Filenames, Keywords and Identifiers

This lesson describes what a basic Go program usually comprises of.

We'll cover the following

- Filename
- Keyword
- Identifiers
  - Blank identifier
  - Anonymous
- The basic structure of a Go program

Filename#

Go source code is stored in **.go** files. Their filenames consist of lowercase-letters, like *educative.go*. If the name consists of multiple parts, they are separated by underscores '_', like *educative_platform.go*. Filenames cannot contain spaces or any other special characters. A source file contains code lines whose length has no intrinsic limits.

Keyword#

A reserved word, with a special meaning in a programming language, is called a **keyword**. Below is the set of *25 keywords*, or reserved words, used in Go-code:

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

Keywords

Identifiers#

An **identifier** is a name assigned by the user to a program element like a variable, a function, a template, and a class, etc. Nearly all things in Go-code have a name or an *identifier*. Like all other languages in the C-family, Go is case-sensitive. Valid identifiers begin with a letter (a letter is every letter in Unicode UTF-8) or _ and are followed by 0 or more letters or Unicode digits, like X56, group1, _x23, i, and ɵε12.

The following are **NOT** valid identifiers:

- `1ab` because it starts with a digit
- `case` because it is a keyword in Go
- `a+b` because operators are not allowed

Apart from the keywords, Go has a set of **36 predeclared identifiers** which contain the names of elementary types and some basic built-in functions (see Chapter 4 (https://www.educative.io/collection/page/10370001/6289391964127232/5402231337648128)); all these will be explained further in the next chapters:

append	false	iota	recover
bool	float32	len	string
byte	float64	make	true
cap	imag	new	uint
close	int	nil	uint8
complex	int8	panic	uint16
complex64	int16	print	uint32
complex128	int32	println	uint64
copy	int64	real	uintptr

Predeclared Identifiers

Blank identifier#

The `_` itself is a special identifier, called the **blank identifier**. Like any other identifier, `_` can be used in declarations or variable assignments (and any type can be assigned to it). However, its value is discarded, so it can no longer be used in the code that follows. We will demonstrate its use later (https://www.educative.io/collection/page/10370001/6289391964127232/5099431512768512/#blank-identifier) in the course.

Anonymous#

Sometimes it is possible that even functions have no name because it is not really necessary at that point in the code and not having a name even enhances flexibility. Such functions are called **anonymous**. We will demonstrate their use later (https://www.educative.io/collection/page/10370001/6289391964127232/4955786063118336/#using-function-literals) in the course.

The basic structure of a Go

program#

Programs consist of *keywords*, *constants*, *variables*, *operators*, *types* and *functions*. It is also important to know the delimiter and punctuation characters that are a part of Golang.

The following *delimiters* are used in a Go program:

- Parentheses ()
- Braces {}
- Brackets []

The following *punctuation characters* are used in a Go program:

- `.`
- `,`
- `;`
- `:`
- `...`

The code is structured in statements. A statement doesn't need to end with a `;` (like it is imposed on the C-family of languages). The Go compiler automatically inserts semicolons at the end of statements. However, if multiple statements are written on one line (a practice which is not encouraged for readability reasons), they must be separated by `;`.

That's it about the basic structure and elements of Golang. In the next lesson, we'll see a basic component of the Go program.

← Back

Test Yourself

Next →

Import Functionality

Mark as Completed