Introduction to Swift

Dr. Sarah Abraham

University of Texas at Austin
CS329e
Spring 2020

What is Swift?

- Programming language for developing OSX, iOS, WatchOS, and TvOS applications
- Best of C and Objective-C without compatibility issues
- Easier to use
- More flexible to program
- Cleaner syntax

An Apple Language

- Provides seamless access to Cocoa frameworks (the interface to OS X)
    - Systems programming language for the lower level operating system
- Has mix-and-match interoperability with Objective-C
    - Will not be using Objective-C in this class
- Treats everything as an object

Main Function

- `main` function is often the starting point for execution of code
- Swift does not have a `main` function
- Entry point is globally-scoped code
    - Code outside of any function
- Command line applications only have code at global-scope in `main.swift`
- iOS applications have entry point in `AppDelegate.swift`

Data Types

- Data types define what kind of thing a variable is
- Built-in data types:
  - Int, UInt, Float, Double
  - Bool
  - Character, String
  - Optional

Integer Data Types

- Integer types can be signed or unsigned
    - Signed ints have negative to positive range
    - Unsigned ints have positive range
- Size of range determined by number of bits
    - Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64
- Int and UInt default to 32-bit or 64-bit depending on platform

Floating Point Data Types

- Allow for decimal place values
- Float is 32-bits
- Double is 64-bits
- Size of floating points affects its precision

Boolean Data Type

- Must be `true` or `false`
- Comparison operators can evaluate boolean expressions:
  - >, <, ==, >=, <= and `!=`
  - `&&`, || and `!`

String Data Type

- Values must be explicitly converted to another type
- `String(value)` will convert `value` from its initial data type to a String type
- String interpolation allows conversion to a String as well
- `\(value)` will convert `value` from initial data type to a String type

Declaring a Type

- Data type annotation assigns a type to a variable
- Colon followed by type:

```
var name:String = “Yossarian”
```

- Data types not required and can be inferred

```
var name = “Yossarian” //name must be a
string
```

Optionals

- Work with values that might be missing
- Optional value contains a value or contains `nil`
- Question mark after type marks the value as optional

```
var optionalInt:Int? = 9
```

- Unwrapping an optional returns the underlying value
    - Done with an exclamation point after the optional

```
        optionalInt!
```

Variables

- Used to store values for a program
- Swift has constant and mutable variables
- Constants (immutable) cannot change during runtime
- Mutables can be changed during runtime

- `var` declares a mutable variable

```
var numApples = 3
```

- `let` declares an immutable variable

```
let numApples = 3
```

- Example:

```
let numApples = 3  //numApples is now 3
numApples = 5 //throws an error
var numOranges = 3  //numOranges is now 3
numOranges = 5  //numOranges is now 5
```

Initializing a Constant

- Constants do not need to be initialized when declared
    - That is, you do not have to specify the value immediately
- The data type must be defined in this case:

```
let numApples:Int

numApples = 3
```

Control Flow

- Statements that dictate the order of the code that is executed at runtime
- Conditional statements (`if` and `switch`)
- Loop statements (`for-in`, `while` and `do-while`)

If-Statements

- Do not require parenthesis (but they're okay, and I'd encourage you to use them for readability)

```
let n = 20

if (n < 10) { print(“n is small”) }

else if (n > 100) { print(“n is big”) }

else { print(“n is in the middle”) }
```

Switch Statements

- Provides cases for all potential choices and runs all that are true

```
let n:UInt = 5

switch n {

  case 0: print(“n is less than 1”)

  case 1: print(“n is 1”)

  default: print(“n is greater than 1”)

}
```

For-Loops with Ranges

- Range operators preferred over C-style syntax
- Range can be inclusive:

```
  for i in 0...5 { //do task}
```

- Range can be exclusive:

```
  for i in 0..<5 { //do task}
```

- What's the difference?

While and Repeat-While Loops

- Standard C-style loops
- While loops look like this:

```
var index = 0

while index < 3 {index+=1}
```

- Repeat-While loops looks like this:

```
var index = 0

repeat { index+=1 } while index < 3
```