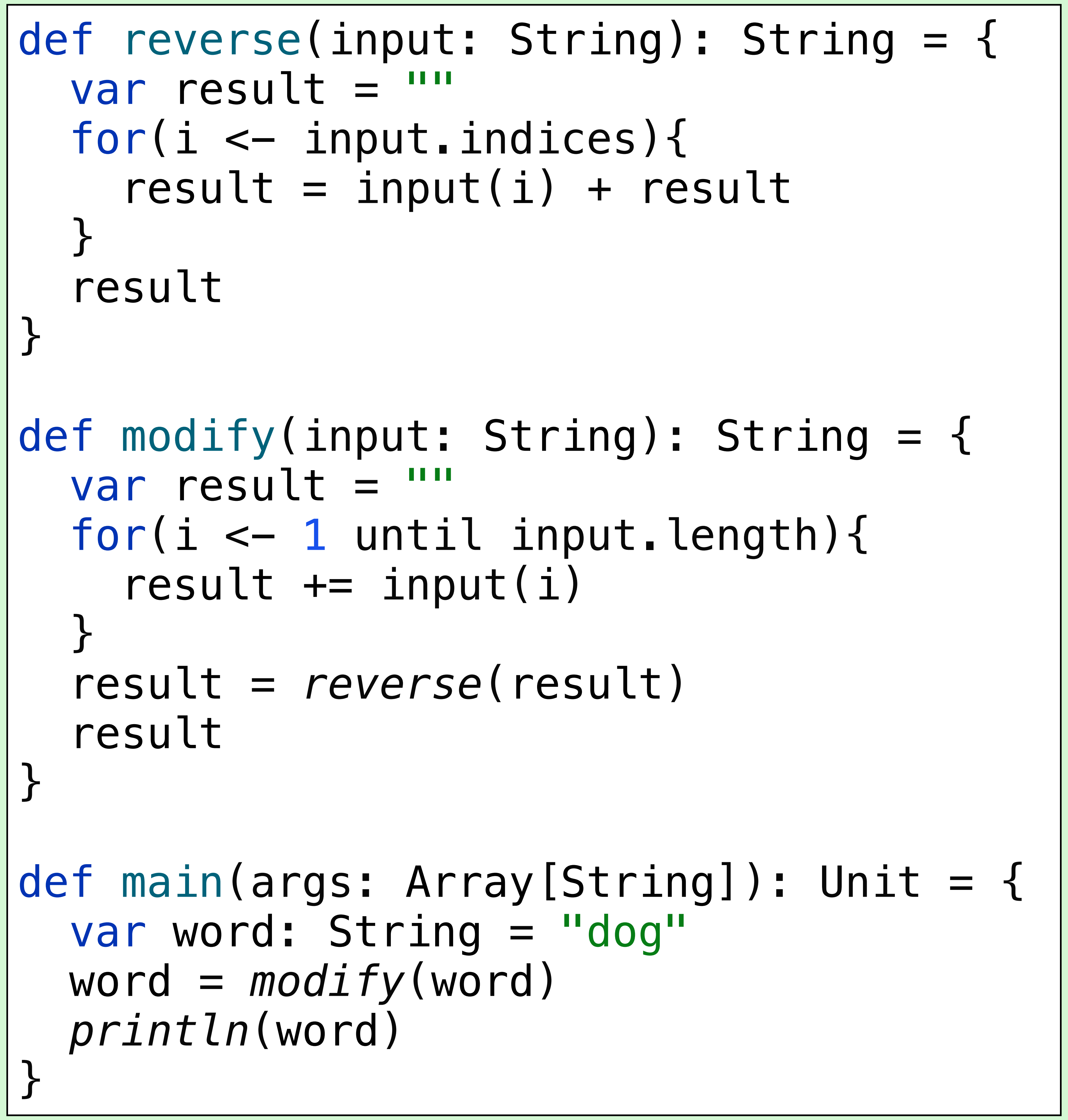
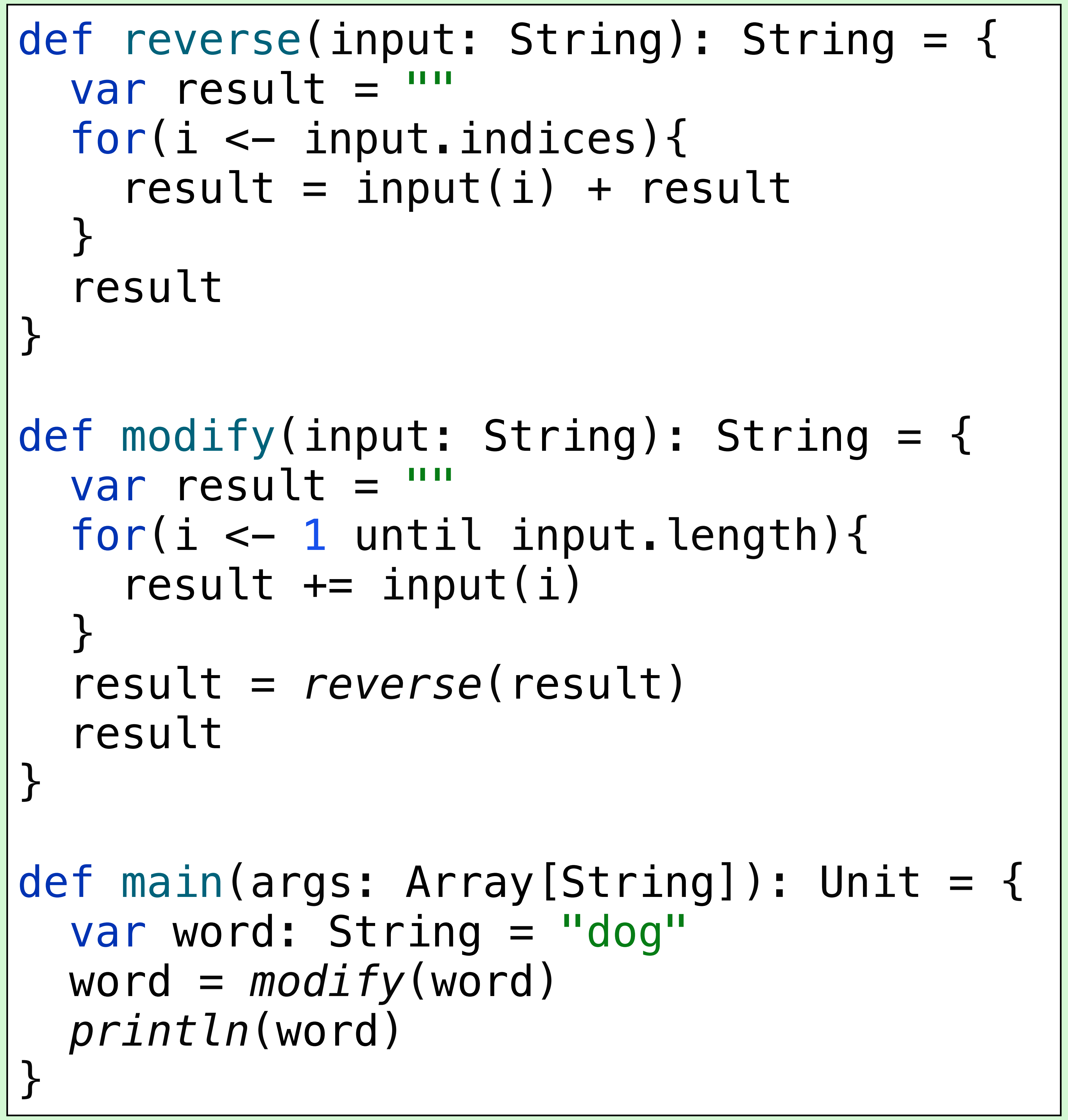
More Memory Diagrams

String Example



```scala
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

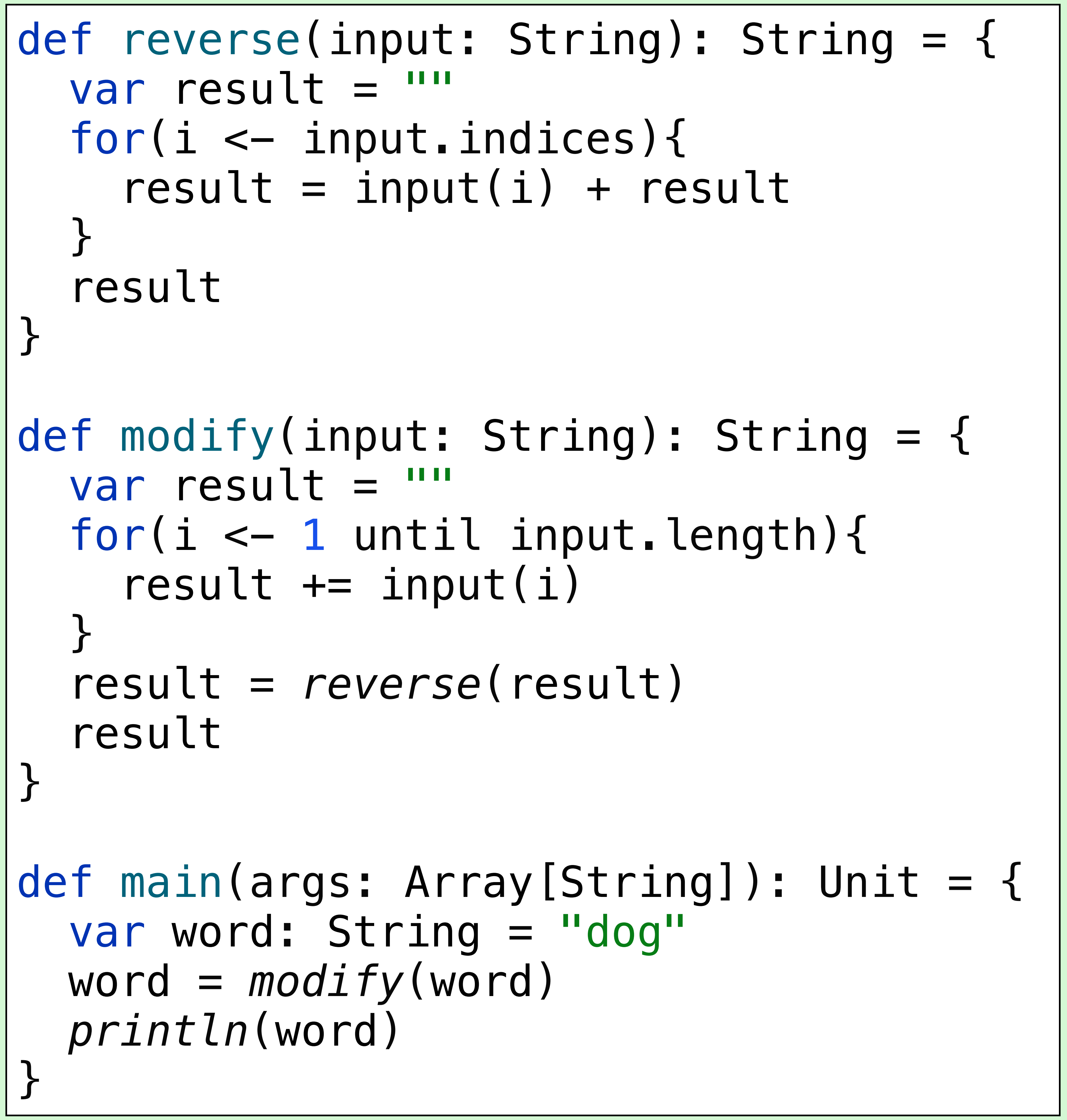


```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Setup the memory diagram

Stack		Heap
Name	Value	

in/out



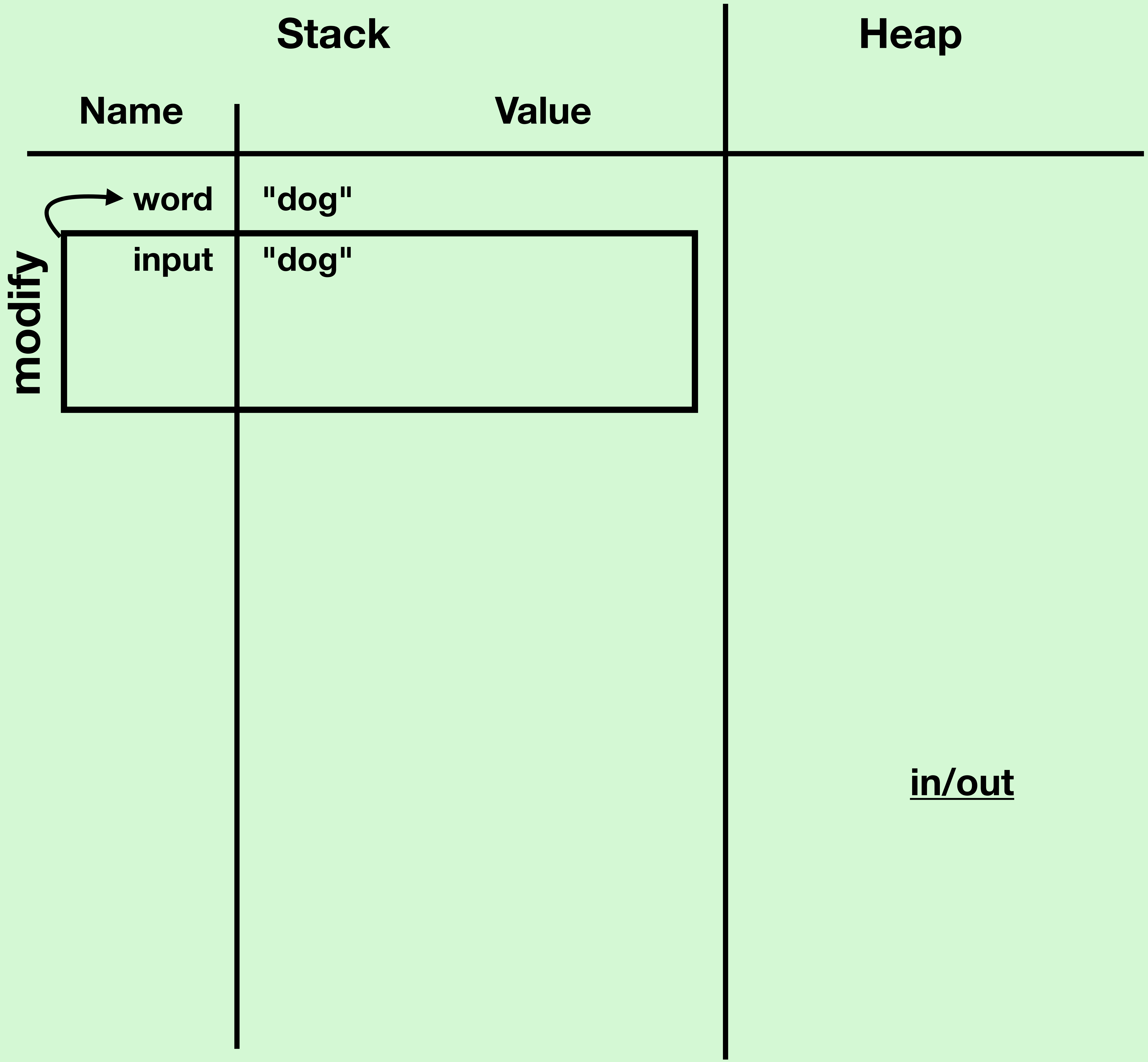
```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Start with the main method
- Add the word variable to the stack

Stack		Heap
Name	**Value**	
word	**"dog"**	

in/out

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



Stack

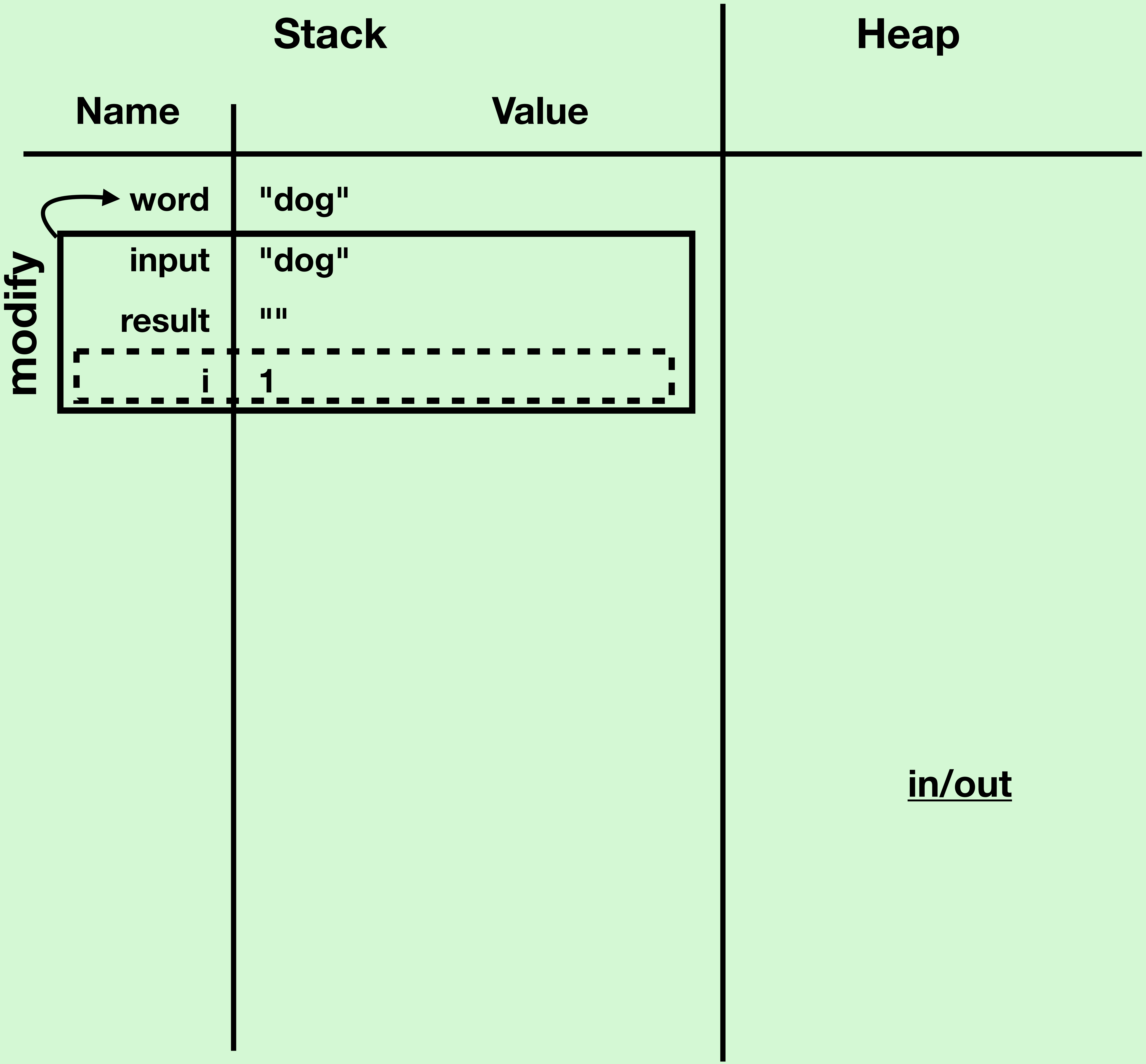
Heap

Name	Value
word	"dog"
input	"dog"

- Add a stack frame for the modify method call

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Add a code block for the loop



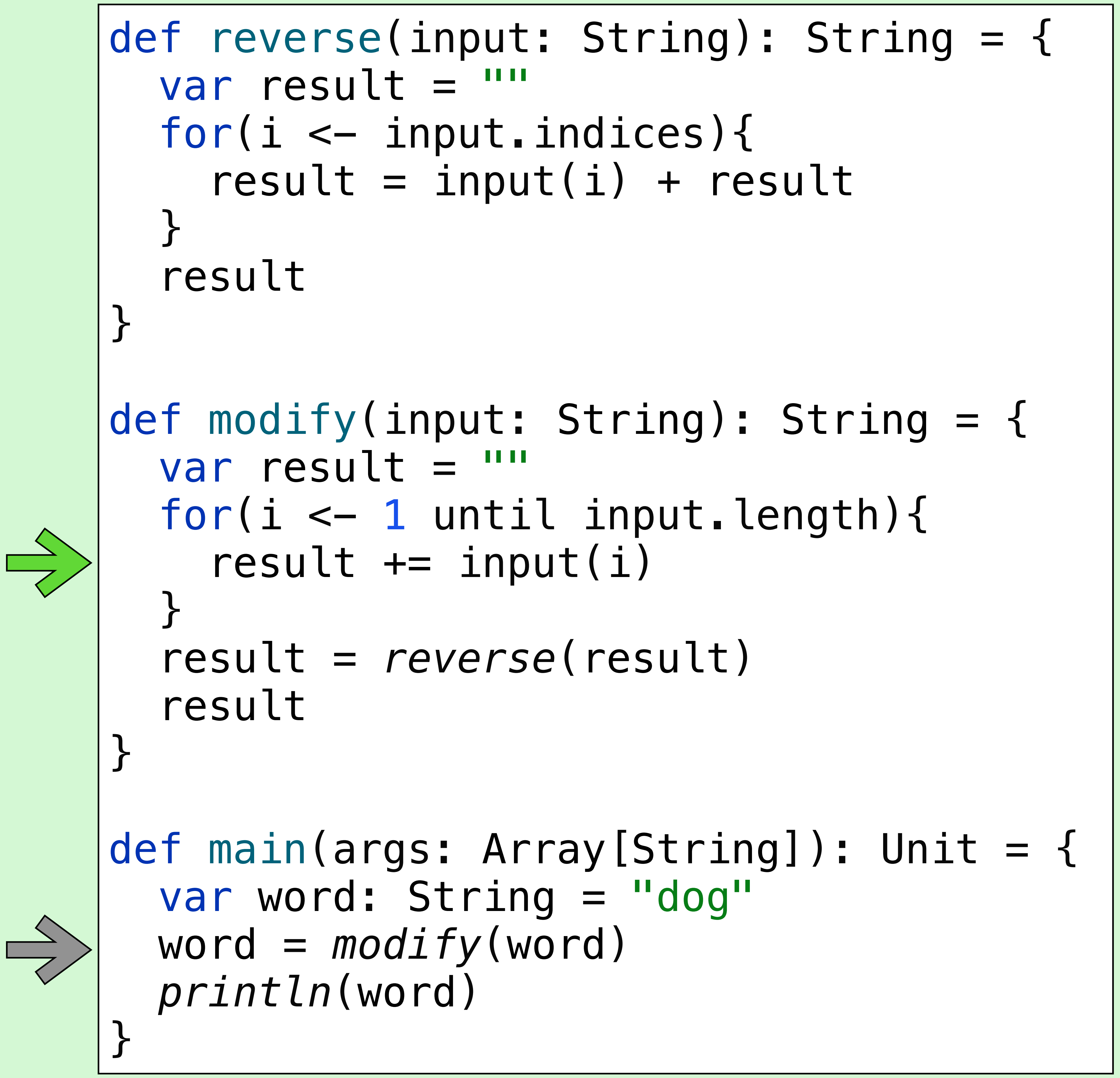
Stack

Heap

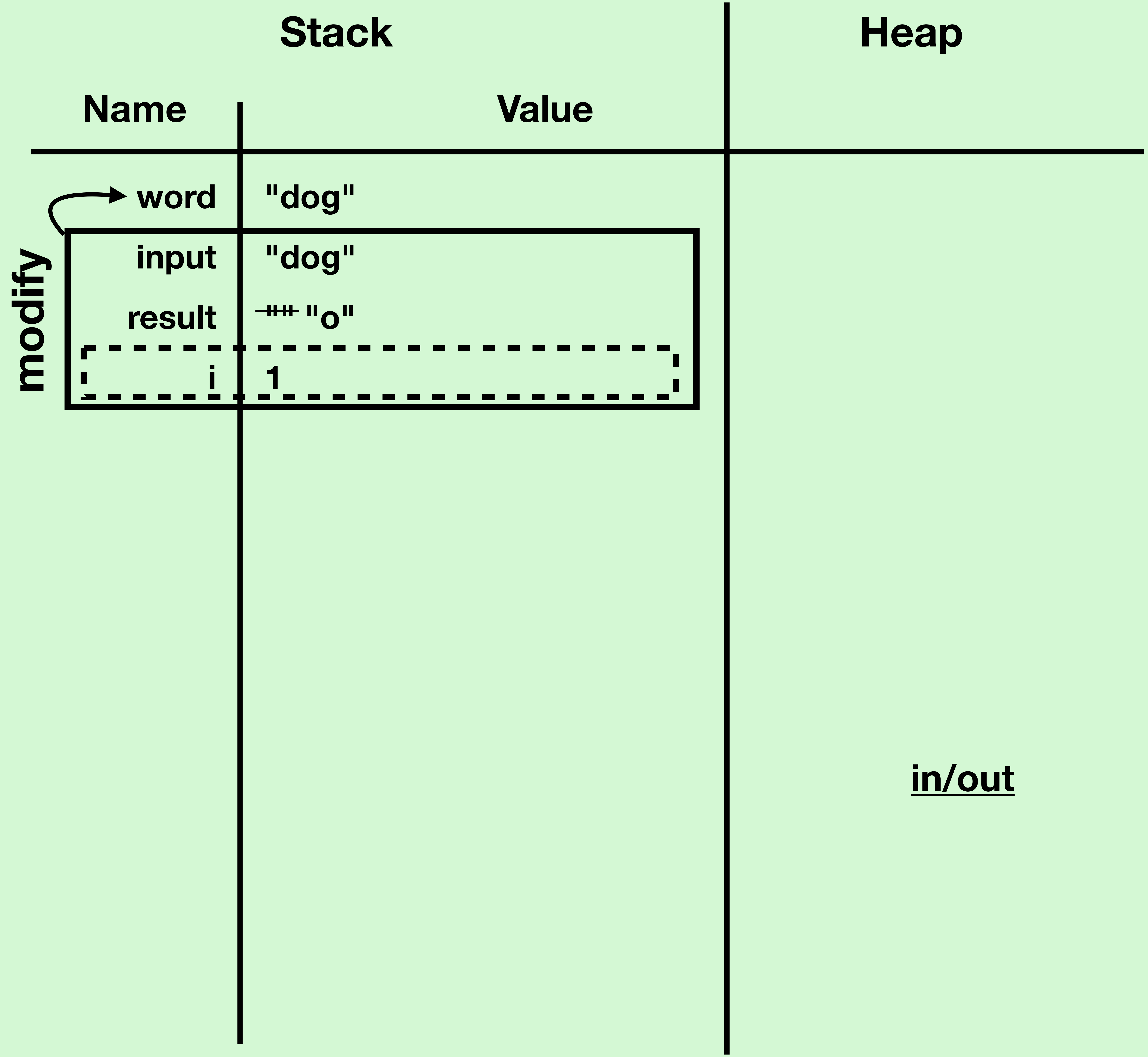
Name	Value
word	"dog"
input	"dog"
result	""
i	1

modify

in/out



```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

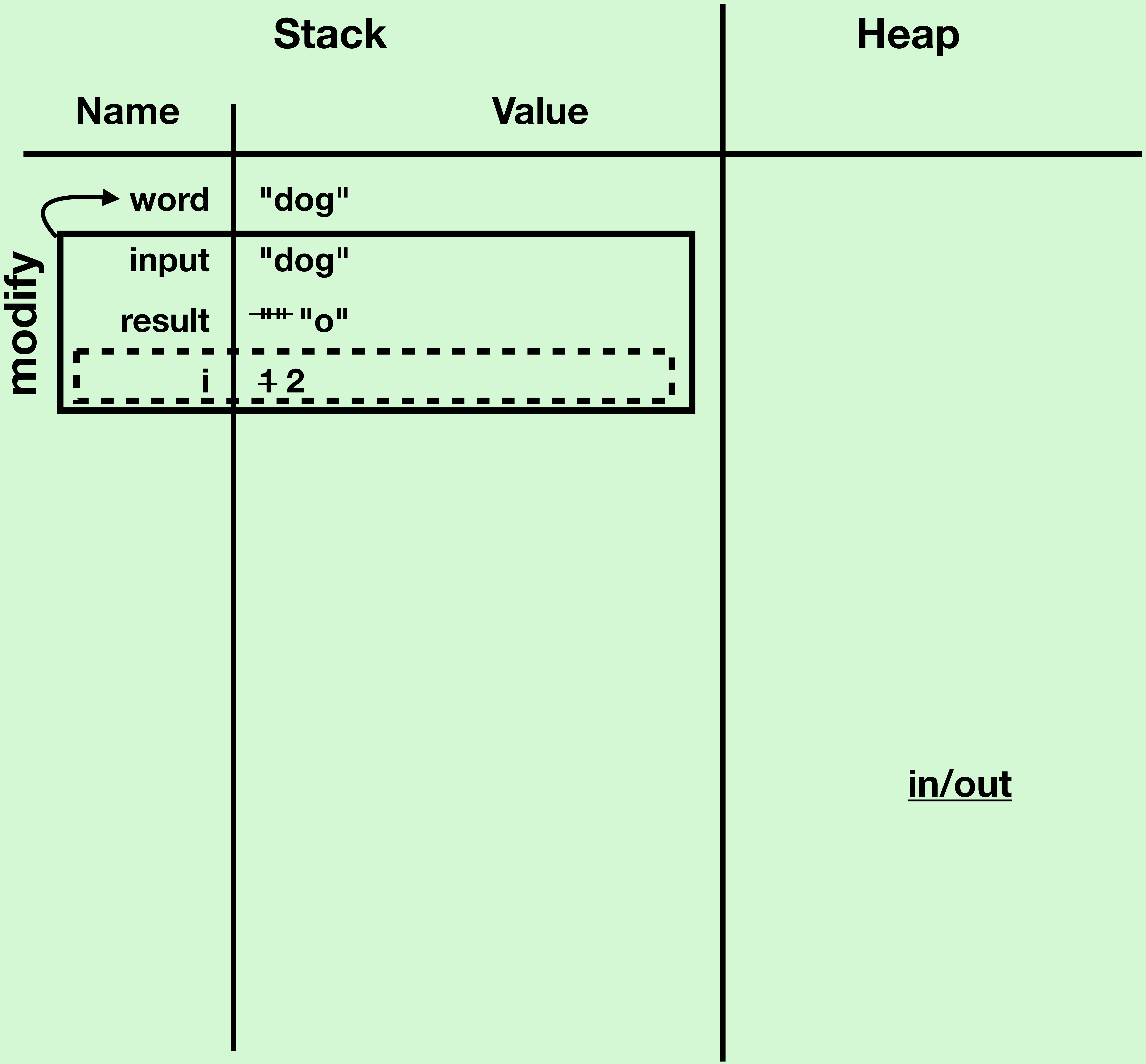


- We are using the indices of the String to access individual characters

- Same syntax as elements of an Array

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Advance to loop variable



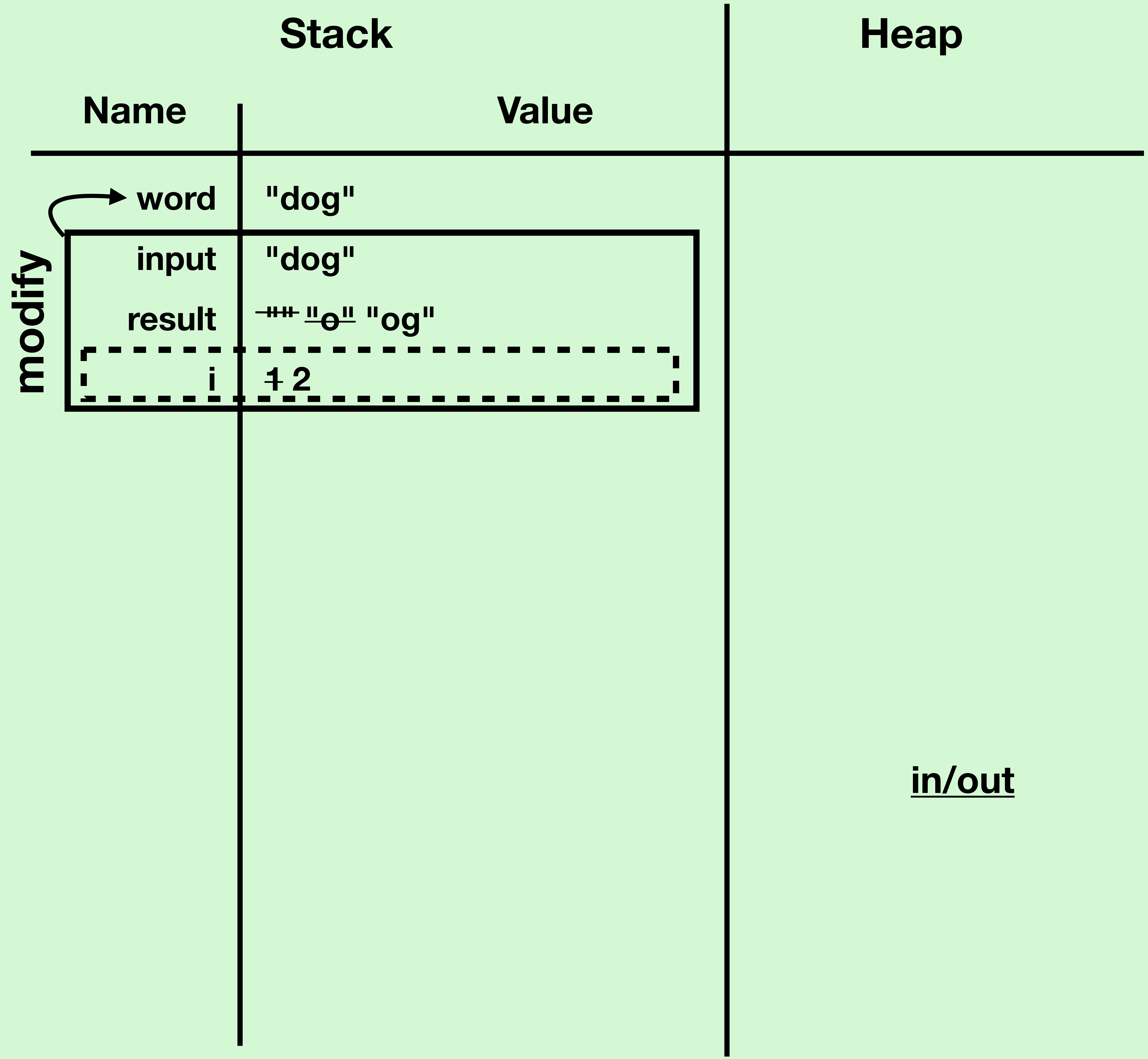
Stack

Heap

Name	Value
word	"dog"
input	"dog"
result	~~""~~ "o"
i	~~1~~ 2

in/out

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



Stack

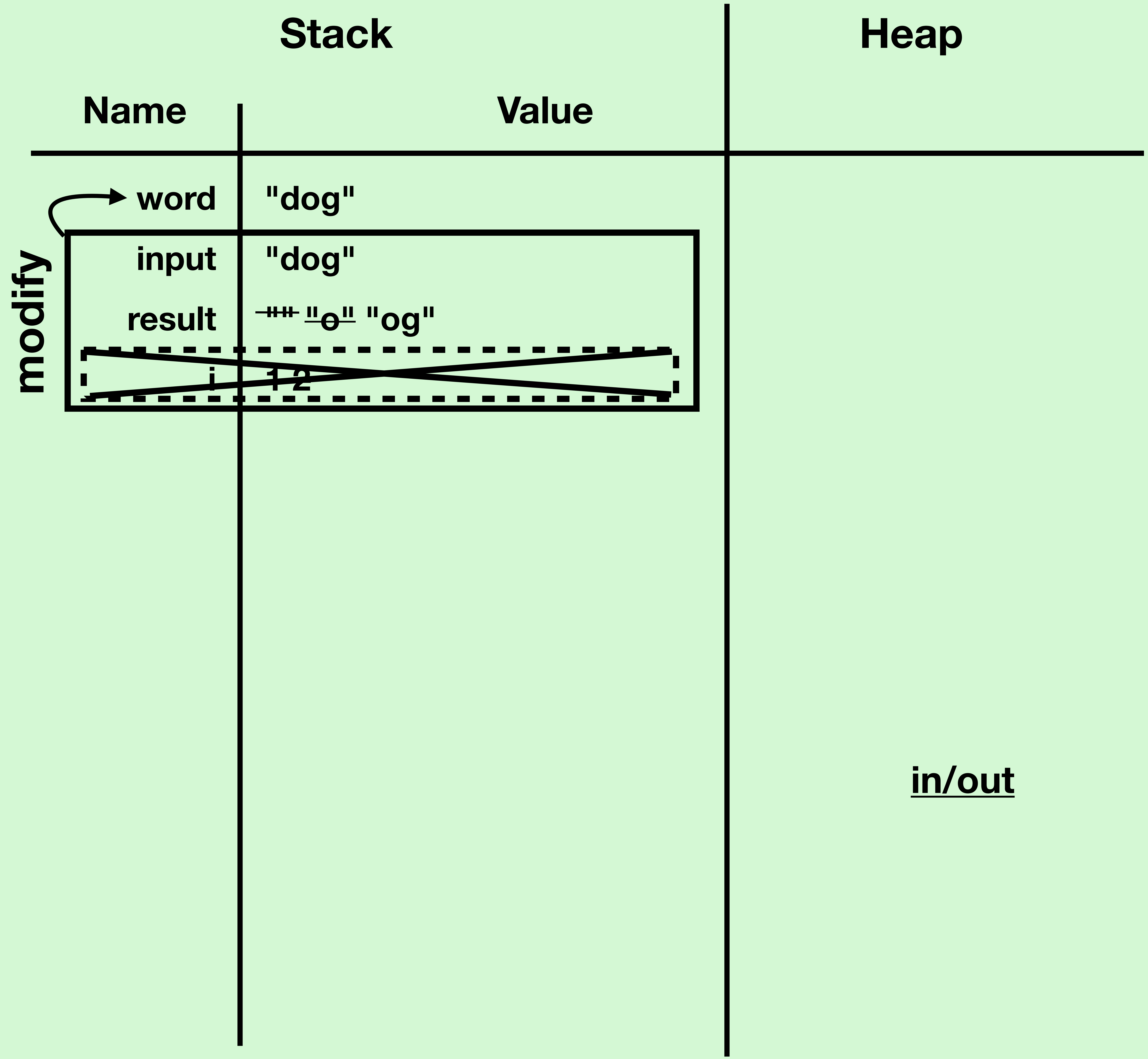
Heap

Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
i	~~1~~ 2

in/out

- Access the character at index 2

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



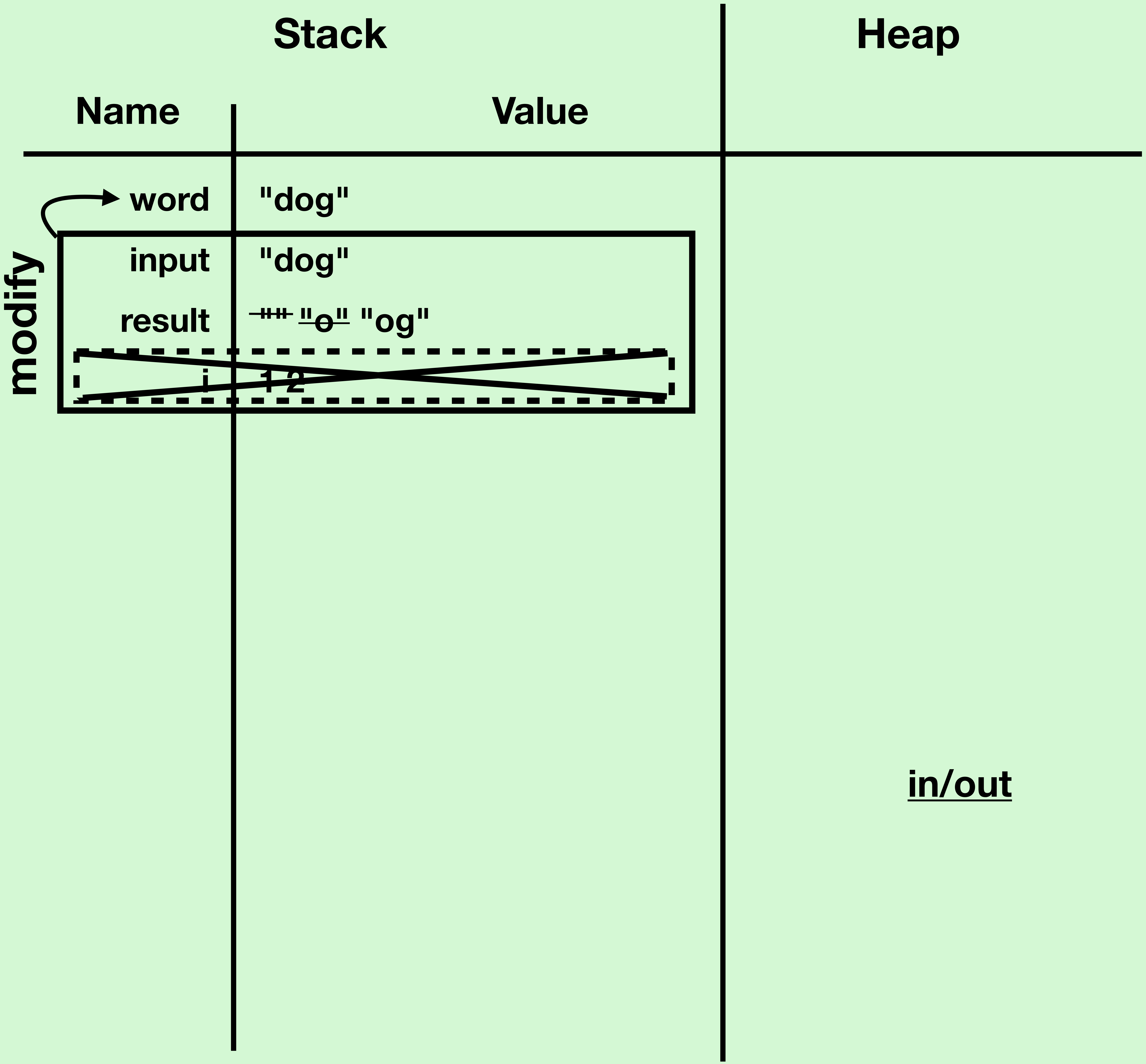
Stack

Heap

Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
~~i~~	~~1 2~~

- Until does not include the endpoint so i does not advance to 3
- The loop block ends

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



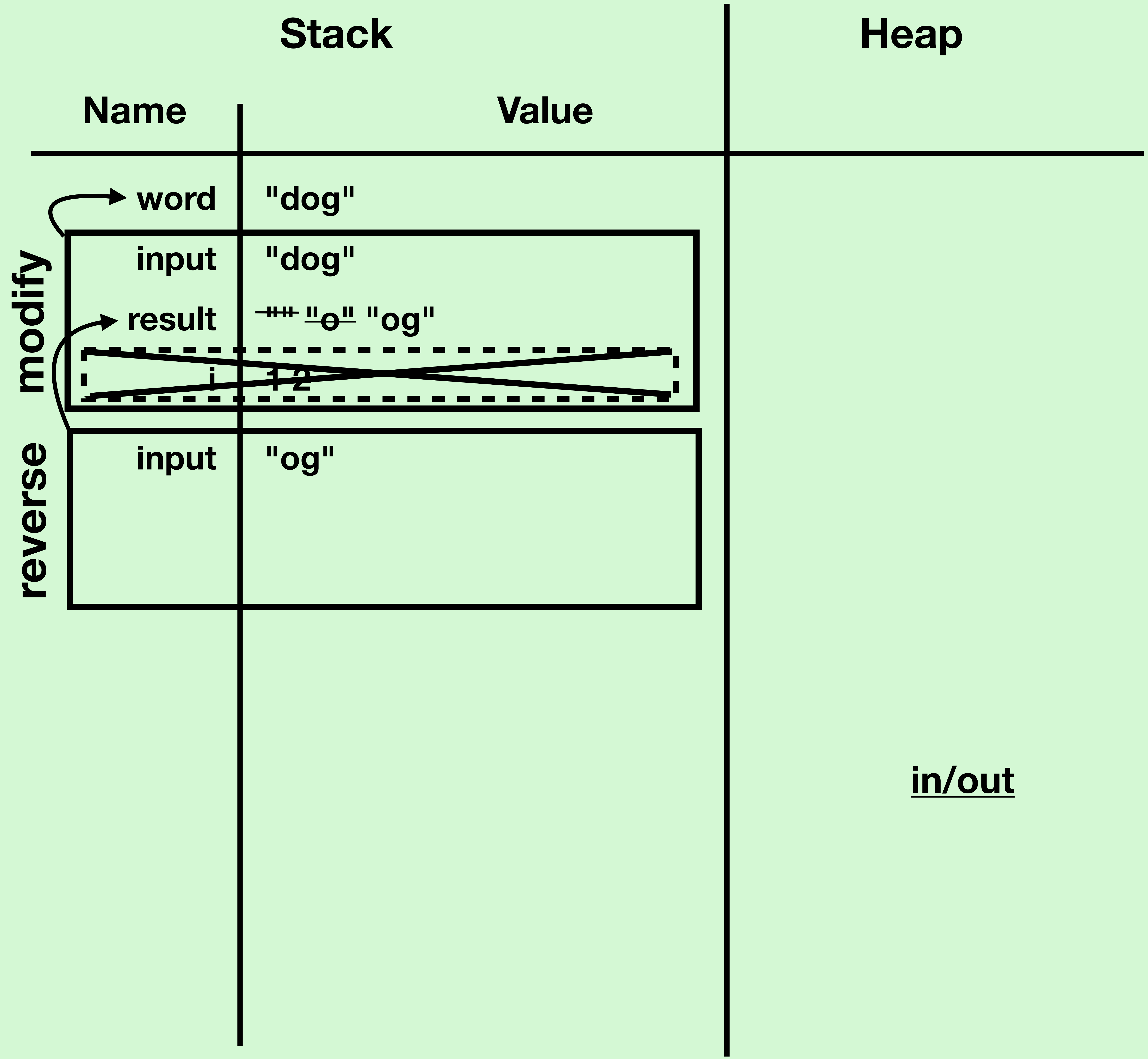
Stack

Heap

Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
~~i~~	~~1 2~~

- We reach another method call
- Add another stack frame to the stack

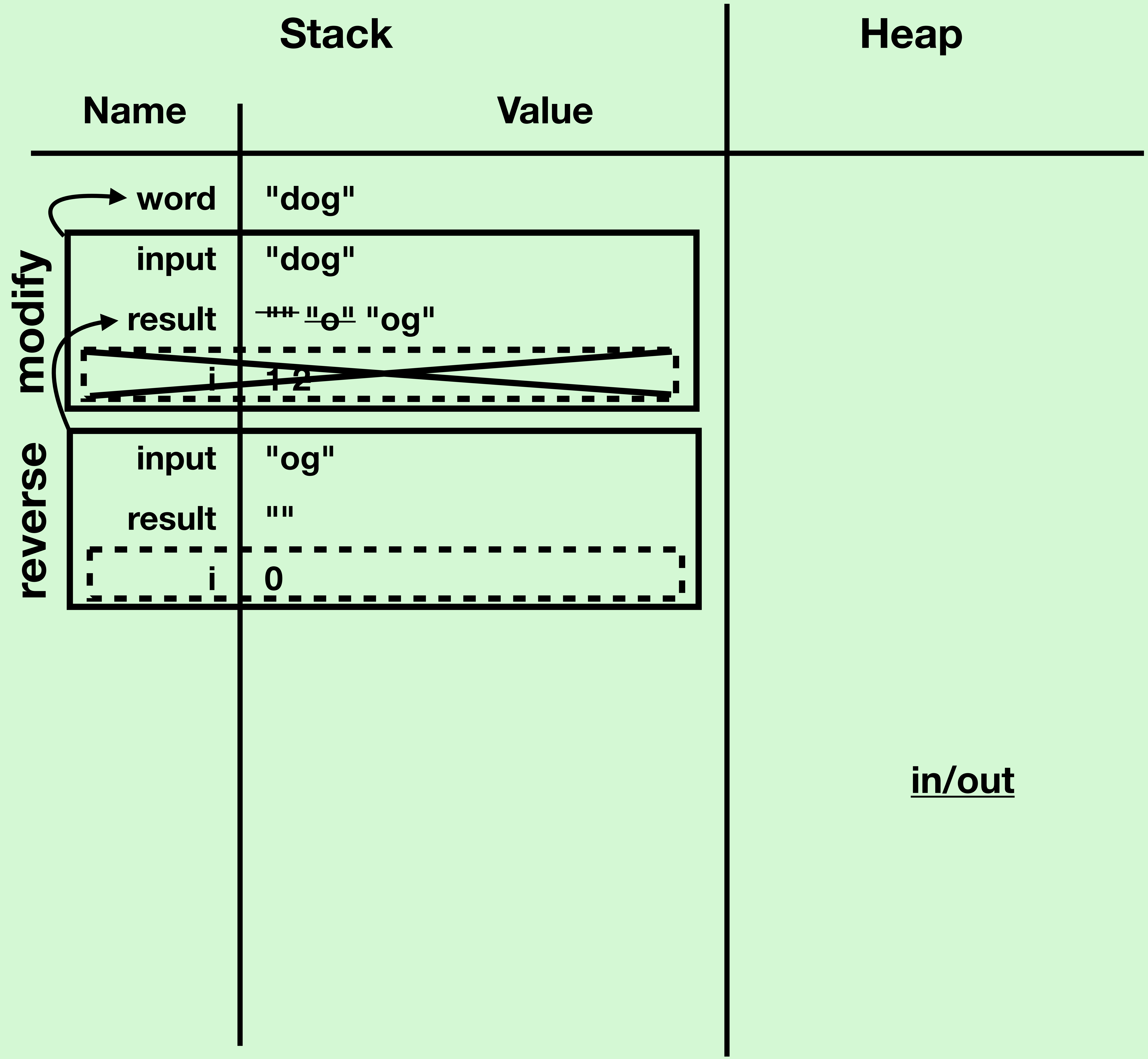
```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
~~i~~	~~1 2~~
input	"og"

- We don't create stack frames inside another stack frames in our memory diagrams
- Add it to the bottom of the stack

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



Stack

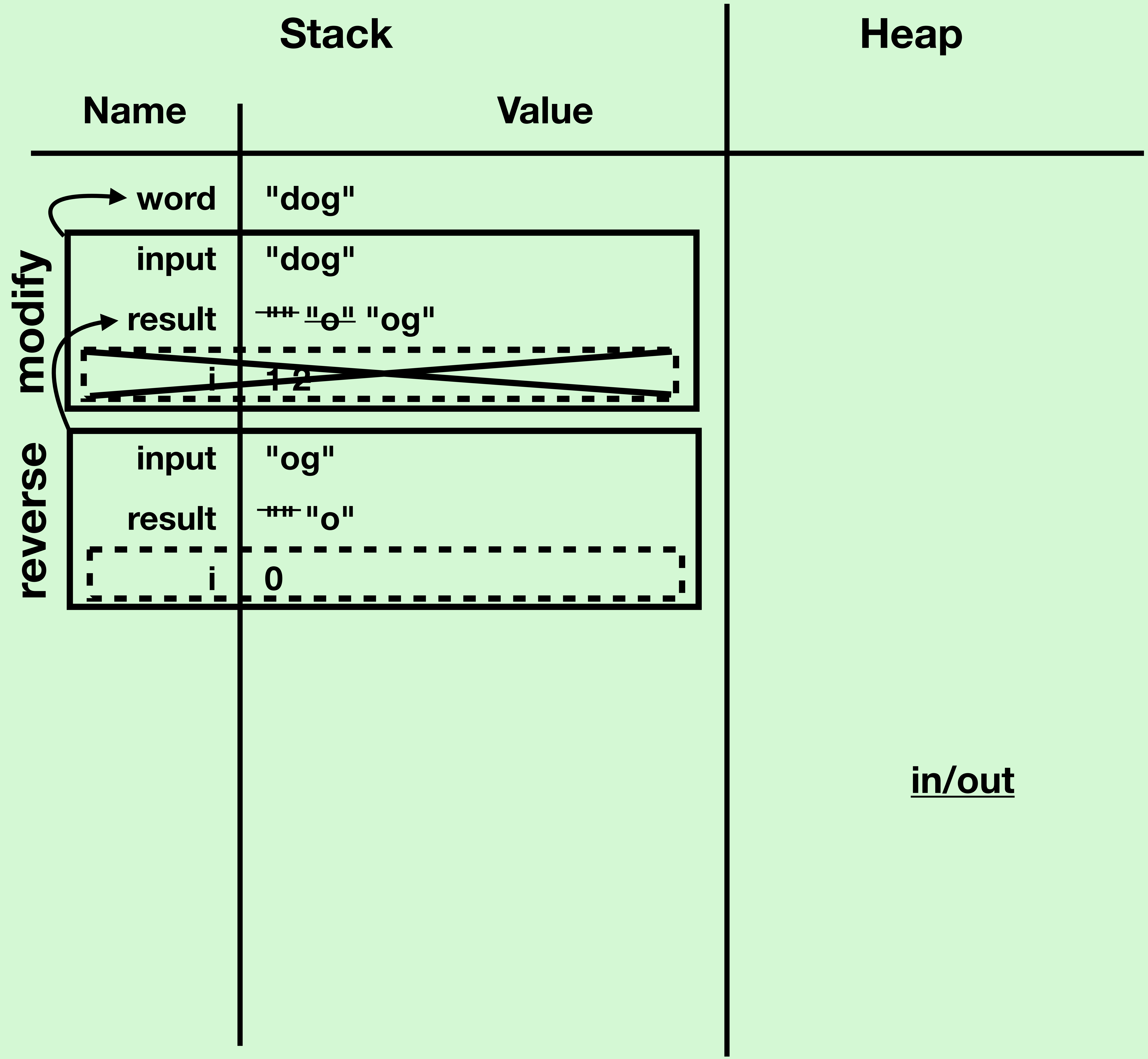
Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
~~i~~	~~1~~ ~~2~~
input	"og"
result	""
i	0

Heap

in/out

- Loop over the indices of the characters of a String just like iterating over the indices of an Array
  - Thick of a String as an Array of Characters

```scala
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



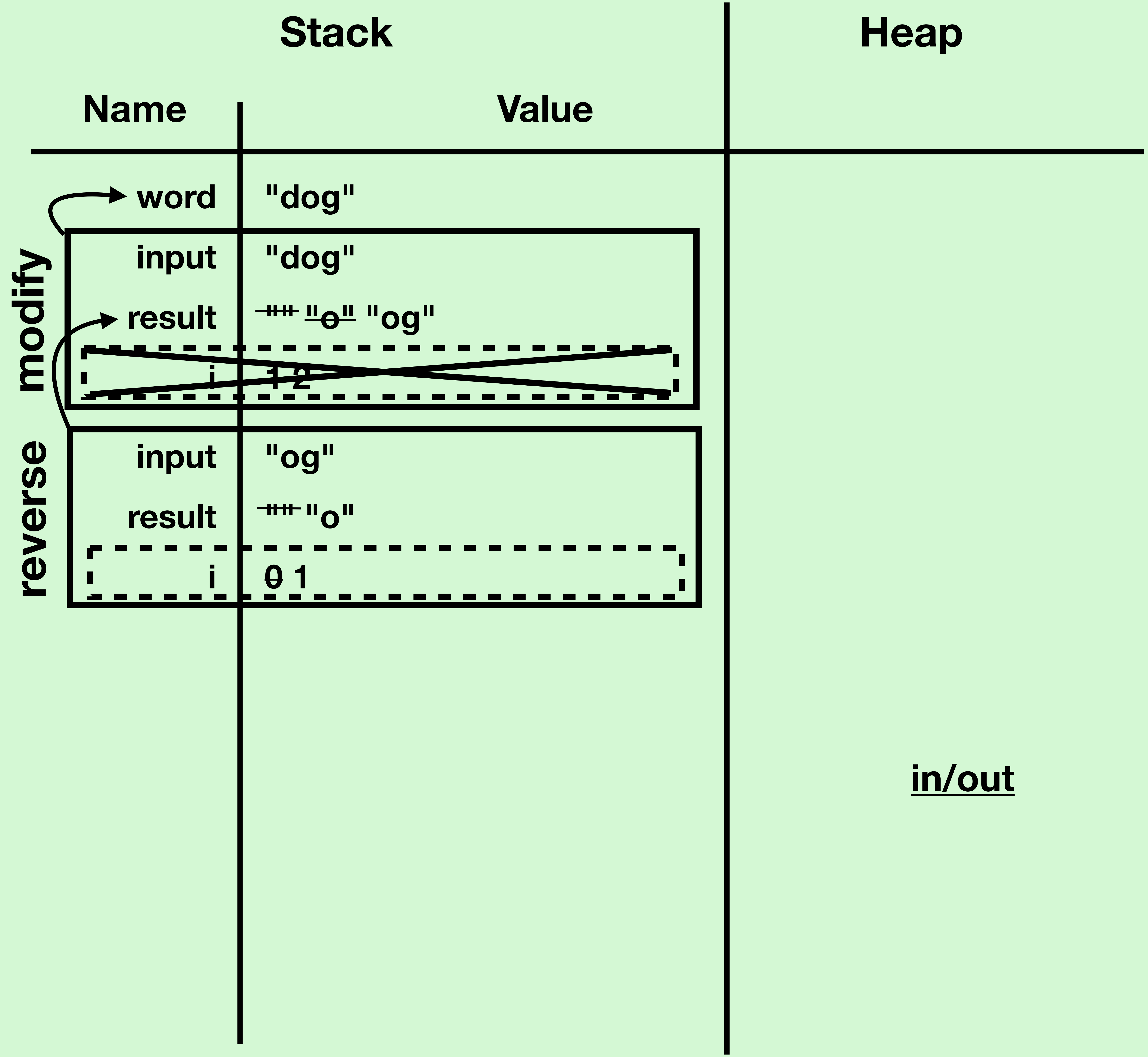
Stack | **Heap**

Name	Value
word	"dog"
input	"dog"
result	~~""~~ ~~"o"~~ "og"
~~i~~	~~1 2~~
input	"og"
result	~~""~~ "o"
i	0

- Update result

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

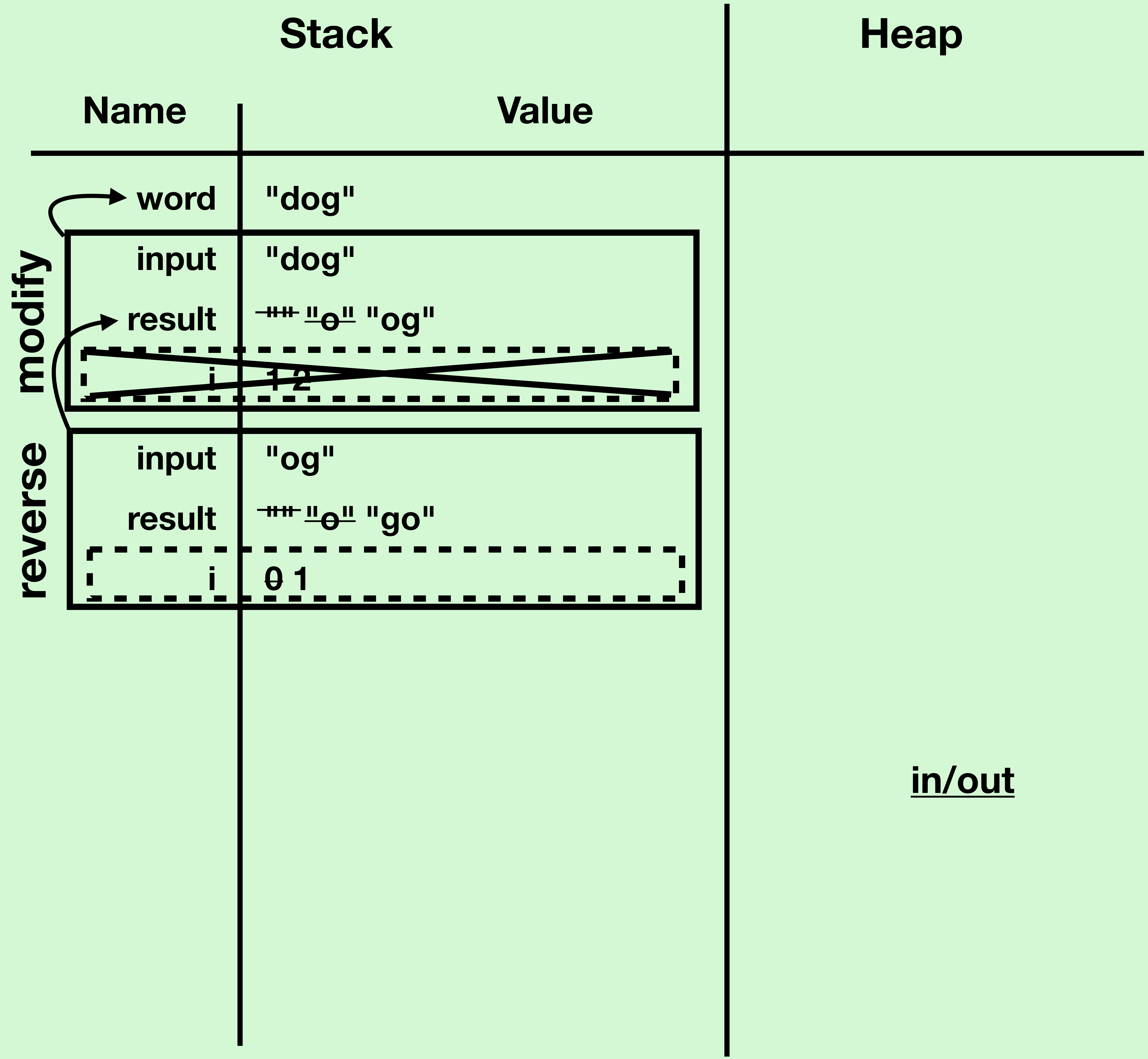
- Advance to the next, and last, index



Stack | **Heap**

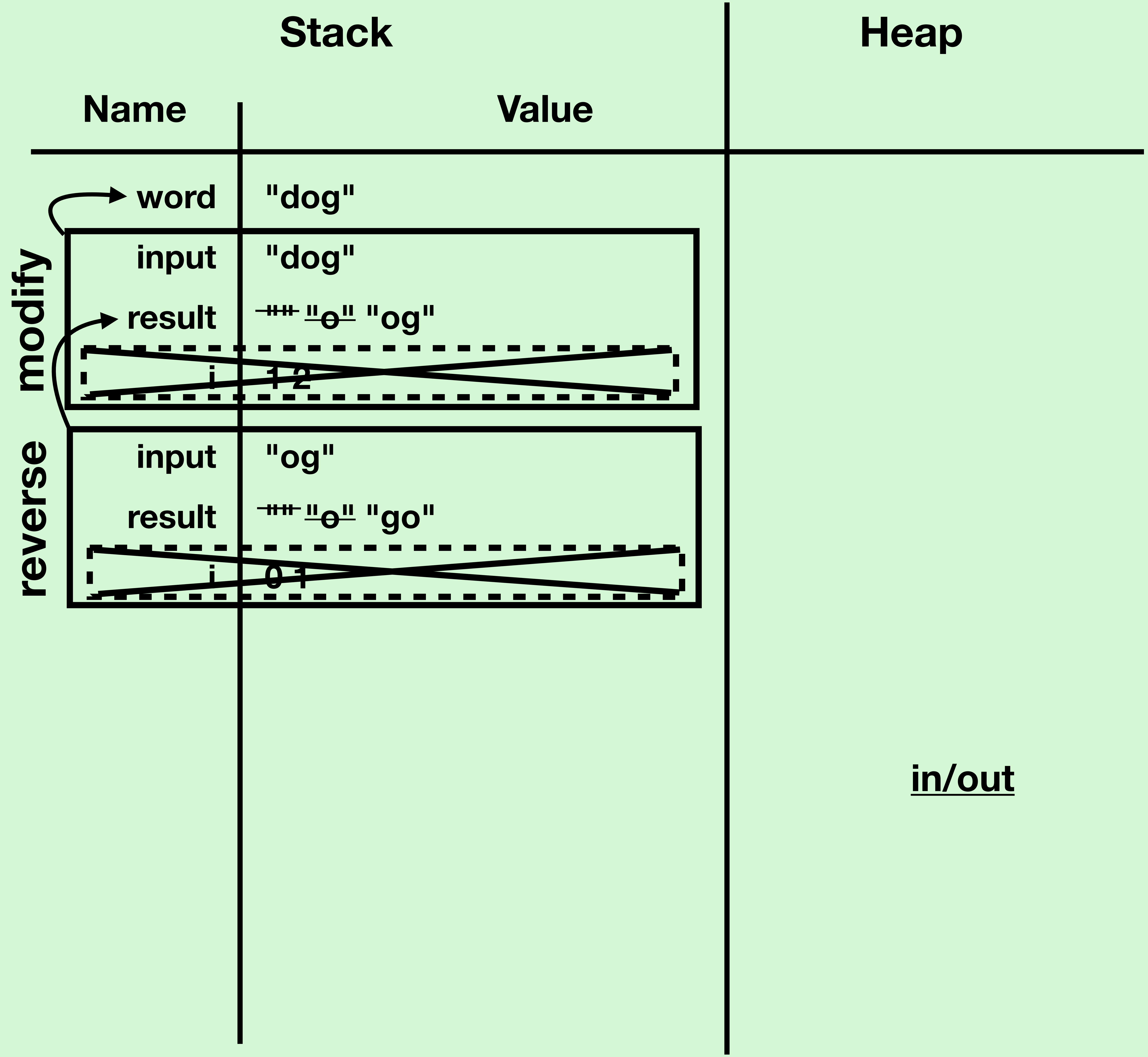
	Name	Value
	word	"dog"
modify	input	"dog"
	result	~~""~~ ~~"o"~~ "og"
	~~i~~	~~1~~ 2
reverse	input	"og"
	result	~~""~~ "o"
	i	~~0~~ 1

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



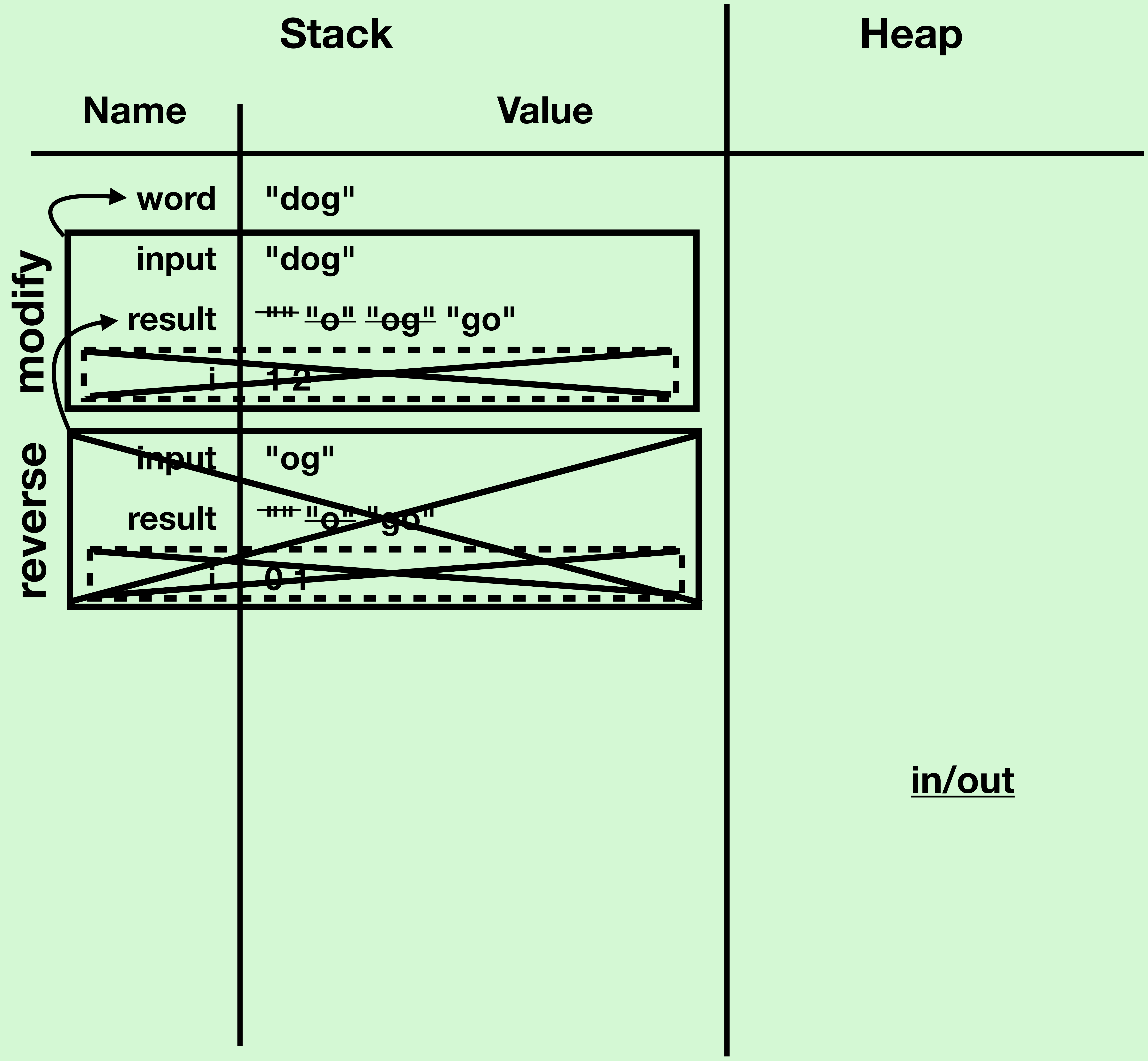
- Concatenate
- Be careful of the order of the concatenation

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



- Loop ends
- Cross out the code block
- i is no longer in memory

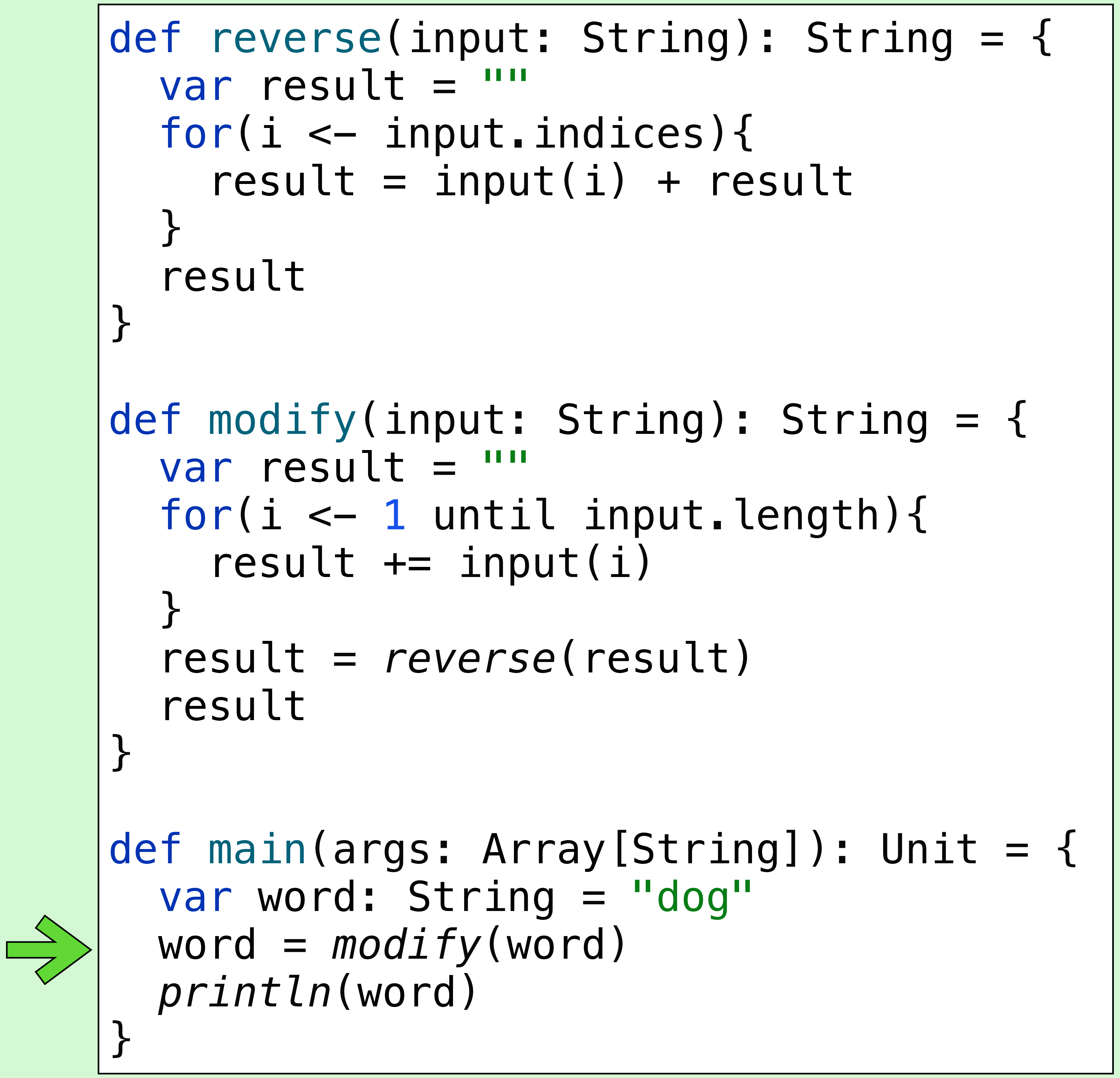
```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



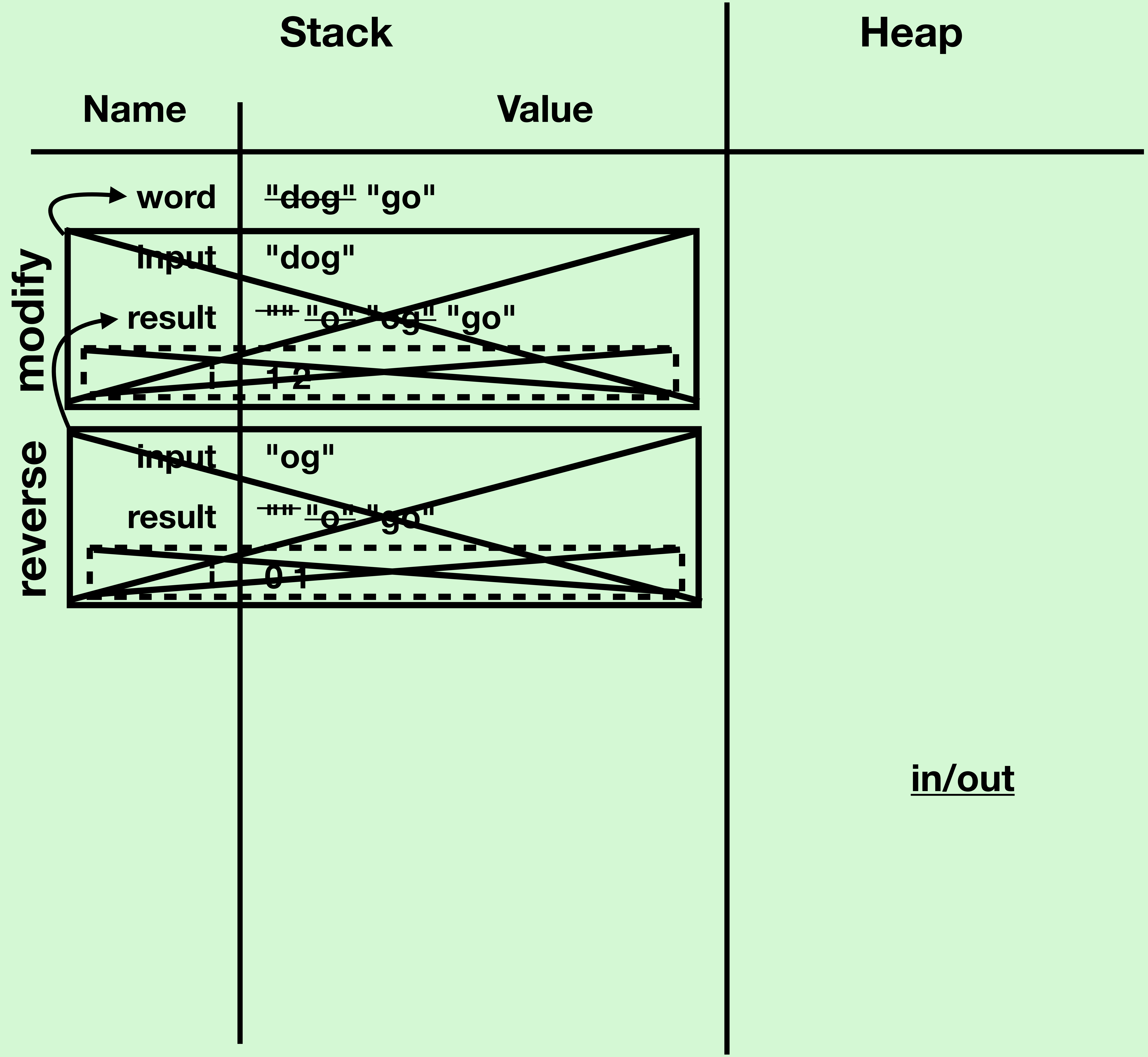
	Name	Value
	word	"dog"
modify	input	"dog"
	result	~~""~~ ~~"o"~~ ~~"og"~~ "go"
	i	1 2
reverse	input	"og"
	result	~~""~~ ~~"o"~~ "go"
	i	0 1

- Return result

- Return control back to the modify stack frame



```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```



Stack

Heap

Name

Value

word ~~"dog"~~ "go"

modify

input "dog"

result ~~""~~ ~~"o"~~ ~~"og"~~ "go"

i 1 2

reverse

input "og"

result ~~""~~ ~~"o"~~ "go"

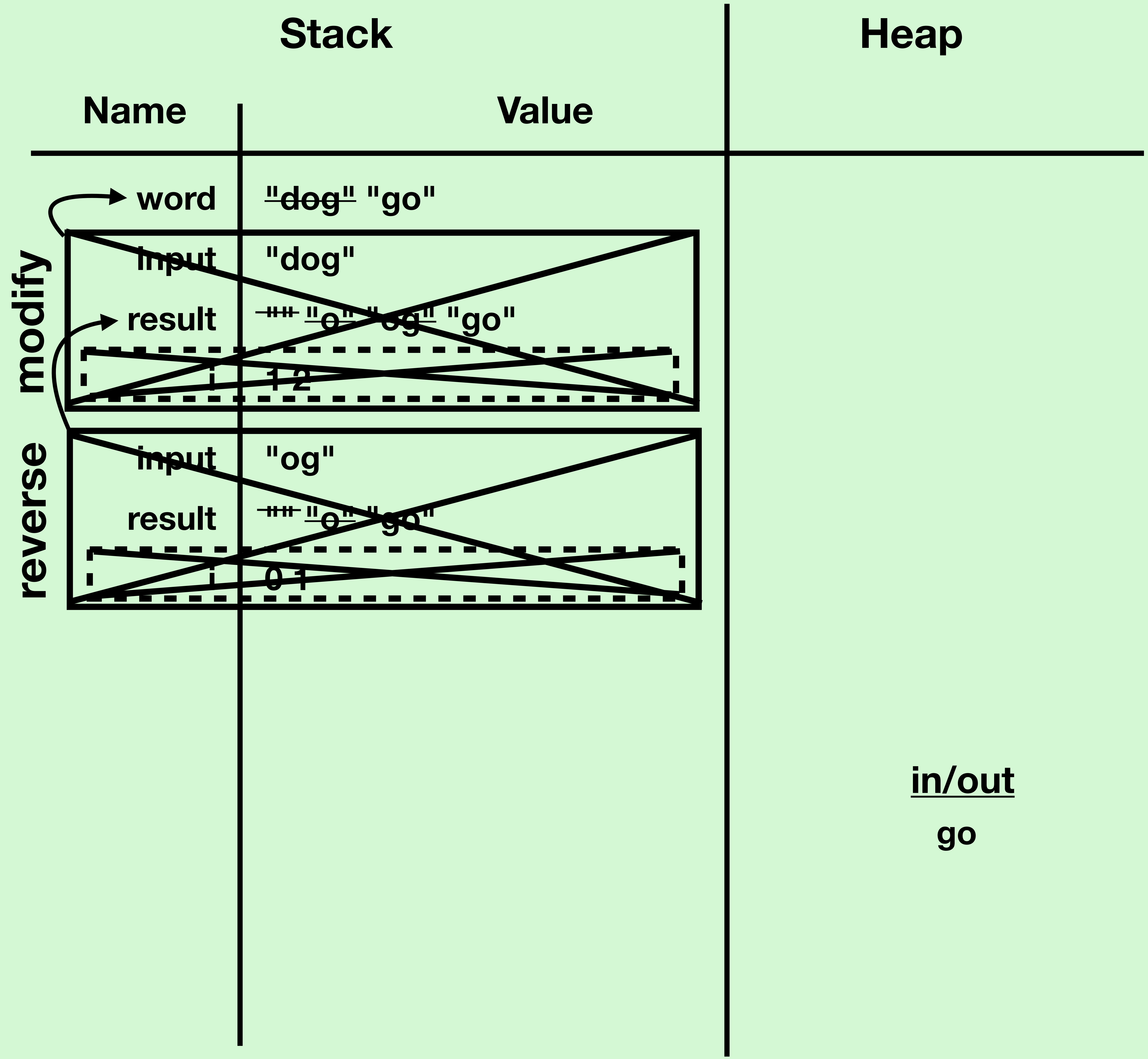
i 0 1

in/out

- Return result from the modify frame
- Return control back to the main stack frame

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Print to the screen



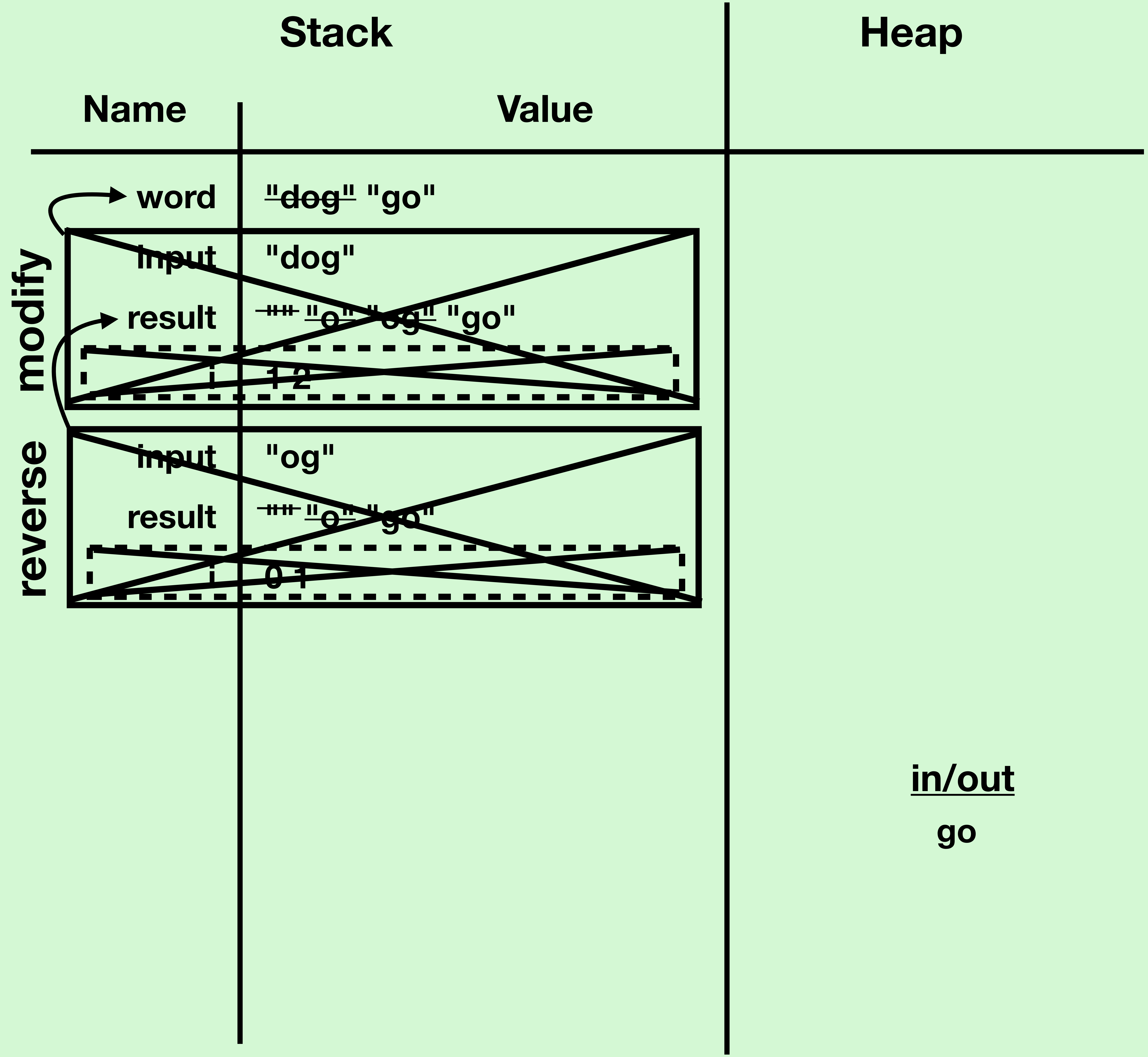
Stack			Heap
	Name	Value	
	word	~~"dog"~~ "go"	
modify	input	"dog"	
	result	~~""~~ ~~"o"~~ ~~"og"~~ "go"	
	i	1 2	
reverse	input	"og"	
	result	~~""~~ ~~"o"~~ "go"	
	i	0 1	

in/out

go

```
def reverse(input: String): String = {
  var result = ""
  for(i <- input.indices){
    result = input(i) + result
  }
  result
}

def modify(input: String): String = {
  var result = ""
  for(i <- 1 until input.length){
    result += input(i)
  }
  result = reverse(result)
  result
}

def main(args: Array[String]): Unit = {
  var word: String = "dog"
  word = modify(word)
  println(word)
}
```

- Program ends



Stack

	Name	Value
	word	~~"dog"~~ "go"
modify	input	"dog"
	result	~~""~~ ~~"o"~~ ~~"og"~~ "go"
	i	1 2
reverse	input	"og"
	result	~~""~~ ~~"o"~~ "go"
	i	0 1

Heap

in/out

go

Debugger Demo