# Learning Outcome 2

Use REPL initially – just execute command swift from a terminal in Mac or Linux

:help to get help in the REPL

:quit to exit the REPL

## Output

Use print

print ( “Hello World” )

Can use escape characters like in Java:

print( "This is a sentence.\n This is another one. This is a tab\t space")

print( "This is a \"sentence\"" )

## Variables

Swift is strongly typed, the data may be inferred but once a variable is created it is typed. There are variables and there are constants.

Variable uses keyword var.

var number = 12

Constants use keyword let.

let PI = 3.14

let ∏ = 3.14

(In Linux do Ctrl-Shift-u – then the code for the character – pi is 3c0 )

(In Mac command+ctrl+space is supposed to open the emoji and symbols, but if doesn’t work just go to Edit – Emoji and Symbols, There is a search functionality in this window when it is opened, try searching for delta)

to include a variable value in print do:

print(“Your age is \(age)” ) where age is a variable name or you can do println( age )

Variable names cannot start with a number. Otherwise, any unicode character can be used for names of constants or variables so we could do: (On Mac go to Edit – Emoji and Symbols to get Unicode stuff)

let Δ = 2 //Delta symbol represents difference or change -delta is 394

Sample code:

let Δ = 2

println( "Change is \(Δ)")

When data type is not included like two examples above it is inferred based on the initialization value. So delta is an int – if we try to allocate a floating point number it is will be an error. To explicitly specify the data type use : then data type. For example:

var number : Double

You can declare multiple variables/constants on same line.

var numb1, numb2, numb3 : Double

Same runnable code:

var number : Double

var numb : Double = 12

var number1, number2, number3 : Double

//Note cannot initialize to a value when multiple variables are included

Data Types Available:

Int – standard type for integers. Has native word size of current platform. Or you can force certain size by using Int16, Int32, or Int64. Unsigned integers also supported: Uint, Uint16, etc. **Properties** provided – min and max.

Float and Double – Float 32 bit floating number, Doble 64 bit floating point number. No min or max properties.

Bool – true/false

When inferring data types Swift always chooses Double for values with a decimal and Int for ones without.

var num1 : 12

var num2 : 12.2

So num1 will be Int data type and num2 will be Double.

Note: Data types are really objects – they can have properties and methods associated with them.

Practice: Write Swift code to:

* display the min and max value for Int.
  + Int.max, .min (or UInt8, UInt16, UInt32 for unsigned)
  + Try assigning -5 to a UInt16
* Create a constant using any unicode character – get creative! Declare and display it.

## Named Types

Any type that is given a name. This could be user defined classes, structures and enumerations. Swift provides String, Character. (And Number? Not sure about Number). Swift also provides Array and Dictionary

# Character

let dollarSign = "\u{24}"

let blackHeart = "\u{2665}"

let sparklingHeart = "\u{1F496}"

print( dollarSign )

print( blackHeart )

print( sparklingHeart )

Characters use double quotes too:

let letter : Character = "D"

But if you try to do this

let letter : Character = "DD"

you will get an error since Character can only hold one single character.

Also, cannot use numbers like you can in Java to set character values. Instead use something like this if you want to set a character based on its decimal value:

let letter : Character = Character( UnicodeScalar( 65 ) )

println( letter )

This will print the letter A.

# String

var name : String

var myName = “Sharon”

Sample Code: (Keep for later with loops?)

var name : String

var myName = "Sharon"

println( myName )

var letter : Character

letter = "S"

println( letter )

By default var l = “M” will be String data type, not Character.

Doing letter = “SM” will be an error since it was a Character data type.

Strings are really an array of Characters. So you can treat like an array.

1. Get length of String using property on Characters array

print( name.characters.count ) - get length of String by getting count of Character array.

//Print individual characters in the array

print( Array ( name.characters ) )

1. Reverse String using method on Characters array

//Method to reverse - based on character array

let reversed = String( name.characters.reversed() )

print(reversed )

1. Determine if String is empty using isEmpty property on String. Note: This is much more efficient than using count property on Character array – since that would require iterating through the array.

print( "Is String empty? ")

print( name.isEmpty )

let emp : String = ""

print( emp.isEmpty )

1. Comparing Strings using ==, < , > , <=. >=

let cafe1 = "Cafe\u{301}"

let cafe2 = "Café"

print( cafe1 )

print( cafe2 )

print(cafe1 == cafe2)

The Unicode code point "\u{301}" modifies the preceding character to include an accent, so "e\u{301}" has the same canonical representation as the single Unicode code point "é".

Basic string operations are not sensitive to locale settings. This ensures that string comparisons and other operations always have a single, stable result, allowing strings to be used as keys in Dictionary instances and for other purposes

//Compare using > < >= <=

var a = "apples"

var b = "bananas"

var aCap = "Apples"

print( a <= aCap )

print( a < b )

print( a <= b )

print( a >= b )

print( a == aCap )

## Compound Types

A compound type has no name and is defined in Swift itself. There are two types of compound types: Tuples and Function Types. Further information on Tuples and Function Types later.

## Optional Values

A variable can be optional meaning it may hold a value or it may not. This is quite useful in some cases. For example, when reading input – we may or may not get a value that we can use when inputing an int for example. Option variables are indicated by the ?. For example:

var opValue : Int?

If an optional variable does not hold a value then it is nil.

var opValue : Int?

print( opValue )

opValue = 10

print( opValue )

Version 3.0 and up give a warning message for the above. To get rid of you must do:

print( opValue as Any )

Forced unwrapping can be used to check if an optional variable has a value:

let num = opValue! //forced unwrapping

println( num )

Forced unwrapping will case a run time error if the optional variable is nil so you should generally check that it is not nil first:

if opValue != nil

{

let num = opValue!

}

Optional binding – use an if statement to try to bind a regular variable to an optional variable:

if let numb = opValue

{

print( "numb is assigned a value of \(numb) " )

}

else

{

print( "opValue was nil ")

}

Try other way.

## The ?? Operator

The ?? operator allows handling of optional values by providing a default to use if nil:

Print(someOpVar ?? “hello”)

## Input

Does not work in playground or online playgrounds. Need to create a swift file and compile.

var name : String

print( "Enter your name" )

name = readLine()!

print( "Input was " + name )

print("Enter a number: " )

let inStr = readLine()!

var numb = Int( inStr )

print( name + " is \( numb ) " )

print( "enter a floating point number" )

var fNumb = Double( readLine()! )

print("You entered \(fNumb)" )

Try entering a non integer value/non number value and see what happens!