Introduction to Swift Programming

- First Swift Program: Printing a line of Text - using playground or swift project
- Data Types, Constants and Variables
- Type Inference
- Strings and strings Interpolation
- Adding integer values
- Arithmetic – Arithmetic Overflow Checking and Operator Precedence
- Decision Making : The if conditional statement and switch.. Case
- Loops – for repetitive tasks
- Optionals – A type that can contain nil if no value provided.

Introduction to Swift Programming

- **A First Swift Program: Printing a line of Text**

print(" Welcome to Swift Programming"); // use of print function, part of Standard Swift Library

Note: *Swift does not have a main function or method. In Xcode project, file that contains the entry point for a Swift project must be named – main.swift (extension - .swift)*

Any globally spaced statements in main.swift – that is statements that are not written inside a function, method or type definitions serve as app's entry point.

- **Commenting your code** – Comments in code are ignored by compiler

// -- single line comment

/* your multi-line comments

here … */

Introduction to Swift Programming

- Function **print** displays a line of text to the standard output
- It is a Swift Standard Library function.
- Where standard output appears depends on the type of program and where you execute it.
  - If you execute print in the playground, the result displays in playground's editor window
  - If you execute any app from a Xcode project, output appears in the Debug area at the bottom of the Xcode window.
  - If you execute an macOS app outside of Xcode, the result is sent to a log file that you can view in the console app
  - If you excute an iOS app on a device, the result is sent to a log file that you can view in Xcode's **Devices** window

Introduction to Swift Programming

- **Escape Sequences** – special characters
  - **\n** (new line) , **\t** (tab), **** (backslash, **\r** (carriage return, **\"** (double quotes), **\'** (single quote), **\0** (null character), **\u{n}** (Unicode character, n is between one and 15 hexadecimal digits)
- Semicolons are not required in Swift, but if you place more than one statement on the same line, they must be separated by semicolon
- Executing the Application/Project
  - If it is playground, it executes immediately when you load it.
  - If it is a Xcode project (.swift file), then you see output in Debug window
- Compilation and syntax errors
  - As you write your code in a playground or in a .swift file that is part of Xcode project, the compiler continuously compiles your code.
  - Any compilation errors are indicated by stop-sign-shape symbol displayed to the left of lines of the code in which error occurs.

Types, Constants and Variables

- **Data Types**
  - **Integers** – denoted by keyword - **Int**
  - Examples:

```
var number1 = 14;   // var keyword indicates a variable. Makes use of type inference


var student_id = 1001;   // var keyword indicates a variable. Makes use of type inference
var account: Int = 1234;   // var keyword indicates a variable. Here type is Integer.
print( number1);
print(student_id );
print(account );
print( 10 + 20); print( 30 – 5);  print(  5 * 6); // If you are writing them in the same line.
```

Types, Constants and Variables

- **Types**
  - **Integers** – denoted by keyword - **Int**
  - Various Integers Types:

Int : Default – 4 bytes or 8 bytes Int8: 8 bits (I byte)

Int16: 16 bits (2 bytes) Int32: 32 bits (4 bytes)

Each type's minimum and maximum values can be determined with its **min** and **max** properties.

For examples: Int.min and Int.max for type Int.

```
print(" The maximum Int value is \(Int.min)")
print(" The maximum Int value is \(Int.max)")
```

Types, Constants and Variables

- Floating-point Data Types (conforms to IEEE 754)
- **Float** – This is used to represent a 32-bit floating-point number and numbers with smaller decimal points. For example, 3.14, 0.1, and -23.15.

```
var salary: Float = 1234.40
var grades:Float
print(salary )
```

- **Double** – This is used to represent a 64-bit floating-point number and used when floating-point values must be very large. For example, 3.14159, 0.1, and -273.158.

```
var balance: Double = 234.567
var totalSales = 23456.45  // It is implicitly double
print(balance)
```

Types, Constants and Variables

- String Data Type
- **String** – This is an ordered collection of characters. For example, "Hello, World!“

```
var courseName = “Swift”
var campusName: String = “Davis”
var  description = “This course is about Swift Programming”
```

Types, Constants and Variables

- **Character** − This is a single-character string literal. For example, "C“

  let exclamationMark: Character = "!" // let makes it constant

- **Boolean** − This represents a variable that can hold true or false value. Keyword - **Bool**

  var mallOpen: Bool = true

  var hasPostOffice: Bool = false

- **Optional** − This represents a variable that can hold either a value or can have no value (nil)

  var courseName:String // Here type is String

  print(courseName) // It will give you an error. Variable must be initialized

  var courseName:String? // String followed by question mark will make it a String Optional

  print(courseName) // It will give you nil. No compilation error

  courseName = “Mobile”

print(courseName) // It will print – optional(“Mobile”)

print(courseName!) // It will print – Mobile. With “!” , it is unwrapped. It is called unwrapping

Constants – use keyword let

Examples:

```
let name = "iOS Programming"
let SIN: Int = 100
let pi: Float = 3.14
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
let numberOfCampuses = 4
    numberOfCampuses += 2
```

Note: Here compiler will issue an error because constants won't vary.

Automatic Arithmetic Overflow Checking

Examples:

```
let y: Int8 = 120
let z = y + 10   // this will give you an error due to over flow because compiler infers the type of z to be Int8
```

- -

Using an overflow operator:

```
let y: Int8 = 120
let z = y &+ 10
print("120 &+ 10 is \(z)")
```

For other operators: &- for minus, &* for multiplication etc.

Converting between Integer Types

Examples:

```
let a: Int16 = 200
let b: Int8 = 50
let c = a + b // so this is not allowed, compile type error
                // There is no automatic conversion in Swift
 let c = a + (Int16) b  // Explicit type conversion
```

- -

Examples of String Interpolation

To perform string interpolation, insert a backslash(\) followed by a set of parenthesis containing the constant, variable, expression or literal value that you would like to insert at that position in the String literal

//**Example #1:**

```
import Cocoa
let name = "Sujeet Lohan"
print("Welcome to Swift Programming, \(name)")
```

//Example #2:

```
import Cocoa
var name = "Banyan tree"
var age:Int = 200
let country = "India"
print("Life of \(name) is more than \(age) years in \(country)")
```

Use of keyword let for constants

```
// Addition program that displays the sum of two integers
import Cocoa
let number1 = 45 // keyword let declares a constant
let number2 = 72
let sum = number1 + number2
print("number1 = \(number1)")
print("number2 = \(number2)")
print("sum = \(sum)")
```

Data Types, declaration, Initialization and Displaying Values

```
11  // Variable declaration and initialization
12
13  var firstName:String = "Sujeet"
14  var lastName:String
15  lastName="Lohan"
16  let employeeID = 1234 // constant by use of let keyword
17  var salary:Double = 2500
18  var gender:Character = "M"
19  var activeStatus:Bool = true
20  let taxRate:Float = 12.5
21
22  //... Displaying the above values
23  print("\nDisplaying the values:")
24  print(firstName)
25  print(lastName)
26  print(employeeID)
27  print(salary)
28  print(gender)
29  print(activeStatus)
30  print(taxRate)
```

Decision Making and if conditional statement

```
//Compare integers using if statements,  relational operators and equality operators
import cocoa
let number1 = 1000;  let number2 = 2000
if number1 == number2  {   print("\(number1) == \(number2)") }
if number1 != number2 {
        print("\(number1) != \(number2)") }
if number1 < number2  {
        print("\(number1) < \(number2)") }
if number1 > number2  {
        print("\(number1) > \(number2)") }
if number1 <= number2  {
        print("\(number1) <= \(number2)") }
if number1 >= number2  {
        print("\(number1) >= \(number2)") }
```

Decision Making and if ...else conditional statement

```
import cocoa
 var population: Int = 5500
 var message: String
if population < 10000  {
        message = "\(population) is a small town" }
 else {
      message = "\(population) is pretty big" }
  print(message)
```

Decision Making and nested if conditional statement

```
//Use of nested if
import cocoa
 var population: Int = 5500
 var message: String
if population < 10000  {
        message = "\(population) is a small town" }
 else {
         if population > 10000  && population < 50000  {
        message = "\(population) is a medium town" }
       else {
         message = "\(population) is pretty big" }
  }
        print(message)
```

Introduction to Swift Programming – Logical Operators

```swift
// Logical AND (&&) operator

var seniorFemales = 0
var gender = "Male"
var age = 65

if (gender == "Female") && (age >= 65) {
    seniorFemales += seniorFemales
}

//... Logical OR (||) operator

var semesterAverage = 91
var finalExam = 97

if (semesterAverage >= 90) || (finalExam >= 90) {
    print("Student grade is A")
}

//... Logical NOT (!) Operator

if !(age > 65) {
    print("Age is less than or equal to 65")
}

if age <= 65 {
    print("Age is less than or equal to 65")
}

```

Introduction to Swift Programming - Switch

- **Switch**

Basic syntax of a switch statement:

```
switch aValue {
    case someValueToCompare:
        // Do something to respond
    case anotherValueToCompare:
        // Do something to respond
default:
    // Do something when there are no matches
}
```

Introduction to Swift Programming - Switch

```
//Listing 5.1 Your first switch  //Create a new Xcode project/playground called Switch and set up a switch
import Cocoa
        var statusCode: Int = 404
        var errorMessage: String
switch statusCode {
        case 400:
                errorMessage = "Bad request"
        case 401:
                errorMessage = "Unauthorized"
        case 403:
                errorMessage = "Forbidden"
        case 404:
                errorMessage = "Not found“
default:
         errorMessage = "None“   }  // end switch
```

Introduction to Swift Programming - Switch

```
//Listing 5.1 Your first switch  //Create a new Xcode project/playground called Switch and set up a switch
import Cocoa
let semesterCode: Character = "W"
switch semesterCode {
case "S", "s":
    print("Summer Semester")
case "W", "w":
    print("Winter Semester")
case "F", "f":
    print("Fall Semester")
default:
    print("Incorrect Semester Code")
}
```

Introduction to Swift Programming – Loops – for loop

```swift
// Examples based on for loops

//.. Example 1.0
for count in 1...5 {
    print("\(count) ")
}

//... Example 2.0
for values in 11...20 { // supposed to loop 10 times

    print("\(values)  ")

} //end for

//... Example 3.0
// you can omit variable name if you are not using it's value
for _ in 1...5 {

    print("* ")
}

//... Example 4.0

for num in 1..<5 {
    print("\(num) ")
}

```

Introduction to Swift Programming – Loops – for loop

```swift
//... Example 5.0
for counter in stride(from: 11, through: 1, by: -2) {
    print("\(counter) ")
}

//... Example 6.0 - Displays values from 10 to 50 by the increment of 10
for numbers in stride(from: 10, to: 50, by: 10) {
    print("\(numbers) ")
}
//... Example 8.0
// printing odd numbers between 1 and 10
for items in 1...10
{
    if (items % 2) == 0
    {
        print("\(items) ")
    }
} // end for

//... Example 9.0
// finding sum of even numbers between 10 and 20
var sum:Int = 0
for intValues in 10...20
{
    if (intValues % 2) == 1
    {
        sum += intValues
    }
} // end for
print("\(sum) ")
```

Introduction to Swift Programming – Loops break and continue

```
//... Example 10.0
// use of break statement
for count in 1...10 { // supposed to loop 10 times
    if (count == 5) {
        break // terminates loop if count is 5
    }

    print("\(count)  ")
}

//... Example 11.0
// Example: continue statement exiting a for...in statement
for count in 1...10 { // supposed to loop 10 times
    if (count == 5) {
        continue // move to next iteration of the loop
    }

    print("\(count)  ")
}
```

Introduction to Swift Programming – Loops while

```
var product = 3


while (product <= 100)
{
    product *= 3
}


print(product)
```

Introduction to Swift Programming – Loops repeat .. while

```
repeat {

print("\(counter)  ")

  counter += 1

} while counter <= 10
```

Introduction to Swift Programming - Switch

- https://developer.apple.com/swift/
- https://docs.swift.org/swift-book
- **Swift for Programming by Deitel and Deitel**
- **Swift Programming 2nd Edition by Matthew Mathias and John Gallagher**
- **IOS Programming 6th Edition by Christian Keur and Aaron Hillegass**
- https://www.tutorialspoint.com/swift/index.htm
- www.udemy.com
- **Edureka.co**
- **Code Repositories on GitHub**
- **Resources on internet**