Go

Go (also known as Golang) is a statically-typed, compiled programming language that was created by Google in 2007. It was initially designed by Robert Griesemer, Rob Pike, and Ken Thompson, who sought to create a language that was efficient, scalable, and easy to use. Go was officially announced in 2009 and has since become increasingly popular among developers.

Go is known for its simplicity, efficiency, and readability. Some of its key features include:

- Built-in concurrency support: Go has a unique approach to concurrency that makes it easy for developers to write code that can execute multiple tasks simultaneously.
- Fast compilation: Go code can be compiled very quickly, which makes it a popular choice for projects that require rapid development.
- Garbage collection: Go has an automatic garbage collector that helps manage memory usage and reduces the risk of memory leaks.
- Strong typing: Go is a statically-typed language, which means that type errors are caught at compile-time rather than run-time.
- Cross-platform compatibility: Go can be compiled to run on a variety of operating systems, including Linux, macOS, Windows, and more.

Go is often used for building network services and web applications, but it can also be used for a variety of other projects. Some common use-cases for Go include:

- Building APIs and microservices
- Writing system-level software
- Developing command-line tools
- Building data pipelines and processing large amounts of data
- Creating network tools and utilities

Overall, Go is a versatile and powerful programming language that is well-suited for a variety of use-cases. Its unique features and ease-of-use have made it a popular choice among developers, particularly those working on projects that require high performance and concurrency.

1. Getting Started

- Learn about Go and its history.
- Understand how to download and install Go on your computer.
- Learn the basic syntax of Go: variables, data types, operators, and control structures.
- Create a simple “Hello, World!” program in Go.

Tasks

1. Learn about Go and its history

- ☐ Read about the history of Go and its design goals.
- ☐ Watch introductory videos or read articles about Go to gain a basic understanding of the language.

2. Install Go on your computer

   - ☐ Download and install the latest version of Go on your computer.
   - ☐ Verify that the installation was successful by running a simple Go program, such as "Hello, World!".

3. Learn the basic syntax of Go

   - ☐ Read and understand the basic syntax of Go, such as variables, data types, operators, and control structures.
   - ☐ Follow tutorials or watch videos to learn how to write simple Go programs.

4. Create a simple "Hello, World!" program in Go

   - ☐ Write a program that prints "Hello, World!" to the console using the fmt package.
   - ☐ Compile and run the program to make sure it works.

2. Functions and Packages

- Understand functions in Go: declaration, parameters, and return values.
- Learn how to use built-in packages in Go: fmt, math, and time.
- Create your own custom packages and use them in your programs.
- Write a program that makes use of the math package to perform simple calculations.

Tasks

1. Understand functions in Go

   - ☐ Read and understand the concept of functions in Go, including how to declare, call, and return values from functions.
   - ☐ Follow tutorials or watch videos to learn how to write and use functions in Go.

2. Learn how to use built-in packages in Go

   - ☐ Read and understand how to use built-in packages in Go, such as fmt, math, and time.
   - ☐ Follow tutorials or watch videos to learn how to use these packages to perform common tasks.

3. Create your own custom packages

   - ☐ Learn how to create your own custom packages in Go, including how to define package names, export functions, and import packages in your code.

- ☐ Write your own custom package and use it in a simple program.

4. Write a program that makes use of the math package
   - ☐ Write a program that makes use of the math package to perform simple calculations, such as calculating the area of a circle or the square root of a number.
   - ☐ Compile and run the program to make sure it works.

3. Arrays, Slices, and Maps

- Understand arrays, slices, and maps in Go.
- Learn how to work with arrays: declaration, initialization, and manipulation.
- Learn how to work with slices: declaration, initialization, and manipulation.
- Learn how to work with maps: declaration, initialization, and manipulation.
- Write a program that makes use of arrays, slices, and maps to store and manipulate data.

Tasks

1. Understand arrays, slices, and maps in Go
   - ☐ Read and understand the concepts of arrays, slices, and maps in Go, including how to declare, initialise, and manipulate them.
   - ☐ Follow tutorials or watch videos to learn how to use arrays, slices, and maps in Go.
2. Learn how to work with arrays
   - ☐ Learn how to work with arrays in Go, including how to declare, initialise, and manipulate them.
   - ☐ Write a program that makes use of arrays to store and manipulate data.
3. Learn how to work with slices
   - ☐ Learn how to work with slices in Go, including how to declare, initialise, and manipulate them.
   - ☐ Write a program that makes use of slices to store and manipulate data.
4. Learn how to work with maps
   - ☐ Learn how to work with maps in Go, including how to declare, initialise, and manipulate them.
   - ☐ Write a program that makes use of maps to store and manipulate data.
5. Write a program that makes use of arrays, slices, and maps

- ☐ Write a program that makes use of arrays, slices, and maps to store and manipulate data, such as a simple database or a list of items.

4. Pointers and Structs

- Understand pointers and structs in Go.
- Learn how to work with pointers: declaration, initialization, and manipulation.
- Learn how to work with structs: declaration, initialization, and manipulation.
- Write a program that makes use of pointers and structs to store and manipulate data.

Tasks

1. Learn about pointers in Go
   - ☐ Read and understand the concept of pointers in Go, including how to declare, use, and manipulate them.
   - ☐ Follow tutorials or watch videos to learn how to work with pointers in Go.
2. Learn how to work with structs in Go
   - ☐ Learn how to work with structs in Go, including how to declare, initialise, and manipulate them.
   - ☐ Write a program that makes use of structs to store and manipulate data.
3. Learn how to work with interfaces in Go
   - ☐ Learn how to work with interfaces in Go, including how to declare, implement, and use them.
   - ☐ Write a program that makes use of interfaces to create a more modular and extensible code.
4. Learn how to handle errors in Go
   - ☐ Learn how to handle errors in Go, including how to use the built-in error type, how to return and handle errors from functions, and how to use the panic and recover mechanisms.
   - ☐ Write a program that handles errors properly and gracefully.
5. Write a program that makes use of pointers, structs, and interfaces
   - ☐ Write a program that makes use of pointers, structs, and interfaces to create a more sophisticated and efficient code, such as a simple game or a web server.

5. Concurrency and Channels

- Understand concurrency and channels in Go.
- Learn how to create goroutines and use channels to communicate between them.
- Understand how to use select statements to synchronise between goroutines.
- Write a program that makes use of concurrency and channels to perform a task in parallel.

Tasks

1. Learn how to work with concurrency in Go
    - ☐ Read and understand the concept of concurrency in Go, including how to use goroutines, channels, and select statements.
    - ☐ Follow tutorials or watch videos to learn how to write concurrent programs in Go.
2. Learn how to work with files in Go
    - ☐ Learn how to work with files in Go, including how to read and write files, create and delete files and directories, and use the file system API.
    - ☐ Write a program that makes use of file I/O in Go.
3. Learn how to work with network in Go
    - ☐ Learn how to work with network in Go, including how to create and use TCP and UDP sockets, how to send and receive data over the network, and how to use the net package.
    - ☐ Write a program that makes use of network I/O in Go.
4. Learn how to work with databases in Go
    - ☐ Learn how to work with databases in Go, including how to connect to a database, how to query and manipulate data, and how to use the database/sql package.
    - ☐ Write a program that makes use of a database in Go.
5. Write a complete project in Go
    - ☐ Choose a small project, such as a web server, a command-line tool, or a game, and write it from scratch in Go, making use of the concepts and techniques you've learned during the previous days.
    - ☐ Make sure the project is well-structured, well-documented, and tested.

Resources

Here are some free online resources that you can use to learn Go programming:

The official Go documentation - The official documentation for the Go programming language is a great place to start. It includes a tutorial for beginners as well as detailed references and examples.

A Tour of Go - A Tour of Go is a hands-on tutorial that takes you through the basics of Go programming. It includes interactive code examples and exercises to help you practice.

Go by Example - Go by Example is a collection of practical examples that cover the basics of Go programming. Each example includes a short explanation and runnable code.

Learn Go with Tests - Learn Go with Tests is a free online book that teaches Go programming through the use of tests. It covers the basics of the language as well as more advanced topics like concurrency and web development.

Gophercises - Gophercises is a collection of small programming exercises designed to help you practise and improve your Go programming skills. Each exercise includes a description and a solution for you to compare your code against.

Just for Func - Just for Func is a YouTube channel by Go expert Francesc Campoy that covers various topics related to Go programming. The channel includes tutorials, code reviews, and live coding sessions.

Projects

Here are some project ideas that can help you test your understanding of Go:

Simple Web Application - Create a simple web application using the net/http package. You can create a basic CRUD (Create, Read, Update, Delete) application for a specific domain.

RESTful API - Build a RESTful API using the net/http package and a third-party router library like Gorilla Mux. You can create an API for a specific domain and implement CRUD functionality.

CLI Tool - Develop a command-line interface (CLI) tool using the os and flag packages. The tool can perform some specific tasks like file operations, data processing, or system management.

Concurrency - Create a concurrent program that performs some tasks like downloading files, processing data, or managing network requests. Use goroutines and channels to handle concurrency.

Web Scraper - Build a web scraper that extracts data from websites and saves it to a file or database. Use a third-party library like GoQuery to scrape HTML data.

Chat Application - Create a simple chat application using the WebSocket protocol and a third-party library like Gorilla WebSocket. Users can join chat

rooms and exchange messages with each other.

Authentication and Authorization - Implement authentication and authorization in a web application or RESTful API using third-party libraries like JWT or OAuth2.

Image Processing - Build an image processing application that can perform various operations like cropping, resizing, or converting image formats. Use third-party libraries like Imaging or GoCV.

Machine Learning - Use the GoCV library to build a machine learning application that can perform tasks like face recognition, object detection, or image classification.

Game Development - Create a simple game using a third-party game engine like Engo or Ebiten. You can create a 2D or 3D game and use Go's concurrency features to handle game logic.

These are just a few examples of the many project ideas you can build to demonstrate your understanding of Go. You can choose the one that interests you the most and use your creativity to add your own features and functionalities to make it unique.