--------------------------------------**Syntex/Grammer of Swift** ----------------------------------------------

Alphabets, symbols, Alphabets, symbols

Words keywords, identifiers

Sentence statement

Paragraph function / block

Chapter/article class/program

Book project/App

**Operators. ->**

1. Arithmetic -….. +, -, /, \*, %(mode operator to find remainder)

applicable on only numbers

2. Relational/Comparison - >, <, >=, <=, ==(exact equal to), !=(Not equal to)

applicable on all data types

3. Logical operation -…. &&(AND), || (OR), !(NOT)

A. && - gives true if both input true otherwise false

B. || - gives true if atleast one input is true else false

C. ! – gives opposite of the input value(it only works for Bool value-True/false)

applicable on only Bool

**Data Type -** data types **specify the type of data that can be stored inside a variable**

Arithmetic operators are **used to perform mathematical operations like addition, subtraction, multiplication etc.**

The character data type is used to represent a single-character string

The string data type is used to represent textual data.

An integer data type is used to represent a whole number with no fractional component.

A boolean data type is used to represent logical entities.

A float data type is used to represent a number with a fractional component.to store the number with less precision (up to 6 decimal places)

A double data type is also used to represent a number with fractional components. Double supports data up to 15 decimal places.

(keyword) Any(accept all data types)

(keyword) Int - 45, 989,3546,257

Numeric (keyword) Long - 6456767,467887578,

Data Types (keyword) Float - 5.7, 2312.56, 8875.34

(keyword) Double - 78.224466, 6568.225315

String - “Ankita”, “Indore”, “I love indore”

Character - single letter or symbol - “a”, “m”, “@”, “p”, “!”

Bool - true/false

|  |  |  |
| --- | --- | --- |
| Character | "s","a" | a 16-bit Unicode character |
| String | "hello world!" | represents textual data |
| Int | 3, -23 | an integer number |
| Float | 2.4, 3.14, -23.21 | represents 32-bit floating-point number |
| Double | 2.422342412414 | represents 64-bit floating-point number |
| Bool | true and false | Any of two values: true or false |

--------------------------------------------------------------------------------------------------------------------------

Common Formulas : -

1. Area = length\*width

2. Average = sum of variables /total variables

3.Simple Interest => SI =principle\*rate\*Time/100

4. Area of circle => AOC = Pi\*r\*r

--------------------------------------------------------------------------------------------------------------------------

**If else / if : -** for checking condition

If (condition)

{

CODE

}

else

{

CODE

}

----------------------------------------------------------------------------------------------------------------

**Switch Case** : - for selection

Switch (name)

{

Case 1 : CODE

Case 2 : CODE

Case 3……any : CODE

Default : CODE

}

Use fallthrough to move to remaining case – fallthrough

--------------------------------------------------------------------------------------------------------------------------

**While Loop** : - to check Condition in repeated manner

While(Condition)

{

CODE

}

Break : - to stop current Loop

--------------------------------------------------------------------------------------------------------------------------

**For Loop : -** loop on counting bases (decide how many time loop has to run)

For name in CountStart . . . CountEnd

{

CODE

}

.reversed() : - is used to reverse the code

stride : - stride(from:1, to: 10, by:2) is used to run code by gape (by:gape)

terminator : - print(“message”,terminator:”message”)to print next statement/value on same line

--------------------------------------------------------------------------------------------------------------------------

**Function : -** group of statement

func name ()

{

CODE

}

name()

**Function with parameters : -**

func name (name:DataType,name:DataType,……..)

{

CODE

}

name(name:value,name:value,……)

**Function with Return Value : -**

func name (name:DataType,name:DataType,……..) -> DataTpe

{

CODE

return name

}

name(name:value,name:value,……)

--------------------------------------------------------------------------------------------------------------------------

**Nested Function : - function inside function (nested)**

func name ()

{

CODE

func name()

{

CODE

}

name()

and so on……….

--------------------------------------------------------------------------------------------------------------------------

**Closure Expresion** : - function without prefir func , it can pass in argument or initialiser

var closure() -> Void

{

CODE

return name

}

name

----------------------------------------------------------------------------------------------------------------

**Tuple** : - count start from 0 index

var name : (Int:Float:String) = (1,5.6,”Hello”)

//access part of tuple

print(t1.0) //1

print(t1.1) // 5.6

print(t1.2) //Hello

//modify value of tuple

t1.0 = 7

--------------------------------------------------------------------------------------------------------------------------

**Array** :- similer type data collection (count start from 0 index)

----------------------------------------------------------------------------------------------------------------

Var name : [DateType] = [Data,Data,Data,…]

//access part of tuple

print[name.0)

//replace any value in array

name[0] = newvalue

//remove any element from array

let name = Arrayname.remove(at: 1)

//remove all elements from array

//name.removeAll()

//add new element in array at the end

name.append(indexvalue)

//append multiple elements at end

name.append(contentsOf: [value,value,……])

/find size array

print(name.count)

//insert anywhere in array (selected place value displace to next place and so on )

name.insert(23, at: indexValue)

//insert multiple elements at a position

name.insert(contentsOf: [14,28], at: indexValue)

//check if element exist in array

let exist = name.contains(8) //true or false

//replace multiple elements at once with another elements

name.replaceSubrange(indexCount...indexCount, with: [12,9,11])

**Set** : - To stored uniqe value but it stored in un-ordered list set Keyword ka use krke

**Dictionary** : - collect data in keys and values (it is unordered )

Var name :[keyDataType:ValueDataType] = [value:value]

//get value for a particular key

print(name[key])

//add new pair to dictionary

name[key] = "value"

//number of pairs in dictionary

print(name.count)

//update value of key in dictionary

name[key] = "value"

//remove a pair from dictionary

var str:String? = name.removeValue(forKey: key)

//empty dictionary - store subject code and marks in dictionary

var name:[String:Float] = [:]

//get all keys

var names = name.keys

//get all values

var name = name.values

print(name)

//array of dictionary

var students:[ [String:Any] ] = [stu1, stu2,stu3,stu4,stu5]

print(students)

--------------------------------------------------------------------------------------------------------------------------

class name

{

CODE

}

Let name : name = name() (name is the reference of memory of type classname)

--------------------------------------------------------------------------------------------------------------------------

**Initializer** : - default initializer to initialize value to variable

Init(name:DataType,name:DataType,name:DataType)

{

CODE

}

--------------------------------------------------------------------------------------------------------------------------

**Protocol** : -

protocol name

{

CODE

}

let name = name()

--------------------------------------------------------------------------------------------------------------------------

**Extension** : -

extension name

{

CODE

}

Let name = name()

Data Storage Types in Swift

* UserDefaults
* Keychain
* Saving files to disk
* Core Data
* SQLite
* Property Lists