programs

Assessment Strategy

Formative Assessment (Throughout)

- **Auto-graded exercises**: Compilation and output checking
- **Peer code review**: Students review each other's solutions
- **Self-check quizzes**: After each module

Summative Assessment

- **Mini Calculator Project** (45 minutes):
    - Add, subtract, and multiply two hardcoded numbers
    - Use separate functions for each operation
    - Include basic division with zero-check
    - Starter template provided

Rubric (Pass/Fail)

Criteria	Pass Requirement
Compilation	Program compiles cleanly without errors
Basic Functions	Add, subtract, multiply functions work correctly
Output Format	Results print clearly with proper labels
Zero Division	Division by zero shows error message

Additional Resources

Required Reading

- Chapter 1: “Up and Running” from C++ Crash Course

Supplementary Materials

- cppreference.com - C++ reference
- Compiler Explorer - See assembly output
- C++ vs Python vs Java Cheat Sheet - Quick syntax reference

Office Hours Topics

- Troubleshooting environment setup
- Understanding compiler errors
- Transitioning from interpreted to compiled languages

Teaching Notes

Common Challenges

1. **Environment Setup**: Have alternative online compilers ready
2. **Compilation Errors**: Students from Python struggle with syntax strictness
3. **Static Typing**: Emphasize this is for performance and safety

Differentiation

- **For Advanced Students**: Explore compiler flags, optimization levels
- **For Struggling Students**: Provide more scaffolded exercises, pair programming

Time Management

- Keep environment setup to time limit (provide pre-configured options)
- Focus on concepts over syntax memorization
- Use live coding to demonstrate common errors