Golang Basics

Outlines basic structures, syntax and functions in this note.

Hello world

```
package main

import "fmt"

func main() {
        fmt.Println("Hello, 世界")
}
```

Hello world and a demonstration of the time module

```
package main

import (
        "fmt"
        "time"
)

func main() {
        fmt.Println("Welcome to the playground!")

        fmt.Println("The time is", time.Now())
}
```

Random number generator example

```
package main

import (
        "fmt"
        "math/rand"
)
```

```
func main() {
        fmt.Println("My favorite number is", rand.Intn(10))
}
```

Exported and imported names

In Go, a name is exported if it begins with a capital letter. For example, `Pizza` is an exported name, as is `Pi`, which is exported from the `math` package.
`pizza` and `pi` do not start with a capital letter, so they are not exported.
When importing a package, you can refer only to its exported names. Any "unexported" names are not accessible from outside the package.
Run the code. Notice the error message.
To fix the error, rename `math.pi` to `math.Pi` and try it again.

```
package main

import (
        "fmt"
        "math"
)

func main() {
        fmt.Println(math.pi)
}
```

Function declaration

```
package main

import "fmt"

func add(x int, y int) int {
        return x + y
}

func main() {
        fmt.Println(add(42, 13))
}
```

The type of the output comes after the function declaration. It is like `int main(type arg, type arg)` in C, except its `func main(type arg, type arg) int`, it comes at the end.

Multiple function outputs

```
package main

import "fmt"

func swap(x, y string) (string, string) {
        return y, x
}

func main() {
        a, b := swap("hello", "world")
        fmt.Println(a, b)
}
```

Each of the outputs have their own unique type, and hence you need to declare `(string, string)` at the end to denote that.

Named return

```
package main

import "fmt"

func split(sum int) (x, y int) {
        x = sum * 4 / 9
        y = sum - x
        return
}

func main() {
        fmt.Println(split(17))
}
```

You can see that instead of just giving the function output a type and forgetting about it, we have declared the type and the name of the variable which will be the

output of the function `sum` in this case.
This is very useful for short functions, but it impacts readability in the long run.