Built-in Functions

This lesson describes some built-in functions of Go and explains their tasks.

WE'LL COVER THE FOLLOWING

- Introduction
- Explanation
  - `close`
  - `len` and `cap`
  - `new` and `make`
  - `copy` and `append`
  - `panic` and `recover`
  - `print` and `println`
  - `complex`, `real`, and `imag`

Introduction

The predefined functions, which can be used without having to import a package to get access to them, are called **built-in** functions. They sometimes apply to different types, e.g. `len`, `cap`, and `append`, or they have to operate near system level like `panic`. That's why they need support from the compiler.

Explanation

> **Note:** We'll only list the name of the functions and their functionalities in this lesson. Some of the functions are covered before in detail and used multiple times, so we won't run them from scratch. Functions that were not discussed previously, their running examples will be provided later in the course in detail, section by section. So don't panic if you don't have a clear idea.

The following are some common and important built-in functions provided by Go:

`close`

It is used in channel communication.

`len` and `cap`

The function `len` gives the *length* of a number of types (strings, arrays, slices, maps, channels). Whereas, `cap` is the *capacity*, the maximum storage (only applicable to slices and maps).

`new` and `make`

Both `new` and `make` are used for allocating memory. The function `new` is used for value types and user-defined types like structs. Whereas, `make` is used for built-in reference types (slices, maps, channels). They are used like functions with the type as its argument:

```
new(type)
make(type)
```

`new(T)` allocates *zeroed* storage for a new item of type `T` and returns its address. It returns a pointer to the type T (details are in Chapter 8), and it can be used with primitive types as well:

```
v := new(int) // v has type *int
```

The function `make(T)` returns an initialized variable of type `T`, so it does more work than `new`.

> **Remark:** `new()` is a function; don't forget its parentheses.

`copy` and `append`

These are used for copying and concatenating slices (see Chapter 5).

`panic` and `recover`

These both are used in a mechanism for handling errors (see Chapter 11).

`print` and `println`

These are low-level printing functions. Use the `fmt` package in production programs.

`complex`, `real`, and `imag`

These are used for making and manipulating *complex numbers*.

> **Note:** We won't be covering `complex()`, `real()`, and `imag()` anywhere because they don't fall within the scope of this course. If you want to study them in detail, you can follow their documentation. For `complex()` click here. For `real()` click here. For `imag()` click here.

Built-in functions are never hard to understand. We can always learn them when we feel a need to use them. You just need to call them and see what they do. The next lesson brings an important function type known as *recursive function*.