Arrays and Slices

Nigel Poulton

@nigelpoulton | www.nigelpoulton.com

Arrays and Slices

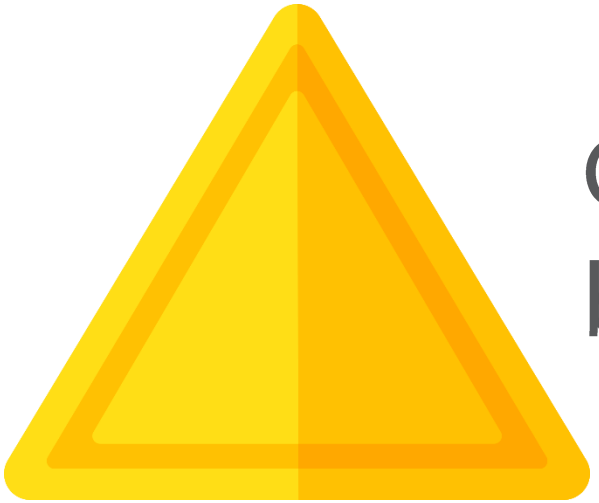
Theory

Syntax

Examples

Manipulation

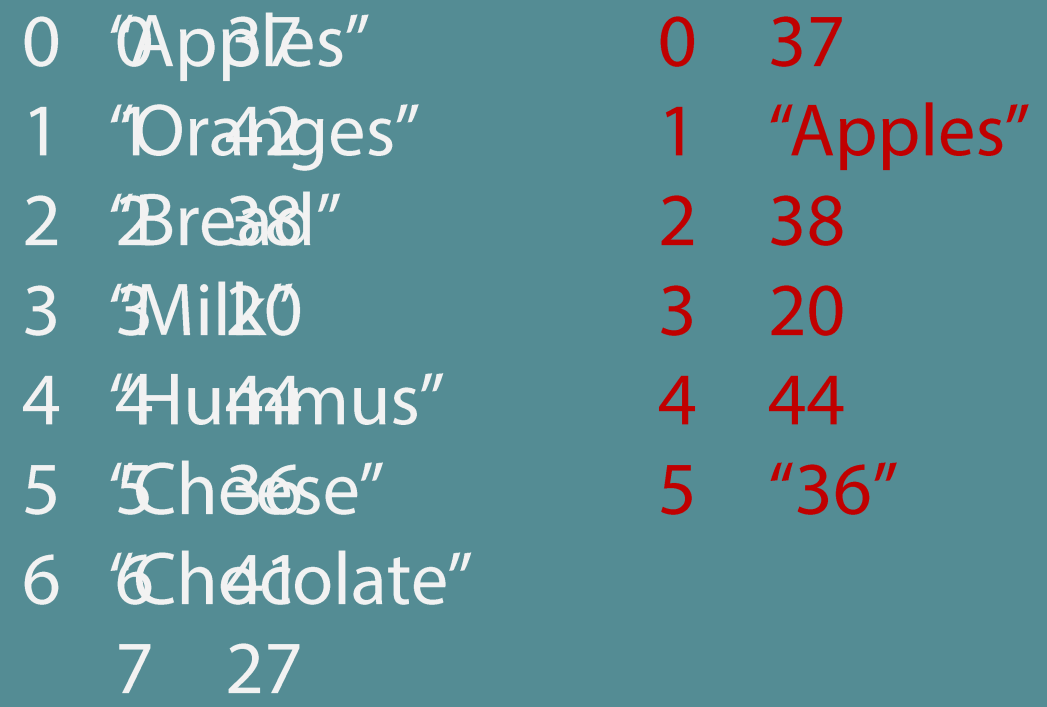
The Theory



Go's handling of Arrays and Slices may be different to what you expect

“**Numbered** lists containing elements of the **same type**”

0 “Apples”
1 “Oranges”
2 “Bread”
3 “Milk”
4 “Hummus”
5 “Cheese”
6 “Chocolate”



Numbered lists of a **single type**

Arrays vs Slices

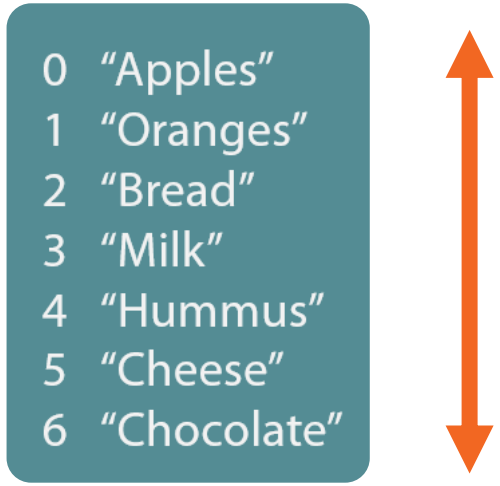
Slices win hands-down!

Arrays vs Slices

Arrays

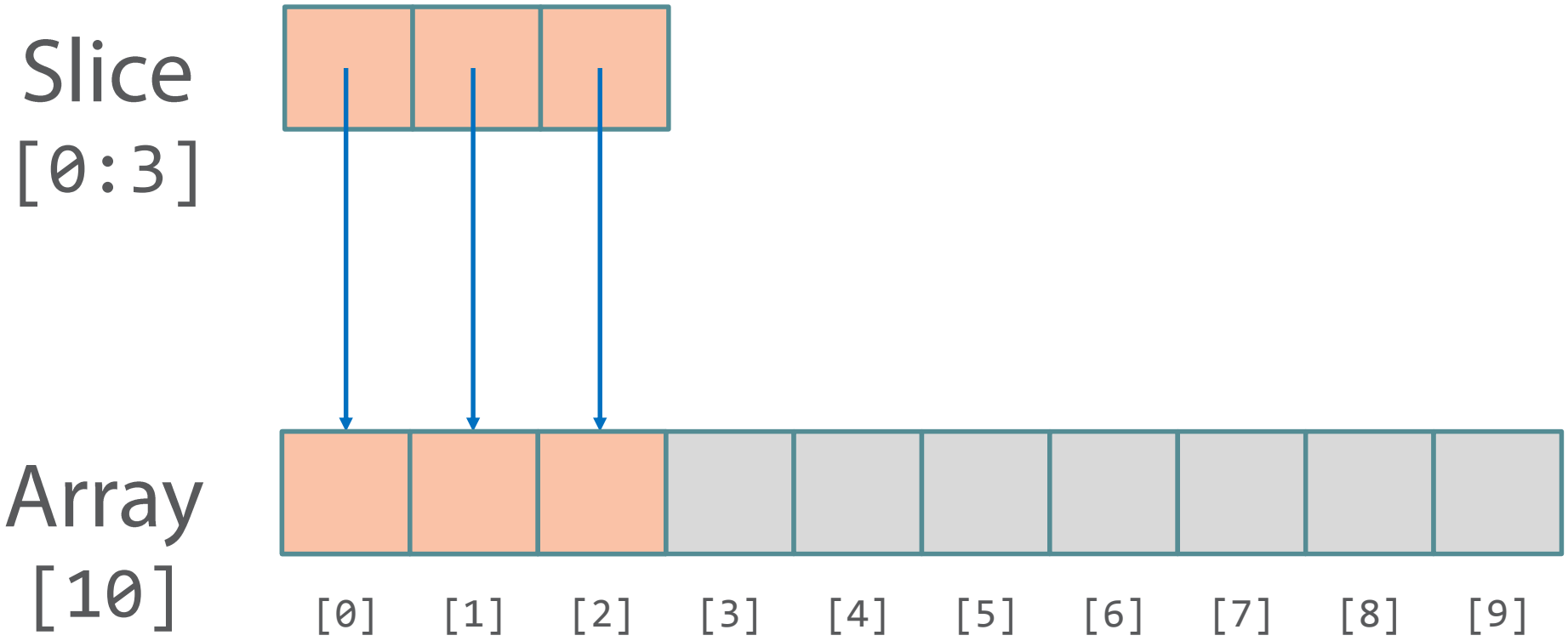
- Numbered list of a single type
- Fixed length

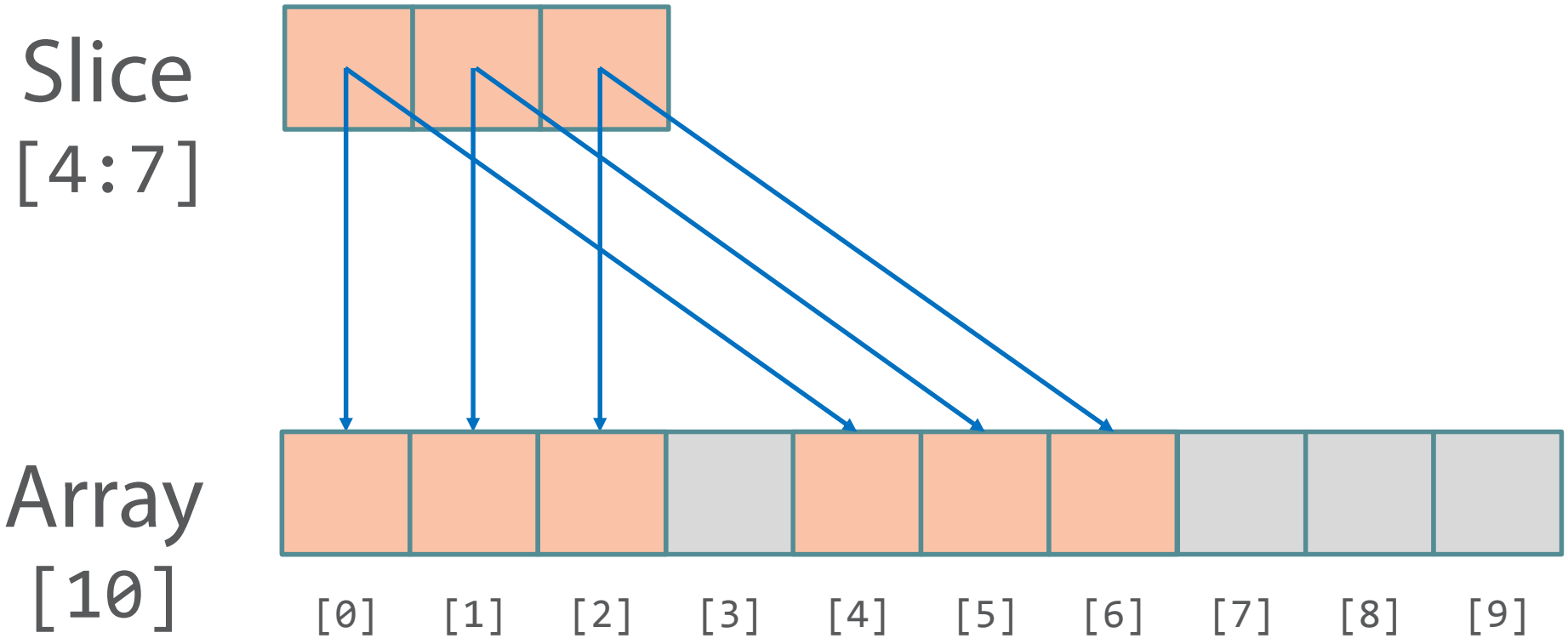
Slices

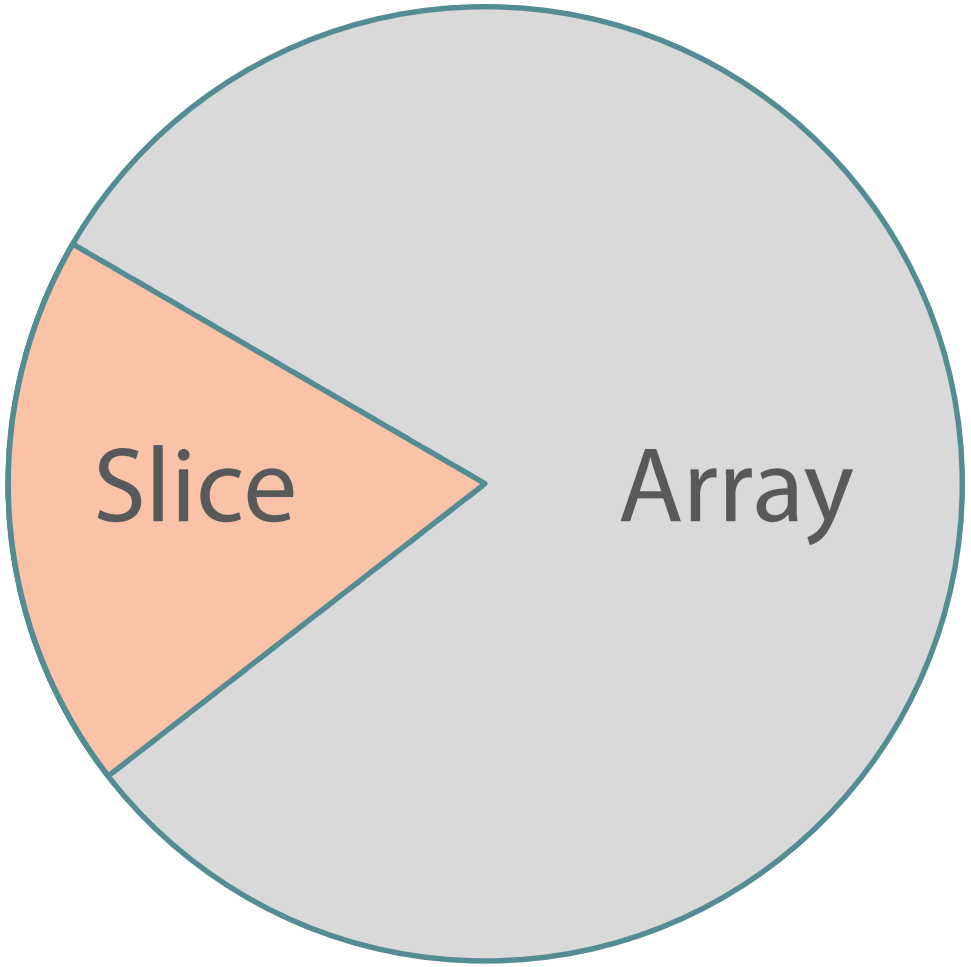


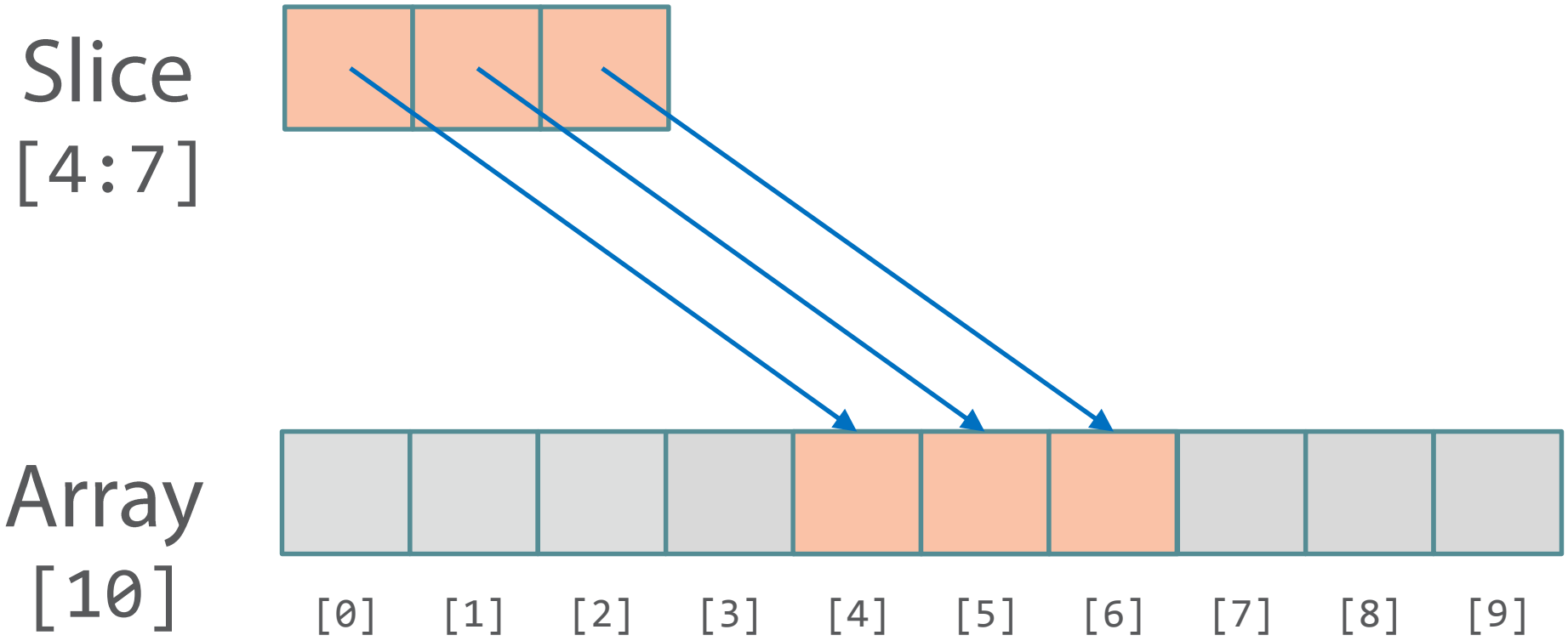
- Numbered list of a single type
- Can be resized

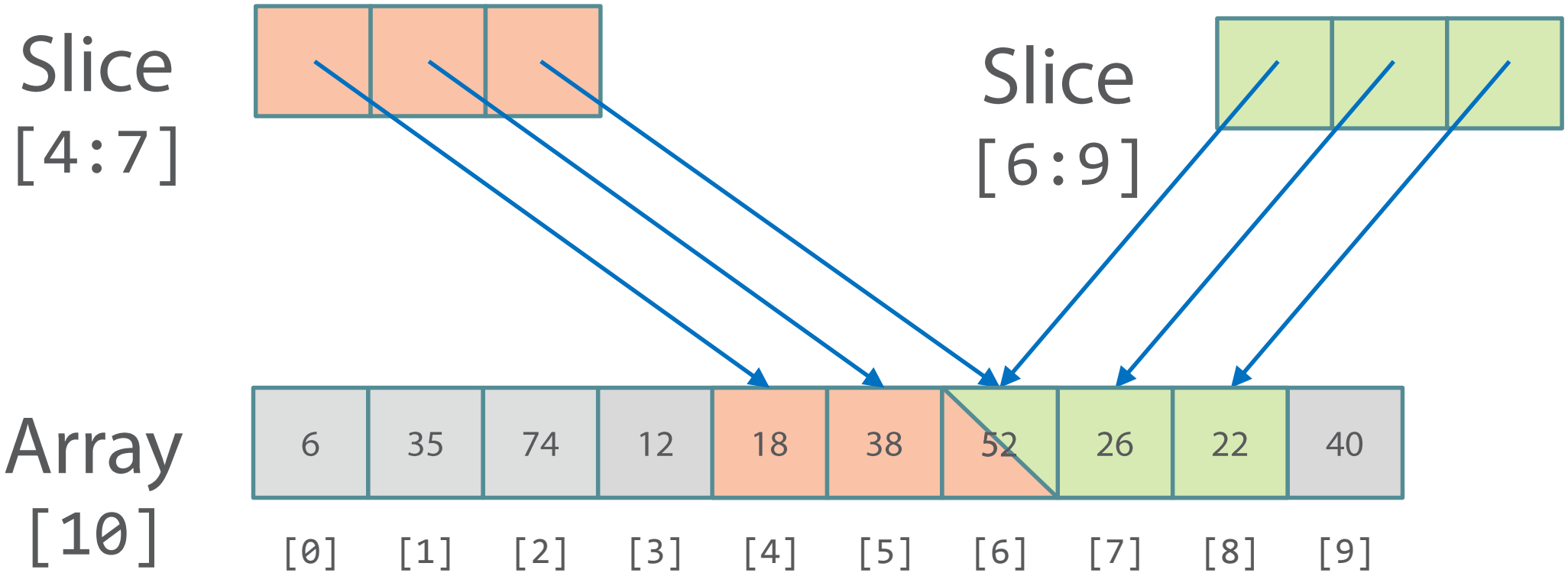
Slices are built on top of arrays











Slices are **references**

- Passed by reference

Reference contiguous portion of an array

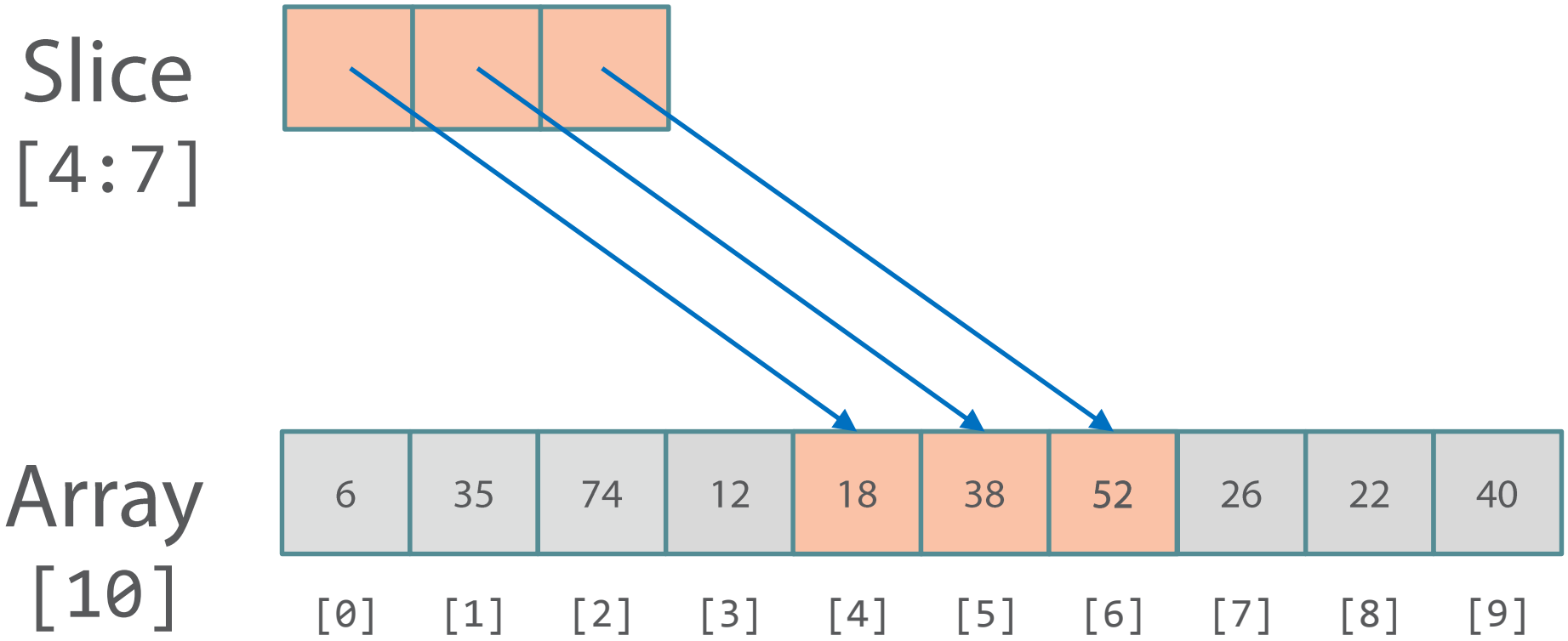
Flexible length

- Up to the length of its associated array

```
myCourses := make(<type>, <len>, <cap>)
```

```
Index:   0  1  2  3  4  5  6  7  8  9
Values:  1  2  3  4  5  6  7  8  9  10
len():   1  2  3  4  5  6  7  8  9  10
```

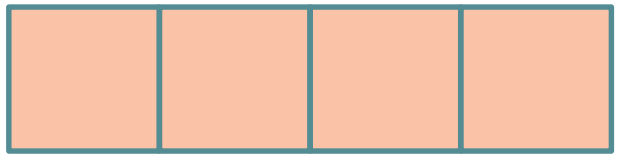
[2:]


Slice
[4:7]
Array
[10]
6
35
74
12
18
38
52
26
22
40
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]

Slice

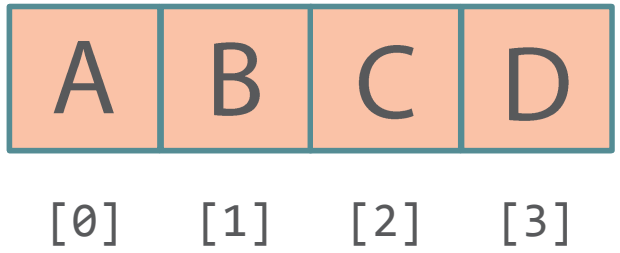
len=1

cap=4



Array

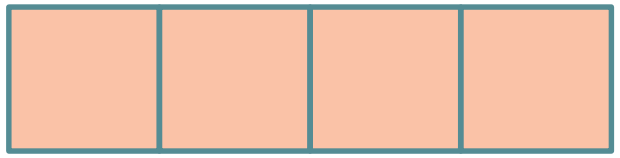
[4]



Slice

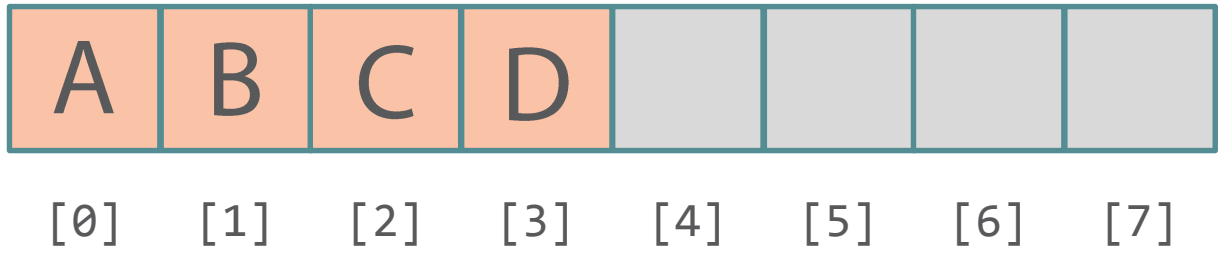
len=1

cap=4



Array

[4]



Referencing a slice by its variable name references the entire slice

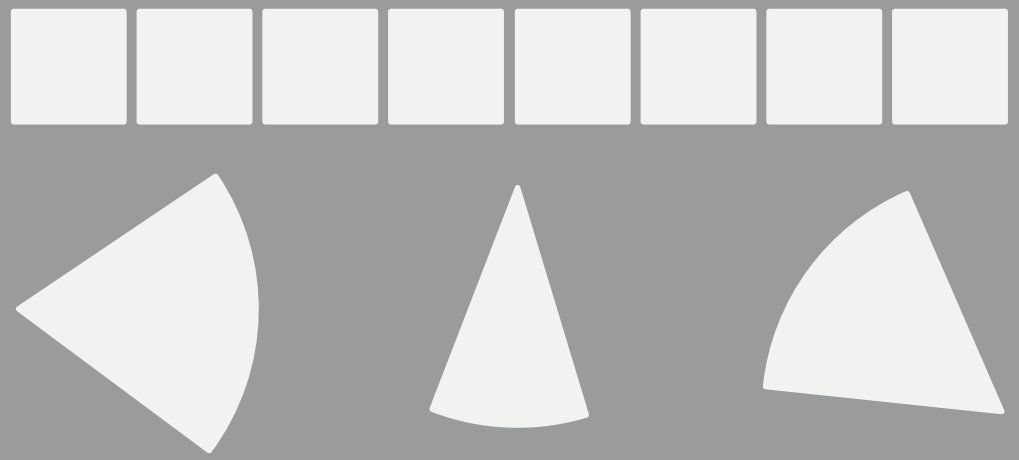
`for range` loops iterate slices

`for range` returns two values –

- index
- data

Can **`append()`** slices to slices with the ellipses

Summary



Arrays

Fixed length

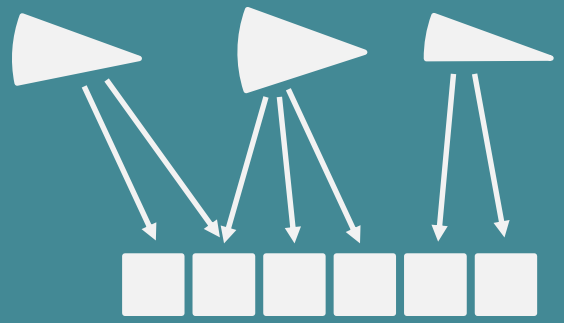
Length is part of **type**

Slices

Flexible length

Always use slices in your code

Slices are **references**



```
slice := make([]int, 5)

slice := []int{1,2,3,4,5}
```