EECS 581

M, F 4:00

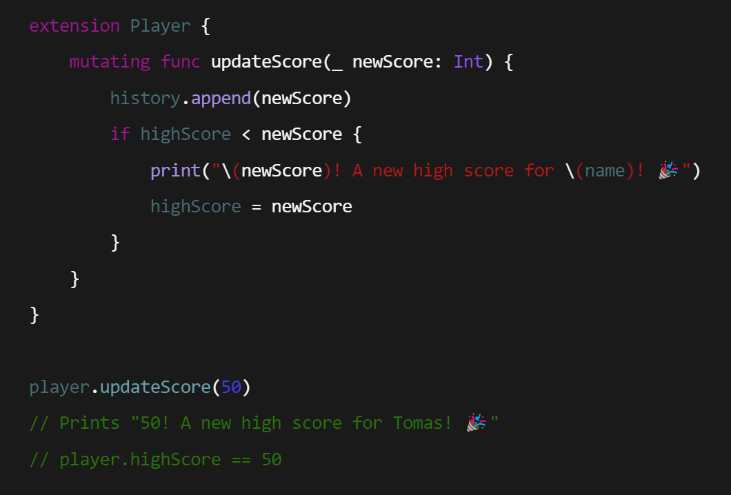
Swift Research and Installation

Research below:

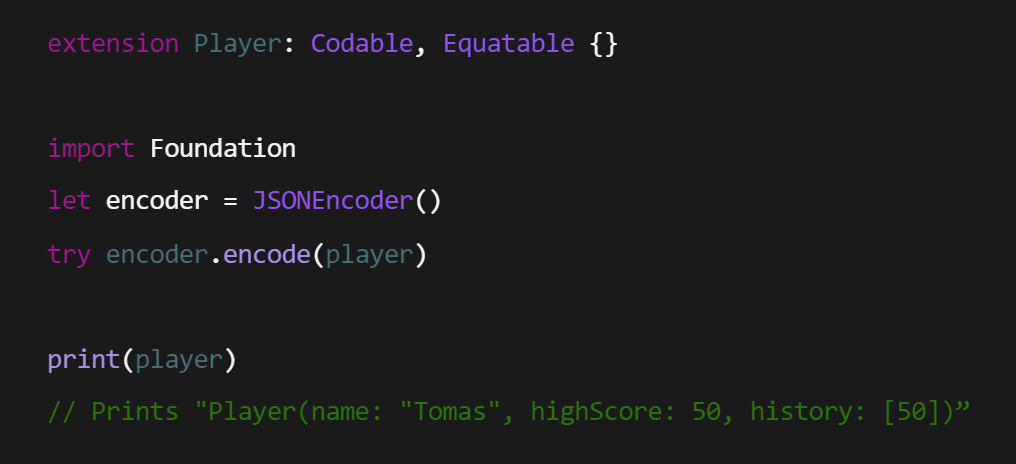
* Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. and the open-source community. Swift is a robust and intuitive programming language for building apps for iOS, Mac, Apple TV, and Apple Watch. It’s designed to give developers more freedom than ever. Swift is easy to use and open source, so anyone with an idea can create something incredible.
* Features of Swift
  + Generics that are powerful and simple to use
  + Protocol extensions that make writing generic code even easier
  + First class functions and a lightweight closure syntax
  + Fast and concise iteration over a range or collection
  + Tuples and multiple return values
  + Structs that support methods, extensions, and protocols
  + Enums can have payloads and support pattern matching
  + Function programming patterns such as map and filter
  + Built-in error handling using try/catch/throw
* Swift if fast and powerful
  + Using a high-performance LLVM compiler technology, Swift code is transformed into optimized machine code that gets the most out of modern hardware. The syntax and standard library have also been tuned to make the most obvious way to write your code also perform the best whether it runs in the watch on your wrist or across a cluster of servers. Swift is a successor to both the C and objective-C languages. It includes low-level primitives such as types, flow control, and operators. It also provides object-oriented features such as classes, protocols, and generics, giving developers the performance and power they demand.
* Examples of code in Swift
  + Declare new types with modern, straightforward syntax. Provide default values for instance properties and define custom initializers.



* Add functionality to existing types using extensions, and cut down on boilerplate with custom string interpolations



* Quickly extend your custom types to take advantage of powerful language features, such as automatic JSON encoding and decoding



* Perform powerful custom transformations using streamlined closures.



Installation steps:

1. Download and install the compiler and other required features and ensure Swift is added to your $PATH
2. On Linux install required dependencies
   1. Git (used by Swift Package Manager)
   2. Python (used by debugger – lldb)
   3. Windows SDK (provides Windows header and import libraries)
   4. Visual Studio (provides the Visual C++ SDK/Build Tools for additional headers)
3. Enable Developer Mode
   1. In order to develop applications, particularly with the Swift Package Manager, you will need to enable developer mode.
4. Start up a new command prompt and install the Python library six
5. Install swift – Swift toolchain will be installed at %SystemDrive%\Library\Developer\Toolchains\unknown-Asserts-development.xctoolchain. A compatible Swift SDK will be installed at %SystemDrive%\Library\Developer\Platforms\Windows.platform\Developer\SDKs\Windows.sdk
6. Verify Swift version by running –version in command line

* Using REPL
  + If you run swift repl without any other argument, you’ll launch the REPL, an interactive shell that will read, evaluate, and print the result of any Swift code you enter
  + You can use the up-arrow and down-arrow keys to cycle through previous lines entered into REPL. This allows you to make a slight change to previous expression without replying the entire line
  + REPL can automatically suggest functions and methods that can be used in a particular context
  + All the functionality of Swift is available to you from the REPL, from writing control flow statements to declaring and instantiating structures and classes.
  + You must ensure that Python 3.7 is available in the path
* Using Package Manager
  + Purpose: Swift package manager provides a convention-based system for building libraries and executables, and sharing code across different packages.
  + Creating a package -> **$ mkdir Hello** **$ cd Hello** (create directory Hello)
  + Every package must have a manifest file called Package.swift in its root directory. You can create a minimal package named Hello using: **$ swift package init**
  + By default, the init command will create a library package directory structure

A picture containing text

Description automatically generated

* + You can use ***swift build*** to build a package. This will download, resolve, and compile dependencies mentioned in the manifest file *Package.swift*.
  + To run the tests for a package, use: ***swift test***
  + Building an executable
    - A target is considered as an executable if it contains a file named ***main.swift***. The package manager will compile that file into a binary executable.
    - Example: **$ mkdir Hello $ cd Hello $ swift package init –type executable $ swift run Hello $ swift build**
  + Working with Multiple Source Files
    - Create a new file in the Sources/Hello directory called ***Greeter.swift***, and enter the following code

Text

Description automatically generated with medium confidence

* + - * sayHello(name: ) function takes a single *String* argument and prints our “Hello” greeting before, substituting the word “World” with the function argument
    - Open *main.swift* again, and replace the existing contents with the following code

Text

Description automatically generated with medium confidence

* + - * *Main.swift* reads from the command line arguments rather than using a hardcoded name as before. And instead of invoking print(\_:) directly, *main.swift* now calls the sayHello(name: ) method. Since the method is part of the Hello module, no import statement is needed.
* Using the LLDB Debugger to run Swift programs step-by-step, set breakpoints, and inspect and modify program state
  + Use print **(p)** command to inspect the value of the parameter
    - Print command can evaluate Swift expressions as well
  + Use the backtrace **(bt)** command to show the frames leading to the function
  + Use the continue **( c)** command to resume the process until breakpoint is hit again