Introduction to Programming in Go

Student Manual

Document Information

Course Title:	Introduction to Programming in Go
Author:	Rod Davison
e-mail:	rod@exgnosis.com
Version:	1.0
Release Date:	2016-10-01
ProTech ID:	PT10491

"The primary objective of copyright is not 'to reward the labor of authors, but to promote the Progress of Science and useful Arts.' To this end, copyright assures authors the right to their original expression, but encourages others to build freely upon the ideas and information conveyed by a work. This result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art."

Justice Sandra Day O'Connor
Feist Publications, Inc. v. Rural Telephone Service Co.
499 US 340, 349(1991)

Student Preface

This is the student manual for the course *Introduction to Programming in Go* offered at Capital One and presented by Protech Training. This manual is intended to serve a couple of functions:

Provide you with a more complete record of the class content

There is only so much you can cram into a slide or powerpoint presentation. This manual covers the same topics as in the slides but provides more depth on each of the topics. This means that you can go back and revisit the material later without having to remember (and you won't) what the instructor actually said about that slide.

Ensure that you get all the intended content

Sometimes in a class not all of the topics are covered equally. This may be due to the skill level of the class, or the fact the class wants to go into more depth on certain topics and less on others, or any of a number of other legitimate reasons for the content as delivered in class to not match up the with the content that was planned. If this happens in your class, and hopefully it won't, at least you have all the planned content in the manual to take away with you.

Add more interesting content beyond what is presented in class.

At some points in the manual I have refrenced extra content, often from the blogs and presentations by the developers of the Go language which, I think, gives some interesting insights and background to what is being discussed in the lectures. This extra material is generally not included in the presentation simply because of the time restrictions we have on the class time available.

What this manual is not

This manual is not a textbook or a brilliant and incisive exploration of Go. It is merely intended to be a useful resource for the course. There is a substantial amount of material floating around about Go – in fact, there is so much that it would be very difficult for me to actually come up with any really novel content.

Go has a fantastic amount of reference material that is easily available including the Go language specification, Go blogs, the Go tour and many other presentations and articles, many of which are written by Rob Pike and the other inventors of the Go language.

This manual is not intended to be a restatement of those available materials. I will reference them and provide instructions on how to access them, but getting them and actually using them will be your responsibility

Prerequisite Expectations

This course is intended for experienced programmers who have spent time programming in a C-type language, by which we mean any language that uses a syntax and structure derived from C such as C#, Java, C++ or even JavaScript. If your programming background is in another language like Python, you may have to work a bit harder to understand the examples but if you have a good solid understanding of programming constructs, then you should be okay.

This is not an introductory programming course. There will be no remedial work done in class to get people up to speed on programming concepts – there is just no time to do that. Consider this a fair warning that if you are not comfortable with writing code then you will find the class hard going. One possible strategy that you might explore if you find yourself out of your depth is to partner up with another student who is a good programmer to at least be able to see how a programmer works with the exercises.

Expected Technical Setup

This section only applies if you are not using one of the supplied virtual machines.

It is assumed that you have the Go programming environment installed and tested on your computer before class starts. There will be no time during the class to go through any installation procedures although one is described in the first module of this manual. The procedure described here is the installation onto a generic machine from the Go project website and other public websites, however Capital One may have their own internal sites from which Go should be installed instead if you are installing it on a Capital One computer.

Course Objectives

The purpose of this course is to get you familiar enough with Go, both in terms of how the language works, and the Go ecosystem that you can go back and start working with Go in your operational environment. The course will have a series of traditional modules covering a number of Go topics, each of which is followed by a lab session where you can work with the concepts from the presentation. The latter part of the course will be a selection of programming projects that should allow you to pull together the concepts and techniques of the course in some hands on programming challenges.

Keep in mind that this is an introductory survey course. That means that we will touch on a wide range of Go related topics without going into a great deal of depth on many of them. For example, we are going to learn about concurrency in Go but we will not learn everything about the topic. Some realistic expectations are that after the class is that you should be able to:

1. Write clean Go code and be able to read Go code and understand the Go code design and techniques used.
2. Be able to use the Go programming environment and utilities (the Go ecosystem) and understand how they work.

3. Get a feel for what the “Go way” of doing things is, especially some of the Go idioms and practices.

This course will not make you into an expert Go programmer – to reach that goal will take a lot more study and practice beyond what we will be able to do in these three days. Programming expertise in any programming language is made up of two parts: first, understanding how the language works and then developing your skill at designing solutions and writing code to implement those solutions in that language.   Three days is not enough time for you to develop that depth of Go programming skill, that will be something that you do yourself through practice after this session is done. Think of this course as an orientation into the Go programming paradigm and the starting point for your Go skill development in the months ahead.

You also have to keep in mind that this course is not comprehensive or intensive. There are many topics we just don’t have the time to get into like the cryptography libraries and working with integrating Go with legacy C code, as interesting as these topics might be. There are also topics we can only touch on superficially because of the same time constraints for the course – we could probably spend two days alone exploring concurrency in Go.

I often use the analogy that learning a programming language is like learning to speak a foreign language like French. This course is akin a one semester college course that introduces you to French grammar, pronunciation and vocabulary (and assumes that you already speak Spanish). But learning about how the French the language works, while a critical step in becoming fluent, is not the same as learning to speak French, which is a skill that only comes by going out and speaking French.

Hands On Work

Hands on work in a course like this is critical for learning. In the first module, along with the instructions for installing Go, is a recommendation for a couple of IDEs that Go programmers often use. I strongly recommend that you get one and use it during the class since tools like colored syntax highlighting, automatic formatting and expression completion make the process of learning the basic of the language a lot smoother and easier

During the instructor’s presentation, there will be a number of code samples used to illustrate concepts. The code for all of these examples is available to you in the appropriate directories in the provided github repository (or some other place will be provided to you at the start of the class), as well as being available on the virtual machine you will be using. Very often, we learn programming concepts most effectively by writing code, playing with code and trying to break code. I would encourage you to load the example code, run it and play with it during the presentation.

The code examples are not intended to show off some sort of nifty programming expertise – they are kept lean and sparse so that the concept being demonstrated is easily seen. It may make the code seem a bit simplistic but that is the whole point of an example. Personally I find it frustrating when I’m learning a new programming language and the author illustrates a simple idea by throwing out a 500 line cargo scheduling application. Yes the application is cool, but it is so complex that I have a hard time finding exactly the point in all the code that is supposed to be exemplified. The more complex code samples will be part of the labs.

Lab Work

I have tried to get away from the usual, step-by-step, follow-the-instructions-by-rote, get-the-answer-in-the-back-of-the-book labs. I've tried to mix it up a bit with some discussion questions, some labs that are mini-projects and some that are where you experiment with code and then try and explain the results. The last part of the course has labs that are fairly significant in size and complexity where we build a couple of interesting applications. Most of these applications variations on standard Go mini-projects so you will be able to find other implementations of them just by a bit of Googling.

We are not looking for "right answers" in the lab. The labs are for experimentation, practicing and getting that hands on component that is often the most critical part in learning a programming language. You might think you "got it" during the lecture but the only way to be sure is to work with it.

The labs are not intended to be a solitary activity. I would encourage you to collaborate and confer with your classmates. Often a lot of learning can take place just in the process of trying to explain how something works to one of your colleagues. It's also enlightening to see how other people are approaching a problem and compare that to how you are doing it. Think of the labs as a team or class activity if you want, but if it is more your style to work on your own, just do you own thing.

Above all, try to have fun with the labs. You learn best when you are enjoying yourself.

If you are not an experienced programmer, you may find some of the more challenging problems too advanced in which case I would suggest again looking at the provided solutions or work with someone who is more experienced. I would suggest the latter as the preferred approach

"The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires."

William Arthur Ward