Starting a Project

Overview

- Installing and configuring Go
- Using the Go command
- Writing Your First Golang Program
- Setting up an editor

Installing and configuring Go

INSTALL, CONFIGURE AND USING THE GO COMMAND

download

https://golang.org/dl/

https://golang.org/dl/

Stable versions

go1.4.2

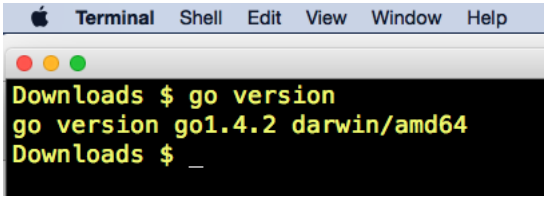
File name	Kind	OS	Arch	Size	SHA1 Checksum
go1.4.2.src.tar.gz	**Source**				460caac03379d746c473814a65223397e9c9a2f6
go1.4.2.darwin-386-osx10.6.tar.gz	Archive	OS X 10.6+	32-bit		fb3e6b30f4e1b1be47bbb98d79dd53da8dec24ec
go1.4.2.darwin-386-osx10.8.tar.gz	Archive	OS X 10.8+	32-bit		65f5610fdb38febd869aeffb0426c83b650bb408
go1.4.2.darwin-386-osx10.6.pkg	Installer	OS X 10.6+	32-bit		3ed569ce33616d5d36f963e5d7cefb55727c8621
go1.4.2.darwin-386-osx10.8.pkg	Installer	OS X 10.8+	32-bit		7f3fb2438fa0212febef13749d8d144934bb1c80
go1.4.2.darwin-amd64-osx10.6.tar.gz	Archive	OS X 10.6+	64-bit		00c3f9a03daff818b2132ac31d57f054925c60e7
go1.4.2.darwin-amd64-osx10.8.tar.gz	Archive	OS X 10.8+	64-bit		58a04b3eb9853c75319d9076df6f3ac8b7430f7f
go1.4.2.darwin-amd64-osx10.6.pkg	Installer	OS X 10.6+	64-bit		3fa5455e211a70c0a920abd53cb3093269c5149c
go1.4.2.darwin-amd64-osx10.8.pkg	**Installer**	**OS X 10.8+**	**64-bit**		8fde619648864cb1c77ddc2a1aec0b7b30406b38
go1.4.2.linux-386.tar.gz	Archive	Linux	32-bit		50557248e89b6e38d395fda93b2f96b2b860a26a

install

https://golang.org/doc/install

Test Install

go version



“go” commands

https://golang.org/cmd/go/

https://golang.org/cmd/go/

Usage:

```
go command [arguments]
```

The commands are:

```
build       compile packages and dependencies
clean       remove object files
env         print Go environment information
fix         run go tool fix on packages
fmt         run gofmt on package sources
generate    generate Go files by processing source
get         download and install packages and dependencies
install     compile and install packages and dependencies
list        list packages
run         compile and run Go program
test        test packages
tool        run specified go tool
version     print Go version
vet         run go tool vet on packages
```

Use "go help [command]" for more information about a command.

```
~ $ go help
Go is a tool for managing Go source code.

Usage:

        go command [arguments]

The commands are:

    build       compile packages and dependencies
    clean       remove object files
    env         print Go environment information
    fix         run go tool fix on packages
    fmt         run gofmt on package sources
    generate    generate Go files by processing source
    get         download and install packages and dependencies
    install     compile and install packages and dependencies
    list        list packages
    run         compile and run Go program
    test        test packages
    tool        run specified go tool
    version     print Go version
    vet         run go tool vet on packages

Use "go help [command]" for more information about a command.

Additional help topics:

    c           calling between Go and C
    filetype    file types
    gopath      GOPATH environment variable
    importpath  import path syntax
    packages    description of package lists
    testflag    description of testing flags
    testfunc    description of testing functions

Use "go help [topic]" for more information about that topic.
```

```
~ $ go help
Go is a tool for managing Go source code.

Usage:

        go command [arguments]

The commands are:

    build       compile packages and dependencies
    clean       remove object files
    env         print Go environment information
    fix         run go tool fix on packages
    fmt         run gofmt on package sources
    generate    generate Go files by processing source
    get         download and install packages and dependencies
    install     compile and install packages and dependencies
    list        list packages
    run         compile and run Go program
    test        test packages
    tool        run specified go tool
    version     print Go version
    vet         run go tool vet on packages

Use "go help [command]" for more information about a command.

Additional help topics:

    c           calling between Go and C
    filetype    file types
    gopath      GOPATH environment variable
    importpath  import path syntax
    packages    description of package lists
    testflag    description of testing flags
    testfunc    description of testing functions

Use "go help [topic]" for more information about that topic.
```

Hello Golang

WRITING YOUR FIRST GOLANG PROGRAM

So Let's write the "Hello, World" program in Go and understand how it works. Open your favorite text editor, create a new file named `hello.go`, and type in the following code -

```go
// My first Program
package main

import "fmt"

func main() {
    fmt.Println("Hello, World")
}
```

You can run the above program using `go run` command like so -

```
$ go run hello.go
Hello, World
```

The `go run` command does two things - It first compiles the program to machine code and then runs the compiled code.

If however, you want to produce a standalone binary executable from your Go source that can be run without using the Go tool, then use the `go build` command -

```
$ go build hello.go
$ ls
hello       hello.go
```

You may now run the built binary file like this -

```
$ ./hello
Hello, World
```

Understanding the internals of the "Hello, World" program

Let's go through each line of the "Hello, World" program one by one and understand what it does-

- **Line 1:** The first line that starts with `//` is a comment -

  ```
  // My first Program
  ```

  Comments are ignored by the Go compiler. They are used to make it easier for others to understand your code.

  Go supports two different styles of comments -

  1. Single-line comment

     ```
     // This is a Single line Comment. Everything in this line is ignored by the compiler
     ```

  2. Multi-line comment

     ```
     /*
       This is a Multi line Comment.
       As the name suggests, It can span multiple lines.
     */
     ```

Understanding the internals of the “Hello, World” program

- **Line 2:** The second line is a package declaration -

```
package main
```

Every Go program starts with a package declaration. Packages are used to organize related go source code files into a single unit and make them reusable.

The package “main” is a special go package that is used with programs that are meant to be executable.

There are two types of programs in Go - Executable Programs and Libraries. Executable Programs are programs that can be run from the command line. Libraries are reusable pieces of code that are used by other programs to perform some task.

Understanding the internals of the “Hello, World” program

- **Line 3:** The third line is an import statement -

```
import "fmt"
```

The `import` keyword is used to import reusable pieces of code from other packages to use in our program.

The “fmt” package contains code for dealing with I/O.

Understanding the internals of the “Hello, World” program

- **Line 4:** The fourth line is a function declaration -

```
func main() {
    // ...
}
```

A function is a unit of code that contains one or more instructions to perform a task. We typically break a program into smaller functions that take some input, do some processing on the input, and produce an output.

A function in go is declared using the `func` keyword.

The `main()` function is the entry point of an executable program in Go. It is the first thing that is invoked when you run an executable program.

Understanding the internals of the “Hello, World” program

- **Line 5:** This line contains a call to the `Println()` function of the `fmt` package -

```
fmt.Println("Hello, World")
```

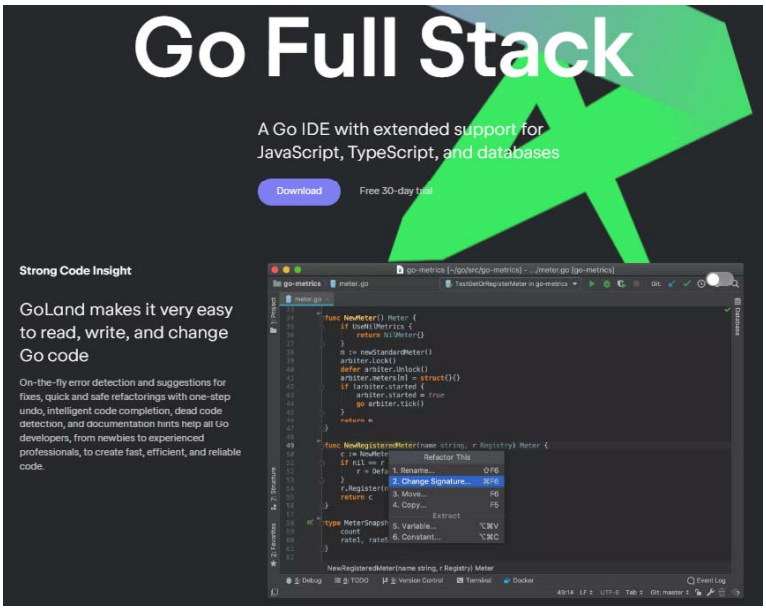
We pass the String “Hello, World” to the `Println()` function which prints it to the standard output along with a new line.

Go IDE's (Golang Editors)

SETTING UP AN EDITOR

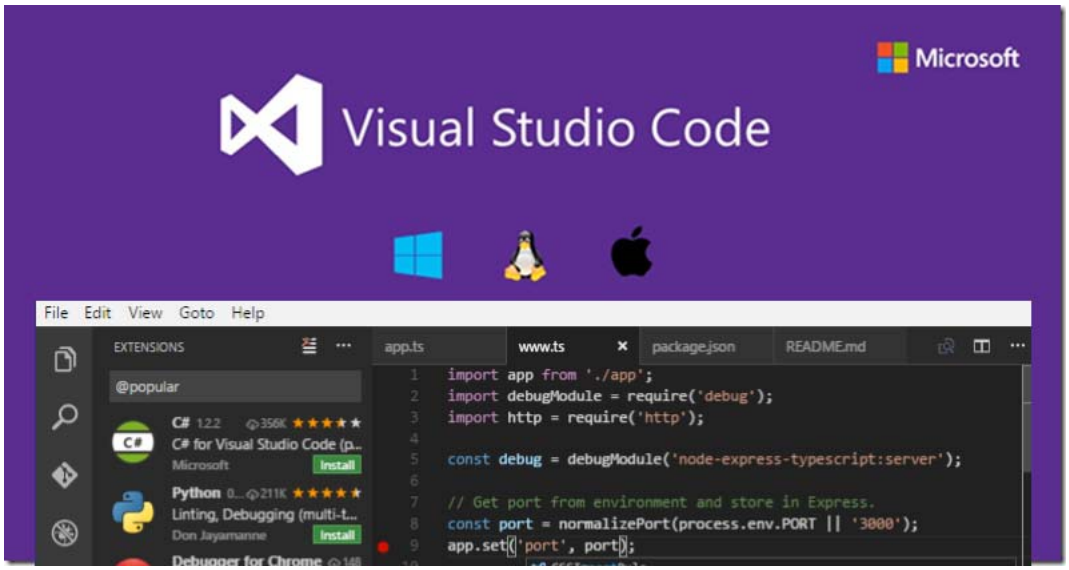
GoLand

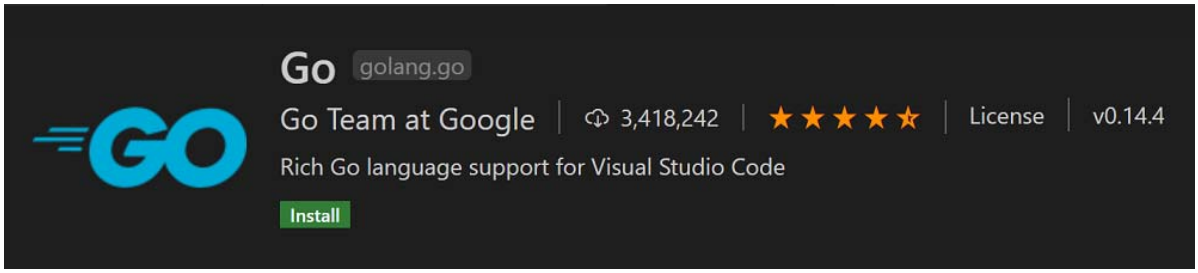
https://www.jetbrains.com/go/



Visual Studio Code

https://code.visualstudio.com/


Microsoft
Visual Studio Code
File Edit View Goto Help
EXTENSIONS
@popular
C# 1.2.2
C# for Visual Studio Code (p...
Microsoft
Install
Python
Linting, Debugging (multi-t...
Don Jayamanne
Install
app.ts
www.ts
package.json
README.md
import app from './app';
import debugModule = require('debug');
import http = require('http');
const debug = debugModule('node-express-typescript:server');
// Get port from environment and store in Express.
const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);


Go
golang.go
Go Team at Google
3,418,242
License
v0.14.4
Rich Go language support for Visual Studio Code
Install

atom

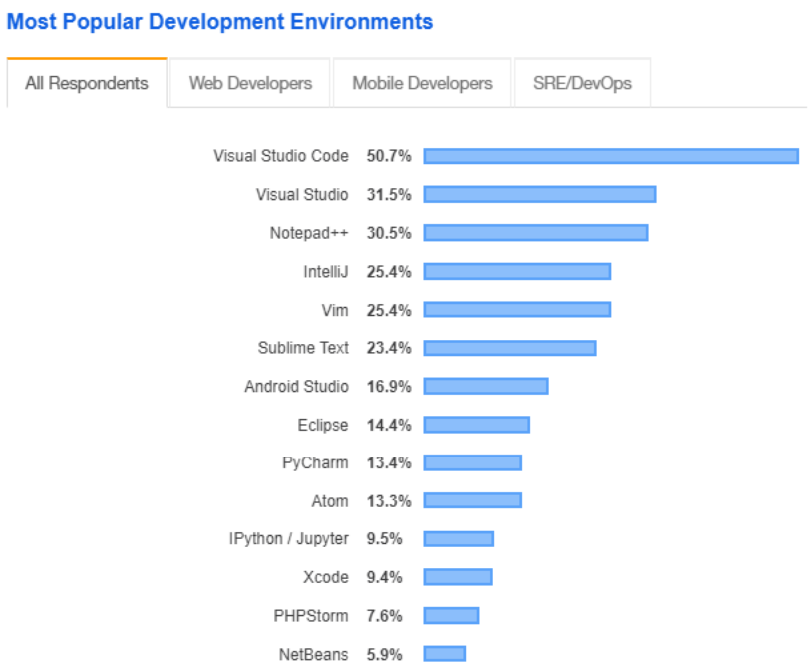
https://atom.io/

atom

made by Github

go-plus

https://atom.io/packages/go-plus



Summary

- Installing and configuring Go
- Using the Go command
- Writing Your First Golang Program
- Setting up an editor