Seneca College of Applied Arts & Technology

PLAIN HTML VERSION: lecture1.161.txt.html
Xcode Project Archive: lecture1.161.zip

```
/*
file:    lecture1.161.xcodeproj
author:  Danny Abesdris
date:    January 13, 2016
purpose: MAP523AA/DPS923AA lecture #1
         Introduction to Apple programming concepts, Xcode 7, and the
         Swift programming language.
*/
/*
Welcome to iOS programming!
Course:  MAP523/DPS923
         Wed. Lecture T3131 1:30 - 3:15
         Fri. Lab     S2114 8:00 - 9:50

Instructors:
Danny Abesdris (CPA/D)
Office: A4003 (Newnham)
Web:    https://scs.senecac.on.ca/~danny.abesdris

Peter McIntyre (BSD)
Office: T2081 (Seneca@York)
Web:    https://scs.senecac.on.ca/~peter.mcintyre

Tools you will need to get started:
1. An Intel based Mac (Mac Mini, MacBook Air, Macbook Pro, iMac, or Mac Pro)
   running OS X 10.10 (Yosemite) or OS X 10.11 (El Capitan), and
2. Apple's developer tools (XCode v7.0+ which is freely available from the App Store).
3. An iPhone 4+, iPad, or iPod Touch running iOS 8.0+

For students that do not have their own Mac, various labs at Seneca (S2114 / Library) a
equipped with modern Intel-based Macs running OS X 10.10+ and XCode 7.0+ that can
be used for development.

But first, a little bit of History!
Apple was founded in 1976 by Steve Jobs and Steve Wozniak in the basement of 2066 Crist
in Los Altos California.
http://cicorp.com/Apple/garage/

The first Macintosh computer was released in January 1984 featuring a graphical user
interface and a mouse. It featured a Motorola 68000 CPU running at 7.83 MHz with 128K o
in memory, a 400Kb single-sided 3.5-inch floppy disk drive and cost $2495 U.S.

Steve Jobs left Apple in 1985 and founded NeXT Inc. to develop computers for use prima
colleges. In 1988, NeXT licensed Objective-C (developed by Brad Cox and Tom Love at St
from StepStone and went on to develop the NeXTSTEP operating system and Interface Build

In 1997, Apple bought NeXT which gave rise to Apple's Mac OS X (now simply called OS X,
and is a descendant of the NeXTSTEP operating system). Apple's mobile operating system
derived from OS X and is used in the iPhone, iPod, iPod Touch, and Apple TV.

The Swift programming language was created by the Apple Developer Tools team and releas
```

```
at Apple's WWDC on June 2, 2014.
Swift is a modern language with simpler syntax than Objective-C and includes popular
programming features adopted from languages like Objective-C, Java, C#, Ruby, Python,
and others.
Swift is a multi-paradigm, compiled programming language created for iOS, OS X, watchO
Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large
existing Objective-C code written for Apple products.
Swift is intended to be "safer" than Objective-C and also more concise. It is built wi
LLVM (Low Level Virtual Machine) compiler framework included in Xcode 6 and later and
Objective-C runtime, which allows C, Objective-C, C++ and Swift code to run within a s
program!

Sample code differences:
Objective-C:

NSString *str = @"hello,";
str = [str stringByAppendingString : @" world"];
NSLog("%@", str);

Swift:
var str = "hello,"
str += " world"
print(str)

To compile a swift program from the command line, you can use the swiftc command.
>swiftc myProg.swift

Apple Development Overview:
Q. What is Xcode?
A. Xcode is an Integrated Development Environment (IDE).
   Xcode contains all of the tools you will need to build and test iOS apps
   including: editors, compilers, linkers, syntax checkers, cryptographic signers,
   resource compilers, debuggers, simulators, performance analyzers, and more!

Q. What is an SDK?
A. An SDK is a Software Development Kit. It consists of the a collection of files that
   supply a compiler with what it needs to build an app for a particular operating syst
   (like iOS 9.0 or OS X 10.10, or 10.11 for example).
   Xcode supplies an SDK to build iOS apps as well as an SDK to build OS X apps.
   An SDK consists of one or more "frameworks".

Q. What is a framework?
A. A framework (also known as an Application Programming Interface or API),
   consists of a bundle of files in a folder that your app needs to use a particular
   segment of the operating system.
   For example, all of the functions, constants, classes, and resources needed to draw
   things on the screen are in the "Core Graphics" framework.
   The "AVFoundation" framework contains the classes that let you record and playback
   audio. To know where you are geographically, you would require using the
   "CoreLocation" framework, etc.

Foundation Framework:
The Foundation frameworkin both Cocoa and Cocoa Touchincludes class NSObject
for defining object behavior. Foundation also has classes for basic types, storing data
working with text and strings, filesystem access, calculating differences in dates and
times, inter-app notifications and much more.

AppKit Framework (for OS X):
Cocoas AppKit framework is for developing the GUIs of OS X apps. AppKit provides
controls, windows, menus, buttons, panels, text fields, dialogs, event-handling capabi
gesture support and more.
```

```
UIKit Framework (for iOS):
Cocoa Touchs UIKit framework is similar to AppKit, but optimized for developing iOS ap
GUIs for mobile devices. UIKit includes multi-touch interface controls that are appropr
for mobile apps, event handling for motion-based events, event handling for sensors an



A comprehensive list of iOS frameworks can be found on Apple's Developer website here:
https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneO

As of Dec. 2015, the iOS 9.2 Foundation Framework currently consists of:
2542 classes, 25807 methods, 7652 functions, 20715 constants, and more!
(source: Dash app for OS X).

The OS X 10.11.2 Foundation Framework currently consists of:
2708 classes, 36293 methods, 21797 functions, 36373 constants, and more!
(source: Dash app for OS X).

Q. What is Cocoa and Cocoa Touch?
A. Cocoa is Apple's native object-oriented application programming interface (API) for
   OS X operating system.
   Cocoa Touch is Apple's API for iOS which includes gesture recognition, animation,
   and different sets of graphical control elements used on Apple devices such as:
   an iPhone, iPad, iPod Touch and Apple TV.
   Cocoa and Cocoa Touch are implemented using the Objective-C language.

Some of the Cocoa and Cocoa Touch frameworks include:
Foundation Kit, Application Kit (Cocoa) or UIKit (Cocoa Touch),
Core Data (Cocoa), GameKit and MapKit (Cocoa Touch).

Review:
IDE:          Integrated Development Environment (Xcode).
SDK:          Software Development Kit (iOS 9 SDK).
API:          Application Programming Interface (A published set of classes and functio
              that describe how your app can use a particular service).
Cocoa:        Apple's API for OS X.
Cocoa Touch: Apple's API for iOS.
Objective-C: Apple's objected oriented language based on C.
Swift:        Apple's new (2014) language implemented to be easier to learn and safer t
              than Objective-C.



Becoming an iOS developer? What do you need?
In order to become an "official" iOS developer, you will need to join Apple's iOS
Developer Program in order to do any of the following:
1. Sell, or give away your apps through Apple's App Store.
2. Gain access to Apple's Developer Forums and other resources.
3. Give your apps to people directly (outside of the App store).
4. Develop apps that use the Game Kit, in-app purchases, push notifications,
   or similar technologies.
5. Test your apps on a real iOS service.

The cost of joining is $99, however you DO NOT have to join to build, test,
and run your apps in Xcode's simulator!
In addition, free registration also allows you to search through all of Apple's
published documentation, download example projects, read technology guides,
technical notes, and much more!

To learn more, you can visit: http://developer.apple.com

Introduction to the Swift Programming Language (current version 2.1 (as of 01/2016):
```

```
Swift Key Language Features:

Type inference: Swift can infer the type of variable being declared based on the
variable's or constant's initializer value in spite of the fact that
the Swift language itself is strongly typed.

Tuples:         Tuples are an aggregate collection of values.

Closures:       Swift supports closures (anonymous functions that some languages refer
                lambdas). Closures can be manipulated as data, assigned to variables, p
                to functions as arguments, and returned from functions. Several of the
                Standard Library's global functions receive closures as arguments (eg.
                the sort function that receives a closure for comparing two objects to
                determine their sort order).

C langauge example:

#include <stdio.h>
#include <stdlib.h>

// qsort prototype:
// void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const

int cmpfunc (const void *a, const void *b) {
   return ( *(int *)a - *(int *)b ); // casting generic pointers a and b to (int *)
}

int main( ) {
   int i;
   int values[ ] = { 88, 56, 100, 2, 25 };

   printf("Before sorting, the list is: \n");
   for(i = 0 ; i < 5; i++) {
      printf("%d ", values[i]);
   }

   qsort(values, 5, sizeof(int), cmpfunc);

   printf("\nAfter sorting, the list is: \n");
   for(i = 0 ; i < 5; i++) {
      printf("%d ", values[i]);
   }
   return(0);
}

switch:         Unlike switch statements in other C-based languages (which can test onl
                integral expressions), Swift's switch statement can test values of any
                Also, its cases are much more flexible than those in other languages (y
                have cases for individual, sets and ranges of values) as well as multip
                conditions.

Optionals:      Swift optionals enable you to define variables and constants that MIGHT
                have a value. The language provides mechanisms for determining whether
                has a value and, if so, obtaining that value. Optionals work for any St
                and are not relegated to pointers that point to an object (as is the us
                in Objective-C).
                Useful in situations where some piece of data may not be present (like
                name, searching for a match in an array, etc.).

Dictionary:     Swift provides built-in support for storing key-value pairs of data by
```

```
                Dictionary data type.

Array, String,
Dictionary:     Swift's Array, String, and Dictionary types are value types that are i
                as structs and are therefore copied when assigned to variables or cons
                However, the Swift compiler typically optimizes value-type copy operat
                performing them only when required.

Array bounds
checking:       In Swift, a runtime error occurs if you attempt to access an Array out

Functions:      Swift allows the capability of having functions return multiple values
                or different types via Tuples.

Generics:       Allows the use of writing code once by using place holders to handle i
                on different data types. The placeholders are replaced with actual dat
                the generic function is called or when a generic object is created.
                Swift's Array and Dictionary types are generics.

Operator
Overloading:    As in C++, Swift makes it possible to define functions that overload e
                operators to work with new types as well as defining new operators.

Overflow:       Swift automatically checks for arithmetic overflow and generates a run
                when any overflow occurs.

String
interpolation:  Enables the ability to build String objects by inserting variable, con
                expression values into placeholders directly in String literals.

Nested
functions:      Functions can be defined within other functions and are callable withi
                of the existing function and can be returned from that function for us

Interoperability:
                You can combine Swift and Objective-C in the same app, so you can use
                code portions of existing apps without rewriting all your Objective-C
                the Cocoa and Cocoa Touch APIs are still written in Objective-C.

Swift Error Prevention Features:
a. Curly braces required around EVERY control statement's body.
b. Unlike Objective-C, Swift does not include pointers.
c. The = operator does NOT return a value and a compilation error results if the =
is used in place of the comparison operator ==.
d. Semicolons are optional (needed if coding multiple statements on a single line).
e. Parentheses ( ) around conditions in control statements are optional.
f. Variables and constants MUST be initialized before they can be used.
g. Arithmetic operations checked for overflow.
h. Swift does NOT allow for implicit conversion between numeric types.
i. Array subscripts (indices) bounds are checked at runtime, generating an error
if the bounds are violated.
j. Automatic memory management eliminates most memory leaks (achieved by way of ARC).


Xcode Overview:
The Xcode integrated development environment (IDE), supports Swift, Objective-C, C++ a
Xcodes editor supports syntax coloring, auto-indenting, auto-complete and more.

Xcode Key Features:
a. Playgrounds: A playground is an interactive Swift coding environment that evaluates
                and displays results as updates are made, without the need to create a
```

```
b. REPL:        Xcode introduces yet another way to experiment with Swift in the form
                Read Eval Print Loop, or REPL. To get started, launch Terminal.app (fo
                /Applications/Utilities) and type swift at the prompt in OS X Yosemite,
                or xcrun swift in OS X Mavericks. You'll then be in the Swift REPL:
                Welcome to Apple Swift version 2.1.1 (swiftlang-700.1.101.15 clang-700
                1>

                All you need to do is type Swift statements and the REPL will immediat
                Expression results are automatically formatted and displayed along wit
                the results of both variable and constant declarations. Console output
                interactive session:

c. Interface Builder, Storyboarding, and Auto Layout:
                Xcode's Interface Builder (IB) enables you to create and add functiona
                application's GUI using drag and drop techniques. You create the GUI's
                that graphically map the paths a user can take through your app (inclu
                and their transitions). By default, new iOS and OS X appls use IB's Au
                allow you to create GUI's that function based on the device chosen.
d. iOS Simulator:
                The iOS 8 or 9 simulator enables you to test your app on your Mac with
                deployment to an actual physical device (like an iPhone or iPad).
*/

import Foundation

// To create a new project in Xcode 7: Xcode->New Project->OSX->Application->Command L
// Enter the application, organization, and organization identifier names and choose S
// as the language
// Next, chooose a location to save your project, then choose "Create".

// constants are declared using the "let" keyword
// using the "let" keyword to store data is considered "best practice"
// because it allows the compiler to perform optimizations that it wouldn't
// ordinarily perform

let PI = 3.14159

// displaying a line of output with print( )

print("Hello Swift World!")
print("Am I going to pass this course? ")
print("Yes? \u{1F600} No? \u{1F622} Maybe? \u{1F914}") // String literals have the Swi
                                                       // Unicode literals are permitt

// Unlike other C-based programming languages, Swift statements are not required to en
// a semicolon (;), though you can use them if you wish.

// decalring a variable (mutable data type) with the keyword var
let str1 : String  = "Hello Playground!" // In Swift, a String has type: struct String
                                         // and is an arbitrary Unicode string value.
                                         // Here, str1 is EXPLICITLY decalred

let str2 = " and welcome to Swift programming"
                                         // In this example, str2's data type is INFER

var str3 = "In French that would be: Bonjour, Playground "

                                         // calling an explicit initializer init( )
                                         // comparable to a C++ constructor
str3 += String(count: 3, repeatedValue: Character("\u{1F601}"))
print(str3)
```

```
                                        // operator overloading with strings
var str4 = str1 + str2
print(str4)

var str5 = ""
str5 += (str4 + " " + str3)
print(str5)

// Other String initializers:
/*
init(_ c: Character)
   Construct an instance containing just the given Character.

init(_ cocoaString: NSString)
   Construct an instance containing just the given Character.

init(contentsOfFile path: String, encoding enc: NSStringEncoding)
   [Foundation] Produces a string created by reading data from the file at a given pat
   interpreted using a given encoding. (see example 1a. below)

init(contentsOfURL url: NSURL, encoding enc: NSStringEncoding)
   [Foundation] Produces a string created by reading data from a given URL interpreted
   using a given encoding. Errors are written into the inout error argument.
   (see example 1b. below)

init(count: Int, repeatedValue c: Character)
   Construct an instance that is the concatenation of count copies of repeatedValue.

init(stringLiteral value: String)
   Create an instance initialized to value.

init?(CString: UnsafePointer<CChar>, encoding enc: NSStringEncoding)
   [Foundation] Produces a string containing the bytes in a given C array, interpreted
   according to a given encoding.
*/

/*
// example 1a.
do {
   // this code enclosed around a do { } block and requires the keyword "try"
   // because the init( ) class method throws an exception
   var stringFromFile = try String(contentsOfFile: "/Users/danny.abesdris/Desktop/qsor
   if(!stringFromFile.isEmpty) {
      print(stringFromFile)
   }
}

// example 1b.
var baseURL = NSURL(string: "https://scs.senecac.on.ca/~danny.abesdris/mrna.txt")
do {
   var stringFromURL = try String(contentsOfURL: baseURL!, encoding: NSUTF8StringEncod
   if(!stringFromURL.isEmpty) {
      print(stringFromURL)
   }
}
*/

// String instance variables:
/*
var capitalizedString: String
```

```
   [Foundation] Produce a string with the first character from each word changed to the
   corresponding uppercase value.

var endIndex: Index
   The "past the end" position in self.characters.

var isEmpty: Bool
   true iff self contains no characters.
var lowercaseString: String
var uppercaseString: String

*/


// using a method
str4 = str4.capitalizedString // NOTE: this method does NOT have a side-effect
//       (i.e. does not change the object)
print(str4)

// String instance methods:
/*
mutating func append(c: Character)

mutating func appendContentsOf(other: String)

func componentsSeparatedByString(separator: String) -> [String]
   [Foundation] Returns an array containing substrings from the String that have
   been divided by a given separator.

func containsString(other: String) -> Bool
   [Foundation] Returns true iff other is non-empty and contained within self
   by case-sensitive, non-literal search.

mutating func insert(newElement: Character, atIndex i: Index)
   Insert newElement at index i.

mutating func removeAtIndex(i: Index) -> Character

mutating func removeRange(subRange: Range<Index>)

mutating func replaceRange(subRange: Range<Index>, with newElements: String)

func stringByAppendingString(aString: String) -> String

func substringFromIndex(index: Index) -> String
   [Foundation] Returns a new string containing the characters of the String
   from the one at a given index to the end.

func substringToIndex(index: Index) -> String
   [Foundation] Returns a new string containing the characters of the String
   up to, but not including, the one at a given index.

func substringWithRange(aRange: Range<Index>) -> String
   [Foundation] Returns a string object containing the characters of the
   String that lie within a given range.

mutating func write(other: String)
   Append other to this stream.

func writeToFile(path: String, atomically useAuxiliaryFile: Bool,
   encoding enc: NSStringEncoding) throws
```

```
   [Foundation] Writes the contents of the String to the URL specified by url
   using the specified encoding.
*/

// Swift numeric types:
/*
Int8 and UInt8
Int16 and UInt16
Int32 (also simply Int native Word size) and UInt32
Int64 and UInt64
Float  (size: 32-bit, precision: <=  6 decimal digits)
Double (size: 64-bit, precision: <= 15 decimal digits)
*/

// Converting Strings to Numbers and formatting
let firstText : String = "123456"
let secondText : String = "789"

let a = Int(firstText)
let b = Int(secondText)

print("a is: \(a!)") // NOTE: a and b are implicitly created as OPTIONALS and must the
print("b is: \(b!)") //       be explicitly unwrapped using the !

print(String(format: "%@ string: %9X %-8.2f", "HEX", 123456, 99.1234), separator: ":",
```