Session 14:
Scala Basics 1
Assignment 1

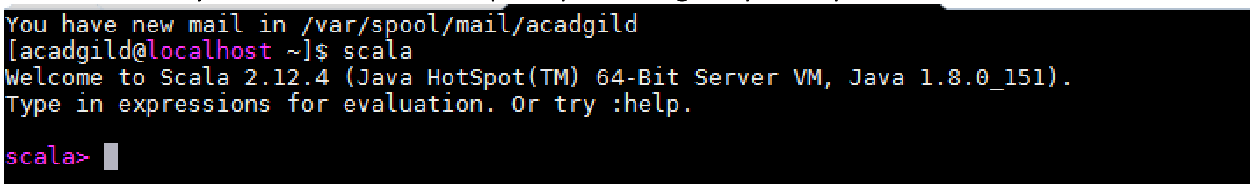
- Prachi Mohite

Some Basics of Scala Programming Language

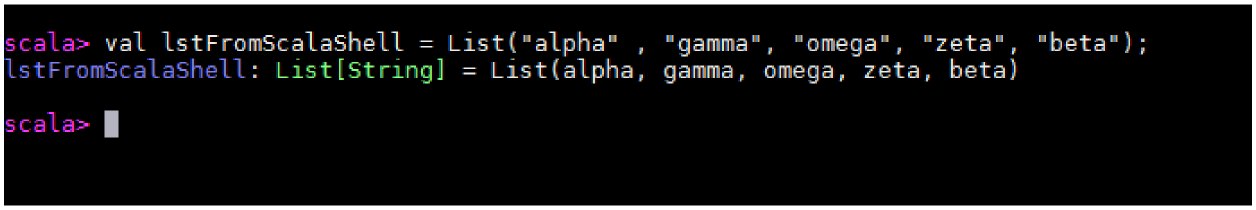
- Scala stands for “Scalable language”
- Runs on java platform. Interoperates with java libraries
- Blend of OOP and functional programming language
- Functions are treated as first class citizens
- Functions can take other functions as arguments
- Functions can return other functions as result

<u>There are different ways to execute the Scala Code</u>

- **Using Scala REPL**
    - To start Scala REPL, open command prompt and simply type *Scala.* After that, you will see new Scala prompt waiting for your input as shown below

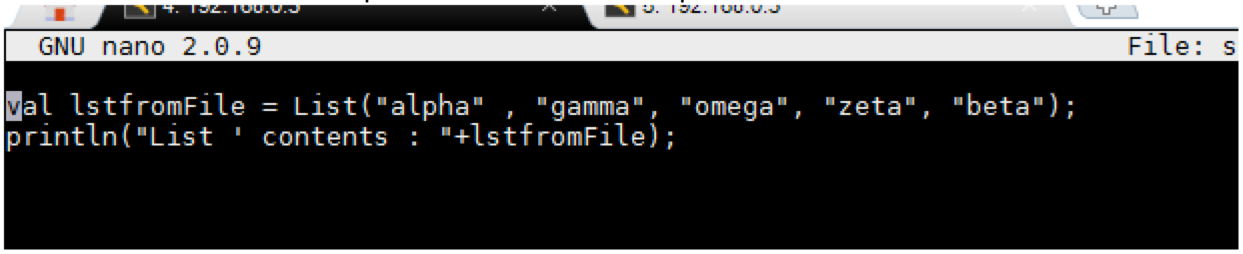


```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ scala
Welcome to Scala 2.12.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151).
Type in expressions for evaluation. Or try :help.

scala>
```

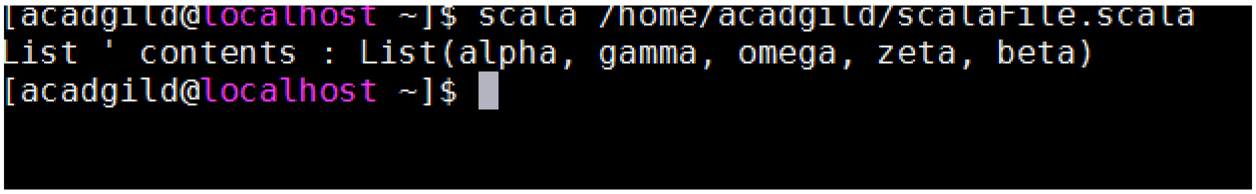


```
scala> val lstFromScalaShell = List("alpha" , "gamma", "omega", "zeta", "beta");
lstFromScalaShell: List[String] = List(alpha, gamma, omega, zeta, beta)

scala>
```

- **Using Scala interpreter to run scala script**
    - You can save your Scala code in the file with .scala extension (basically with any file extension but prefered .scalaextension) and to run, provide file name with extension as parameter to Scala interpreter



```
GNU nano 2.0.9                                              File: s

val lstfromFile = List("alpha" , "gamma", "omega", "zeta", "beta");
println("List ' contents : "+lstfromFile);
```



```
[acadgild@localhost ~]$ scala /home/acadgild/scalaFile.scala
List ' contents : List(alpha, gamma, omega, zeta, beta)
[acadgild@localhost ~]$
```

- **Using Scala interpreter**
    - Usual Scala program contains lot code chunks spread of across lot of files, for running these programs we need to go through two stages, compile the Scala

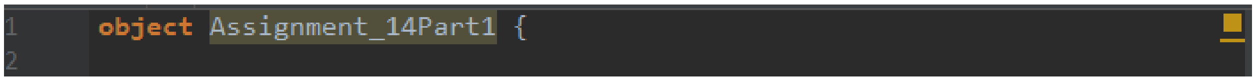
source code using Scala compiler and run the compiled bytecode using Scala interpreter.

- **Using Scala worksheet**

    - A worksheet is a Scala file that is evaluated on save, and the result of each expression is shown in a column to the right of your program. Worksheets are like a REPL session on steroids, and enjoy 1st class editor support: completion, hyperlinking, interactive errors-as-you-type, auto-format, etc.
    - For creating new Scala worksheet in Scala IDE / IDEA IntelliJ , first create Scala project then right click on Scala project and go to following *New > Scala WorkSheet*

Here we will be using IntelliJ Idea to create scala Projects instead of Scala IDE.
We have installed Scala plug in in Idea IDE.

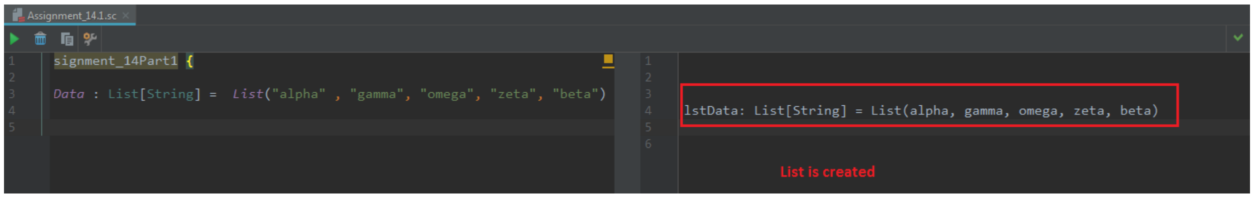
- Added worksheet in Scala Project
-



```
object Assignment_14Part1 {
```

Task 1
Given a list of strings - List[String] (“alpha”, “gamma”, “omega”, “zeta”, “beta”)

- List
    - Immutable sequence of objects of same type can be represented using lists
    - A List[Int] contains only integers.
    - Scala Lists are always immutable (whereas Java Lists can be mutable).
    - While creating a list we can define the data collection getting into that list
        - Val lstData : List[DataType] = List (Collection_of_data)

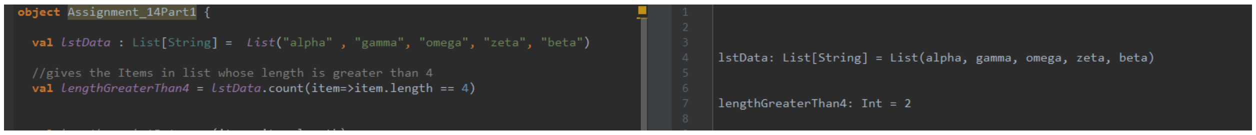


```
signment_14Part1 {

Data : List[String] =  List("alpha" , "gamma", "omega", "zeta", "beta")
```

```
lstData: List[String] = List(alpha, gamma, omega, zeta, beta)
```

1.1

Find count of all strings with length 4.



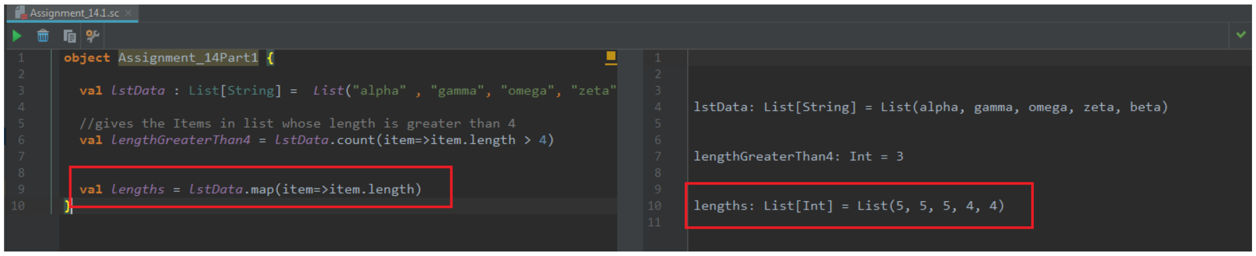
```
object Assignment_14Part1 {

  val lstData : List[String] =  List("alpha" , "gamma", "omega", "zeta", "beta")

  //gives the Items in list whose length is greater than 4
  val lengthGreaterThan4 = lstData.count(item=>item.length == 4)
```

```
lstData: List[String] = List(alpha, gamma, omega, zeta, beta)

lengthGreaterThan4: Int = 2
```

1.2 Convert the list of string to a list of integers, where each string is mapped to its corresponding length.

Here we need to use Map function of Scala list

Map - Returns a list resulting from adding a "y" to each string element in the list



```
object Assignment_14Part1 {

  val lstData : List[String] =  List("alpha" , "gamma", "omega", "zeta"

  //gives the Items in list whose length is greater than 4
  val lengthGreaterThan4 = lstData.count(item=>item.length > 4)


  val lengths = lstData.map(item=>item.length)
}
```

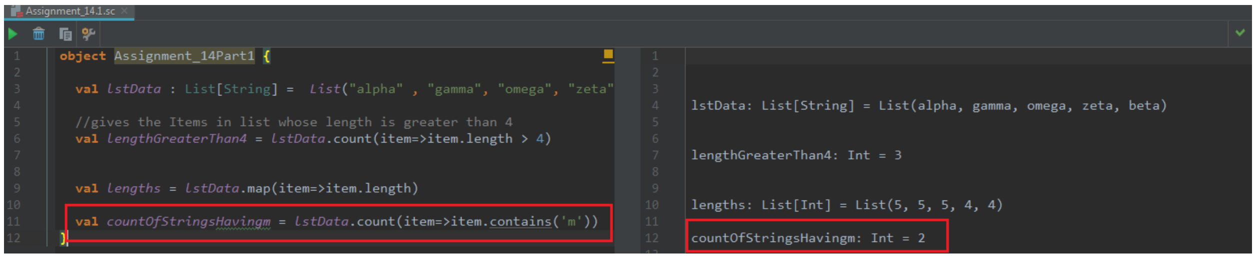
```
lstData: List[String] = List(alpha, gamma, omega, zeta, beta)

lengthGreaterThan4: Int = 3

lengths: List[Int] = List(5, 5, 5, 4, 4)
```

1.3 Find count of all strings which contain alphabet ‘m’.

Here to iterate through every record we will be using lambda expression

To get count we will be using count method of List

And to check if any item from List contains ‘m’, we will be using string operator ‘contains’



```
object Assignment_14Part1 {

  val lstData : List[String] =  List("alpha" , "gamma", "omega", "zeta"

  //gives the Items in list whose length is greater than 4
  val lengthGreaterThan4 = lstData.count(item=>item.length > 4)


  val lengths = lstData.map(item=>item.length)

  val countOfStringsHavingm = lstData.count(item=>item.contains('m'))
}
```

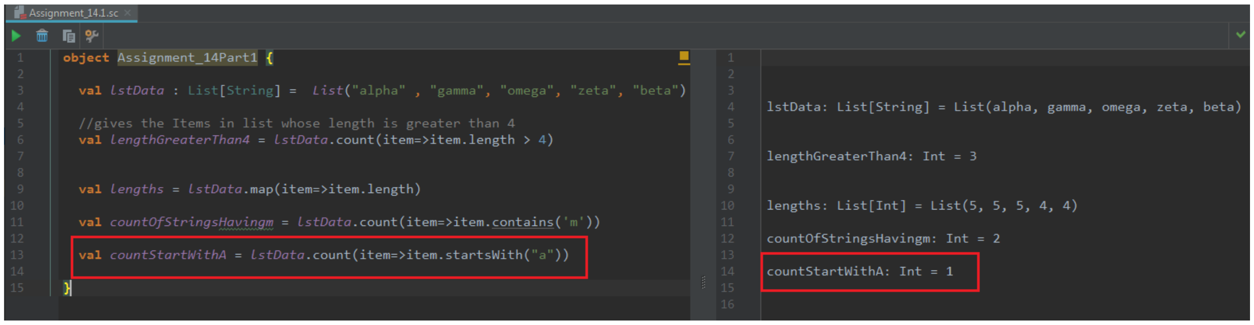
```
lstData: List[String] = List(alpha, gamma, omega, zeta, beta)

lengthGreaterThan4: Int = 3

lengths: List[Int] = List(5, 5, 5, 4, 4)
countOfStringsHavingm: Int = 2
```

1.4 Find the count of all strings which start with the alphabet ‘a’.

Here to iterate through every record we will be using lambda expression

To get count we will be using count method of List

And to check if any item from List starts with ‘a’ we will be using startsWith string operator.


Assignment_14.1.sc

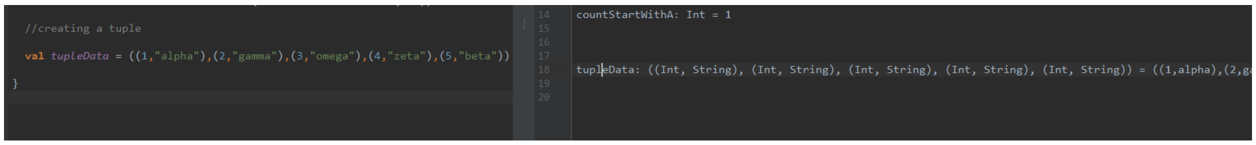
```
object Assignment_14Part1 {

  val lstData : List[String] =  List("alpha" , "gamma", "omega", "zeta", "beta")

  //gives the Items in list whose length is greater than 4
  val lengthGreaterThan4 = lstData.count(item=>item.length > 4)


  val lengths = lstData.map(item=>item.length)

  val countOfStringsHavingm = lstData.count(item=>item.contains('m'))

  val countStartWithA = lstData.count(item=>item.startsWith("a"))

}
```

```
lstData: List[String] = List(alpha, gamma, omega, zeta, beta)

lengthGreaterThan4: Int = 3

lengths: List[Int] = List(5, 5, 5, 4, 4)

countOfStringsHavingm: Int = 2

countStartWithA: Int = 1
```

Task 2

2.1 Create a list of tuples, where the 1st element of the tuple is an int and the second element is a string.

- Tuples are immutable, but unlike lists, tuples can contain different types of elements
- Tuple can have both an integer and a string at the same time



```
  //creating a tuple

  val tupleData = ((1,"alpha"),(2,"gamma"),(3,"omega"),(4,"zeta"),(5,"beta"))

}
```

```
countStartWithA: Int = 1



tupleData: ((Int, String), (Int, String), (Int, String), (Int, String), (Int, String)) = ((1,alpha),(2,ga
```

2.1 For the above list, print the numbers where the corresponding string length is 4.

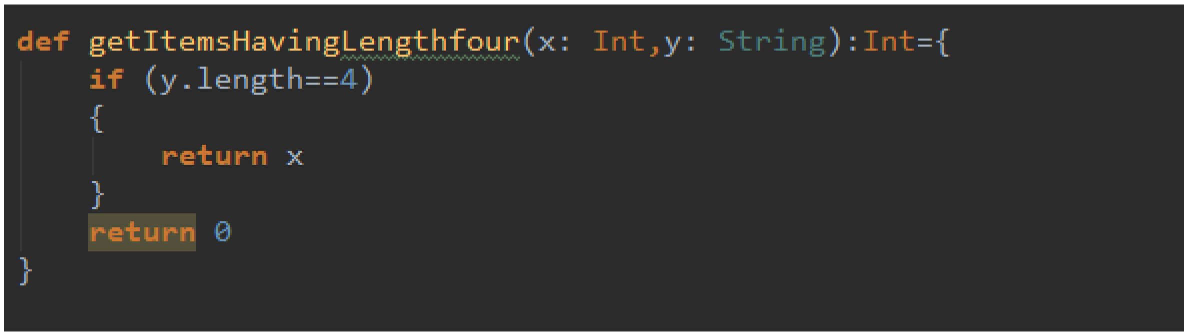
- Created list of tuple data

Solution Approach 1

 - This approach is just to explain the use of functions.

- As we have to iterate through every tuple of list and get the number in associated with the string having length = 2

 - created a function which accepts tuple as argument and return the number from tuple where string length = 4 else return 0

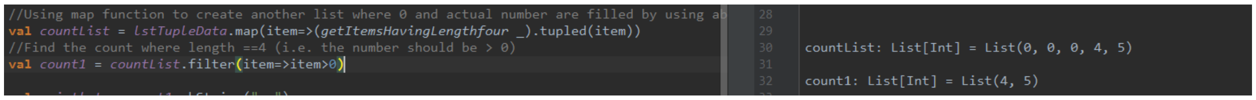


```
def getItemsHavingLengthfour(x: Int,y: String):Int={
    if (y.length==4)
    {
        return x
    }
    return 0
}
```

Used this function on every tuple in list to create new list where 0 will indicate that length <> 4 and actual number will indicate that length ==4

Iterated through list and used map function of list to get new list

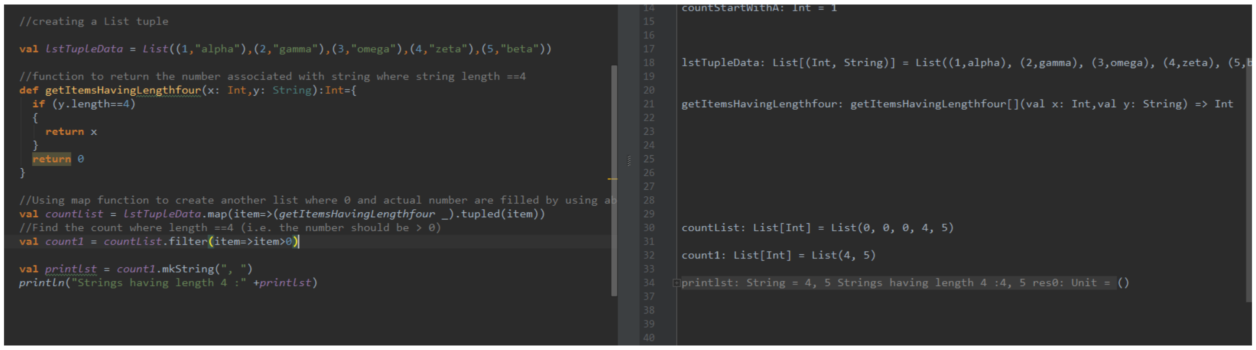
Created string from the newly created list using filter fucntion



```
//Using map function to create another list where 0 and actual number are filled by using ab
val countList = lstTupleData.map(item=>(getItemsHavingLengthfour _).tupled(item))
//Find the count where length ==4 (i.e. the number should be > 0)
val count1 = countList.filter(item=>item>0)
```

```
countList: List[Int] = List(0, 0, 0, 4, 5)

count1: List[Int] = List(4, 5)
```

Finally used mkstring function of string to get final result.

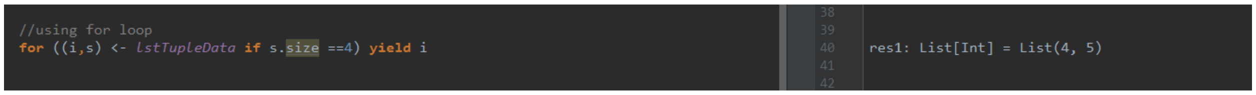


```
//creating a List tuple

val lstTupleData = List((1,"alpha"),(2,"gamma"),(3,"omega"),(4,"zeta"),(5,"beta"))

//function to return the number associated with string where string length ==4
def getItemsHavingLengthfour(x: Int,y: String):Int={
  if (y.length==4)
  {
    return x
  }
  return 0
}

//Using map function to create another list where 0 and actual number are filled by using ab
val countList = lstTupleData.map(item=>(getItemsHavingLengthfour _).tupled(item))
//Find the count where length ==4 (i.e. the number should be > 0)
val count1 = countList.filter(item=>item>0)

val printlst = count1.mkString(", ")
println("Strings having length 4 :" +printlst)
```

```
countStartWithA: Int = 1


lstTupleData: List[(Int, String)] = List((1,alpha), (2,gamma), (3,omega), (4,zeta), (5,b

getItemsHavingLengthfour: getItemsHavingLengthfour[](val x: Int,val y: String) => Int






countList: List[Int] = List(0, 0, 0, 4, 5)

count1: List[Int] = List(4, 5)

printlst: String = 4, 5 Strings having length 4 :4, 5 res0: Unit = ()
```

Solution Approach 2

- With for loop

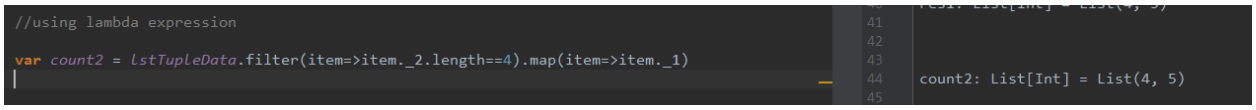


```
//using for loop
for ((i,s) <- lstTupleData if s.size ==4) yield i
```

```
res1: List[Int] = List(4, 5)
```

Solution Approach 3

- With Lambda Expressions

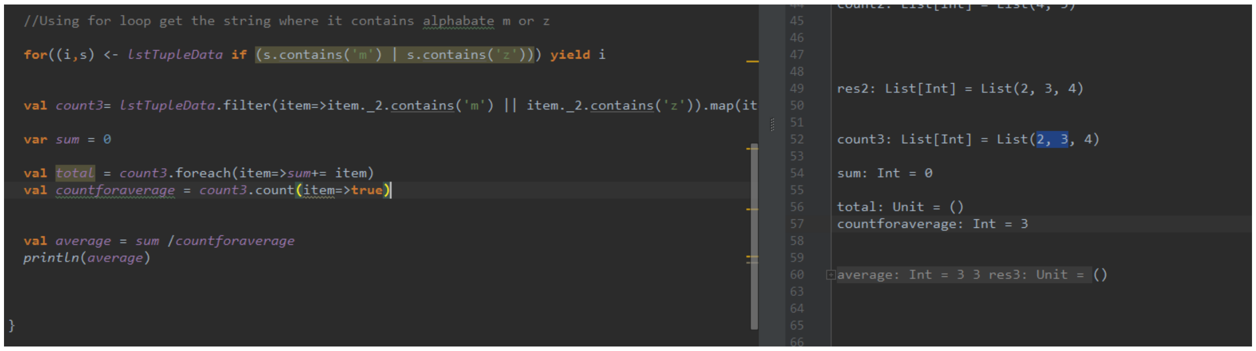


```
//using lambda expression

var count2 = lstTupleData.filter(item=>item._2.length==4).map(item=>item._1)
```

```
count2: List[Int] = List(4, 5)
```

- find the average of all numbers, where the corresponding string contains alphabet 'm' or alphabet 'z'.

We will be using for lambda Expression to get the above task done



```scala
  //Using for loop get the string where it contains alphabate m or z

  for((i,s) <- lstTupleData if (s.contains('m') | s.contains('z'))) yield i


  val count3= lstTupleData.filter(item=>item._2.contains('m') || item._2.contains('z')).map(it

  var sum = 0

  val total = count3.foreach(item=>sum+= item)
  val countforaverage = count3.count(item=>true)


  val average = sum /countforaverage
  println(average)



}
```

```
res2: List[Int] = List(2, 3, 4)


count3: List[Int] = List(2, 3, 4)

sum: Int = 0

total: Unit = ()
countforaverage: Int = 3


average: Int = 3 3 res3: Unit = ()
```