Swift Function Overloading

In this tutorial, we will learn about the function overloading in Swift with examples.

In Swift, two or more functions (/swift-programming/functions) may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both).

These functions are called overloaded functions and this feature is called function overloading. For example:

```
// same function name different arguments
func test() { ... }
func test(value: Int) -> Int{ ... }
func test(value: String) -> String{ ... }
func test(value1: Int, value2: Double) -> Double{ ... }
```

Here, the `test()` function is overloaded. These functions have the same name but accept different arguments.

> **Note:** The return types of the above functions are not the same. It is because function overloading is not associated with return types. Overloaded functions may have the same or different return types, but they must differ in parameters.

Example1: Overloading with Different Parameter Types

```
// function with String type parameter
func displayValue(value: String) {
    print("String value:", value)
}

// function call with String type parameter
displayValue(value: "Swift")

// function call with Int type parameter
displayValue(value: 2)
```

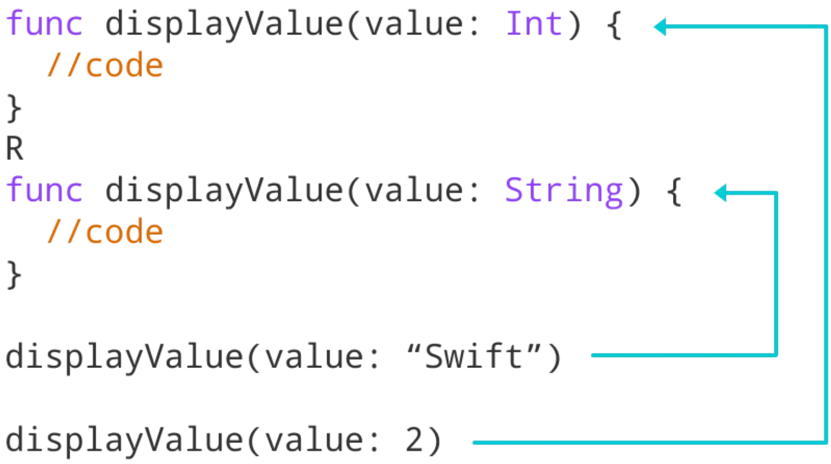
Output

```
String value: Swift
Integer value: 2
```

In the above example, we have overloaded the `displayValue()` function:

- one function has `Int` type parameter
- another has `String` type parameter

Based on the type of the argument passed during the function call, the corresponding function is called.



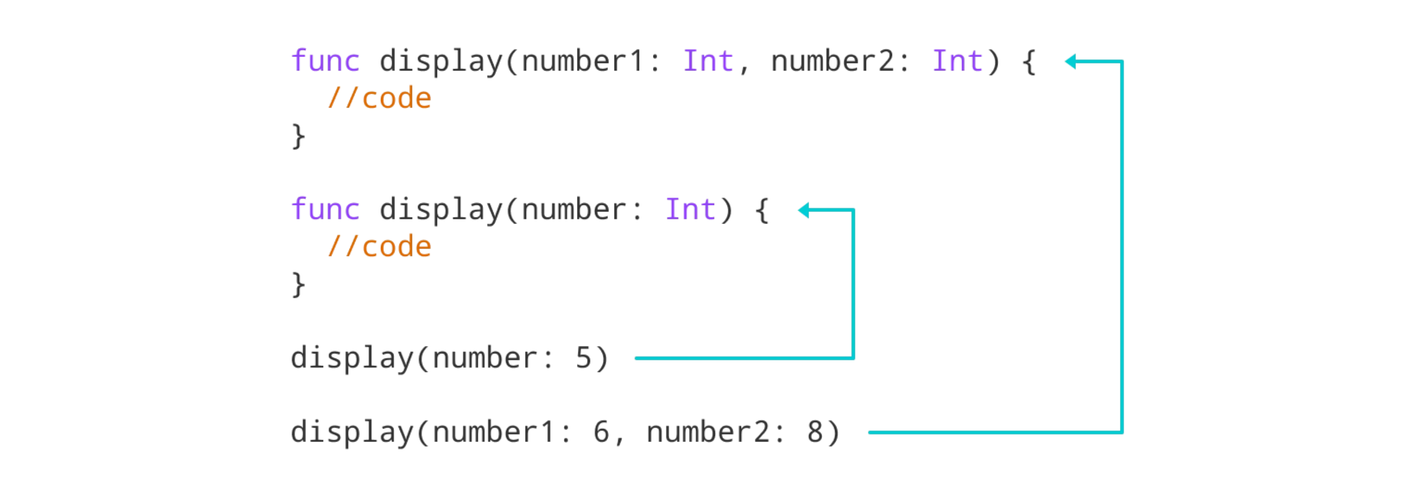
```
func displayValue(value: Int) {
  //code
}
R
func displayValue(value: String) {
  //code
}

displayValue(value: “Swift”)

displayValue(value: 2)
```

Working of Swift Function Overloading for displayValue() function

```
// function with two parameters
func display(number1: Int, number2: Int) {
   print("1st Integer: \(number1) and 2nd Integer: \(number2)")
}

// function with a single parameter
func display(number: Int) {
   print("Integer number: \(number)")
}


// function call with single parameter
display(number: 5)

// function call with two parameters
display(number1: 6, number2: 8)
```

Output

```
Integer number: 5
1st Integer: 6 and 2nd Integer: 8
```



```
func display(number1: Int, number2: Int) {
  //code
}

func display(number: Int) {
  //code
}

display(number: 5)

display(number1: 6, number2: 8)
```

Working of Swift Function Overloading for display() function

Example 3: Function overloading with Argument Label

```
func display(person1 age:Int) {
    print("Person1 Age:", age)
}

func display(person2 age:Int) {
    print("Person2 Age:", age)
}

display(person1: 25)

display(person2: 38)
```

Output

```
Person1 Age: 25
Person2 Age: 38
```

Here, two functions have different [argument labels](/swift-programming/function-parameter-return-values#argument-label) (/swift-programming/function-parameter-return-values#argument-label). And, based on the argument label used during the function call, the corresponding function is called.

Next Tutorial:
Swift Closures (/swift-programming/closures)

Previous Tutorial:
Swift Ranges (/swift-programming/ranges)

Share on:

(https://www.facebook.com/sharer/sharer.php?u=https://www.programiz.com/swift-programming/function-overloading)

(https://twitter.com/intent/tweet?text=Check%20this%20amazing%20programming/function-overloading)

Did you find this article helpful?

😃	🙁

Related Tutorials

Swift Tutorial

Swift Function Parameters and Return Values

(/swift-programming/function-parameter-return-values)

Swift Tutorial

Swift Nested Functions

(/swift-programming/nested-functions)

Swift Tutorial

Swift Initializer

(/swift-programming/initializer)

Swift Tutorial

Swift Closures

(/swift-programming/closures)