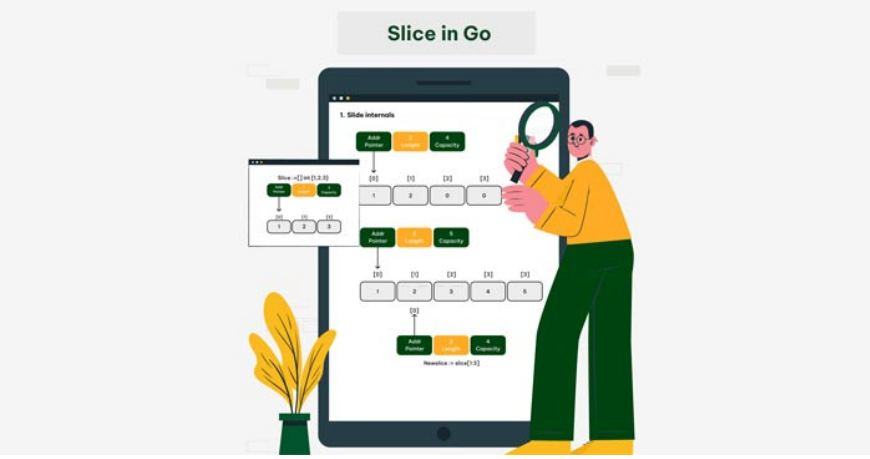
How to copy one slice into another slice in Golang

To copy one slice into another, Go provides a built-in function called copy(). This function allows you to copy elements from one slice (source slice) into another slice (destination slice).

In Golang , a Slice is a variable-length sequence that can contain elements of the same type. You cannot mix different types of elements in a slice. **To copy one slice into another, Go provides a built-in function called copy(). This function allows you to copy elements from one slice (source slice) into another slice (destination slice).**



For example:

```
package main import "fmt" // Slice c? b?n ???c dùng trong m?i ví d?
 var source = []int{10, 20, 30, 40, 50} func main() { fmt.Println("Source Slice:"
```

Syntax to pass one slice into another slice in Golang

```
func copy(dst, src []Type) int
```

1. **dst:** Destination slice where elements will be copied.
2. **src:** The source slice where the elements will be copied.

The function returns the number of elements copied, which will be the smallest of the lengths of the source and destination slices.

Using copy() function in Golang

Here is how to copy the source slice to the destination slice using the copy() function:

```
package main import "fmt" // Slice c? b?n ???c dùng trong m?i ví d?
 var source = []int{10, 20, 30, 40, 50} func main() { // T?o slice ?ích có
?? dài gi?ng nh? slice ngu?
n destination := make([]int, len(source)) // Sao chép các thành ph?n t? ngu
?n t?i ?
ích count := copy(destination, source) // In slice fmt.Println("Source:", source
```

Result:

```
Source: [10 20 30 40 50] Destination: [10 20 30 40 50] Elements copied: 5
```

Detailed explanation

1. **Creating Slice:** We create a destination slice named destination using **make()** which can contain the same number of elements as the source slice.
2. **Copying elements:** We use the **copy()** function to copy elements from the source slice to the destination slice. The **copy()** function returns the number of elements copied.
3. **Display the result:** Finally, we print both the source and destination slices, along with the number of elements copied.

Manual copy using loop

You can manually copy slices using a loop.

Syntax:

```
for i := 0; i  len(source); i++ { destination[i] = source[i]}
```

For example:

```
package main import "fmt" // Slice c? b?n cho m?i ví d?
 var source = []int{10, 20, 30, 40, 50} func main() { destination := make([]int,
? tay sao chép t?ng ph?n t?
 for i := 0; i  len(source); i++ { destination[i] = source[i] } fmt.Println("Sou
```

Result:

```
Source: [10 20 30 40 50] Destination: [10 20 30 40 50]
```

Use Literal Slice

If you want to make a copy of a slice while initializing it, you can use a **slice literal** with the **append()** function .

Syntax

```
destination = append(destination, source.)
```

For example:

```
package main import "fmt" // Slice c? b?n ???c dùng trong m?i ví d?
 var source = []int{10, 20, 30, 40, 50} func main() { // Sao chép b?ng m?
t slice literal destination := []int{} destination = append(destination, source.
```

Result:

```
Source: [10 20 30 40 50] Destination: [10 20 30 40 50]
```

Note: Make sure that the destination slice is equal to or greater than the source slice when using **copy()** ; otherwise, the function will only copy up to the length of the destination slice.