# Solution Review: Sum of Squares

Let's go over the solution of the Sum of Squares problem using `select` statements.

```
package main
import "fmt"

func SumOfSquares(c, quit chan int) {
  y := 1
  for {
    select {
    case c <- (y*y):
      y++
    case <-quit:
      return
    }
  }
}

func main() {
  mychannel := make(chan int)
  quitchannel:= make(chan int)
  sum:= 0
  go func() {
    for i := 1; i <= 5; i++ {
      sum += <-mychannel
    }
    fmt.Println(sum)
    quitchannel <- 0
  }()
  SumOfSquares(mychannel, quitchannel)
}
```

Sum Of Squares

Let's go over the changes we made to the `SumofSquares` function.

```
func SumOfSquares(c, quit chan int) {
  y := 1
  for {
    select {
    case c <- (y*y):
      y++
    case <-quit:
      return

    }
  }
}
```

First of all, we declare a variable `y` and then jump to the `For-Select` Loop. We have two cases in our select statements:

1. `case c <- (y*y):` This is to send the square of `y` over the channel `c` which is being received in the goroutine created in the main routine.

2. `case <-quit:` This is to receive a message from the main routine which, when received, will return from the function.

This wasn't that hard, right? I accept that there can be numerous other approaches to solve the `SumOfSquares` function but I wanted you to practice with the `select` statement. I hope you had fun and let's meet again in the next lesson!