Go Break-Continue

Explaination

> This Go program demonstrates the use of for loops with break and continue statements. Below are the key points explained:
>
> First Loop:
>
> for i := 0; i < 10; i++ {}: This loop runs from i = 0 to i = 9. if i == 5 { break }: When i equals 5, the break statement terminates the loop. fmt.Println("i ke : ", i): Prints the current value of i for each iteration until i equals 5. Second Loop:
>
> for idx := 0; idx < 10; idx++ {}: This loop runs from idx = 0 to idx = 9. if idx%2 == 0 { continue }: If idx is even, the continue statement skips the rest of the loop body and proceeds to the next iteration. fmt.Println("idx ke : ", idx): Prints the current value of idx for each iteration where idx is odd. Final Print Statement:
>
> fmt.Println("Selesai"): Prints "Selesai" after both loops have completed their execution. This program provides a clear example of how to use break and continue within for loops in Go, demonstrating how to control the flow of loop execution based on specific conditions.

```go
package main

import "fmt"

func main() {
    // Loop from 0 to 9
    for i := 0; i < 10; i++ {
        // If i equals 5, break out of the loop
        if i == 5 {
            break
        }
        // Print the current value of i
        fmt.Println("i ke : ", i)
    }

    // Loop from 0 to 9
    for idx := 0; idx < 10; idx++ {
        // If idx is even, skip the rest of the loop and continue with the
next iteration
        if idx%2 == 0 {
            continue
        }
        // Print the current value of idx
        fmt.Println("idx ke : ", idx)
    }
    // Print "Selesai" after both loops are done
    fmt.Println("Selesai")
}
```