Swift Functions

In this tutorial, we will learn about the Swift function and function expressions with the help of examples.

A function is a block of code that performs a specific task.

Suppose we need to create a program to create a circle and color it. In this case, we can create two functions:

1. a function to draw the circle
2. a function to color the circle

Hence, a function helps us break our program into smaller chunks to make our code reusable and easy to understand.

In Swift, there are two types of function:

- **User-defined Function** - We can create our own functions based on our requirements.
- **Standard Library Functions** - These are built-in functions in Swift that are available to use.

Let's first learn about user-defined functions.

Swift Function Declaration

Here,

- **func** - keyword used to declare a function
- **functionName** - any name given to the function
- **parameters** - any value passed to function
- **returnType** - specifies the type of value returned by the function

Let's see an example,

```
func greet() {
print("Hello World!")
}
```

Here, we have created a function named `greet()`. It simply prints the text `Hello World!`.

This function doesn't have any parameters and return type. We will learn about return type and parameters later in this tutorial.

Calling a function in Swift

In the above example, we have declared a function named `greet()`.

```
func greet() {
  print("Hello World!")
}
```

Now, to use this function, we need to call it.

Here's how we can call the `greet()` function in Swift.

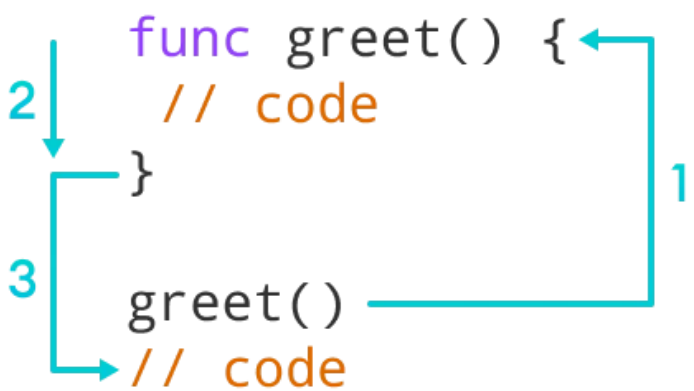
Example: Swift Function

```
// declare a function
func greet() {
  print("Hello World!")
}

// call the function
greet()

print("Outside function")
```

Output

```
Hello World!
Outside function
```

In the above example, we have created a function named `greet()`. Here's how the program works:



Working of functions in Swift

Here,

Function Parameters

As mentioned earlier, a function can also have parameters. A parameter is a value that is accepted by a function. For example,

```
// function with two parameters
func addNumbers(num1: Int, num2: Int) {
  var sum = num1 + num2
  print("Sum: ",sum)
}

// function with no parameter
func addNumbers() {
  // code
}
```

If we create a function with parameters, we need to pass the corresponding values while calling them. For example,

```
// function call with two values
addNumbers(num1: 3, num2: 4)

// function call with no value
addNumbers()
```

Here, `num1: 3, num2: 4` specifies that parameters `num1` and `num2` will get values **3** and **4** respectively.

> **Note**: To learn more about function parameters, visit Swift Function Parameters (/swift-programming/function-parameter-return-values).

```
  print("Sum: ", sum)
}

// calling function with two values
addNumbers(num1: 3, num2: 4)
```

Output

```
Sum: 7
```


In the above program, we have created a function named `addNumbers()` with parameters: `num1` and `num2`.

Working of Function with Parameters

> **Note:** The data type of function parameter should always match the data passed during the function call. Here, the data type of `num1` and `num2` is `Int`, so we have passed integer values **3** and **4** during a function call.

Swift Function Return Type

A Swift function may or may not return a value. If we want our function to return some value, we can use the **return statement**. For example,

```
func addNumbers() -> Int {
  ...
  return sum
}
```

Here, we are returning the variable `sum`. The `-> Int` in the function definition specifies that the return type of the function is an integer.

If a function returns any value, the return type of the value should be specified in the function definition (**-> returnType**). Otherwise, it will generate an error.

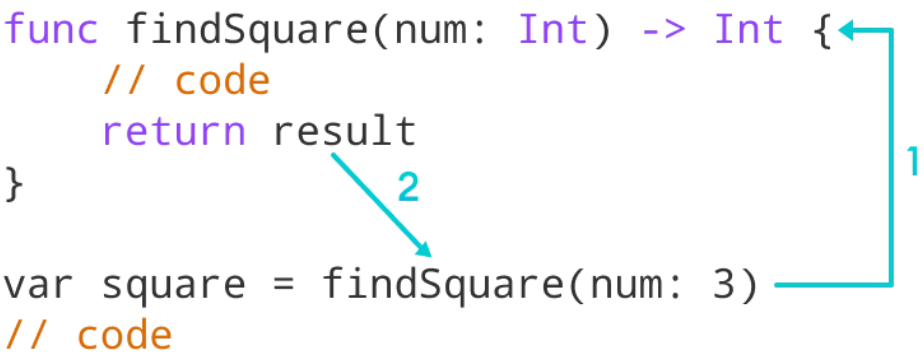
Example: Function Return Type

```
// function call
var square = findSquare(num: 3)

print("Square:",square)
```

Output

```
Square: 12
```

In the above example, we have created a function named `findSquare()`. The function accepts a number and returns the square of the number.



Working of Swift function with return values

Example: Add Two Numbers

```
// calling function with two values
var result = addNumbers(num1: 3, num2: 4)

print("Sum: ", result)
```

Output

```
Sum = 7
```

Swift Library Functions

In Swift, standard library functions are the built-in functions that can be used directly in our program. For example,

- `print()` - prints the string inside the quotation marks
- `sqrt()` - returns the square root of a number
- `pow()` - returns the power of a number

These library functions are defined inside the framework. And, to use them we must include the framework inside our program.

For example, `sqrt()` and `pow()` are defined inside the `Foundation` framework.

Example: Swift Library Function

```
print("Square Root of 25 is",squareRoot)

// pow() comptes the power
var power = pow(2, 3)

print("2 to the power 3 is",power)
```

Output

```
Square Root of 25 is 5.0
2 to the power 3 is 8
```

In the above example, we have used

- `sqrt(25)` - to compute the square root of **25**
- `pow(2, 3)` - computes the power of a number i.e. 2^3

Here, notice the statement,

```
import Foundation
```

Since `sqrt()` and `pow()` are defined inside the `Foundation` library, we need to include it in our program.

Benefits of Using Functions

1. Code Reusable - We can use the same function multiple times in our program which makes our code reusable. For example,

```
for i in 1...3{

  // function call
  var result = getSquare(num: i)
  print("Square of \(i) =",result)
}
```

Output

```
Square of 1 = 1
Square of 2 = 4
Square of 3 = 9
```

In the above program, we have created the function named `getSquare()` to calculate the square of a number. Here, the function is used to calculate the square of numbers from **1** to **3**.

Hence, the same method is used again and again.

2. Code Readability - Functions help us break our code into chunks to make our program readable and easy to understand.

Next Tutorial:
Swift Function Parameters
(/swift-programming/function-parameter-return-values)

Previous Tutorial:
Swift Dictionary
(/swift-programming/dictionary)

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 Tutorial

Swift Methods

(/swift-programming/methods)