

A photograph of a server room with rows of server racks illuminated by blue light. The racks are filled with server units, and the perspective shows a long aisle leading into the distance. The text "Introduction to Programming in Go" is overlaid in white on a dark horizontal band across the top of the image.

Introduction to Programming in Go

3. Go Program Control Structures

Module Topics

1. Operators
2. **if** statements
3. **for** loops
4. **switch** statements

Operators

Go Operator Differences

1. Standard C type operator behavior except for....
2. No mixed mode operations.
3. No exponent operator.
4. Increment/decrement (`x++` and `x--`) are statements, not expressions.
5. Prefix notation not allowed - ie. `++x` is illegal.
6. Multiple assignments are done in parallel.

Mixed Mode Operations

```
// Example 03-01  Operators
package main

import "fmt"

func main() {
    a, b, c := uint8(1), uint16(1), int8(1)
    fmt.Println("1. a + b =", a + b)
    fmt.Println("2. a + c =", a + c)
    fmt.Println("3. a + uint8(b)=", a + uint8(b))
    fmt.Println("4. uint8(a) + c=", int8(a) + c)
}
```

```
[Module03]$ go run ex03-01.go
./ex03-01.go:10: invalid operation: a + b (mismatched types uint8 and uint16)
./ex03-01.go:11: invalid operation: a + c (mismatched types uint8 and int8)
```

Mixed Mode Operations

```
// Example 03-01  Operators
package main

import "fmt"

func main() {
    a, b, c := uint8(1), uint16(1), int8(1)
    //fmt.Println("1. a + b =", a + b)
    //fmt.Println("2. a + c =", a + c)
    fmt.Println("3. a + uint8(b)=", a + uint8(b))
    fmt.Println("4. uint8(a) + c=", int8(a) + c)
}
```

```
[Module03]$ go run ex03-01.go
3. a + uint8(b)= 2
4. uint8(a) + c= 2
```

Parallel Assignment

```
// Example 03-02  Parallel Assignment
package main

import "fmt"

func main() {
    first, last := "York", "New"
    fmt.Println(first, last)
    first, last = last, first
    fmt.Println(first, last)
}
```

```
[Module03]$ go run ex03-02.go
York New
New York
```

Conditionals – if statements

Conditionals - Differences in Go

1. Standard C type conditional behavior except for....
2. No parentheses "(..)" around test condition.
3. Local variables can be defined in the `if` statement itself.
4. Braces "{ .. }" are mandatory for all then and else blocks.
5. Opening "{" for each block cannot start on a new line.
6. The `else` keyword cannot appear at the start of a new line.

Basic Conditional Statement

```
// Example 03-03 Basic if Statement
package main

import "fmt"

func main() {
    x := 22
    if x == 0 {
        fmt.Printf("%d is zero\n", x)
    } else if x % 2 == 0 {
        fmt.Printf("%d is even\n", x)
    } else {
        fmt.Printf("%d is odd\n", x)
    }
}
```

```
[Module03]$ go run ex03-03.go
22 is even
```

Local Variables

```
// Example 03-04 Local Variables in Conditionals
package main

import "fmt"

func main() {
    if x, y := 22, "hi" ; x == 0 {
        fmt.Println("Value of x=", x, " y=", y)
    } else if x % 2 == 0 {
        fmt.Println("Value of =", x, " y=", y)
    } else {
        fmt.Println("Value of x=", x, " y=", y)
    }
}
```

```
[Module03]$ go run ex03-04.go
Value of x= 22 y= hi
```

Variable Assignment

```
// Example 03-05 Non-local variables in Conditionals
package main

import "fmt"

var x = 1
var y = "bye"

func main() {
    if x, y = 22, "hi" ; x % 2 {
        fmt.Println("Value of x=", x, " y=", y)
    } else {
        fmt.Println("Value of x=", x, " y=", y)
    }
}
```

```
[Module03]$ go run ex03-05.go
Value of x= 22 y= hi
```

For Loops

Loops - Differences in Go

1. The only loop construct in Go is the `for` loop.
2. Can function as a while loop.
3. No parentheses "`(..)`" allowed in the for clause.
4. Braces "`{..}`" are mandatory for the loop body.
5. The pre and post terms in the for clause can be empty.

Basic For Loop

```
// Example 03-06 Basic for loop

package main

import "fmt"

func main() {
    var total = 0
    for count := 0; count < 100; count++ {
        total += count
    }
    fmt.Println("total = ", total)
```

```
[Module03]$ go run ex03-06.go
total = 4950
```

Basic For Loop - Multiple Variables

```
// Example 03-07 Basic for loop
....

func main() {
    var total = 0
    for i, m := 0, "abort" ; i < 100; i++ {
        total += i
        if total > 100 {
            fmt.Println(m)
            break
        }
    }
    fmt.Println("total = ", total)
}
```

```
[Module03]$ go run ex03-07.go
abort
total = 105
```

For Loop - Non-local Variables

```
// Example 03-08 Non-local Variables
....

func main() {
    var total, i = 1000, 1000

    for i, total := 0, 0 ; i < 100; i++ {
        if total > 200 {
            continue
        } else {
            total += i
        }
    }
    fmt.Println("total = ", total)
}
```

```
[Module03]$ go run ex03-08.go
total = 205
```

For Loop as a While Loop

```
// Example 03-09 For Loop as While Loop
....

func main() {

    count := 0

    for count < 2 {
        count++
    }
    fmt.Println("count = ", count)
}
```

```
[Module03]$ go run ex03-09.go
count = 2
```


Looping Using Ranges

```
// Example 03-10 Looping Using Range
....

func main() {

    test := "Hi!"

    for i, c := range test {
        fmt.Printf("Letter %d is %#U\n", i, c)
    }
}
```

```
[Module03]$ go run ex03-10.go
Letter 0 is U+0048 'H'
Letter 1 is U+0069 'i'
Letter 2 is U+0021 '!'
```

Switch Statement

Switch - Differences in Go

1. Go **switch** statements are more general.
2. Test values can be any expression, not just integers.
3. Cases break automatically unless the **fallthrough** keyword is used.
4. The **break** statement breaks out of a clause at any point.
5. Test values are optional, cases will execute on a true result.

Simple Switch Statement

```
// Example 03-11  Simple Switch Statement
```

```
...
```

```
func main() {  
    for i := 0; i < 3; i++ {  
        switch i {  
            case 0:  
                fmt.Println("Case 0")  
            case 1:  
                fmt.Println("Case 1")  
            fallthrough  
            default:  
                fmt.Println("Default")  
        }  
    }  
}
```

```
[Module03]$ go run ex03-11.go  
Case 0  
Case 1  
Default  
Default
```

Switch Statement Break

```
// Example 03-12  Switch Statement Break
...

func main() {
    for i := 0; i < 3; i++ {
        switch i {
        case 0:
            fmt.Println("Case 0")
        case 1:
            fmt.Println("Case 1")
            break
            fmt.Println("After break")
        default:
            fmt.Println("Default")
        }
    }
}
```

```
[Module03]$ go run ex03-12.go
Case 0
Case 1
Default
```


Switch - Non-integral Test Value

```
// Example 03-13 Non-integral Test
...

func main() {
    os := "fedora"

    switch os {
    case "fedora", "redhat":
        fmt.Println("Open Source")
    case "Windows":
        fmt.Println("Proprietary")
    default:
        fmt.Println("unknown")
    }
}
```

```
[Module03]$ go run ex03-13.go
Open Source
```

Switch - No Test Value

```
// Example 03-14 No Test Value
...

func main() {
    x := 22

    switch {
    case x == 0:
        fmt.Println("zero")
    case x % 2 == 0:
        fmt.Println("even")
    default:
        fmt.Println("odd")
    }
}
```

```
[Module03]$ go run ex03-14.go
even
```

Lab 3: Control Structures