Swift Ranges

In this article, we will learn about Swift ranges and its type with the help of examples.

In Swift, a range is a series of values between two numeric intervals. For example,

```
var numbers = 1...4
```

Here,

- `...` is a range operator
- `1...4` contains values **1, 2, 3, 4**
- **1** is **lower bound** (first element)
- **4** is **upper bound** (last element)

Types of Range in Swift

In Swift, there are three types of range:

- Closed Range (/swift-programming/ranges#closed)
- Half-Open Range (/swift-programming/ranges#half-open)
- One-Sided Range (/swift-programming/ranges#one-sided)

```
// 1...4 is close range
for numbers in 1...4 {
  print(numbers)
}
```

Output

```
1
2
3
4
```

In the above example, we have created a closed range `1...4`.

Since it is a closed range, it contains all the numbers between **1** to **4** including the lower bound (**1**) and upper bound (**4**).

Here, we have used the Swift for loop (/swift-programming/for-in-loop) to access all the values in the range.

---

2. Half-Open Range

A half-open range includes all the values from the lower bound to the upper bound. However, it excludes the upper bound (last number).

It is declared using the `..<` operator. For example,

```
for numbers in 1..<4 {
  print(numbers)
}
```

In the above example, we have created a half-open range `1..<4`. Since it is a half-open range, it excludes the upper bound element **4**.

---

3. One-sided Range

We can create a one-sided range using either of the `...` or the `..<` operator.

A one-sided range contains elements up to infinite in one direction. For example,

```
let range1 = ..<2
```

Here, `...<2` is a one-sided range. It contains all elements from **2** to **-∞**. Similarly, the range

```
let range2 = 2...
```

```
// one-sided range using
// ..< operator
let range1 = ..<2

// check if -9 is in the range
print(range1.contains(-1))

// one-sided range using
// ... operator
let range2 = 2...

// check if 33 is in the range
print(range2.contains(33))
```

Output

```
true
true
```

Here, we have used `contains()` method to check if a certain number is present in the range.

> **Note:** With a one-sided range, we only set either upper bound or lower bound.

Access Array Elements Using Swift Range

We can also use the Swift range to access array (/swift-programming/arrays) elements. For Example,

Output

```
["Swift", "Java", "C"]
```

Here, the range `0...2` acts as array indices and helps us access all array elements.

Things to Remember About Swift Range

1. The lower bound value must be smaller than the upper bound value. For example,

```
// Invalid Range
3...1

// Valid Range
1...3
```

2. The lower bound and upper bound value can be negative. For example,

```
// negative lower bound
-3...1

// negative upper and lower bound
-9...-2
```

Next Tutorial:
Swift Function Overloading (/swift-programming/function-overloading)

Previous Tutorial:
Swift Recursion (/swift-programming/recursion)

Did you find this article helpful?

Related Tutorials

Swift Tutorial

Swift for-in Loop

(/swift-programming/for-in-loop)

Swift Tutorial

Swift Operator precedence and associativity

(/swift-programming/operators)

Swift Tutorial

Swift Data Types

(/swift-programming/data-types)