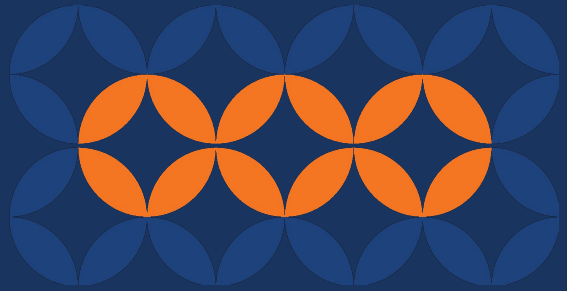
alterra
academy



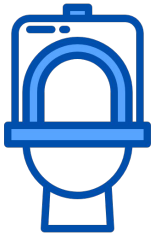
Structure Data (Array, Slice & Map)

alterra academy

OUR RULES

Silent Mode

Ask Question

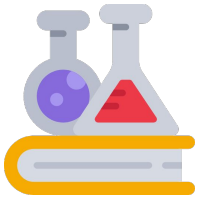


Go Toilet

TIME ALLOCATION

Explanation

90 minute



Challenge

60 minute

Review

30 minute

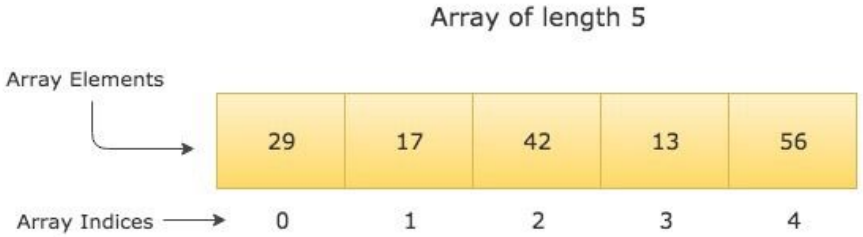
alterra
academy

Array

alterra academy

WHAT IS ARRAY?

Array is a data structure that contains a group of elements, can contain one type of variable with fixed allocation size. Different data types can be handled as elements in arrays such as Numeric, String, Boolean.



alterra academy

ARRAY DECLARATION

To declare an array you need to specify the number of elements it holds in square brackets [], followed by the type of elements the array holds.

```
var <variable_name> [<size_of_array>]<tipe_variable>
```

example

```go
import (
  "fmt"
  "reflect"
)

func main() {
  var primes [5]int
  var countries [5]string

  fmt.Println(reflect.ValueOf(primes).Kind())
  fmt.Println(reflect.ValueOf(countries).Kind())
}
```

output

```
array
array
```

alterra academy

ASSIGN AND ACCESS ARRAY ELEMENT

example

```go
package main

import "fmt"

func main() {
  var countries [2]string

  countries[0] = "India"  // Assign a value to the first element
  countries[1] = "Canada" // Assign a value to the second element

  fmt.Println(countries[0]) // Access the first element value
  fmt.Println(countries[1]) // Access the second element value
}
```

alterra academy

INITIALIZE WITH ARRAY LITERAL

example

```go
package main

import "fmt"

func main() {
  odd_numbers := [5]int{1, 3, 5, 7, 9}   // Intialized with values
  var even_numbers [5]int = [5]int{0, 2, 4} // Partial assignment

  fmt.Println(odd_numbers)
  fmt.Println(even_numbers)
}
```

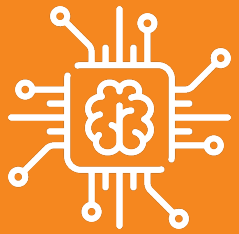
output

```
[1 3 5 7 9]
[0 2 4 0 0]
```

alterra academy

INFER THE LENGTH OF ARRAY

Compiler count the number of elements for you.



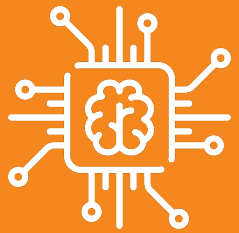
example

```go
package main

import (
  "fmt"
  "reflect"
)

func main() {
  primes := [...]int{2, 3, 3}

  fmt.Println(reflect.ValueOf(primes).Kind())
  fmt.Println(len(primes))
}
```

output

```
array
3
```

Initialize values for specific index



alterra academy

example

```
package main

import "fmt"

func main() {
  even_numbers := [5]int{1: 2, 2: 4}
  fmt.Println(even_numbers)
}
```

output

```
[0 2 4 0 0]
```

ITERATE ARRAY USING FOR LOOP

alterra academy

example

```go
primes := [5]int{2, 3, 5}

// technique 1
for index := 0; index < len(primes); index++ {
  fmt.Println(primes[index])
}
// technique 2
for index, element := range primes {
  fmt.Println(index, "=>", element)
}
for _, value := range primes {
  fmt.Println(value)
}
// technique 3
index := 0
for range primes {
  fmt.Println(primes[index])
  index++
}
```

alterra
academy

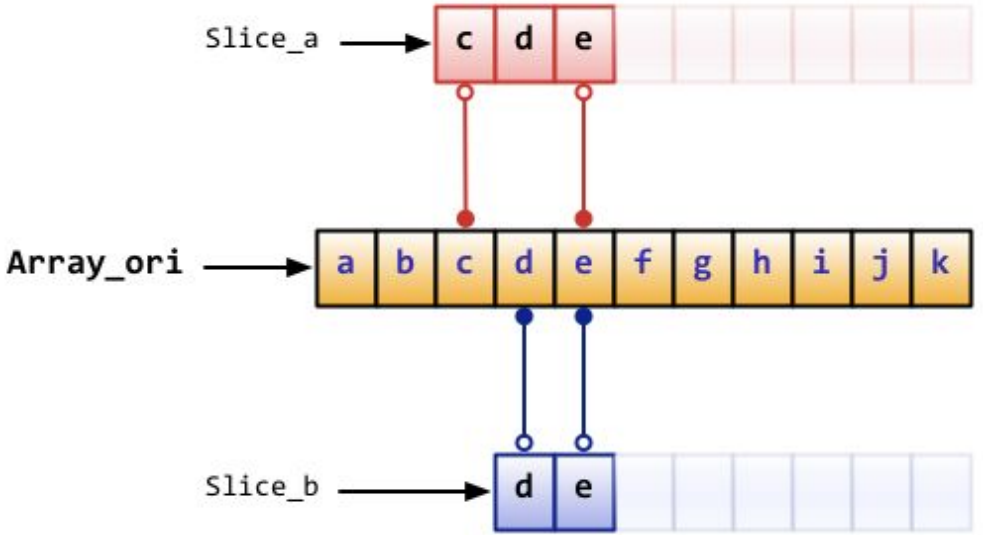
Slice

alterra
academy

WHAT IS SLICE?

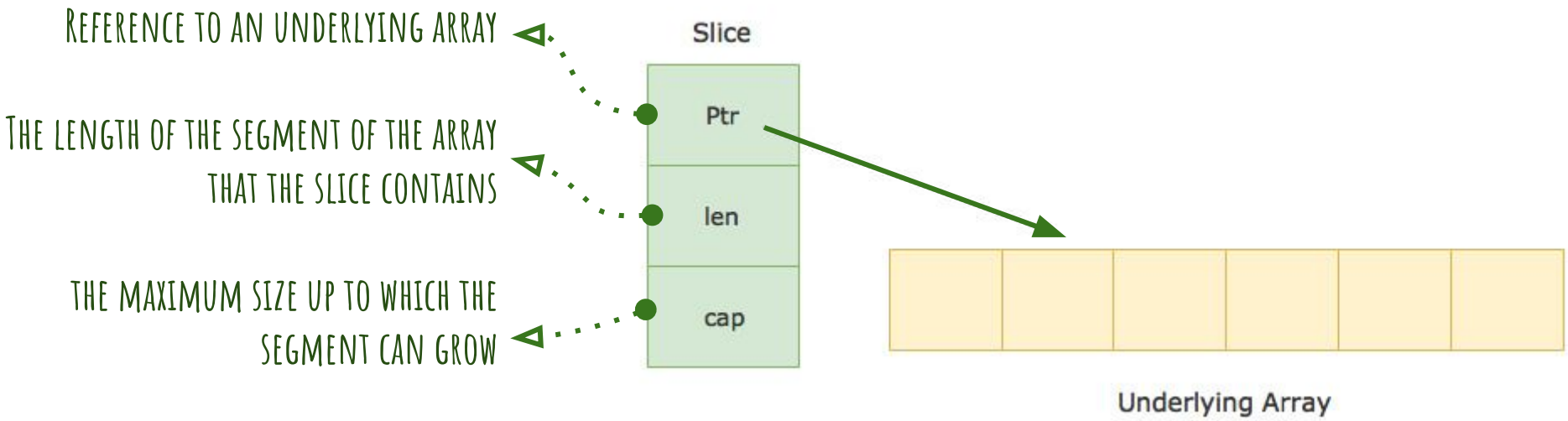
Slice is a data structure that contains a group of elements, can contain one type of variable (like Array) but have dynamic allocation size.

CREATE SLICE FROM ARRAY



Slice is not really a dynamic array, It's a reference type.

LENGTH AND CAPACITY OF A SLICE



Create Slice from Array

alterra academy

```go
package main

import (
  "fmt"
  "reflect"
)

func main() {
  // Obtaining a slice from an array `array`
  // array[low:high]
  var primes = [5]int{2, 3, 5, 7, 11}

  // Creating a slice from the array
  var part_primes []int = primes[1:4]

  // part_primes = append(part_primes, 10000)
  // menambah data ke slice akan menambah data ke
array juga

  fmt.Println(reflect.ValueOf(part_primes).Kind())
  fmt.Println(part_primes)
}
```

Optional

output

```
slice
[3 5 7]
```

alterra academy

IS THERE ANY OTHER WAY?

THE SLICE IS DECLARED JUST LIKE AN ARRAY EXCEPT THAT WE DO NOT SPECIFY ANY SIZE IN THE BRACKETS [].

SLICE DECLARATION

alterra academy

```go
package main

import "fmt"

func main() {
  // long declaration
  var even_numbers []int
  fmt.Printf("elements = %v, len = %d, cap = %d\n", even_numbers, len(even_numbers), cap(even_numbers))

  // long declaration with values
  var odd_numbers = []int{1, 3, 5, 7, 9}
  fmt.Printf("elements = %v, len = %d, cap = %d\n", odd_numbers, len(odd_numbers), cap(odd_numbers))

  // short declaration with values
  numbers := []int{1, 2, 3, 4, 5}
  fmt.Printf("elements = %v, len = %d, cap = %d\n", numbers, len(numbers), cap(numbers))

  // using make function
  var primes = make([]int, 5, 10)
  fmt.Printf("elements = %v, len = %d, cap = %d\n", primes, len(primes), cap(primes))
}
```

alterra academy

USING KEYWORD MAKE()

```
func make([]T, len, cap) []T
```

MAKE() ALLOWS US TO
CREATE A SLICE
WHEN THE UNDERLYING
ARRAY IS NOT DEFINED

alterra academy

Different Type of Slice

Declaration	Reference to defined Array	Dynamic Len	Dynamic Cap	Have Limit
`var numbers []string = a[1:4]`	Y	Y	Y	N
`var numbers []int`	N	Y	Y	N
`var s = make([]int, 5, 10)`	N	Y	Y	N

Append() and Copy()

append(): One can increase the capacity of a slice

copy(): the contents of a source slice are copied to a destination slice

example

```
package main

import "fmt"

func main() {
  var colors = []string{"red", "green", "yellow"}
  colors = append(colors, "purple")

  copied_colors := make([]string, 10)

  copy(copied_colors, colors) // copy from colors to
copied_colors
  fmt.Println(copied_colors)
}
```

SLICE ZERO VALUE

THE ZERO VALUE OF A SLICE IS NIL

alterra academy

example

```go
package main

import "fmt"

func main() {
  var primes []int
  fmt.Printf("s = %v, len = %d, cap = %d\n", primes, len(primes),
cap(primes))

  if primes == nil {
    fmt.Println("s is nil")
  }
}
```

output

```
s = [], len = 0, cap = 0
primes is nil
```

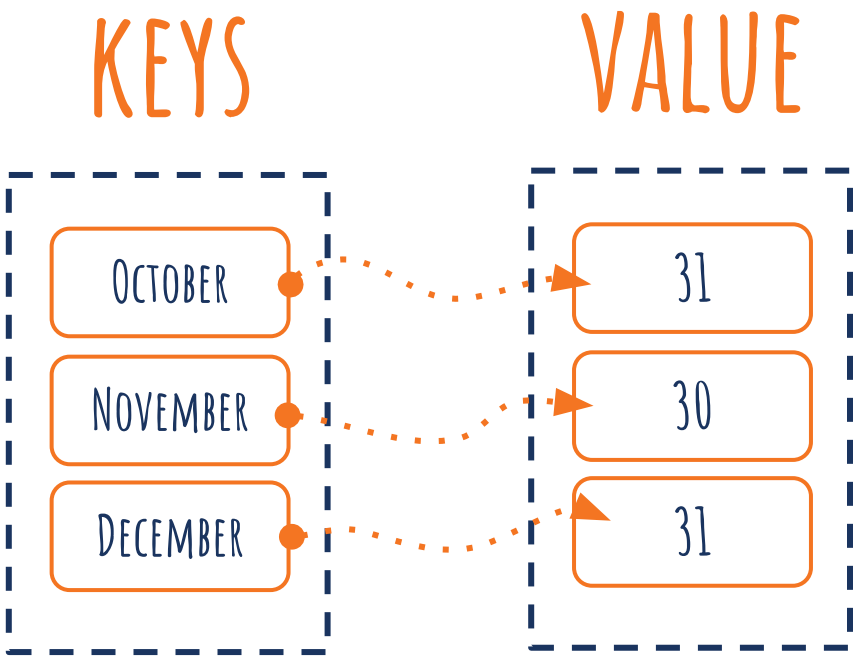
alterra
academy

Map

alterra academy

What is Map?

A map is a data structure stores data in the form of key and value pairs where every key is unique.


KEYS
VALUE
October
November
December
31
30
31

alterra academy

```go
package main

import "fmt"

func main() {
  // long declaration
  var salary = map[string]int{}
  fmt.Println(salary)

  // long declaration with value
  var salary_a = map[string]int{"umam": 1000, "iswanul": 2000}
  fmt.Println(salary_a)

  // short declaration
  salary_b := map[string]int{}
  fmt.Println(salary_b)

  // using make
  var salary_c = make(map[string]int)
  salary_c["doe"] = 7000 // assign value
  fmt.Println(salary_c)
}
```

Working with Maps

alterra academy

example

```go
package main
import "fmt"
func main() {

  // long declaration with value
  var salary_a = map[string]int{"umam": 1000, "iswanul": 2000}
  fmt.Println(salary_a, len(salary_a))

  salary_a["nabilah"] = 7000 // assign value
  fmt.Println(salary_a)

  delete(salary_a, "iswanul") // remove value by key
  fmt.Println(salary_a)

  value, exist := salary_a["umam"] // check key is exist
  fmt.Println(value, exist)

  for key, value := range salary_a { // loop over maps
    fmt.Println("->", key, value)
  }
}
```

alterra
academy

function

WHAT IS FUNCTION?

A function is a piece of code that is called by name. Functions are a convenient way to divide your code into useful blocks. Enable to us to write clean, tidy and modular code.

FUNCTION DECLARATION

func <name_function> () { <statements>}

```
func main() {}
```

func <name_function> () <type_return> { <statements> }

```
func phi() float64 { return 3.14 }
```

func <name_function> (<parameter>) { <statements> }

```
func square(value int) int {
    return value * value
}
```

EXAMPLE Function Without and With Parameter

-code editor

```go
package main

import "fmt"

func sayHello() {
  fmt.Println("Hello")
}

func greeting(hour int) {
  if hour < 12 {
    fmt.Println("Selamat Pagi")
  } else if hour < 18 {
    fmt.Println("Selamat Sore")
  } else {
    fmt.Println("Selamat Malam")
  }
}

func main() {
  hour := 15
  greeting(hour)
}
```

Without parameter

With parameter

Function with Return Value (SINGLE AND MULTIPLE)

alterra academy

```go
package main

import (
  "fmt"
  "math"
)

// singe return value
func calculateSquare(side int) int {
  return side * side
}

// multiple return value
func calculateCircle(diameter float64) (float64, float64) {
  var keliling = math.Pi * math.Pow(diameter/2, 2)
  var luas = math.Pi * diameter
  // return 2 value
  return keliling, luas
}

func main() {
  var side = 5
  wide := calculateSquare(side)
  fmt.Printf("Luas persegi empat: %d \n\n", wide)

  var diameter float64 = 15
  keliling, luas := calculateCircle(diameter)
  fmt.Printf("Luas lingkaran: %.2f \n", keliling)
  fmt.Printf("Keliling lingkaran: %.2f \n", luas)
}
```

FUNCTION WITH Named Return Parameter

alterra academy

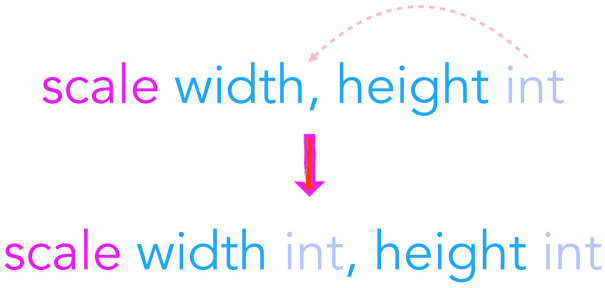
-code editor

```
package main

import "fmt"

// function having named return parameter
func multiplication(a, b int) (mul int) {
  mul = a * b
  return
}

func main() {
  m := multiplication(5, 5)
  fmt.Println("5 x 5 = ", m)
}
```

alterra academy

AUTOMATIC TYPE-ASSIGNMENT

```
func scale(width, height, scale int) (int, int)
func scale(width int, height int, scale int) (int, int)
```



alterra academy

“First, solve the problem.
Then, write the code.”

- John Johnson -