C Programming + Computer Science Foundations — Python to C Transition BluePrint

Introduction

This 4-week blueprint is for learners moving from Python to C.

By the end, learners will understand how C gives low-level control over memory and execution, revealing what Python does behind the scenes.

Week 1 — Getting Started with C

- **Header files**: `#include <stdio.h>`
- **`main()` function**: program entry point
- **`printf`** for output
- **`scanf`** for input (explained in comments)
- **`return 0;`** for program termination
- **No-semicolon cases**: function header, `if`/`while`, `#include`

Python vs C:

- In Python, no header files; built-ins are imported with `import`
- Python prints with `print()`, C uses `printf()` with format specifiers
- Python programs start from the first line; C starts from `main()`
- Python is dynamically typed; C requires explicit data type declarations

---

Week 2 — Data Types, Variables, and Basic I/O

- Data types: `int`, `float`, `double`, `char`, `string` (char array), `bool`
- Variable **declaration** and **initialization**
- Format specifiers: `%d`, `%f`, `%lf`, `%c`, `%s`
- Special characters: `\n` (newline), `' '` (space)

Python vs C:

- Python infers type: `x = 5`; C needs `int x = 5;`
- Python can store different types in one variable (reassign), C cannot
- Python uses `input()` for reading values; C uses `scanf()` with memory address (`&`)

Week 3 — Conditional Logic and Functions

- for, while, and do-while loops
- Nested loops and logical operators
- Defining and calling functions in C
- Passing arguments by value
- Scope and lifetime of variables
- Switch Case

    Python vs C:

- Python functions don't need a return type; C must specify (`void`, `int`, etc.)
- Python lists are dynamic; C arrays have fixed size

Week 4 — Pointers, Pass-by-Reference, and Strings

- Introduction to pointers and addresses
- Arrays and strings in C
- Using pointers with arrays and functions

Python vs C:

- Python variables store references automatically; in C, you must use pointers explicitly
- Python strings are immutable and managed automatically; in C, they're arrays of chars
- Python memory handling is automatic; in C, you must manage addresses and storage manually
- Python indexing and slicing work on strings directly; in C you work with memory positions

---

Final Notes

- Each week's lessons include **hands-on code**, **debugging practice**, and **Python comparisons**
- By Week 4, learners can:
  - Work with memory directly
  - Build modular programs
  - Understand how high-level languages handle variables, functions, and strings internally