For-In Loops

You use the `for-in` loop to iterate over a sequence, such as items in an array, ranges of numbers, or characters in a string.

```
let names = ["Anna", "Alex", "Brian", "Jack"]

for name in names {

print("Hello, \(name)!")

}

// Hello, Anna!

// Hello, Alex!

// Hello, Brian!

// Hello, Jack!
```

You can also iterate over a dictionary to access its key-value pairs. Each item in the dictionary is returned as a `(key, value)` tuple when the dictionary is iterated, and you can decompose the `(key, value)` tuple's members as explicitly named constants for use within the body of the `for-in` loop

```
let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]

for (animalName, legCount) in numberOfLegs {

print("\(animalName)s have \(legCount) legs")

}

// cats have 4 legs

// ants have 6 legs

// spiders have 8 legs
```

You can also use `for-in` loops with numeric ranges. This example prints the first few entries in a five-times table:

```
for index in 1...5 {

print("\(index) times 5 is \(index * 5)")

}

// 1 times 5 is 5

// 2 times 5 is 10

// 3 times 5 is 15

// 4 times 5 is 20

// 5 times 5 is 25
```

f you don’t need each value from a sequence, you can ignore the values by using an underscore in place of a variable name.

```
let base = 3

let power = 10

var answer = 1

for _ in 1...power {

answer *= base

}

print("\(base) to the power of \(power) is \(answer)")

// Prints "3 to the power of 10 is 59049"
```

In some situations, you might not want to use closed ranges, which include both endpoints. Consider drawing the tick marks for every minute on a watch face. You want to draw `60` tick marks, starting with the `0` minute. Use the half-open range operator (`..<`) to include the lower bound but not the upper bound

```
let minutes = 60

for tickMark in 0..<minutes {

// render the tick mark each minute (60 times)

}
```

Some users might want fewer tick marks in their UI. They could prefer one mark every `5` minutes instead. Use the `stride(from:to:by:)` function to skip the unwanted marks.

```
let minuteInterval = 5

for tickMark in stride(from: 0, to: minutes, by: minuteInterval) {

// render the tick mark every 5 minutes (0, 5, 10, 15 ... 45, 50, 55)

}
```

Closed ranges are also available, by using `stride(from:through:by:)` instead:

```
let hours = 12

let hourInterval = 3

for tickMark in stride(from: 3, through: hours, by: hourInterval) {

// render the tick mark every 3 hours (3, 6, 9, 12)

}
```

Conditional Statements

```
temperatureInFahrenheit = 90

if temperatureInFahrenheit <= 32 {

print("It's very cold. Consider wearing a scarf.")

} else if temperatureInFahrenheit >= 86 {

print("It's really warm. Don't forget to wear sunscreen.")

} else {

print("It's not that cold. Wear a t-shirt.")

}

// Prints "It's really warm. Don't forget to wear sunscreen."
```

This example uses a `switch` statement to consider a single lowercase character called.

```
let someCharacter: Character = "z"

switch someCharacter {

case "a":

print("The first letter of the alphabet")

case "z":

print("The last letter of the alphabet")

default:

print("Some other character")

}
```

```
// Prints "The last letter of the alphabet"
```

To make a `switch` with a single case that matches both `"a"` and `"A"`, combine the two values into a compound case, separating the values with commas.

```
let anotherCharacter: Character = "a"

switch anotherCharacter {

case "a", "A":

print("The letter A")

default:

print("Not the letter A")

}

// Prints "The letter A"
```

Interval Matching

Values in `switch` cases can be checked for their inclusion in an interval. This example uses number intervals to provide a natural-language count for numbers of any size:

```
let approximateCount = 62

let countedThings = "moons orbiting Saturn"

let naturalCount: String

switch approximateCount {

case 0:

naturalCount = "no"

case 1..<5:

naturalCount = "a few"
```

```
    case 5..<12:
        naturalCount = "several"
    case 12..<100:
        naturalCount = "dozens of"
    case 100..<1000:
        naturalCount = "hundreds of"
    default:
        naturalCount = "many"
}
print("There are \(naturalCount) \(countedThings).")
// Prints "There are dozens of moons orbiting Saturn."
```