Cheatsheets / **Learn Swift**

Loops

Ranges

Ranges created by the `...` operator will include the numbers from the lower bound to (and includes) the upper bound.

```
let zeroToThree = 0...3

// zeroToThree: 0, 1, 2, 3
```

`stride()` Function

Calling `stride()` with the 3 necessary arguments creates a collection of numbers; the arguments decide the starting number to, the (excluded) ending number, and how to increment/decrement from the start to the end.

```
for oddNum in stride(from: 1, to: 5, by:
2) {
  print(oddNum)
}

// Prints: 1
// Prints: 3
```

`for-in` Loop

The `for - in` loop is used to iterate over collections, including strings and ranges.

```
for char in "hehe" {
  print(char)
}

// Prints: h
// Prints: e
// Prints: h
// Prints: e
```

`continue` Keyword

The `continue` keyword will force the loop to

```
for num in 0...5 {
```

move on to the next iteration.

```
  if num % 2 == 0 {
    continue
  }
  print(num)
}

// Prints: 1
// Prints: 3
// Prints: 5
```

`break` Keyword

To terminate a loop before its completion, use the `break` keyword.

```
for char in
"supercalifragilisticexpialidocious" {
  if char == "c" {
    break
  }
  print(char)
}

// Prints: s
// Prints: u
// Prints: p
// Prints: e
// Prints: r
```

Using Underscore

Use `_` instead of a placeholder variable if the variable is not referenced in the `for - in` loop body.

```
for _ in 1...3 {
  print("Olé")
}

// Prints: Olé
// Prints: Olé
// Prints: Olé
```

`while` Loop

A `while` loop accepts a condition and continually executes its body's code for as long as the provided condition is `true`.

If the condition is never `false` then the loop continues to run and the program is stuck in an infinite loop.

```
var counter = 1
var stopNum = Int.random(in: 1...10)

while counter < stopNum {
  print(counter)
  counter += 1
}

// The loop prints until the stop
condition is met
```