# Swift Loop

In Swift, loops are used to repeatedly execute a block of code as long as a specified condition remains true or for a certain number of iterations. Below are detailed notes on the various types of loops in Swift:

**for-in Loop**

The **for-in** loop is used to iterate over collections, such as arrays, dictionaries, ranges, strings, or any other sequence.

```
for item in collection {
    // Code to execute for each item
}
```

```
let numbers = [1, 2, 3, 4, 5]

for number in numbers {
    print(number)
}
```

In this example, **number** takes on each value in the **numbers** array in turn, and the print statement is executed for each value.

## while Loop

The **while** loop continues to execute a block of code as long as a specified condition is true.

```
while condition {
    // Code to execute as long as the condition is true
}
```

```
var i = 1

while i <= 5 {
    print(i)
    i += 1
}
```

Here, the loop will print the numbers from 1 to 5. The condition is checked before the loop body is executed.

### repeat-while Loop

The **repeat-while** loop is similar to the **while** loop, but it checks the condition after executing the block of code, guaranteeing that the block is executed at least once.

```
repeat {
   // Code to execute at least once and then repeatedly as long as the condition
is true
} while condition
```

```
var i = 1

repeat {
   print(i)
   i += 1
} while i <= 5
```

### forEach Loop

The **forEach** loop is a method provided by sequences like arrays and dictionaries. It allows you to iterate over each element in the collection.

```
collection.forEach { item in
   // Code to execute for each item
}
```

```
let numbers = [1, 2, 3, 4, 5]

numbers.forEach { number in
   print(number)
}
```

### Loop Control Statements

Swift provides several control statements that can change the execution flow within loops:

- **break**: Exits the loop immediately.
- **continue**: Skips the current iteration and proceeds to the next iteration.
- **return**: Exits the current function.

```
for i in 1...10 {
   if i == 5 {
       break  // Exits the loop when i is 5
   }
   if i % 2 == 0 {
       continue  // Skips the current iteration when i is even
   }
   print(i)  // Prints odd numbers less than 5
}
```