ITERATION WITH FOR LOOPS

For loop

```
val list = List(1, 8, -3, 12)
```

```
for (number <- list) {
  println(s"number is $number")
}
// number is 1
// number is 8
// number is -3
// number is 12
```

Without s String interpolator

```
val list = List(1, 8, -3, 12)
```

```
for (number <- list) {
  println("number is $number")
}
// number is $number
// number is $number
// number is $number
// number is $number
```

Other String interpolators

In standard library

```
println(f"PI is ${Math.PI}%1.2f")
// PI is 3.14
```

In other libraries

```
val subject = "functional programming"

println(sql"subject = $subject")
// res: Fragment = Fragment("subject = ?")
```

```
val names = List("Eda", "Bob")

json"{ names : $names }"
// res: Json = "{ "names" : ["Eda", "Bob"] }"
```

Code block: curly braces

Optional

```
for (number <- list)
  println(s"number is $number")
// number is 1
// number is 8
// number is -3
// number is 12
```

Mandatory

```
for (number <- list) {
  val square = number * number
  println(s"number square is $square")
}
// number square is 1
// number square is 64
// number square is 9
// number square is 144
```

Code block: curly braces

Optional

```
for (number <- list)
  println(s"number is $number")
// number is 1
// number is 8
// number is -3
// number is 12
```

Optional in Scala 3

```
for (number <- list)
  val square = number * number
  println(s"number square is $square")
// number square is 1
// number square is 64
// number square is 9
// number square is 144
```

For loop is a syntax for foreach

```
for (number <- list)
  println(s"number is $number")
// number is 1
// number is 8
// number is -3
// number is 12
```

```
list.foreach(number =>
  println(s"number is $number")
)
// number is 1
// number is 8
// number is -3
// number is 12
```

For loop is a syntax for foreach

```
for (number <- list)
  println(s"number is $number")
// number is 1
// number is 8
// number is -3
// number is 12
```

```
list.foreach(number =>
  println(s"number is $number")
)
// number is 1
// number is 8
// number is -3
// number is 12
```

```
trait List[A] {
  def foreach[To](action: A => To): Unit
}
```

For loop is a syntax for foreach

```
for (number <- list)
  println(s"number is $number")
// number is 1
// number is 8
// number is -3
// number is 12
```

```
list.foreach(number =>
  println(s"number is $number")
)
// number is 1
// number is 8
// number is -3
// number is 12
```

```
trait List[A] {
  def foreach[To](action: A => To): Unit
  def map    [To](update: A => To): List[To]
}
```

For loop is a syntax for foreach

```
for (number <- Point(1, 5))
  println(s"number is $number")
// number is 1
// number is 5
```

```
Point(1, 5).foreach(number =>
  println(s"number is $number")
)
// number is 1
// number is 5
```

```
case class Point(first: Int, second: Int) {
  def foreach[To](action: Int => To): Unit = {
    action(first)
    action(second)
  }
}
```

Nested for loop

```
val shapes  = List("triangle", "square")
val colours = List("red", "green", "blue")
```

```
for {
  shape  <- shapes
  colour <- colours
} println(s"The $shape is $colour")
// The triangle is red
// The triangle is green
// The triangle is blue
// The square is red
// The square is green
// The square is blue
```

```
shapes.foreach{ shape =>
  colours.foreach { colour =>
    println(s"The $shape is $colour")
  }
}
// The triangle is red
// The triangle is green
// The triangle is blue
// The square is red
// The square is green
// The square is blue
```

Nested for loop with different data types

```
val shapes = List("triangle", "square")
val point  = Point(2, 5)
```

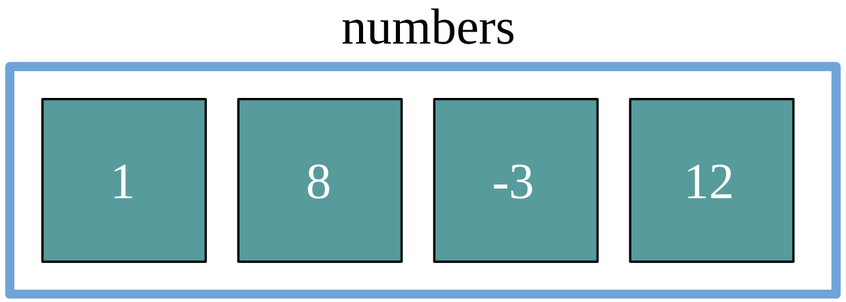
```
for {
  shape  <- shapes
  number <- point
} println(s"There is $number ${shape}(s)")
// There is 2 triangle(s)
// There is 5 triangle(s)
// There is 2 square(s)
// There is 5 square(s)
```

```
shapes.foreach{ shape =>
  point.foreach { number =>
    println(s"There is $number ${shape}(s)")
  }
}
// There is 2 triangle(s)
// There is 5 triangle(s)
// There is 2 square(s)
// There is 5 square(s)
```

Use case

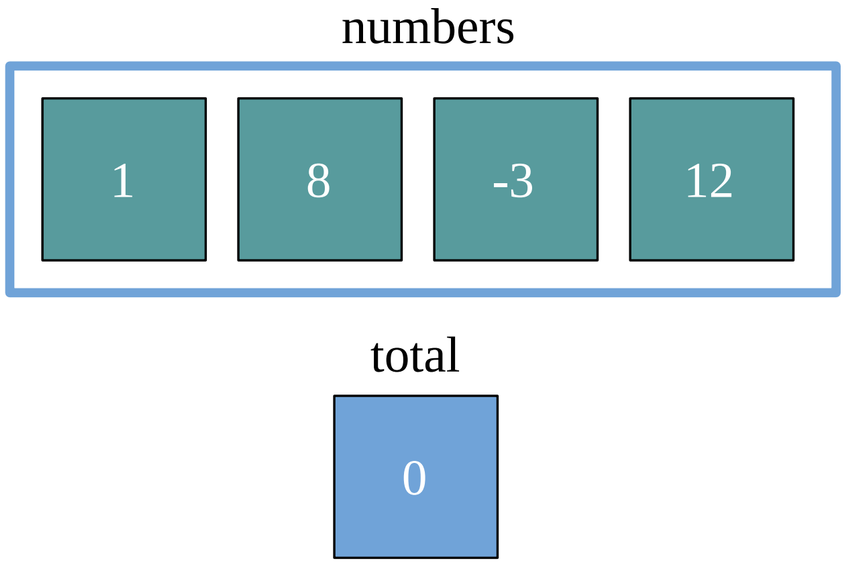
```
def sum(numbers: List[Int]): Int
```

```
sum(List(1, 8, -3, 12))
// res: Int = 18
```



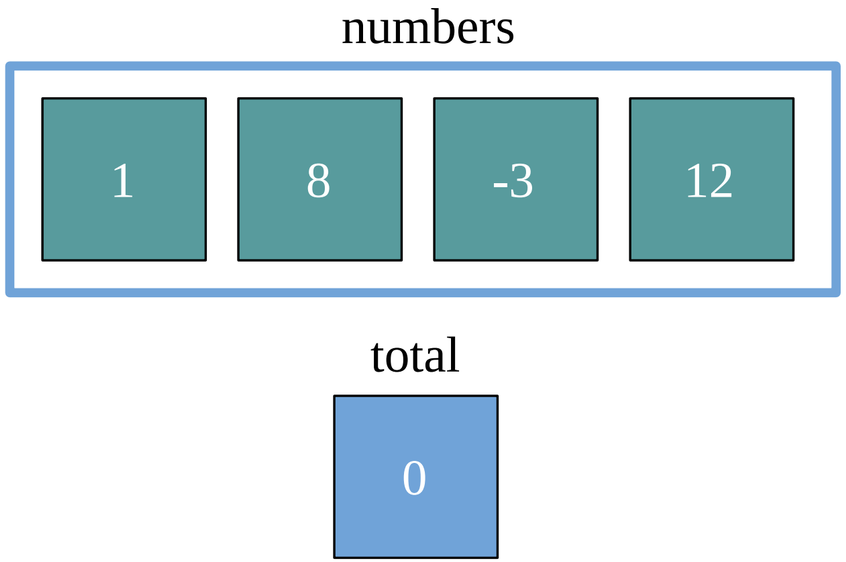
For loop with mutable state

```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total = total + number

  total
}
```



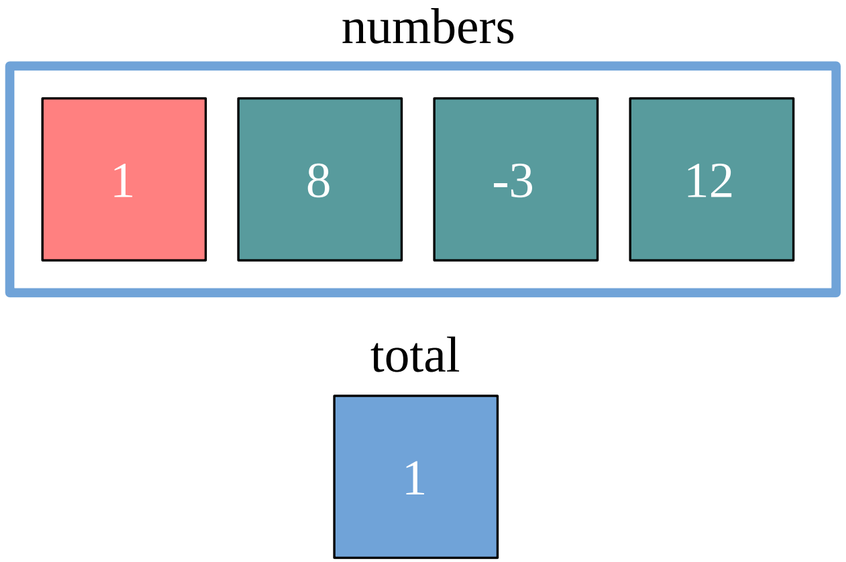
For loop with mutable state

```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total += number

  total
}
```



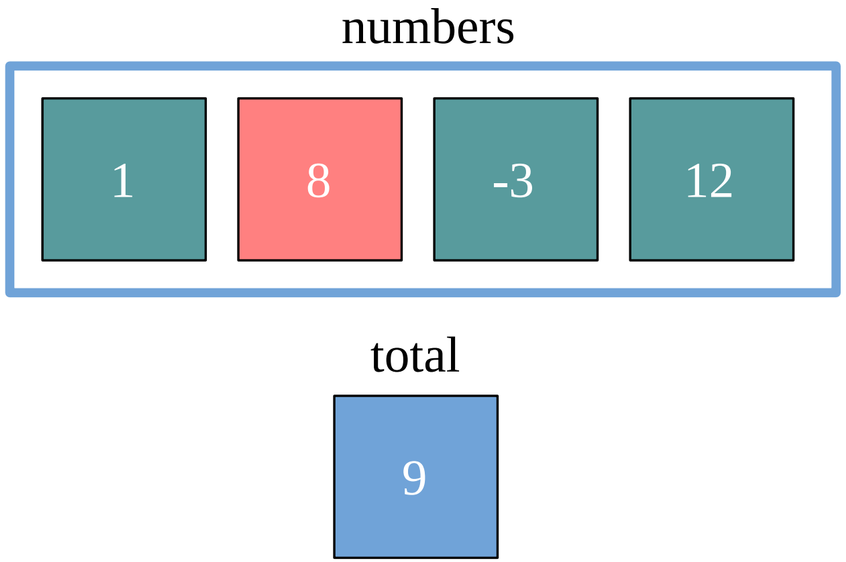
For loop with mutable state

```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total += number

  total
}
```



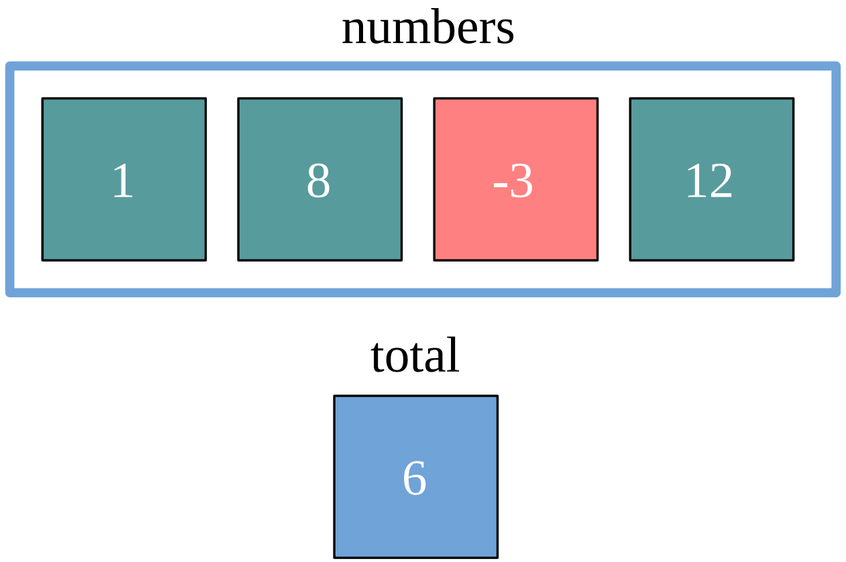
For loop with mutable state

```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total += number

  total
}
```



For loop with mutable state

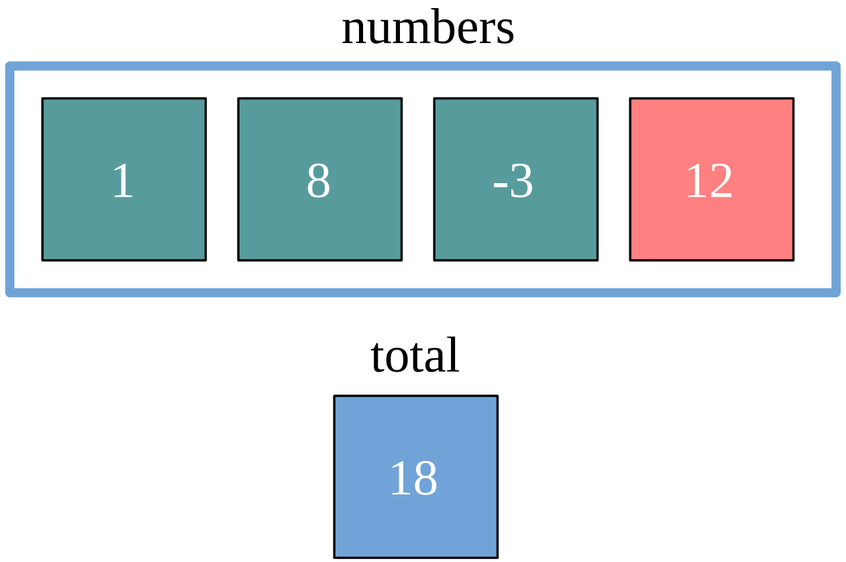
```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total += number

  total
}
```



For loop with mutable state

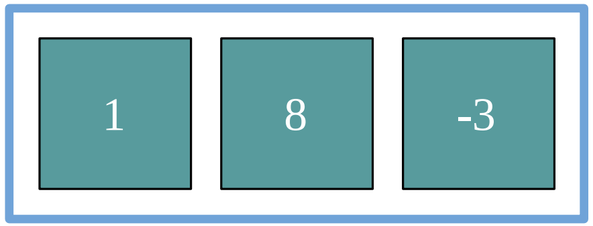
```
def sum(numbers: List[Int]): Int = {
  var total = 0

  for (number <- numbers)
    total += number

  total
}
```

```
sum(List(1, 8, -3, 12))
// res20: Int = 18
```



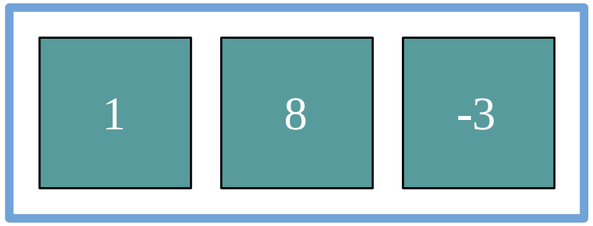
ForLoopExercises.scala

sum

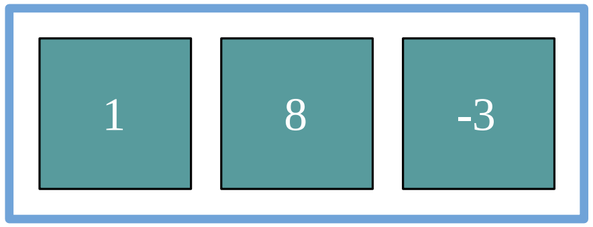


wordCount

min

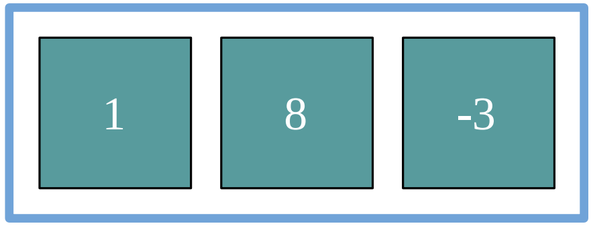


sum

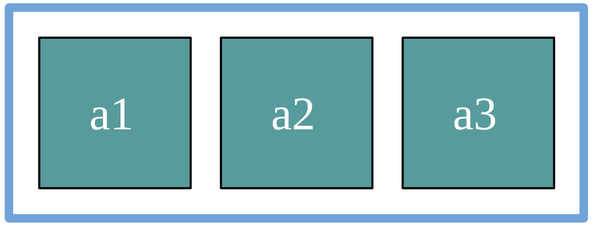


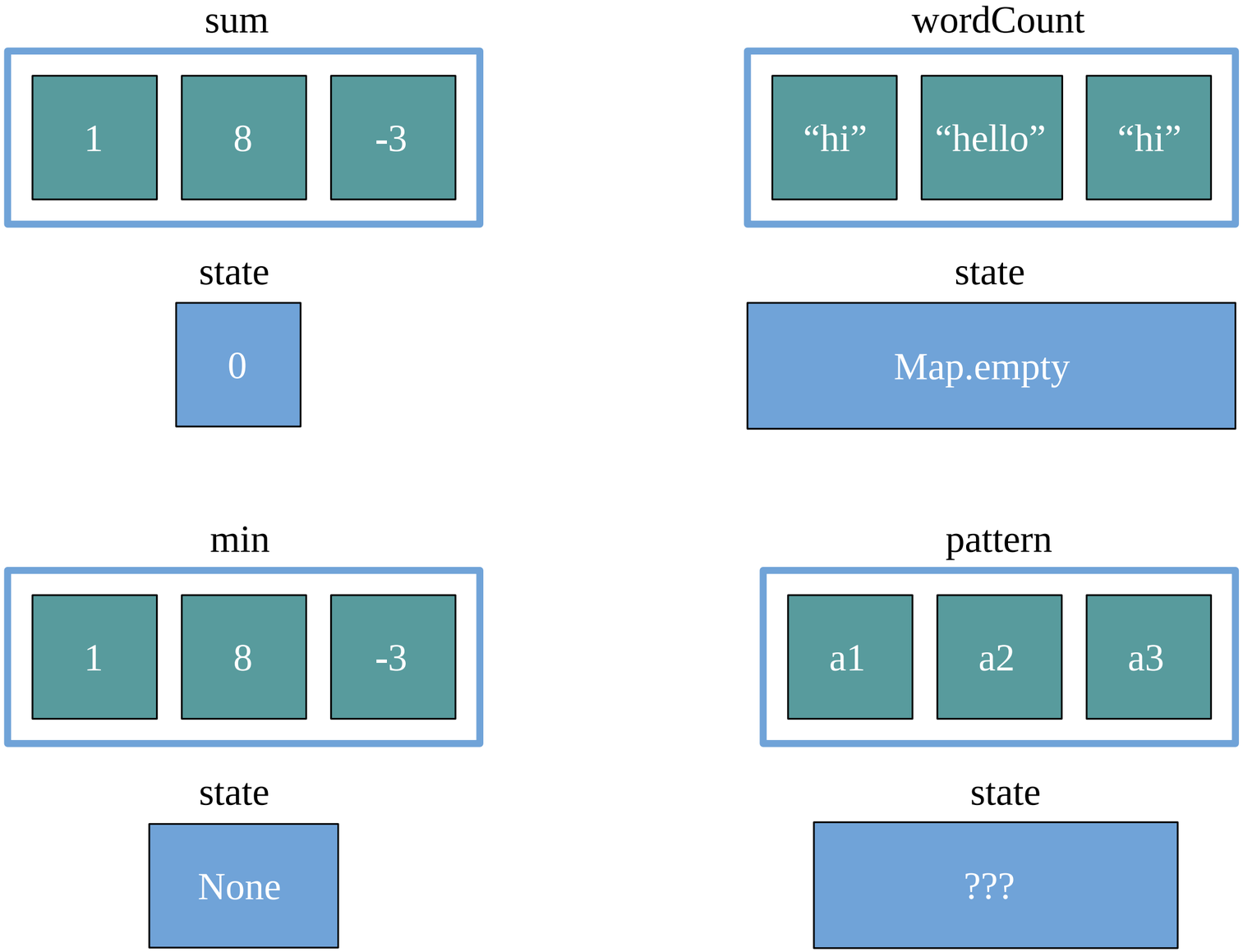
wordCount

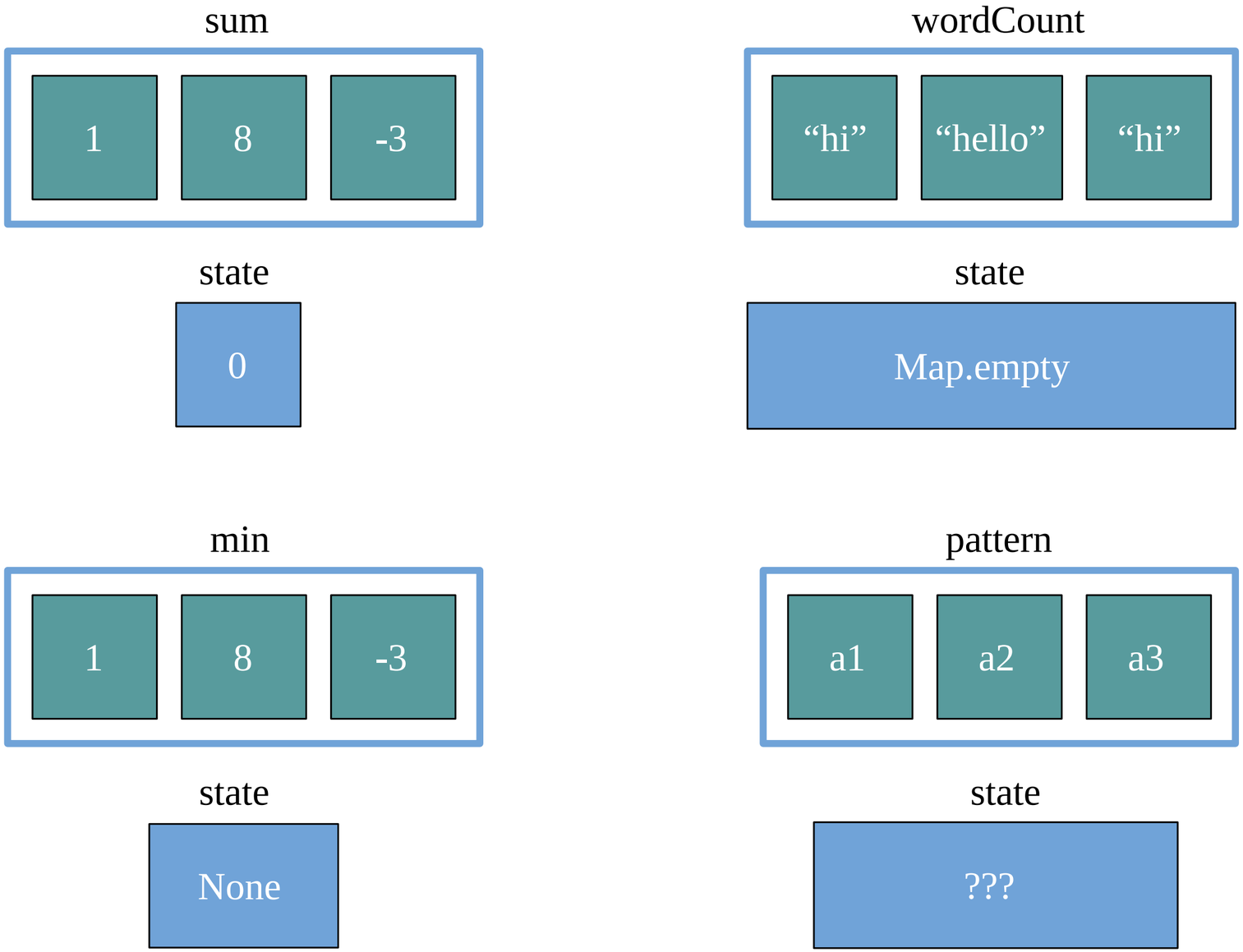
min

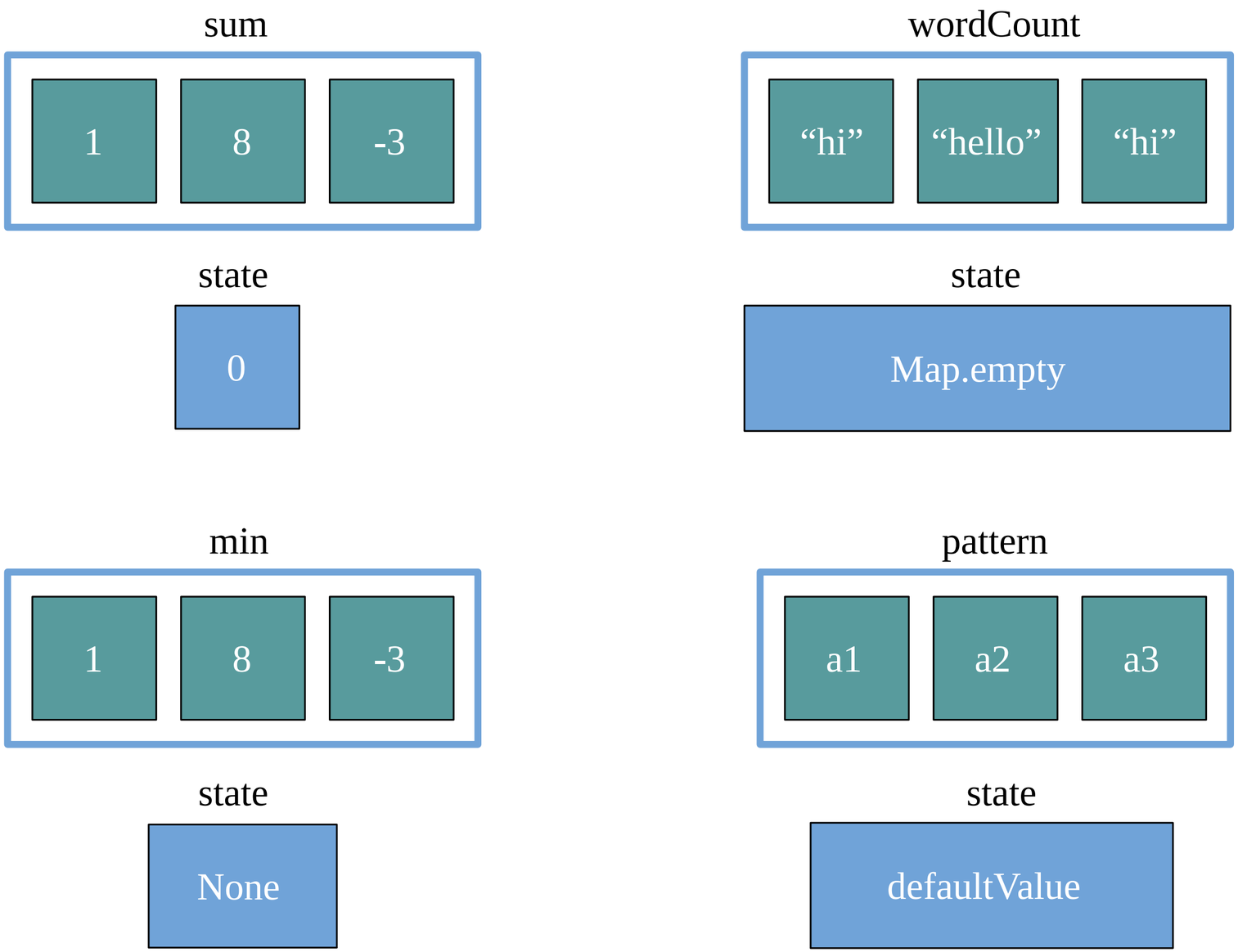


pattern

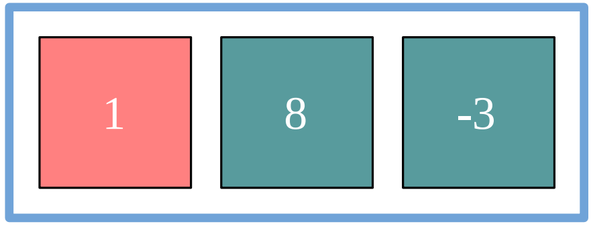



sum
1
8
-3
state
0
wordCount
“hi”
“hello”
“hi”
state
Map.empty
min
1
8
-3
state
None
pattern
a1
a2
a3
state
???

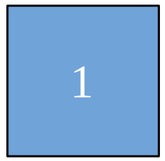

sum
1
8
-3
state
0
wordCount
“hi”
“hello”
“hi”
state
Map.empty
min
1
8
-3
state
None
pattern
a1
a2
a3
state
???


sum
1
8
-3
state
0
wordCount
“hi”
“hello”
“hi”
state
Map.empty
min
1
8
-3
state
None
pattern
a1
a2
a3
state
defaultValue

sum

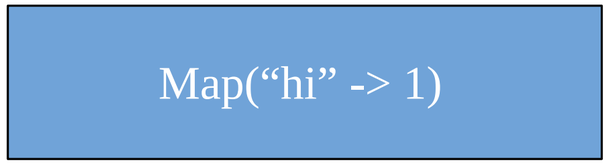


state

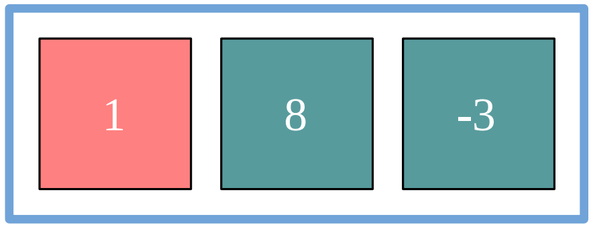


wordCount

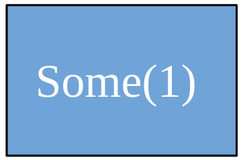
state



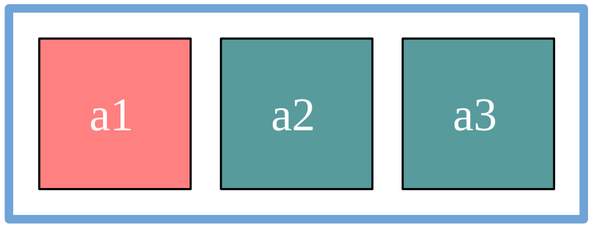
min



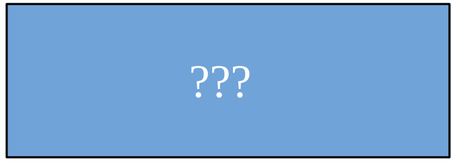
state

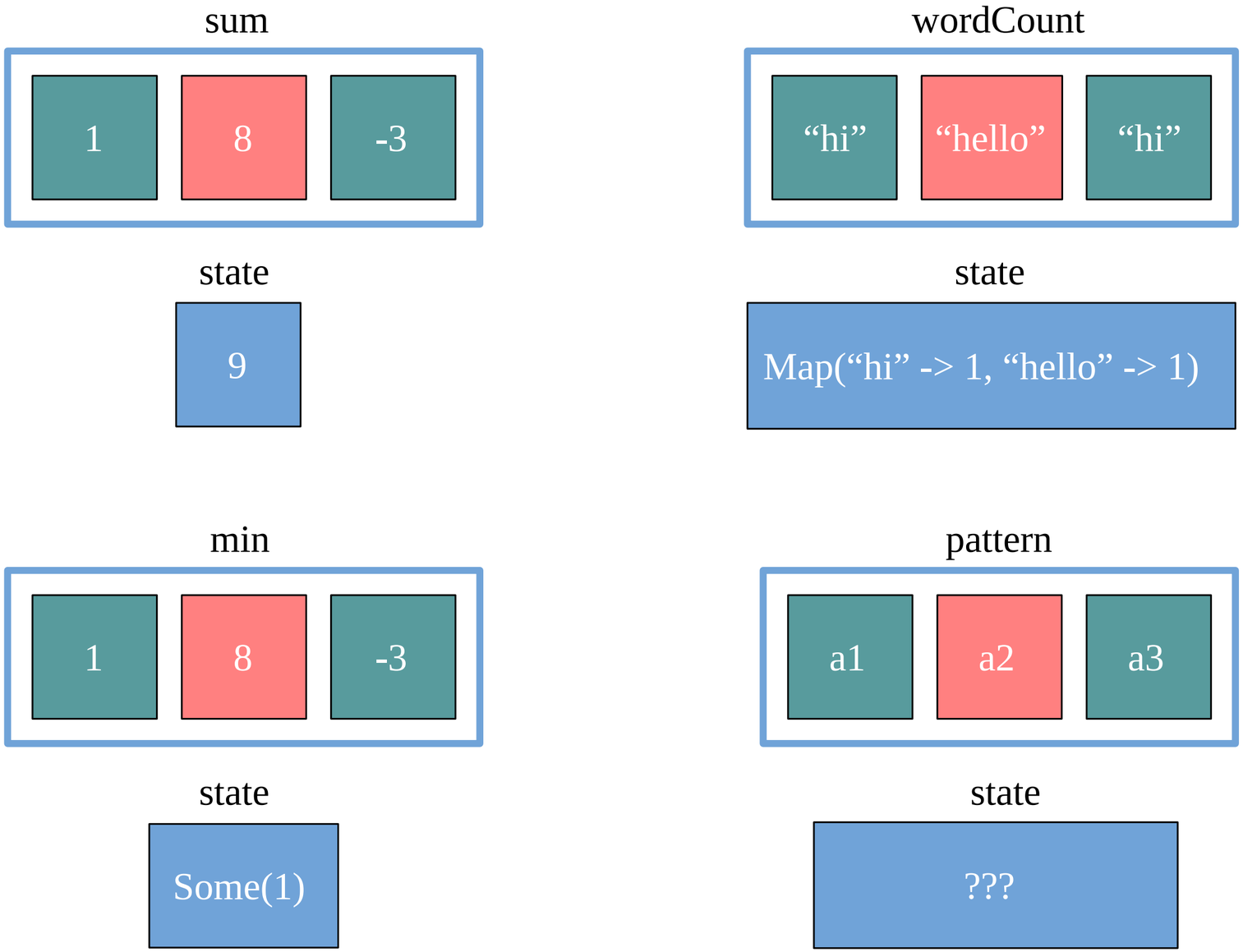


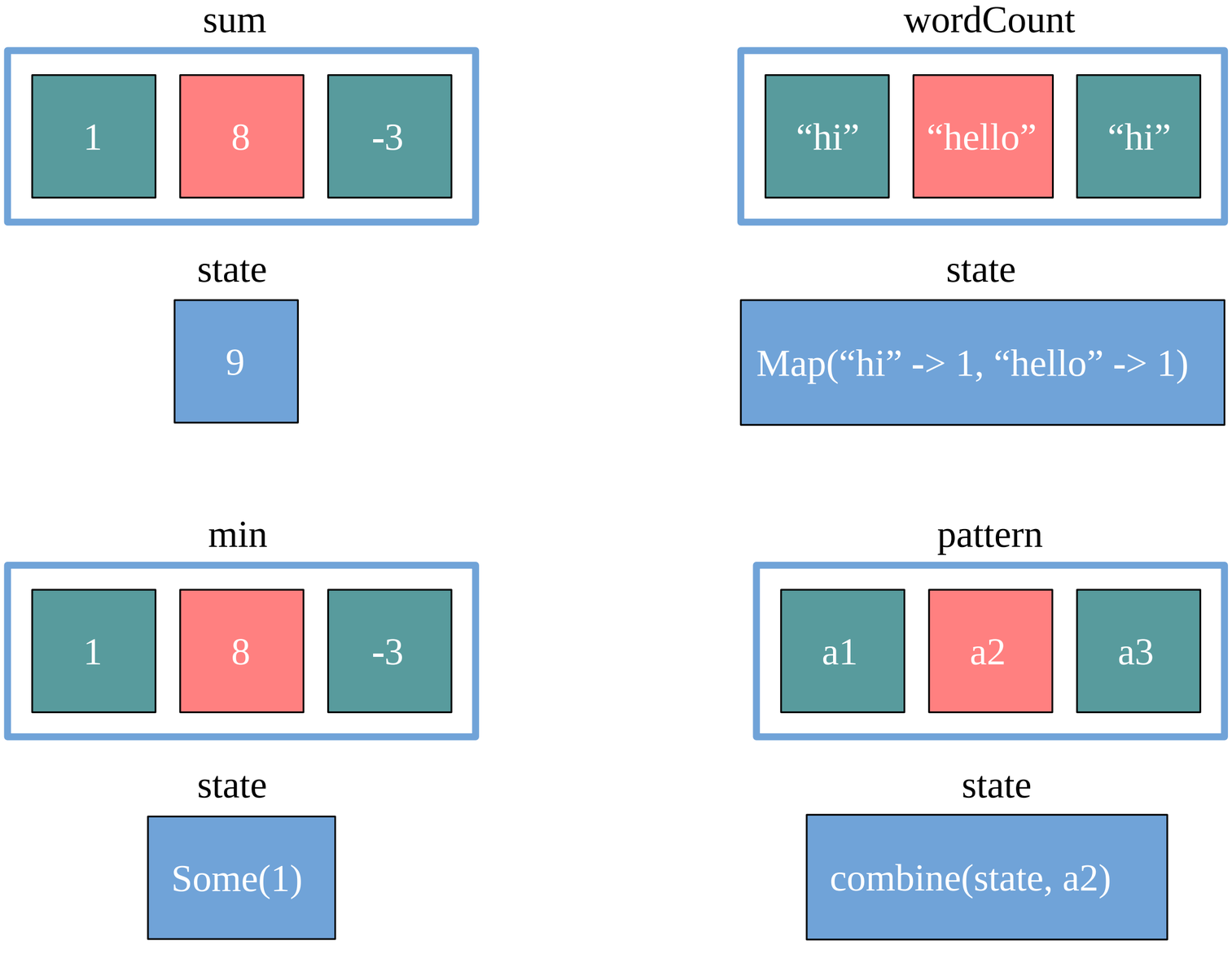
pattern

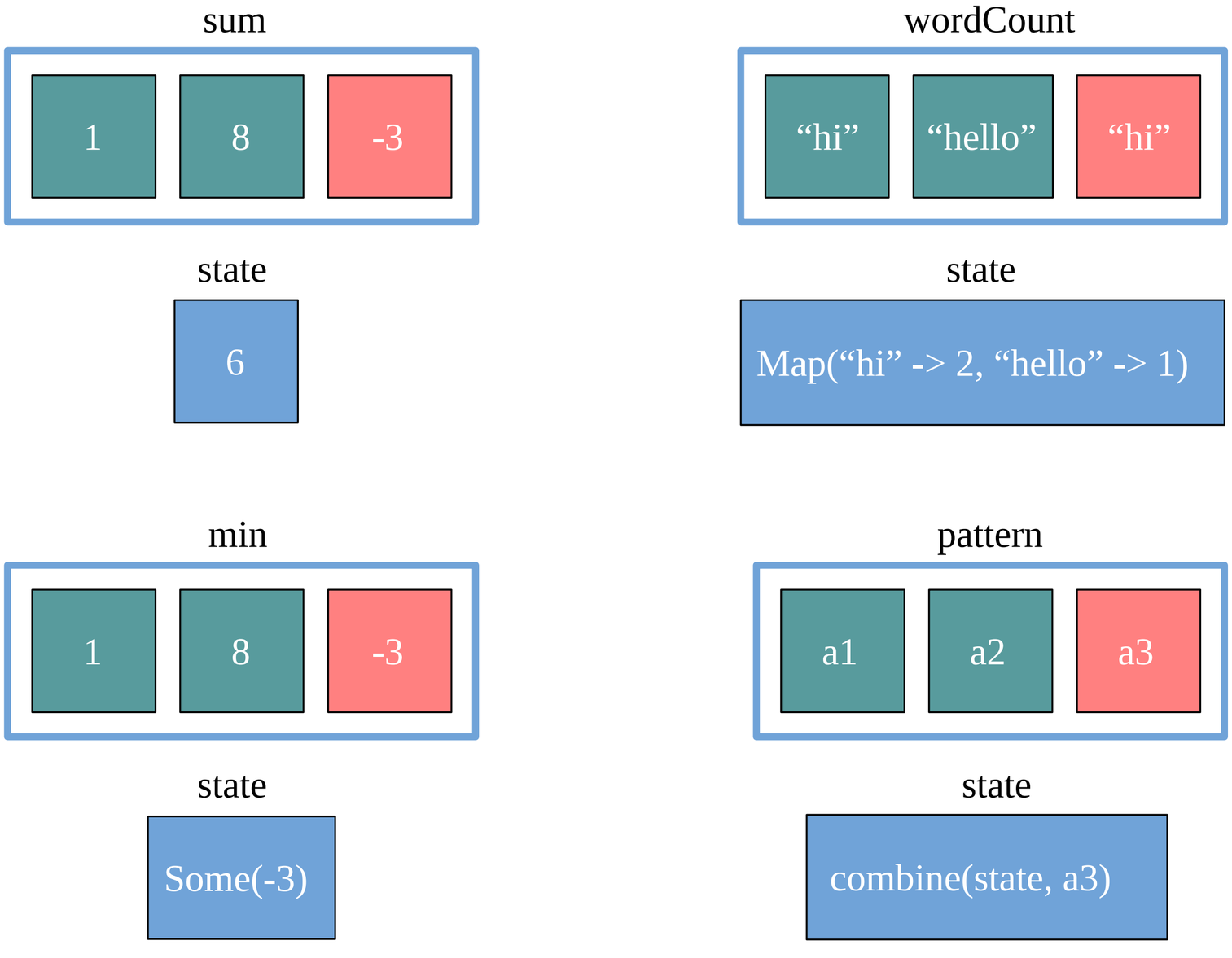


state




sum
1
8
-3
state
9
wordCount
“hi”
“hello”
“hi”
state
Map(“hi” -> 1, “hello” -> 1)
min
1
8
-3
state
Some(1)
pattern
a1
a2
a3
state
???


sum
1
8
-3
state
9
wordCount
“hi”
“hello”
“hi”
state
Map(“hi” -> 1, “hello” -> 1)
min
1
8
-3
state
Some(1)
pattern
a1
a2
a3
state
combine(state, a2)


sum
1
8
-3
state
6
wordCount
“hi”
“hello”
“hi”
state
Map(“hi” -> 2, “hello” -> 1)
min
1
8
-3
state
Some(-3)
pattern
a1
a2
a3
state
combine(state, a3)

ForLoopExercises.scala

Summary

- For loop syntax
- Iterate over a List
- foldLeft