A Swift Kickstart

Daniel H Steinberg
@dimsumthinking

More Functions

Array

```
var numbers = [5, 2, 8, 3, 9, 4]
```

Can't

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(array: [Int]) -> [Int] {
    for element in array {
        element *= 100
    }
    return array
}
```

Can

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(array: [Int]) -> [Int] {
    var tempArray = [Int]()
    for element in array {
        tempArray += [element * 100]
    }
    return tempArray
}
```

Can

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(array: [Int]) -> [Int] {
    var tempArray = [Int]()
    for element in array {
        tempArray += [element * 100]
    }
    return tempArray
}

emphasize(numbers)
```

Nothing changes

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(array: [Int]) -> [Int] {
    var tempArray = [Int]()
    for element in array {
        tempArray += [element * 100]
    }
    return tempArray
}

emphasize(numbers)
numbers
numbers = emphasize(numbers)
```

var parameter

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(var array: [Int]) -> [Int] {
    for i in 0..<array.count {
        array[i] *= 100
    }
    return array
}

emphasize(numbers)
numbers
numbers = emphasize(numbers)
```

inout parameter

```swift
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(inout array: [Int]) {
    for i in 0..<array.count {
        array[i] *= 100
    }
}

emphasize(&numbers)
numbers
```

generalize

```
var numbers = [5, 2, 8, 3, 9, 4]

func emphasize(inout array: [Int],
            modificationOf:(Int) -> Int) {
    for i in 0..<array.count {
        array[i] = modificationOf(array[i])
    }
}
```

generalize

```
func emphasize(inout array: [Int],
            modificationOf:(Int) -> Int) {
    for i in 0..<array.count {
        array[i] = modificationOf(array[i])
    }
}

func times100(number:Int) -> Int {
    return number * 100
}

emphasize(&numbers, times100)
```

Closure

```
func emphasize(inout array: [Int],
            modificationOf:(Int) -> Int) {
    for i in 0..<array.count {
        array[i] = modificationOf(array[i])
    }
}

emphasize(&numbers){ number in number * 100}
```

Types

```
var doubles = [5.0, 2.0, 8.0, 3.0, 9.0, 4.0]

emphasize(&doubles){ number in number * 100}
```

Noooooooooooo

```
func emphasize(inout array: [Int],
               modificationOf:(Int) -> Int) {
    for i in 0..<array.count {
        array[i] = modificationOf(array[i])
    }
}

func emphasize(inout array: [Double],
    modificationOf:(Double) -> Double) {
        for i in 0..<array.count {
            array[i] = modificationOf(array[i])
        }
}
```

Generics

```
func emphasize<T>(inout array: [T],
    modificationOf:(T) -> T) {
        for i in 0..<array.count {
            array[i] = modificationOf(array[i])
        }
}
```

Generics

```
func emphasize<T>(inout array: [T],
    modificationOf:(T) -> T) {
        for i in 0..<array.count {
            array[i] = modificationOf(array[i])
        }
}

emphasize(&numbers){ number in number * 100}

emphasize(&doubles){ number in number * 100}
```

Extension with Mutating Function

```swift
extension Array {
    mutating func emphasize(modificationOf:(T) -> T) {
        for i in 0..<self.count {
            self[i] = modificationOf(self[i])
        }
    }
}
numbers.emphasize{number in number * 100}
numbers
```

Extensions

```
extension Array {
     func emphasize(modificationOf:(T) -> T) -> [T] {
        var tempArray = [T]()
        for element in self {
            tempArray += [modificationOf(element)]
        }
        return tempArray
    }
}
numbers.emphasize{number in number * 100}
numbers
numbers = numbers.emphasize{number in number * 100}
numbers
```

Try this

- Create a function named sigma that accepts an Int for the start and ending numbers and an operation to perform on each number.
- Sigma should return the sum of the operation applied to each number between start and end.
- Provide a default identity value for the operation

Try this

```swift
func sigma(#start:Int,
             #end:Int,
        operation:(Int) -> Int = {a in a} ) -> Int {
    var sum = 0
    for i in start...end {
        sum += operation(i)
    }
    return sum
}

let result = sigma(start: 0, end: 4){ a in a * a}
result

let identity = sigma(start: 0, end: 5)
identity
```

https://itunes.apple.com/us/book/a-swift-kickstart/id891801923?mt=11&uo=4&at=11l56E