**Module 1 – 1 Why Should I Learn Go**

1. **Advantages of Go**

* Code run fast
* Garbage collection : another language don’t have this
* Simpler objects : object oriented, makes it easier to code
* Concurrency is efficient

1. **Software Translation**

* Machine language:
* The lowest level language, executed on the CPU, on the processor
* CPU instructions represented in binary
* Assembly language:
* Is basically machine language, easier to read
* Almost a one-to-one mapping to machine language
* Can be run really fast and really efficient
* CPU instructions with mnemonics
* High level language:
* Essentially humans commonly use to program in
* Much easier to use than assembly language or machine language
* Provide you with lots of abstractions that any program would be used to
* Assembly language and machine language do not have variables
* They have memory, and you can put stuff in it

Note: Go is of course a high-level language

1. **Compiled vs Interpreted**

* Compilation: translate instructions once before running the code
  + C, C++, Java (partially)
  + Translation occurs only once, saves time
  + Interpretation:
* Translate instructions while code is executed:
* Python, Java (partially)
* Translation occurs every execution
* Requires an interpreter

1. **Efficiency vs Ease of Use**

* Compiled code is fast
* Interpreters make coding easier
* Manage memory automatically
* Infer variable types

Note: Go is a good compromise

1. **Garbage Collection**

* Automatic memory management
* Where should memory be allocated ?
* When can memory be deallocated ?
* Manual memory management is hard
* Deallocate too early, false memory accesses
* Deallocate too late, wasted memory
* Go includes garbage collection
* Typically only done by interpreters

**Module 1 – 2 Objects**

1. **Object Oriented Programming**

* Organize your code through encapsulation.
* Group together data and functions which are related
* User-defined type which is specific to an application. Example ints have data (the number) and functions (+,-,\*,etc)

1. **Object Example**

* Implementing an application performing geometry in 3D
* Many functions will operate on points
* Each point has data: x, y, and z cords
* Points also have functions: DistToOrigin(), Quadrant()
* Point class defines data and functions
* Point objects are instances of class

1. **Objects in Go**

* Go does not use the term class
* Go uses structs with associated methods (data and functions, like class)
* Structs simplified implementation of classes (no inheritance, no constructors, no generics)
* Easier to code
* If you like inheritance, constructors, and generics, this is a disadvantage.

**Module 1 – 3 Concurrency**

One of the advantages of Go (speed)

1. **Performance Limits**

* Moore’s Law used to help performance: number of transistors doubles every 18 months
* More transistors used to lead to higher clock frequencies
* Power / temperature constraints limit clock frequencies now

1. **Parallelism**

* Number of cores still increases over time
* Multiple tasks may be performed at the same time on different cores
* Difficulties with parallelism:
* When do tasks start / stop ?
* What if one task needs data from another task ?
* Do tasks conflict in memory ?

1. **Concurrent Programming**

* Concurrency is the management of multiple tasks at the same time such as executing at the same time
* Key requirement for large systems
* Concurrent programming enables parallelism
* Management of task execution
* Communication between tasks
* Synchronization between tasks

1. **Concurrency in Go**

* Go includes concurrency primitives
* Goroutines represent concurrent tasks
* Channels are used to communicate between tasks
* Select enables task synchronization
* Concurrency primitives are efficient and easy to use

**Module 2 – 1 Installing Go**

1. Go to golang.org, Download Go, choose your platform
2. Installation Wizard

Obey the wizard

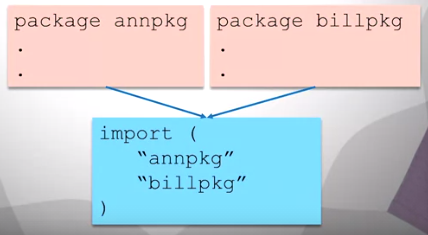
**Module 2 – 2 Workspaces and Packages**

1. **Workspaces**

* Hierarchy of directories
* Common organization is good for sharing
* Three subdirectories:
  + - Src contains source code files
    - Pkg contains packages (libraries)
    - Bin contains executables
* Programmer typically has one workspace for many projects
* Directory hierarchy is recommended, not enforced for example you can have an executable in src
* Workspace directory defined by GOPATH environment variable
* GOPATH is defined during installation

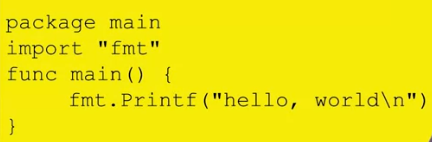
1. **Packages**

* Group of related source files
* Each package can be imported by other packages
* Enables software reuse
* First line of file names the package



1. **Packages Main**

* There must be one package called main
* Building the main package generated an executable program
* Main package needs a main() function
* Main() is where code execution starts



**Module 2 – 3 Go Tool**

1. **Import**

* Import keyword is used to access other packages
* Go standard library includes many packages such as fmt
* Searches directories specified by GOROOT and GOPATH

1. **The Go Tool**

* A tool to manage Go source code
* Several commands
* Go build to compile the program.
  + Arguments can be a list of packages or a list of go files.
  + Creates an executable for the main package, same name as the first .go file
  + .exe suffix for executable in Windows

1. **Go Tool Commands**

* Go doc prints documentation for a package
* Go fmt formats source code files
* Go get downloads packages and installs them
* Go list lists all installed packages
* Go run compiles .go files and runs the executable
* Go test runs tests using files ending in “test.go”

**Module 3 – 1 Variables**

1. **Naming**

* Names are needed to refer to programming constructs: variables, functions
* Names must start with a letter
* Any number of letters, digits, underscores
* Case sensitive
* Don’t use keywords: if, case, package

1. **Variables**

* Data stored in memory
* Must have a name and a type
* All variables must have declarations
* Most basic declaration like keyword name type, such as var x int
* Can declare many on the same line, such as var x, y int

1. **Variable Types**

* Type defines the values a variable may take and operations that can be performed on it.
* Integer: only integral values, integer arithmetic (+,-,\*,…)
* Floating point: fractional (decimal) values, floating point arithmetic (may use different hardware)
* Strings: Byte (character) sequences, string comparison, search

**Module 3 – 2 Variable Initialization**

1. **Type Declarations**

* Defining an alias (alternate name) for a type
* May improve clarity
  + Type Celsius float64
  + Type IDnum int
* Can declare variables using the type alias
  + Var temp Celsius
  + Var pid IDnum

1. **Initializing Variables**

* Initialize in the declaration
  + Var x int = 100
  + Var x = 100
* Initialize after the declaration
  + Var x int
  + X= 100
* Uninitialized variables have a zero value
  + Var x int // x = 0
  + Var x string // x = “”

1. **Short Variable Declarations**

* Can perform a declaration and initialization together with the := operator
* X := 100
* Variable is declared as type of expression on the right hand side
* Can only do this inside a function

**Assignment 1**

1. Open cmd in your folder
2. set GOPATH=path\_folder

set GOPATH= E:\ Assignment\_1

1. set GOBIN=path\_folder\bin

set GOBIN=E:\ Assignment\_1\bin

1. Write the code in main.go

package main

import(

    "fmt"

)

func main() {

    fmt.PrintIn("Hello World!")

}

1. Write on cmd

go run src/main/main.go

1. To make an execute file, write this code.

cd bin

go install ../src/main/main.go

main.exe