Extended

230303-0749

GO1

Get Going on z/OS

Z Xplore

USING GO ON Z/OS

Getting started with Go and familiarizing yourself with a new type of programming language.

The Challenge

Go (or Golang) is an open source programming language that makes it easy to build simple, reliable, and efficient software. During this challenge you will be using the Go extension found in VS Code to familiarize yourself with Go and understand its basics.

Before You Begin

Make sure you have a basic understanding of

001|230303-0749

INVESTMENT

Steps	Duration
6	60 minutes

1 GO ON Z/OS

Go is a new programming language at the forefront of innovation and invented by Google. It is fast and easy to use and it is becoming more commonly used in cloud computing.

Go has a rich ecosystem of packages that enable you to run new applications, especially those that enable the cloud on z/OS.

The ecosystem of Go modules and the small size of the language's syntax mean that application developers typically can deliver Go applications in a shorter time and with fewer new lines of code compared to other languages and frameworks.

Go has been used in an interesting range of applications - crypto currencies, banking, and containers.

Some well-known companies that are using Go are Twitch, Netflix, and Uber.

Benefits of Go:

- Simplicity
  Go combines features of other programming languages into one easy-to-understand language.
- It’s a compiled language
  Go source code for applications is converted into machine-level code that can be executed directly by z/OS, rather than through an interpreter.
- Cloud Native Development
  Go can be used to streamline automation operations for multicloud, hybrid IT and DevOps environment
- Scalability, concurrency and parallelism
  Go is designed to support scalability, with “go routines” and channels to raise concurrency to true parallel programming.
- Garbage collection
  The Go language performs automatic memory management, with extensive control over memory allocation.

Reference: IBM z/OS and Go

For more information, see:

- Go on z/OS
- Case Studies

2 LET'S GET READY TO GO!

2.0.1 Before You Begin

If you are using USS on VS Code:

- You already have Go installed within and do not need to install anything.

When you are not using USS:

- You will need to download the Go add-on. Open up VS Code and look for the extensions tab on the left hand side.

  1. Search for Go in Extensions
  2. Install the add-on

G011230303-0749

2.1 GET GO-ING AND GRAB THE FILES YOU NEED

1. Locate your terminal and log into the z/OS system.

   Hint: Don't forget to SSH

2. If you want to check the version of Go you have, type `go version` into the terminal

3. Before we find the files we need, let's create a directory to store them in. Create a directory called `go` in your home directory.

Now that we have a directory to store our files, let's find the files.

1. Change directories either through the terminal or with Zowe (on the left). Go to `/z/public`

2. Find the three files with `.go` and copy all of them into your new directory called “Go”. You can do this using Zowe, but we recommend using the terminal to continue practicing.

   Hints using terminal:

   Remember `cp` means copy

   Include the path you are taking from and going to

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE                    3: ssh

code1.go  code2.go  code3.go
/z/z28604/go > cd ..
/z/z28604 > ls
animals.sh      directory1      go              members.py      mynewfile       secret.txt
code1.py        dslist.py       hackathons      movie-data.csv  odbc            test
code2.py        fixfiles.sh     marbles.py      mynewdir        scramble.sh     ussout.txt
/z/z28604 > cd go
/z/z28604/go > ls
code1.go  code2.go  code3.go
/z/z28604/go >
```

You should have three files in your “Go” directory.

Now you are ready to GO!

3 LEARN BEFORE YOU GO

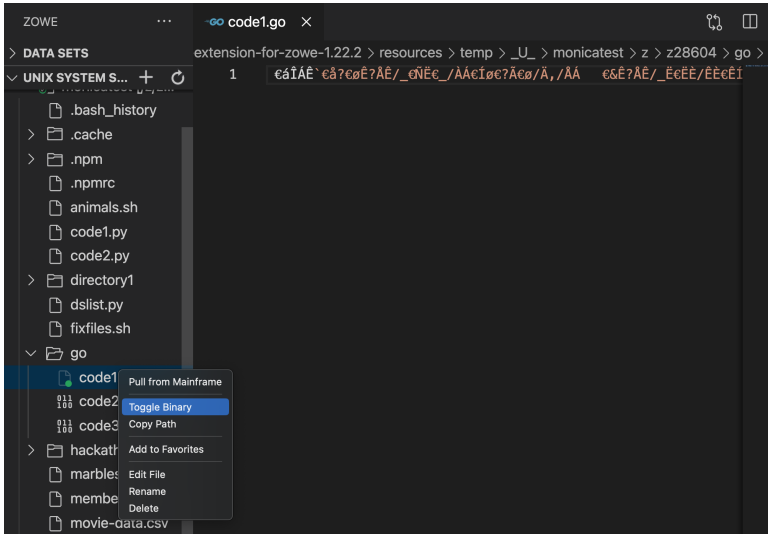
In your home directory, find the file code1.go.

My code is unreadable, what do I do?

If your code is unreadable, it is because it is in EBCDIC, not ASCII. Files you work with will not always be readable. We want you to start to be aware of the tagged nature of text files when you start playing with traditionally ASCII/UTF world of open source languages.

To fix this, go to USS on VS Code on the left and right click on code1.go. Press “toggle binary” and watch as your code now becomes readable.

Make sure you do this for the other .go files as well.



3.0.1 Now That You Can Read It, Dig Into The File

Once you have fixed the readability of your file, read through the code to understand what is happening and what it is producing.

Note: Anything with `//` means it is a comment. These comments are meant to help you understand Go. Make sure you read through them before continuing.

Once you have read it through, it is time to build the code.

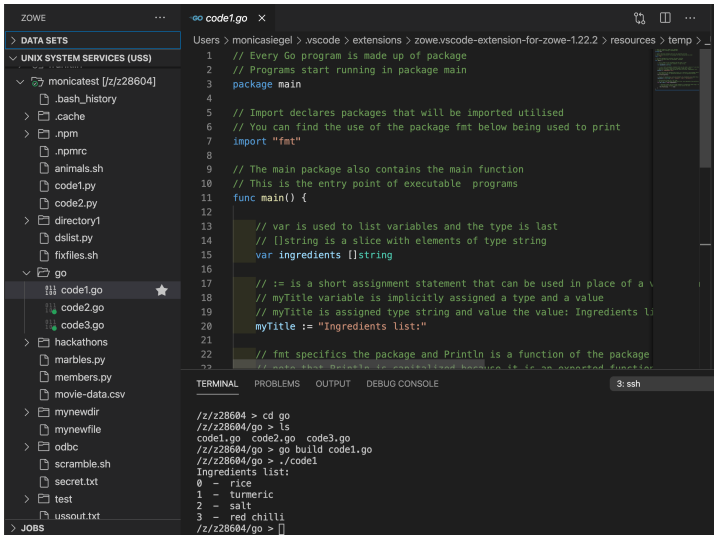
1. Type go build code1.go in the terminal.

This creates an executable named code1. We will discuss what executable means in a bit.

Now, let's run the code we just executed.

1. Enter ./code1

What do you see in the output in your terminal? Is it what you expected?



4 REPETITION THEN GO

Let's try this again with another file.

1. Find code2.go and read through the code to understand it.
2. Again, build the code by typing `go build XX` in the terminal.

   Hint: What should you replace with XX?

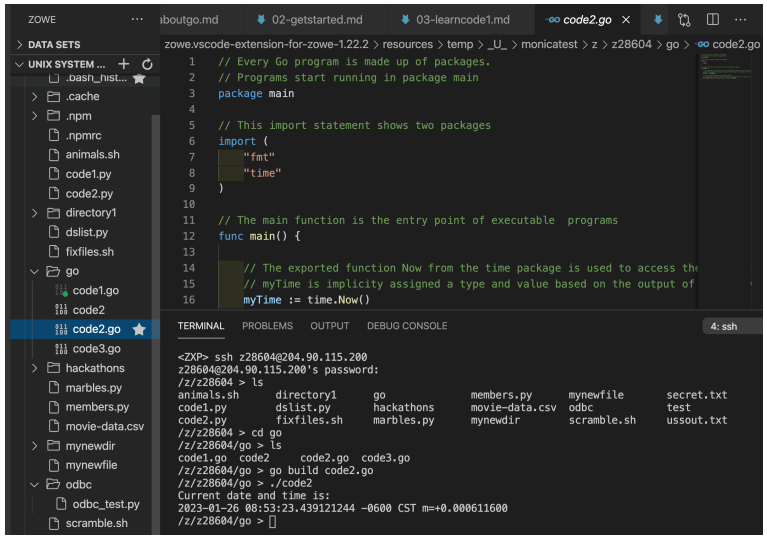
3. Execute the code using `./XX`

This should feel familiar to what you did with code1.go.

Our output looks a little crowded. Let's take it a step further and improve the code to produce a cleaner output.

1. Look again at the code2.go file. Add a newline character `\n` at the end of the `printf` statement. Your output should look like the image below. Notice how this makes the output more readable.

   Hint: Don't forget to save and build again



Note:

You might see an error message like the one below when you save the file. This is because you have not downloaded Go on your personal environment. Go is installed via USS on VS Code so you did not do this step. This is understandable and will still work.

5 GO BUILD VS. GO RUN

So far, we have used `go build xx.go` to create a binary executable for our file. When you are writing code in Go, the computer cannot automatically understand or use the code you've written. `Go build` creates a file of 0s and 1s that the computer can read and understand. After building, you then have used the command `./XX` to run the file.

For your next and final step in this challenge, you will be using `go run`. You will find that it works similarly to `go build` but eliminates a step by building and running your file in one command.

`go run` is better for checking how things are coming along in the short term.

In short:

- If you use `go build`:

  Commands:

  1. `go build xx.go` - builds an executable
  2. `./xx`

  Benefits:

  Good for long-term, when you want to run the code at any time
  Will always be available and you can use it repeatedly

- If you use `go run`:

  Command:

  1. go run xx.go

Benefits:

Good for making small changes and testing to see if they worked

Let’s try out `go run` in the next and final step.

6 READY, SET, GO!

Time to take what you've learned and apply it to another file, `code3.go`!

Follow a *similar* process that you did when working with `code1.go` and `code2.go`, but instead you are going to RUN (compile, execute, discard) instead of BUILD (compile and keep)

When you first run `go run code3`, what is your output?

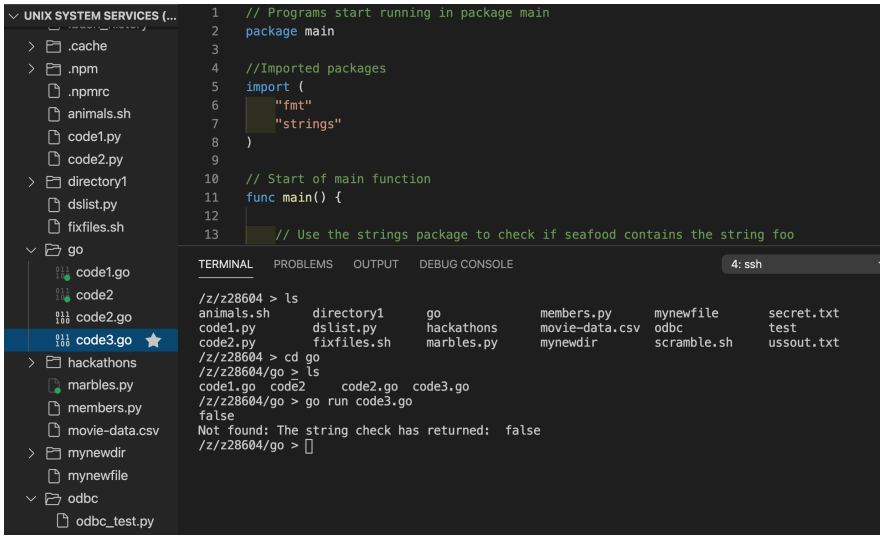
Your challenge is to change your output to say:

```
false
Not found: The string check has returned: false
```

1. Test the code to make sure string.Contains returns false
2. Once confirmed it returns False, make sure that it also returns the "Not found" statement.

- *Hint 1: Use the comments and the code in code3.go to help you along the way.*
- *Hint 2: You will have to make 2 changes to code3.go in order to accomplish this.*

Once you see an output similar to the one below, you've correctly fixed the code!



G01|230303-0749

7 GO ON!

You have now tried both methods for invoking GO language programs -

- run – compile, execute once, and remove the executable
- build - compile and save the executable; run the executable whenever you want or need

To finish off, make sure that you have created the **code3** executable in your ~/go directory before trying the validation.

Submit CHKGO1 to ensure you have done everything correctly before completing the challenge in IBM Z Xplore.

G011230303-0749

Nice job - let’s recap	Next up …
Congratulations! You have just learned a basic introduction to Go. Through these exercises, you have learned: • How to structure and organize Go code using packages • About commonly used Go packages • How to compile and run Go code from the command line • How to debug Go code • How to format print statements in Go	There are many more functions you can do with Go on z/OS. Stay tuned for more challenges on Go. In the meantime, visit https://go.dev/learn/ to learn more about Go and take a tour visit.

G011230303-0749