## QUIZ

Which function returns the number of elements in an array or slice?

`sizeof()`

`len()`

Correct!

`length()`

`elements()`

Fill in the blank to create an empty slice:

```
var numberSlice []int
```

You got it!

Fill in the paragraph below:

In Go, an array is a **fixed size** ordered list of elements with the same data type. Arrays are useful for collecting and accessing **multiple related values** .

You got it!

Fill in the code block below to create an empty array:

```
var my_array [3] int
```

You got it!

Fill in the blanks to match the function names with the description of their purpose:

- `len` : retrieves the current number of elements in a slice.
- `append` : adds an element to the end of a slice.
- `cap` : retrieves the number of elements can hold before it must be resized.

You got it!

Which of the following describes the necessary components to declare an array with specified values in Go?

The number of elements and the data type.

Only the number of elements and a list of elements are required.

Only a list of elements is required

The number of elements, type, and list of values

Correct!

Fill in the blanks in the paragraph below:

In Go, both arrays and slices are collections of `multiple elements of the same data type` . However, a slice can be `resized to hold additional elements` , whereas an array cannot.

You got it!

Fill in the code to change the first element of the array to `3` and then print it to the console.

```
array := [3]int{1,2,3}
array [0] = 3
fmt.Println( array[0] )
```

You got it!

Fill in the blanks:

- A slice's **length** is the current number of elements it holds. This is retrieved using the **len** function.
- A slice's **capacity** is the number of elements it can hold before needing to resize itself. This is retrieved using the **cap** function.

You got it!

Fill in the syntax to create a function which takes an array as a parameter:

```
func printFirstLastArray( array [4]int ) {
    fmt.Println("First", array[0])
    fmt.Println("Last", array[3])
}
```

You got it!

Fill in the blanks in the paragraph below:

In Go, arrays are passed into functions **by value**, which means that changes remain local to the function. Slices, which are pointers, are passed **by reference** so changes affect the original variable. To modify an array within a function, it must be converted into a slice.

You got it!