Day 1)

* The Scala REPL is a tool (scala) for evaluating expressions in Scala.
* The scala command will execute a source script by wrapping it in a template and then compiling and executing the resulting program.
* In interactive mode, the REPL reads expressions at the prompt, wraps them in an executable template, and then compiles and executes the result.
* Previous results are automatically imported into the scope of the current expression as required.
* REPL Shell is the scala interpreter used for executing commands in terminal

**Sample commands in scala REPL SHELL**

scala> print("hello World")

hello World

**Scala determines the Dataype on flow**

scala> 2.5 // Data type is defined based on the value

res0: Double = 2.52+3

scala> 10

res1: Int = 10

**For defining variables in Scala**

scala> def x = 3

scala> def x = 3

x: Int

scala> x+2

res0: Int = 5

**For Defining a function in Scala**

def f2c(x:Int) : Double = (x-32)\*5/9.0

scala> def f2c(x:Int) : Double = (x-32)\*5/9.0

f2c: (x: Int)Double

scala> f2c(100)

res0: Double = 37.77777777777778

**Defining list**

val odds = List(1,3,5,7,9) // val variables are immutable

scala> val odds = List(1,3,5,7,9)

odds: List[Int] = List(1, 3, 5, 7, 9)

odds.map(x=>x+1) // Similar to for each loop in java

scala> odds.map(x=>x+1)

res1: List[Int] = List(2, 4, 6, 8, 10)

scala> odds.reduce((x,y)=>(x+y)) // For aggregating

res0: Int = 25

List.sum

List.sum()

odds.sum

scala> odds.sum

res2: Int = 25

**//Custom function in scala**

def votingright(customerAge:Int)=if(customerAge >= 18) println ("eleigiblevoter") else println("minor”)

scala> votingright(22)

eleigiblevoter

scala> votingright(10)

minor

votingright(10)

scala> def shortNumber(x:String)=x match { // Similar to switch case in java and pattern matching

| case "one" => 1

| case "two" => 2

| case "three" => 3

| case "four" =>4

| }

shortNumber: (x: String)Int

def shortNumber(x:String) = x match {

case "one" => 1

case "two" => 2

case "three" => 3

case "four" => 4

case \_ => "Wrong String"

def shortnumber(x:String) = x match {

case "one" => 1

case "two"=> 2

case "three" => 3

case "four" => 4

case \_ => "Wrong String"

}

scala> shortNumber("one")

res5: Int = 1

scala> shortNumber("two")

res6: Int = 2

scala> shortNumber("three")

res7: Int = 3

def shortnumber(x:String) = x match {

case "one" | "ONE" => 1

case "two" | "TWO" => 2

case "three" | "THREE" =>3

}

shortnumber(one)

shortnumber("TWO")

shortnumber("ONE")

def shortnumber(x:String) = x.lower match {

case "one" => 1

case "two" => 2

case "three" => 3

case "four" => 4

case\_ => "Wrong String"

}

scala> val animals = List("dog","cat","pet")

animals: List[String] = List(dog, cat, pet)animals.foreach(println)

scala> animals.foreach(x =>println("animalname is" +x) //Foreach will iterate among items in loop and prints but doesn’t do any transformation

| )

animalname isdog

animalname iscat

animalname ispet

scala> animals.map(x=>x.length) //map can do the transformations

res9: List[Int] = List(3, 3, 3)

animals.foreach(x =>x.length)

scala> val nums = List(1,2,3)

nums: List[Int] = List(1, 2, 3)

scala> nums.map(x=>x\*x).map(x=>x-1)//output of one map result to another map result

res11: List[Int] = List(0, 3, 8)

scala> odds ++ nums//For concatenating two lists

res12: List[Int] = List(1, 3, 5, 7, 1, 2, 3)

val tokens = List("the","program","halted")

scala> val tags = List("DT","NN","VB")

tags: List[String] = List(DT, NN, VB)

scala> val tokentagpairs = tokens.zip(tags)//for alternative concatenating of two Lists

tokentagpairs: List[(String, String)] = List((the,DT), (program,NN), (halted,VB))

scala> (for((x,y)<- tokens.zip(tags)) yield x + "/" +y)// yield is for output formatting after zipping

res13: List[String] = List(the/DT, program/NN, halted/VB)