Swift Cheat Sheet

Variables and Constants

```swift
/// You can change values with variables
var variable: DataType

/// Values of constants cannot be changed
let constant: DataType
```

Control Flow

If Statements

```swift
/// Use IF statements for simple conditions
/// with only a few possible outcomes
if condition {
  // Runs the code here if condition is true
  // Otherwise falls through the else conditionals
} else if anotherCondition {
  // Runs the code if anotherCondition is true
} else {
  // If none passed, runs the code here=
}
```

Switch Statements

```swift
/// Use SWITCH for more complex conditions
/// with multiple possible outcomes
switch someValue {
case outcome1:
  // Respond to outcome1
case outcome2, outcome3:
  // Respond to either outcome2 or outcome3
default:
  // Otherwise, do something else
}
```

Loops

For-in loop

```
/// Use for-in loops to iterate over a sequence
/// For in with arrays
let fruits = ["Apple", "Banana", "Coconut"]
for fruit in fruits {
  print(fruit)
}

/// For in using the range operator
/// lower...upper
for variable in 1...10 { // 1 to 10
  print(variable) // prints 1 to 10
}
/// lower..<upper
for variable in 0..<10 { // 0 to 9
  print(variable) // prints 0 to 9
}
```

While loop

```
/// Use while loops to perform a set of code
/// until a condition becomes false
while conditionIsTrue {
  doSomething()
}

var count = 0
while count < 2 {
  print(count)
  count += 1 // Increment count by 1
}
```

Repeat-While

```
/// Performs a single pass through the code
/// before considering the loop's condition
repeat {
  doSomething()
} while conditionIsTrue

var count = 0
repeat {
  print(count)
  count += 1 // Increment count by 1
} while count < 2
```

Operators

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

```
1 + 2 // equals 3
2 - 1 // equals 1
1 * 2 // equals 2
5.0 / 2.5 // equals 2.0
```

Conditional Operators

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

```
a == b // Is `a` equal to `b`?
a != b // Is `a` not equal to `b`?
a > b // Is `a` greater than `b`?
a < b // Is `a` less than `b`?
```

Nil-coalesing Operator

```
let defaultColor: Color = .red
var userPickedColor: Color? // defaults to nil

// Will default to `.red` if user didn't pick a Color
var colorToUse = userPickedColor ?? defaultColor
```

Range Operators

Operator	Description
a...b	Closed Range Operator
a..<b	Half-open range Operator

```
1...10 // A range from 1 to 10
0..<10 // A range from 0 to 9
```

Declaring Types

Reference Types

Classes

```
/// Use classes if you want to pass objects by reference
/// or need features such as inheritance or type casting
class MyClass: SuperClass {
  var storedProperty: Type

  init(storedProperty: Type) {
    self.storedProperty = storedProperty
  }
}
```

Declaring Types

Reference Types

Classes

```swift
/// Use classes if you want to pass objects by reference
/// or need features such as inheritance or type casting
class MyClass: SuperClass {
  var storedProperty: Type

  init(storedProperty: Type) {
    self.storedProperty = storedProperty
  }
}
```

Value Types

Structures

```swift
/// Use struct if you want to model data or pass objects
/// by value
struct Model {
  var storedProperty: Type
}
```

Enumeration

```swift
/// Use enumeration to model a range of values
enum Compass {
  case north, south, east, west
}
```

Protocols

```swift
protocol Printable {
  var property: Type { get }
  func print()
}
```