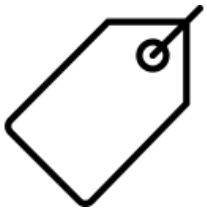
Use of Labels - goto

In this lesson you'll study how a program execution can be controlled with labels.

WE'LL COVER THE FOLLOWING

Introduction

A **label** is a sequence of characters that identifies a location within a code. A code line which starts with a *for*, *switch*, or *select* statements can be preceded with a **label** of the form:



```
IDENTIFIER:
```

This is the first word ending with a colon **:** and preceding the code (*gofmt* puts it on the preceding line).

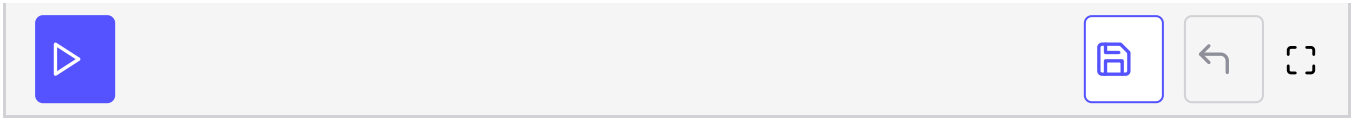
Use of labels in Go

Let's look at an example of using a label in Go:

```
package main
import "fmt"

func main() {
    LABEL1:                          // adding a label for location
    for i := 0; i <= 5; i++ {        // outer loop
        for j := 0; j <= 5; j++ {        // inner loop
            if j == 4 {
                continue LABEL1                  // jump to the label
            }
            fmt.Printf("i is: %d, and j is: %d\n", i, j)
```

```
        }
    }
}
```



Control using Labels

As you can see, we made a label `LABEL1` at **line 5**. At **line 6**, we made an outer loop: `for i := 0; i <= 5; i++` that will iterate **5** times. At **line** 7, we made an inner loop: `for j := 0; j <= 5; j++` that will iterate **5** times for each `i`. You may have noticed that we added `continue LABEL1` at **line 9**. The execution will jump to the label `LABEL1` if condition `j ==4` is satisfied at **line 8**. You can see that cases `j == 4` and `j == 5` do not appear in the output because the label precedes the outer loop, which starts `i` at its next value continuing the outer loop and causing the `j` in the inner for loop to reset to **0** at its initialization.

The `continue` can be very handy when dealing with nested loops. If we replace `continue` with `break` in our program above, our output is only:

```
i is: 0, and j is: 0
i is: 0, and j is: 1
i is: 0, and j is: 2
i is: 0, and j is: 3
```

We see that `break` jumps out of and effectively stops both loops here. `break LABEL` can be used, not only from a for- loop, but also to break out of a `switch`.

The name of a label is case-sensitive, it can be put in uppercase to increase readability, but this is merely a convention.

Use of `goto` keyword

In Go, there is a `goto` keyword, which has to be followed by a label name:

```
goto IDENTIFIER
```

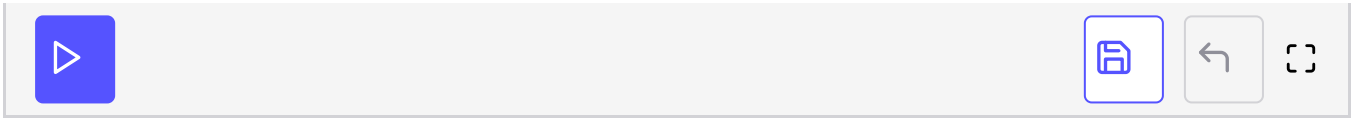
Let's look at an example:

```
package main
import "fmt"
```

```
func main() {
    i:=0
    HERE:           // adding a label for a location

    fmt.Printf("%d",i)
    i++
    if i==5 {
        return
    }
    goto HERE      // goto HERE

}
```



Control using goto

As you can see in the above code, we made a label `HERE` at **line 6**. At **line** 7, we are printing the value of `i` and then incrementing `i` by **1**. Next, we have a condition at **line 9**: `i==5`. If this condition is *true*, we'll return from *main*. Otherwise, control will go at **line 12**, which states: `goto HERE`, and from there, control will transfer from **line 12** to **line 6**. It's evident that `i` will reach the maximum value of **5** because, at `i==5`, the program's execution ends due to the `return` statement.

See the following program and notice the problem:

```
func main() {
  LABEL1:                          // adding a label for location
    for i := 0; i <= 5; i++ {        // outer loop
      for j := 0; j <= 5; j++ {        // inner loop
        if j == 4 {
          goto LABEL1                  // jump to the label
        }
        fmt.Printf("i is: %d, and j is: %d\n", i, j)
    }
  }
}
```

If you use `goto` you get an infinite loop that is stopped by the runtime after a number of iterations. However, a backward `goto` quicky leads to unreadable *spaghetti-code* and should not be used. There is always a better alternative. Use the `goto LABEL` only with `LABEL` defined *after* the goto line! The use of labels, and certainly goto, is discouraged because it can quickly lead to bad program design, and the code can almost always be written more readable

without using them.

That’s it about how control is transferred using labels. There is a quiz in the next lesson for you to solve.