Basic Requirements for a Go Environment

This lesson lists some functionalities successfully covered in Go that are expected by users when programming in any language.

WE'LL COVER THE FOLLOWING

Desired functionalities

What can you expect from an environment that you can accomplish with a simple text editor and the compiler/link tools on the command-line? Here is an extensive list of desired functionalities:

- **Syntax highlighting**: this is, of course, crucial, and every environment cited provides a configuration—or settings files for this purpose—preferably different color-schemes (also customizable) should be available.
- **Automatic saving** of code, at least before compiling.

- **Line numbering** of code
- Good **overview and navigation** in the codebase must be possible; different source-files can be kept open.
- Setting **bookmarks** in code is possible.
- **Bracket matching**
- From a function call, or type use, **go to the definition** of the function or type
- Excellent **find** and **replace** possibilities, the latter preferably with

preview

- Being able to **(un)comment** lines and selections of code
- **Compiler errors**: double-clicking the error-message should highlight the offending code line.
- **Cross-platform**: working on Linux, macOS, and Windows so that only one environment needs to be learned/installed
- Preferably **free**, although some developers willing to pay for a high-quality environment
- Preferably **open-source**
- **Plugin-architecture**: relatively easy to extend or replace a functionality by a new plugin.
- **Easy to use**: an IDE is a complex environment, but still, it must have a lightweight feel
- **Code snippets (templates)**: quick insertion of much-used pieces of code for ease and accelerating the coding.
- Closely related is the concept of a **build system**: it must be easy to compile/link (build), clean (remove binaries) and/or run a program or a project. Running a program should be possible in the console view or inside the IDE.
- **Debugging** capabilities (breakpoints, inspection of values, stepping through executing code, being able to step over the standard library code)
- Easy access to **recent files and projects**
- **Code completion** (IntelliSense) capabilities: syntax-wise (keywords), packages, types and methods within packages, program types, variables, functions, and methods; function signatures
- An **AST-view** (abstract syntax tree) of a project/package code
- **Built-in access to the go tool-chain**, such as `go fmt`, `go fix`, `go install`, `go test`, ...
- Convenient and integrated **browsing of Go documentation**
- Easy switching between **different Go-environments**
- **Exporting code to different formats**, such as pdf, Html or even **printing** of the code
- **Project templates** for special kinds of projects (such as a web application, an App Engine project) to get you started quickly

application, an App Engine project) to get you started quickly

- **Refactoring** possibilities
- **Integration with version control-systems** like hg or git
- Integration with **Google App Engine**

Consultation

There is very good support for Go in a whole stack of editors. Consult this official page, and visit Github for the various options.

Go is becoming a demanding language due to the ease of use and flexibility. In the next lesson, you’ll see some editors and IDE(s) commonly used for programming in Golang.