SOFTWARE DEVELOPMENT AND ENGINEERING PRACTICES

Introduction to Programming

Level: Foundation • 2 days (expandable to 3) • Virtual, In-person

Overview

Programming looks like typing, but it is really a way of thinking: taking a problem apart into steps so small and precise that a machine can execute them. That shift in thinking is the genuinely hard part of learning to code, and it is exactly the part most beginners are left to figure out alone while tutorials rush them toward syntax. Once the thinking clicks, the syntax is the easy part.

This is a hands-on, foundation course. It assumes no programming experience at all and builds in the only order that works: first how programs and computers actually relate, then storing information, then making decisions, repeating work, and organizing code, with each concept practiced before the next one arrives. Rather than race through a language's feature list, the course teaches a small core of ideas deeply, because those ideas transfer to every language you will ever meet. Every module ends with a lab and builds on the one before, and by the end you will have written a complete small program yourself.

Who Should Attend

- People with no programming experience who want to learn to code properly from the start
- Analysts, testers, support engineers, and other technical-adjacent professionals moving toward code
- Anyone who has tried tutorials, gotten stuck, and wants the fundamentals taught in the right order

  Learners who already write working code should start with *Python Programming* or *Object-Oriented Programming with C#* instead.

Prerequisites

- No programming experience required
- Comfortable using a computer: files, folders, and installing an application
- Willingness to get things wrong and try again, which is how programming is actually learned

What You Will Learn

- Explain what a program is and how source code becomes running software
- Break a problem into precise, ordered steps a computer can follow
- Store and work with information using variables and basic data types
- Control what a program does with conditions and loops
- Organize code into functions and work with lists of data
- Write, run, and debug a complete small program on your own

Course Outline

Day one: thinking in steps

- How Programs Work
  - What a program is: precise instructions, executed exactly as written

  - Languages, source code, and what happens when a program runs
  - Your tools: writing code, running it, and reading what comes back
  - Lab: write and run your first program, break it on purpose, and read the error like a programmer
- Storing Information
  - Variables: giving names to values so you can use them later
  - Data types: numbers, text, and true or false
  - Getting input, doing arithmetic, and printing results
  - Lab: build a small calculator that takes input, computes, and reports the answer
- Making Decisions
  - Conditions: asking true-or-false questions about your data
  - If, else, and choosing between paths
  - Combining conditions, and the logic mistakes everyone makes at first
  - Lab: build a program that gives different answers based on the user's input

Day two: repeating, organizing, and building something whole

- Repeating Work with Loops
  - Why loops exist: making the computer do the boring part
  - Counting loops and condition-based loops
  - Loops plus decisions: the pattern behind most real programs
  - Lab: build a guessing game that keeps playing until the user gets it right
- Functions and Lists
  - Functions: naming a chunk of work so you can reuse it
  - Inputs, outputs, and why small functions keep programs understandable
  - Lists: working with many values at once, and looping over them
  - Lab: refactor your game into functions and add a scoreboard kept in a list
- Your First Real Program
  - Turning a problem statement into a plan before you type
  - Debugging: reading errors, forming a hypothesis, and testing it
  - Where to go next: languages, practice habits, and what to build first
  - Lab: plan, write, and debug a complete small program of your own, start to finish

Extended Version

The three-day version keeps the same gradient and adds a gentler pace with more practice:

- More guided practice time on loops and functions, the two concepts beginners struggle with most
- Working with files: reading and saving simple data so programs remember things
- An introduction to objects: a first look at the idea behind *Object-Oriented Programming with C#*
- A capstone: build a complete interactive program in pairs and walk the class through how it works