# Import packages / modules

## I have a helper file that I would like to use within the same project.

If say your directory layout is like below:

```
src
├main.go
    helper.go
```

A very simple directory, you wrote some helper function in `helper.go` and you would like to use it in `main.go`. Now because they are under the same *package* you do not need do anything importing to use the functions / global variables that's defined in `helper.go` in `main.go`, you can just call it as if they are already in the same file.

> However, when you compile you would need to provide either `build / run` with both of the files that you use.
> If you just run `go build main.go` or `go run main.go` it will not work. You must run `go build main.go helper.go` or `go run main.go helper.go`.
> To link all the files you need to explicitly mention it.

## I have a package that I would like to use within the same project

So instead of putting the helper file in the same package you decided to be a little fancy and include it in another directory (package) on it's own like below:

```
src
├main.go
    - helper
   ├├helper.go
```

Now how can I use it in `main.go`? Since it is in another package now you would need to explicitly import it in `main.go` in order to use it. The way that you would import it is by importing it using the `module_path` name.

For example, if this local project's `module_path` is github.com/tamaarine/testproject, or whatever, it doesn't matter if it is published under that URL, you would import the `helper` package by writing

```
import (
	"github.com/tamaarine/testgoproject/helper"
)
```

Then you can access the functions or variables that's defined in that package by using `helper.<function/variable name>`.

> Do keep in mind that those functions or variables that you decide to export through this way need to be explicitly exported, by making their name capitalized. So instead of function name add, it needs to be Add. To tell Go that this function is going to be exported and be used.

> Doing it this way you do not need to add any files to the `build / run` CLI. You can just do `go run main.go` and it will compile without any complains.

# I made a module that I haven't published yet to a repository, but I want to use it in another local project, how do I do it?

Okay cool, say you got yourself a module under the `module_path=github.com/tamaarine/testgoproject` but you haven't published it into any repository but your local project wants to use it because you want your stuff to be modularized. How do you do it?

Well you would use the replace directive within your `go.mod` file.

```
module local/project2

go 1.20

replace github.com/tamaarine/testgoproject => /home/tamarine/GoProject/Project1
```

In this case, say we are making a new project under the `module_path=local/project2` and we want the module that haven't been published to a repository yet. By using the replace directive you can tell Go to find the module in another place, in this case in the local file directory.

Left side consist of the module that you would like to replace, and the right side is the local directory to the module root directory where it contains the `go.mod` file.

Using replace directive basically substitute the module path with another. This can be useful if you're developing a new module but you haven't published the code to a repository yet but just want to test it locally. Or you found an issue with the dependency, so you cloned the repo and fixed it and wanted to test it without pushing the change yet.

---

Revision #1
Created 2023-05-08 01:39:48 UTC by Tamarine
Updated 2023-05-08 02:09:36 UTC by Tamarine