INTEGRATED DEVELOPMENT ENVIRONMENTS

The Basics

Developers store their source code in text files and use specialized tools like compilers to translate it into instructions computers can understand and execute.

In the early days of programming, simple text editors were used to write code, and development tools were run via the command line (such as Windows' Command Prompt or macOS Terminal).

Modern applications can consist of dozens to thousands of source files and millions of lines of code, however, making the process of compiling and managing code far more complex. Text editors and command-line tools alone are insufficient for handling this scale of development.

Today, developers use integrated development environments (IDEs), which help manage large codebases and streamline the building and testing of applications. Common IDE features include:

- Syntax highlighting to make code easier to read by distinguishing key language elements.
- Auto-completion for quick insertion of common code patterns, reducing manual input.
- Real-time error detection to flag coding mistakes instantly.
- Library management tools to efficiently handle external project dependencies.
- Debugging tools to help step through and troubleshoot code.
- Collaboration features enabling multiple developers to work together on the same project.
- Project structuring tools to organize and build complex application components.
- Automated documentation generation for creating reference materials easily.

In the earliest days of computing, programs were physically wired into the machine. The development of stored-program computers in the late 1940s revolutionized this, allowing programs to be written and stored as we understand them today.

A Spectrum of Capabilities

There are many IDEs available, generally categorized by two factors: complexity and specificity.

Complexity: More complex IDEs offer a wider range of features, making them more powerful but also harder to learn and more resource-intensive. For example, Microsoft's Visual Studio includes templates, integrated compilation, debugging, and cloud platform support. In contrast, Python's Integrated Development and Learning Environment (IDLE) is simpler, offering basic syntax-aware editing and runtime support.

Specificity: Some IDEs are tailored for specific tasks, like Android Studio for Android app development, while others, like Eclipse, support multiple platforms and programming languages.

For Apple enthusiasts, developers typically use the Xcode IDE for iOS and macOS development.

A Happy (Yet Still Powerful) Medium

Quantic generally recommends Microsoft's Visual Studio Code (VS Code) as the preferred IDE, except for a few courses. It's a versatile, general-purpose IDE that supports many programming languages and platforms. While not as feature-rich as Visual Studio initially, VS Code offers a wide range of extensions to add specific functionalities.

VS Code runs on both Windows and MacOS, and it's easy to install: just select the "Download for <my OS>" on the home page. There's also a useful getting started tutorial that will quickly have you confident with the basics.

We recommend installing VS Code and completing the tutorial now. Don't worry if anything seems challenging—throughout the courses, we'll provide all the detailed guidance you need to succeed.