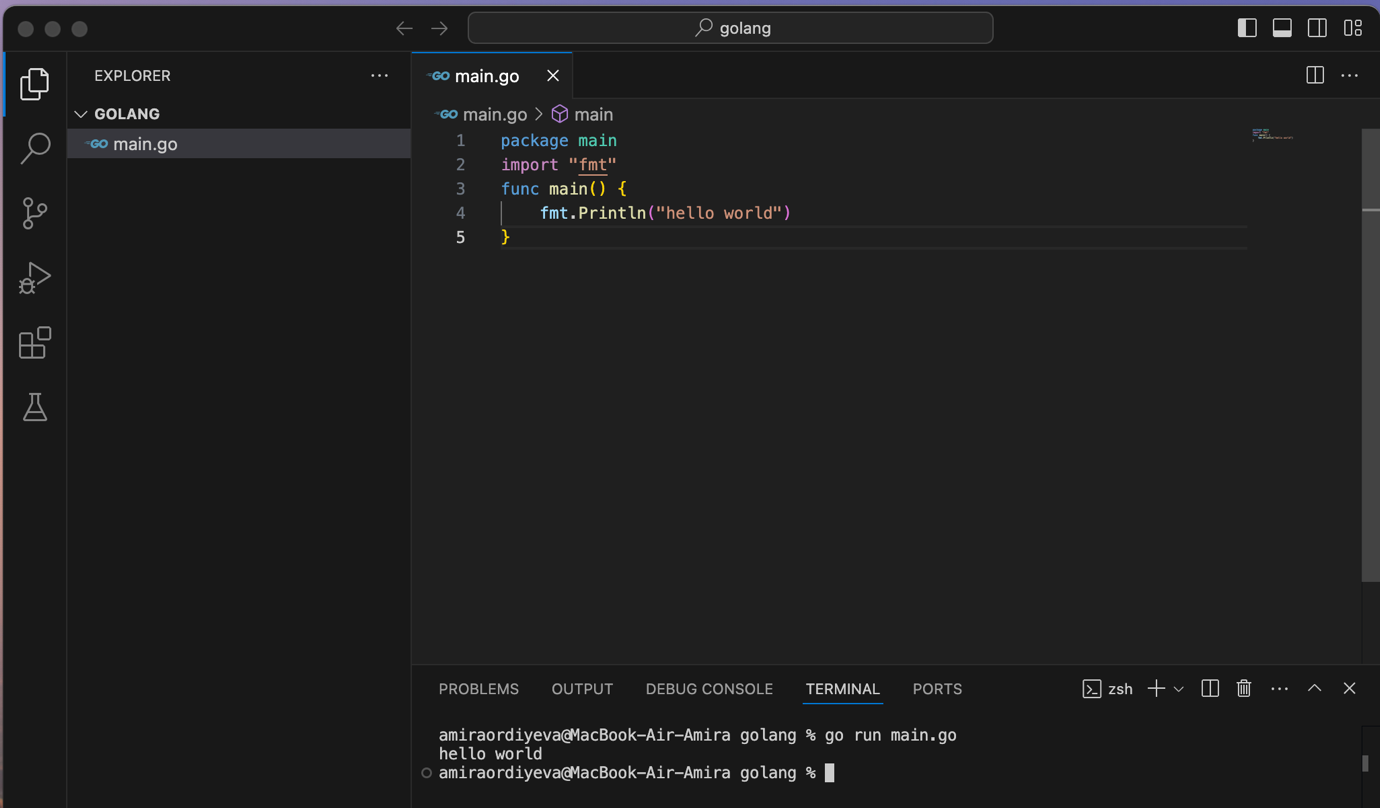
**Assignment 1, Go**

**Exercise 1: Hello, World!**



1. **Questions**:
   * ○  What command did you use to compile and run your Go program?

**i used “go run”**

* + ○  What are the basic components of a Go program (e.g., package, import,

function)?

**Package declaration in go defies the package to which a source file belongs. It is the very first statement in any go file and serves as the organizing structure for go code.**

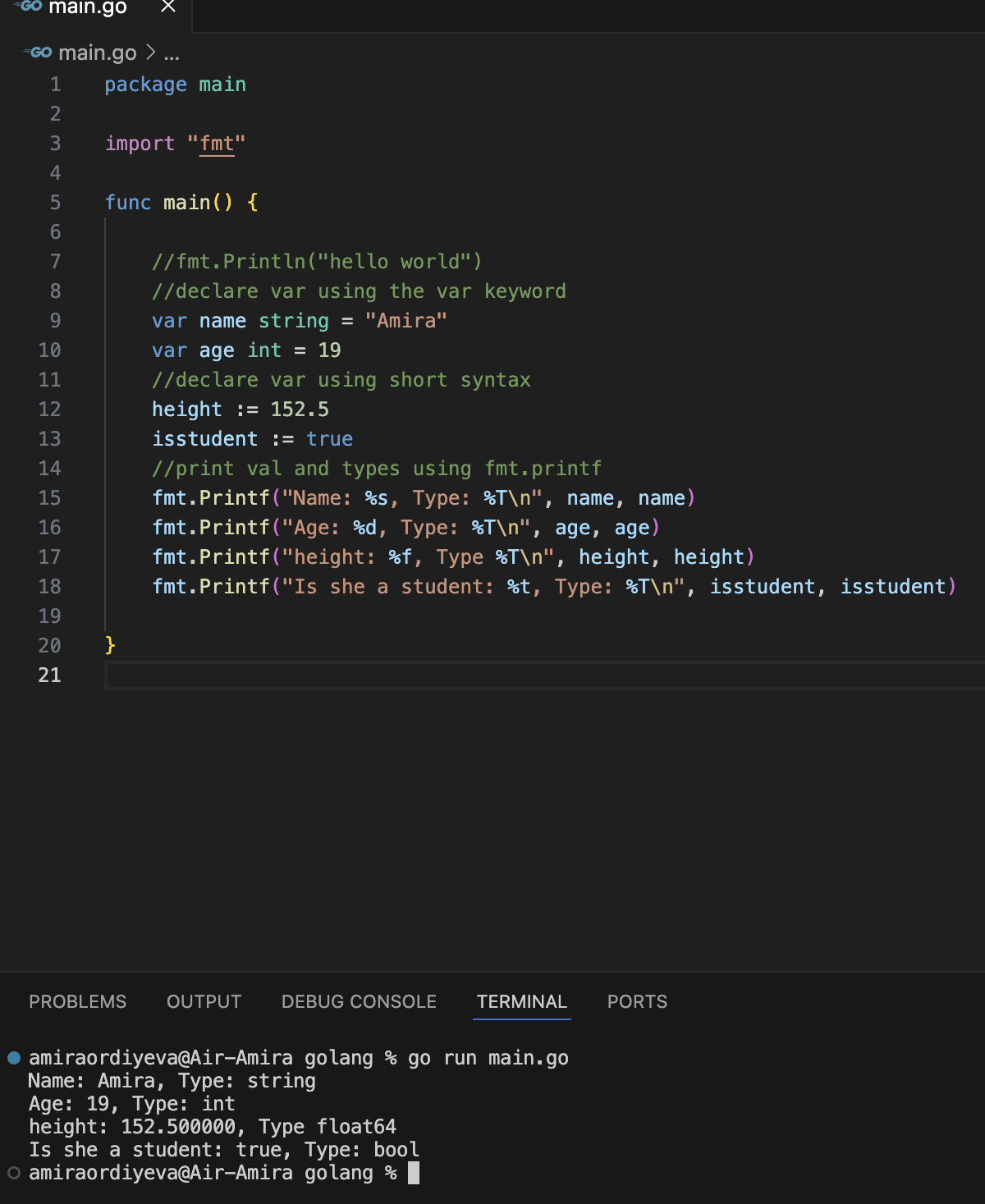
**Import – go programs use import to brinh in necessary packages. In the case, we imported the fmt packages to access the printin function**

**Function – the main functiong is the enrty point of go program. It is where the program starts executing**

* + ○  How does Go handle package imports?

Go program ise the import keyword to bring in external or standart packages.

**Exercise 2: Variables and Data Types**



1. **Questions**:
   * ○  What is the difference between using var and := to declare variables?

**Var can be used at both the package and function level, := only inside function. := is a short declaration syntax that automatically defines a type of the variable**

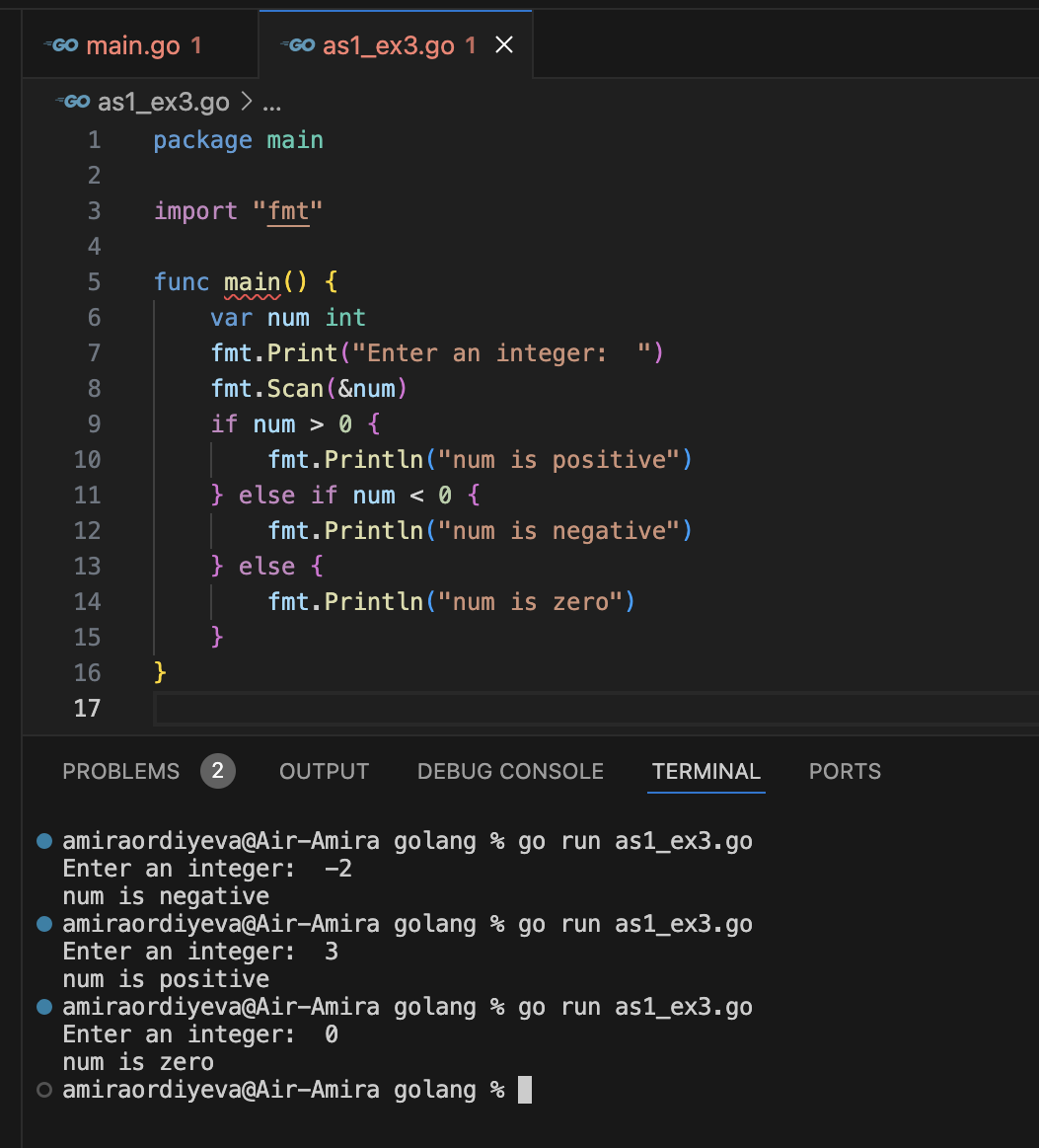
* + ○  How do you print the type of a variable in Go?

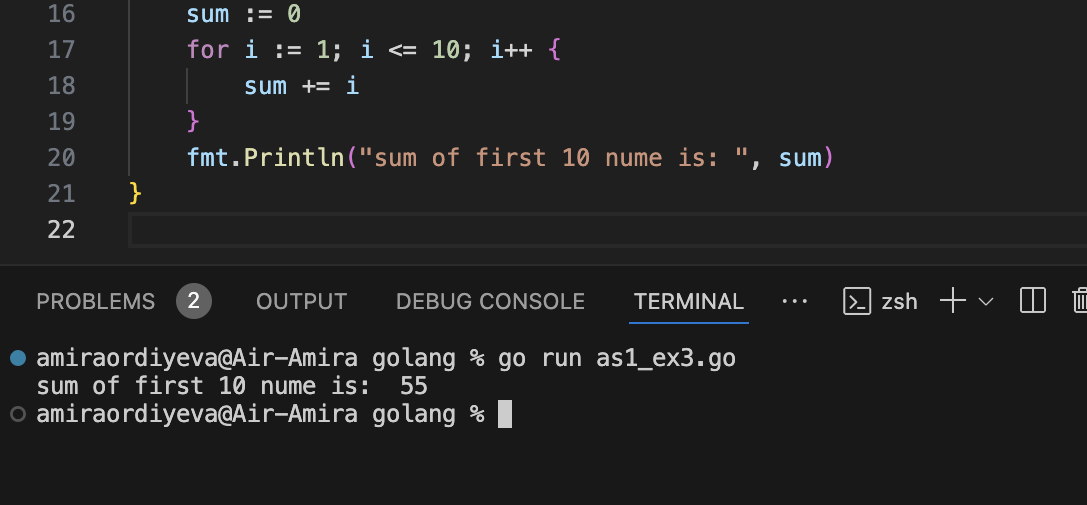
**Ftm.printf(“Type: “%T\n, something)**

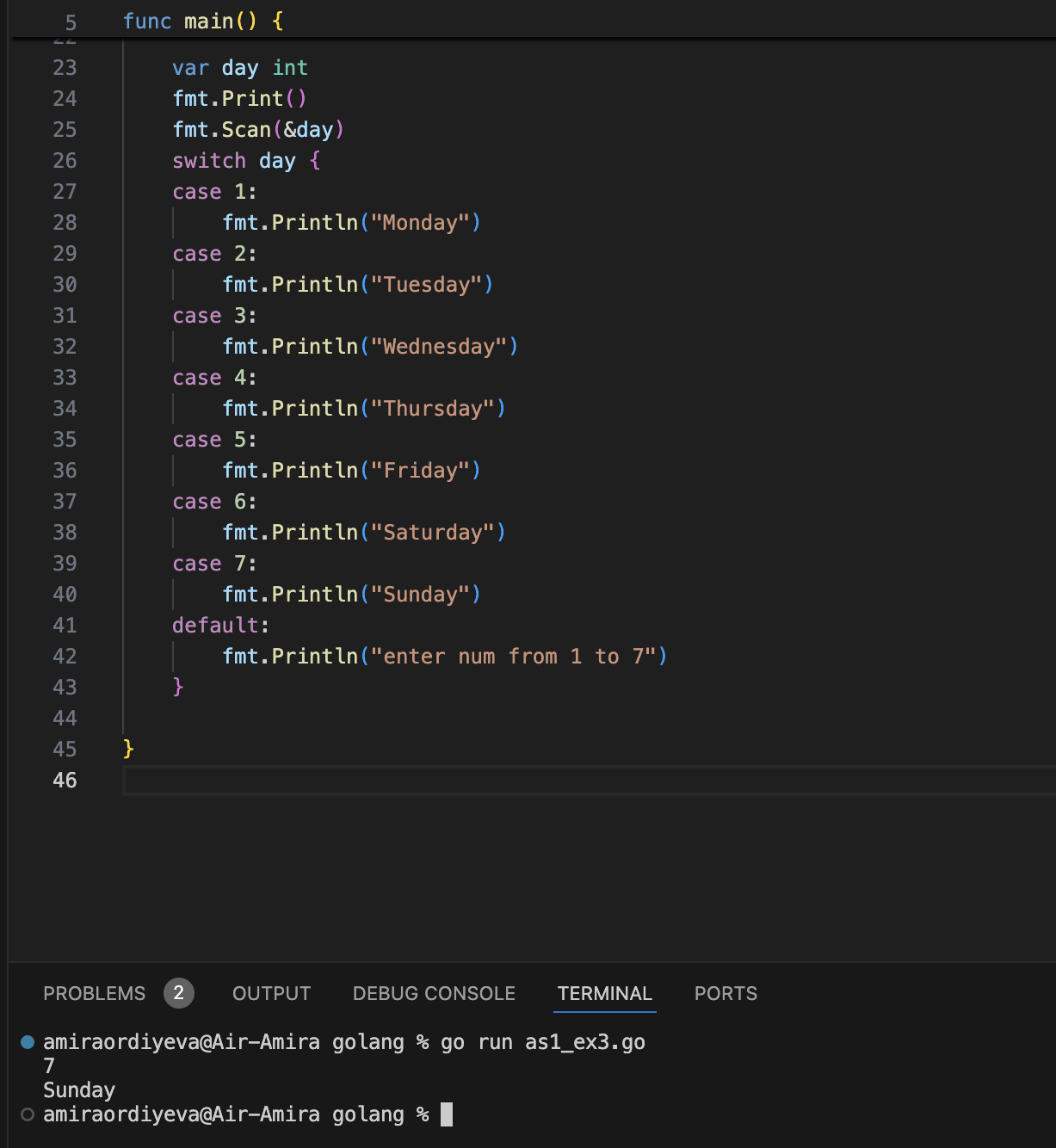
* + ○  Can you change the type of a variable after it has been declared? Why or why

not? **No, because golang statically typed language. If you need a different type, you will have to declare a new variable**

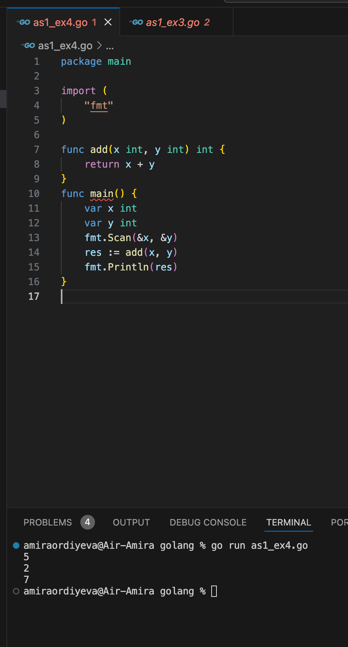
**Exercise 3: Control Structures**

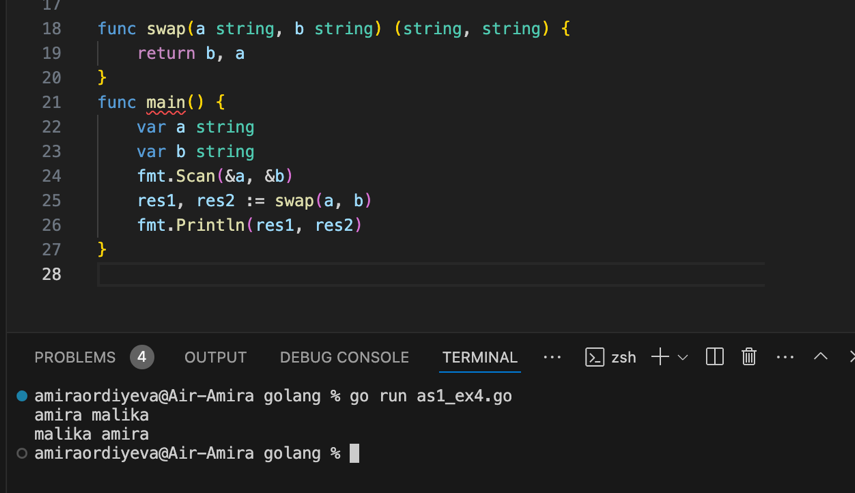


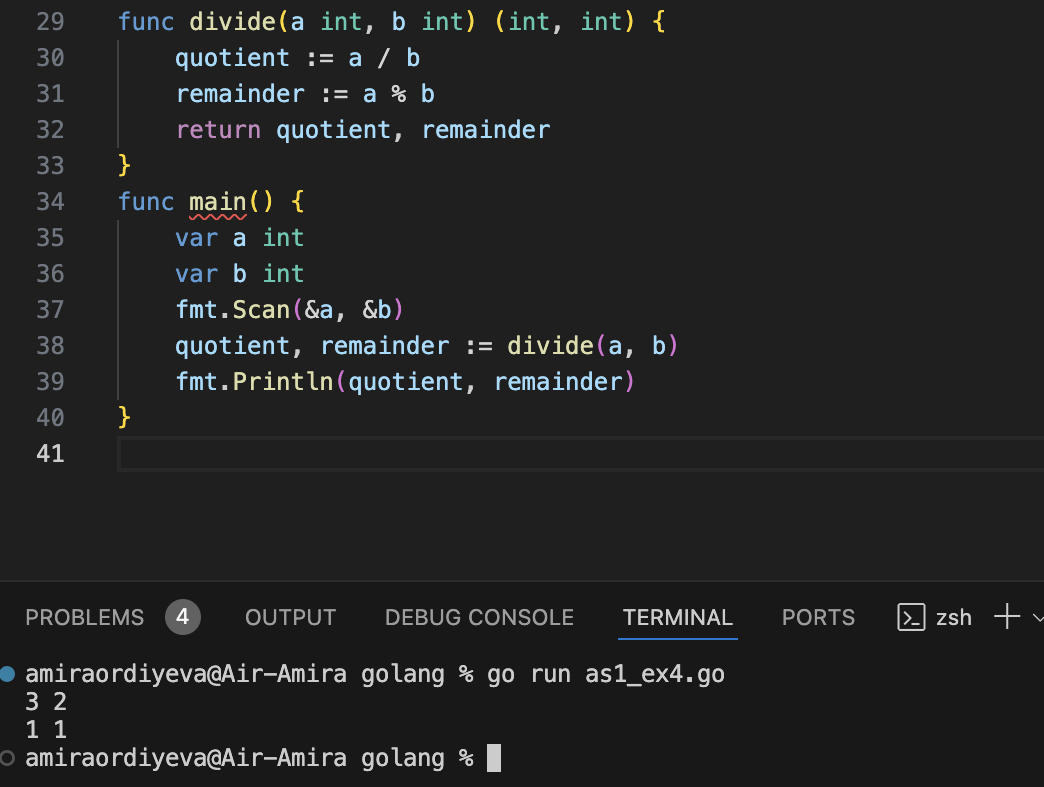




**Exercise 4: Functions and Multiple Return Values**







* ○  How do you define a function with multiple return values in Go?

**By specifying thw types of all return values in the function signature**

* ○  What is the significance of named return values in Go?

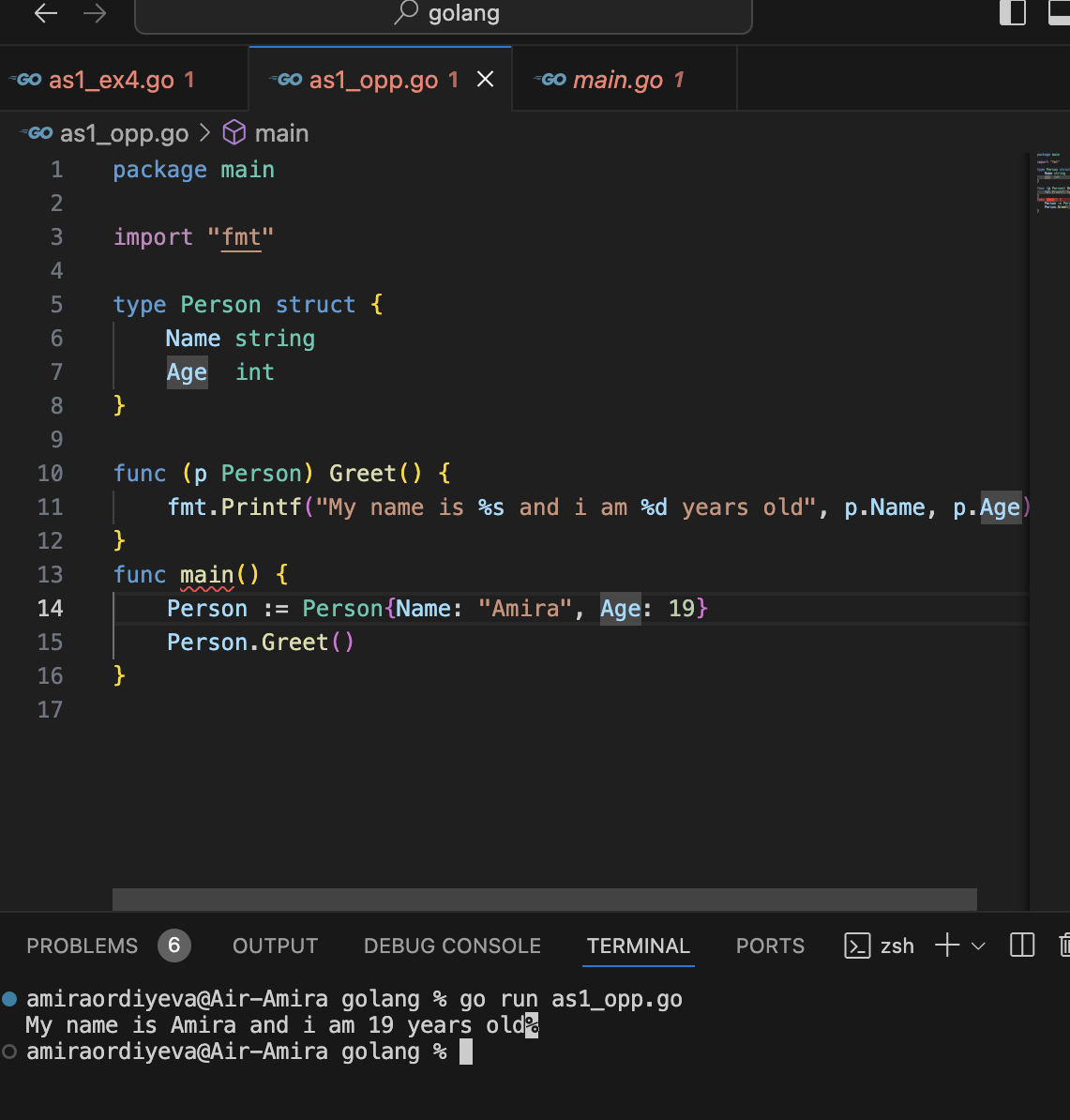
**Can return values without using the return keyword with variables**

* ○  How can you ignore certain return values if you don't need them?

We can use \_

**OOP in Golang**

1. **Exercise 1: Structs and Methods**



* How do you define a struct in Go?

**Defines using the type keyword followed by the name of the struct and the keyword struct.**

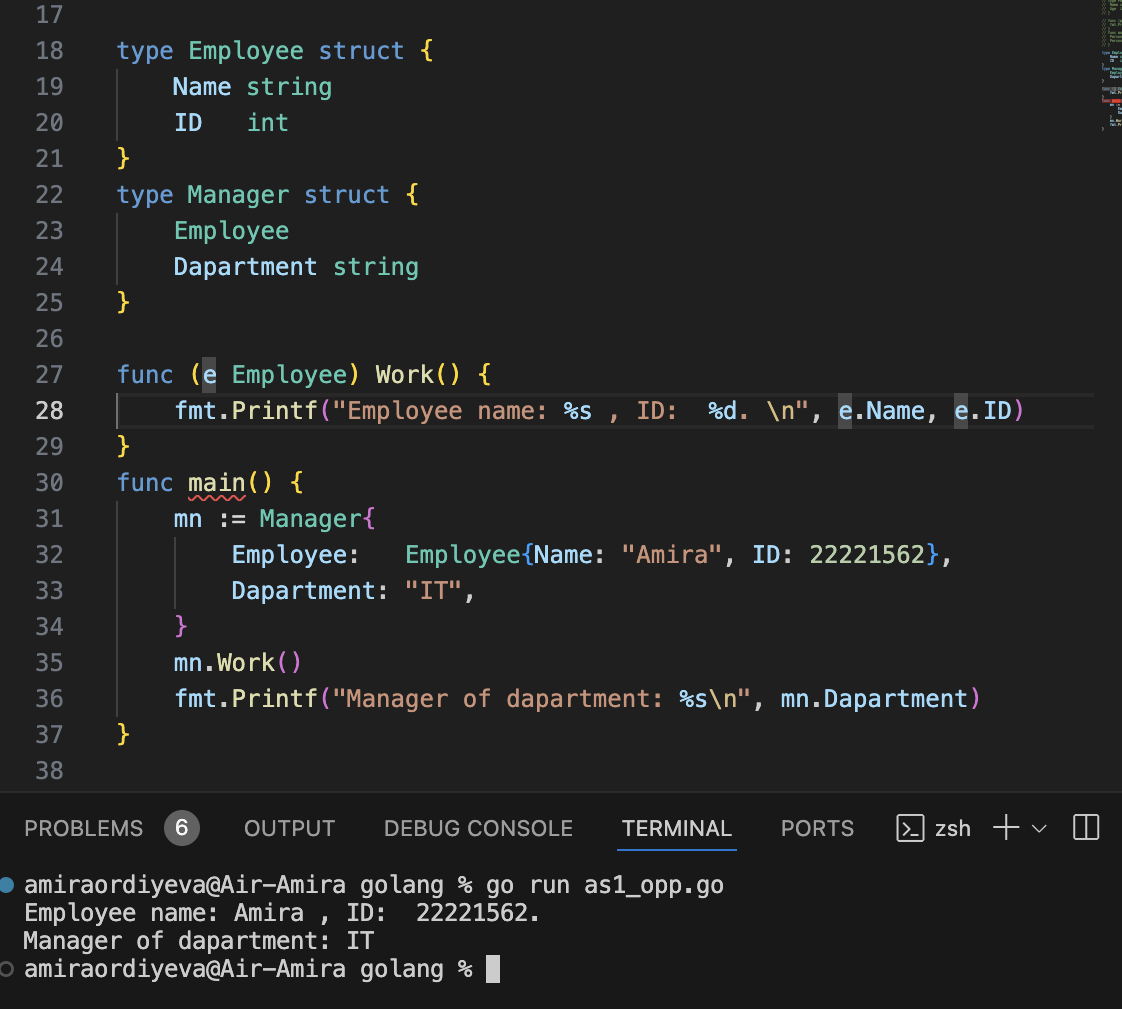
* ○  How do methods differ from regular functions in Go?

**Methods are assosiated with a specfic type , and regulat func are not associated with a specific type and can be used independently of types**

* ○  Can a method in Go be associated with types other than structs?

**No, methods can be only associated with types that are structs.**

Exercise 2: Embedding and Composition

****

* Questions:
* ○  What is embedding in Go, and how does it relate to composition?

**Embedding in go is when one struct placed inside another struct. This allow second struct to use the fields and methids of the first struct, just like it owns them.**

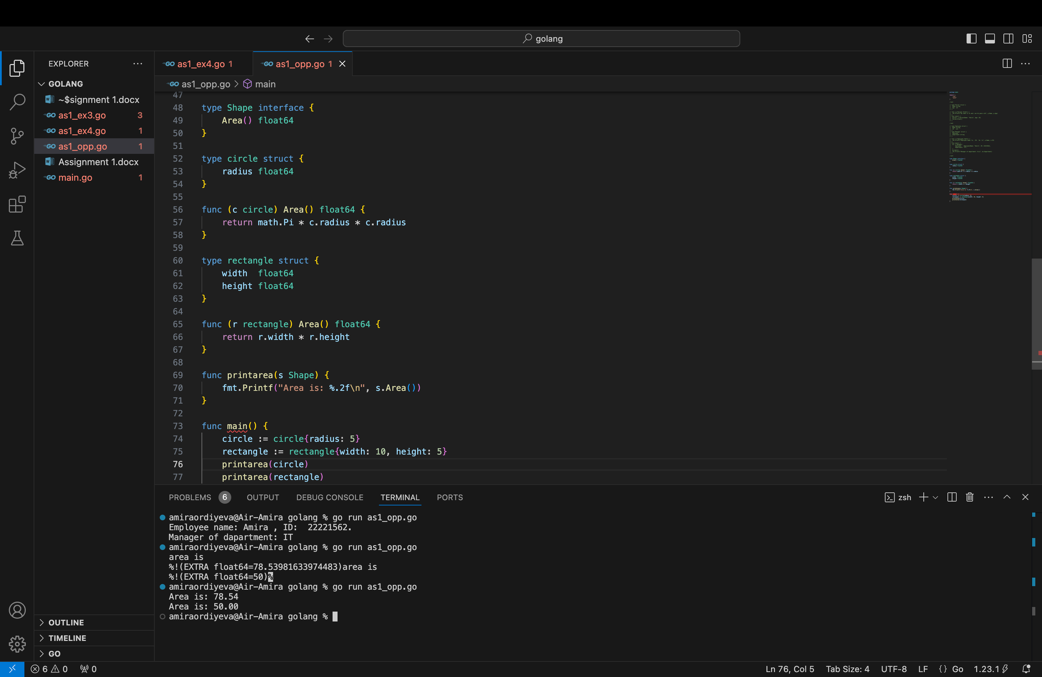
* ○  How does Go handle method calls on embedded types?

**For example when a struct manager embeds employee struct , go automatically allows you to call the embedded struct’s methods(work) directly on the outher manager struct**

* ○  Can an embedded type override a method from the outer struct?

No, but if thw outher struct defines a method with the same name, it will work and be used instead.

Exercise 3: Interfaces and Polymorphism



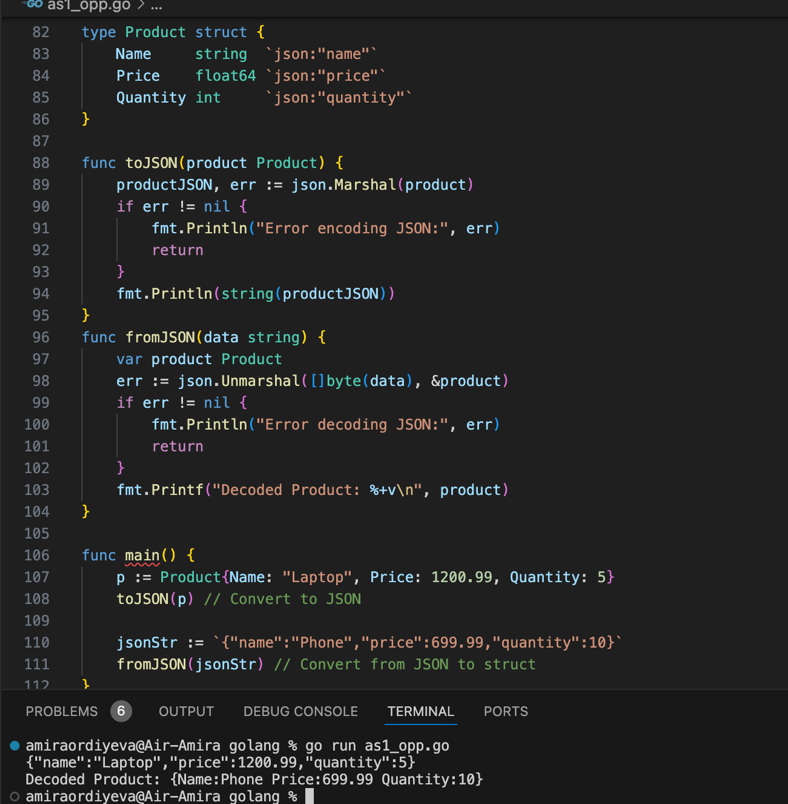
1. **Questions**:
   * ○  How do you define and implement an interface in Go?

**Be spesifing a set of methods signatures. If a type has al the methods that the interface requires, it automatically implements the interface**

* + ○  What is the role of interfaces in achieving polymorphism in Go? **interfaces allow different types to be treated the same way, as long as they implement the same interface.**
  + ○  How can you check if a type implements a certain interface?

**using a type assertion or just try assigning the type to a variable of the interface type. If it doesn't**

**Exercise 4: Working with JSON using Structs**



**Questions**:

* + ○  How do you work with JSON in Go?

By using the `encoding/json` package. Use `json.Marshal` to convert structs to JSON and `json.Unmarshal` to convert JSON to structs.

* + ○  What role do struct tags play in JSON encoding/decoding?

Struct tags (like `json:"name"`) define how struct fields map to JSON keys. They control field names and which fields are included or excluded.

* + ○  How do you handle errors that may occur during JSON encoding/decoding?

Check for errors right after calling `json.Marshal` or `json.Unmarshal`. Handle them by printing an error message or taking appropriate action.