Getting Started with Swift

Key points-

- To create variable:
  var abc = “hello” //by default assigns data type as String - Type Inference
  abc = 123 //this will give an error - TypeSafe language

- For multi - line strings use “””
- String Interpolation:
  var score = 40
  print("score = \(score)”)   // use /() to substitute value

- Constants:
  instead of var use the keyword **let**
  let Taylor = “swift”

- Type annotations:
  Can be explicit about the type of data rather than relying on Swift’s type inference
  let album: String = “Red”
  let today: Bool = true
  let height: Double = 1.78
  let year: Int = 1999

- Arrays:
  let eve = [“yesterday”, “today”, “tmrw”]
  print(eve[1])   // today
  * to use type annotations, arrays are written in brackets [String],[Int] etc.

- Sets:
  1. Items aren’t stored in random order //Unordered
  let colors = Set(["red","green","yellow"])
  print(colors). // ["yellow", "red", "green"]

  2. All items must be unique
  let colors = Set(["red","green","yellow","red"])
  print(colors) // ["green", "red", "yellow"]

- Tuples:
    1. Can’t add or remove items from a tuple //fixed size
    2. Can’t change the type , will always have the same type as when created
    3. Can access items through numerical positions or by naming them
     Accessing non-existent items - violation

    - Ex: var name = (first:”Steve”, last:”Smith”)
         // name.0 (OR) name.first   would print Steve
        var name = (first:"Steve", age:18) //will now give redeclaration
        error

- Dictionaries :

```
let days = [
        “Monday” : 1,
        “Tue” : 2,
        “Wed” :  3
        ]
// days[“Wed”]
```

   * to use type annotations, [String:Int], [String:String]

    days[“Thur”] // o/p : nil
   If we don’t wish to specify the key, use - days[“Thur”, default: “Unknown”]   //now we will get O/P as Unknown instead of nil

- Empty Collections:

    1. Empty dict
    var teams = [String : String]()
    //can add entries later
    teams[“Rudd”] =“Furlenco”

    OR

    var teams = Dictionary<String,String>()

    2. Empty Array
    var res = [Int]()

    OR

    var res = Array<Int>()

    3. Empty set

```
var words = Set<String>()
var num = Set <Int>()
```

- <u>Enumerations:</u>

```
enum Res {
      case success
      case failure
      }
let result1 = Res.failure
```

- <u>Enum associated values :</u>

```
enum Act {
      case bored
      case run(destination : String)
      case talk(topic : String)
      case sing(volume : Int)
}
let talk2 = Act.talk(topic: “bagels” )
```

- <u>Enum raw values:</u>

Swift will automatically assign each of the constituents a number starting from 0.

```
let result = Res( rawValue : 1 )

enum Res {
case success = 3
case failure
}

// Res( rawValue : 4 ) = failure
```

- <u>Operators :</u>

1. Arithmetic: + , - , * , /, %
   Note : + can be used to join strings , arrays
2. Comparison : ==, != , < , <= , > , >=
3. Combine conditions : &&, ||
4. Ternary :

```
let first = 11
let sec = 10
print( first == sec ? “Same” : “Diff” )
```

5. Range : 1 .. <5 contains 1,2,3,4

- <u>Conditions:</u>

1. If-else

```
let roll = 1
let rick = 2
if rick + roll == 3 { //do something }
else { //do something}
```

2. Switch

```
switch dice {
case “one” :
case “two” :
default :
        }
```

3. For

```
let count = 1 … 10
for num in count {
        print(“ number is \(num) “ )
        }
```