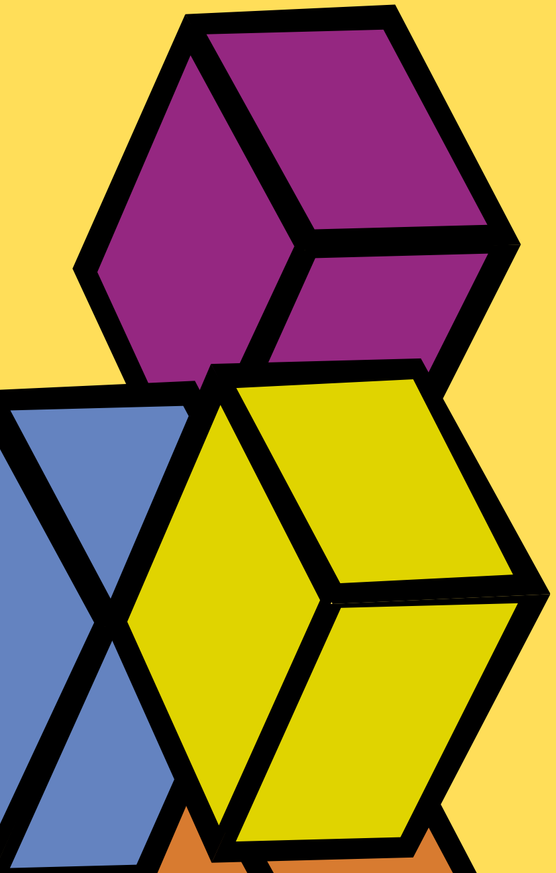
CONTROL STRUCTURES



CONTROL FLOW

THE ORDER IN WHICH A COMPUTER EXECUTES STATEMENTS IN A PROGRAM.

IF STATEMENT

A CONDITIONAL STATEMENT THAT EXECUTES A BLOCK OF CODE ONLY IF A SPECIFIED CONDITION IS TRUE.

`if` : Keyword that starts a conditional block

`else` : Optional keyword that executes when the if condition is false

`else if` : Used for multiple conditional checks

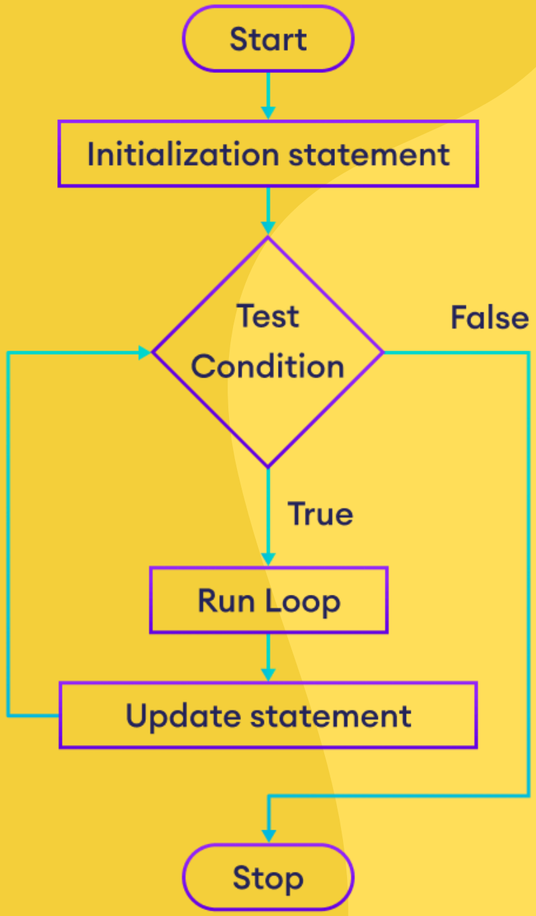
`Condition` : Boolean expression that evaluates to true or false

IF STATEMENT EXAMPLE:

```go
// 1. if statement
age := 18
if age >= 18 {
    fmt.Println("You are an adult")
} else {
    fmt.Println("You are a minor")
}
```

FOR LOOP

A PROGRAMMING CONSTRUCT THAT REPEATS A BLOCK OF CODE UNTIL A CONDITION IS MET.



```go
// 2. for loop (the only loop type in Go)
// Basic for loop
for i := 0; i < 5; i++ {
    fmt.Println(i)
}
```

```go
// While-style for loop
count := 0
for count < 3 {
    fmt.Println("Count is:", count)
    count++
}
```

```go
// Infinite loop with break
for {
    fmt.Println("This will run once")
    break
}
```

SWITCH STATEMENT

A PROGRAMMING CONSTRUCT THAT ALLOWS YOU TO EXECUTE DIFFERENT STATEMENTS BASED ON THE VALUE OF AN EXPRESSION

SWITCH STATEMENT

```go
// 3. switch statement
day := "Monday"
switch day {
case "Monday":
    fmt.Println("Start of work week")
case "Friday":
    fmt.Println("TGIF!")
default:
    fmt.Println("Regular day")
}
```

SELECT STATEMENT

Select statement is specifically for working with channels.

It's commonly used in concurrent programming

SELECT STATEMENT

```go
// 4. select statement (used for channel operations)
ch1 := make(chan string)
ch2 := make(chan string)

go func() {
    ch1 <- "Hello from channel 1"
}()

select {
case msg1 := <-ch1:
    fmt.Println(msg1)
case msg2 := <-ch2:
    fmt.Println(msg2)
case <-time.After(time.Second):
    fmt.Println("Timeout after 1 second")
}
```

THANK YOU