Functions

by Ash Dreyer & Donald Pinckney

Table of Contents

Avoid Repeating Code

Why and How?

- Makes code clearer & easier to understand / edit later
- Previous ways of avoiding senseless repetition:
  - Variables / Constants
  - While / For loops
- However, the above doesn’t help with code that needs to be used at different times more than once

Example of Repeating Code

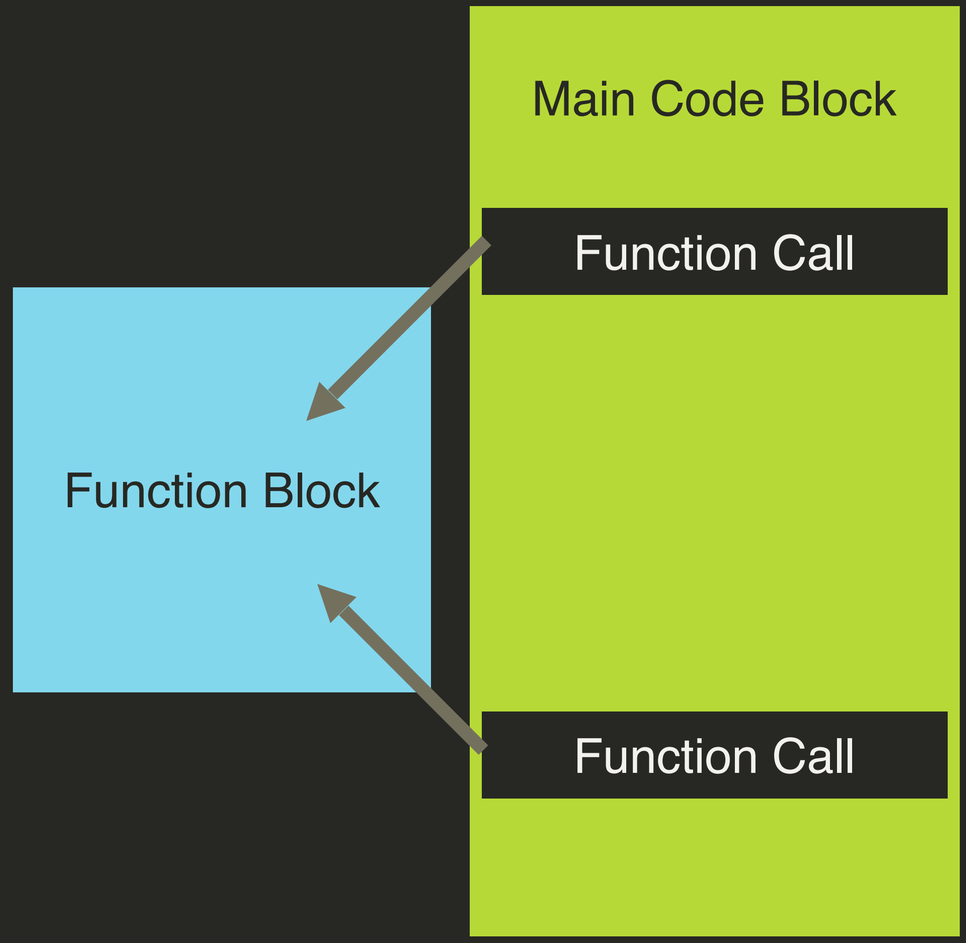
```
let data1 = [1, 2.3, 7.5, -0.2]
let data2 = [0.2, 2.9, 132.784, -0.1, 7.89]
var sum1 = 0
var sum2 = 0

for x in data1 {
    sum1 += x
}

for x in data2 {
    sum2 += x
}
```

How would you fix this?

Functions

What are Functions?

- Blocks of code that can be “called” as many times as needed
- Calling a function runs all the statements in the code block



Example

```
func print_cat() {
    print(“  /\\-/\\”)
    print(“ (=^Y^=)”)
    print(“  (>o<)”)
}

print_cat()
print()
print_cat()
```

Output:

```
 /\-/\
(=^Y^=)
 (>o<)

 /\-/\
(=^Y^=)
 (>o<)
```

Communicating with Functions

Why and How?

- Sometimes you want a function to do something slightly different based on input
- Because of scope, code that calls the function cannot use the function’s variables / constants
  - Need some way of returning values to the main code

Example

```
func greeting(forPerson: String) -> String {
    let message = “Hello, \(forPerson)!”
    return message
}

print(greeting(forPerson: “Ash”))
print(greeting(forPerson: “Donald”))
```

Output:

```
Hello, Ash!
Hello, Donald!
```

Let’s Break It Down

```
func does_something(input_int: Int) -> String {
    // Function does something with input input_int
    // Function returns something of type String
    var something = “You gave me the number
        \(input_int)!”
    return something
}
```

- These parts are needed for every function you create
- does_something is the function’s name
  - Call by name to use function’s code
  - You come up with the function name

Let’s Break It Down

```
func does_something(input_int: Int) -> String {
    // Function does something with input input_int
    // Function returns something of type String
    var something = “You gave me the number
        \(input_int)!”
    return something
}
```

- Whatever is in the function’s parentheses is the function’s parameters
  - Parameters are values taken in / used by the function
  - x is a parameter for f(x)
- input_int is the parameter name
- : Int says input_int is an Int
- When calling the function, put parameters in parentheses
  - Ex. print(does_something(input_int: 42))
  - Parameter here is 42

Let’s Break It Down

```
func does_something(input_int: Int) -> String {
    // Function does something with input input_int
    // Function returns something of type String
    var something = “You gave me the number
        \(input_int)!”
    return something
}
```

- Function returns constant of type String
- -> denotes that the function will actually be returning something
- return ends the function & gives the call the value of something
- Ex. print(does_something(input_int: 42)) prints the value of something

Back at It Again

```
func greeting(forPerson: String) -> String {
    let message = “Hello, \(forPerson)!”
    return message
}

print(greeting(forPerson: “Ash”))
print(greeting(forPerson: “Donald”))
```

Take a minute and explain what is going on here.