

Group A

Lab 3A

Scala

Participants:

1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 35:

```
While(x>=1){  
    println("Hello")  
    x=-1  
}
```

The screenshot shows a desktop environment with a terminal window and a file explorer window. The terminal window is titled 'root@sandbox-hdp~ - Shell In /' and contains the following Scala code and output:

```
scala> var x=5  
x: Int = 5  
  
scala> while (x>=1){  
    |   println("Hello")  
    |   x-=1  
    | }  
Hello  
Hello  
Hello  
Hello  
Hello  
  
scala>
```

The file explorer window is titled 'BDM Group...' and lists the 'Participants' as follows:

Participants

1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 36:

The screenshot shows a desktop environment with a terminal window and a file explorer window. The terminal window is titled 'root@sandbox-hdp~ - Shell In /' and contains the following Scala code and output:

```
scala> var x=5  
x: Int = 5  
  
scala> while (x>=1){  
    |   println("Hello")  
    |   x-=1  
    | }  
Hello  
Hello  
Hello  
Hello  
Hello  
  
scala> 1.to(10)  
res0: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
  
scala> 1 to 10  
res10: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
  
scala> (1 to 10).reverse  
res11: scala.collection.immutable.Range = Range(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)  
  
scala>
```

The file explorer window is titled 'BDM Group...' and lists the 'Participants' as follows:

Participants

1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 37:

```
for(i<-1 to 10)println(i)
for(i<-(1 to 10).reverse)println(i)
```

The screenshot shows a Windows desktop environment. On the left, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "192.168.246.129:4200". The terminal displays Scala code and its output. The code includes creating a range from 1 to 10, reversing it, and then printing each element. On the right, there is a file explorer window titled "BDM_Group..." showing a list of participants for "Group A".

```
res10: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
scala> (1 to 10).reverse
res11: scala.collection.immutable.Range = Range(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

scala> for(i<-1 to 10)println(i)
1
2
3
4
5
6
7
8
9
10

scala> for(i<-(1 to 10).reverse)println(i)
10
9
8
7
6
5
4
3
2
1

scala>
```

Participants
1. Aadarsha chapagain
2.Jyoti shukla
3.Rishi Phaneendra Varma
4.Priti Bhale
5.Sreya Treesa Johny
6.Piyush Bhatia

Slide 38:

```
for(i<-1 until 10)println(i)
```

The screenshot shows a Windows desktop environment. On the left, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "192.168.246.129:4200". The terminal displays Scala code and its output. The code includes creating ranges from 1 to 10 and 1 until 10, and then printing each element. On the right, there is a file explorer window titled "BDM_Group..." showing a list of participants for "Group A".

```
res15: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
scala> 1 until 10
res16: scala.collection.immutable.Range = Range(1, 2, 3, 4, 5, 6, 7, 8, 9)

scala> for(i<-1 until 10)println(i)
1
2
3
4
5
6
7
8
9

scala>
```

Participants
1. Aadarsha chapagain
2.Jyoti shukla
3.Rishi Phaneendra Varma
4.Priti Bhale
5.Sreya Treesa Johny
6.Piyush Bhatia

Slide 39:

```
for(ch<- 0 until x.length)println(x(ch))  
for (ch <- x)println(ch)
```

It takes each character from x which is the string and assign it to “ch” variable then ch become a character variable (of type character)

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp- Shell In / x" with the IP address "192.168.246.129:4200". The terminal displays the following Scala code and its output:

```
scala> for(i<-1 until 10)println(i)  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
scala> var x="Hello"  
x: String = Hello  
scala> for(ch<- 0 until x.length)println(x(ch))  
H  
e  
l  
l  
o  
scala> for (ch <- x)println(ch)
```

Below the terminal, a messaging application window titled "BDM_Group..." is open, showing a list of participants in "Group A":

- 1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 41:

```
for(i<-1 to 5){  
    for(j<-1 to 2){  
        println(i*j)  
    }  
}
```

```
scala> for(i<-1 to 5;j<- 1 to 2) println(i*j)
```

Advance looping in scala rather than traditional approach

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp- Shell In / x" with the IP address "192.168.246.129:4200". The terminal displays the following Scala code and its output:

```
scala> for(i<-1 to 5){  
    | for(j<-1 to 2){  
    |     println(i*j)  
    | }  
    | }  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
scala> for(i<-1 to 5;j<- 1 to 2) println(i*j)  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
scala>
```

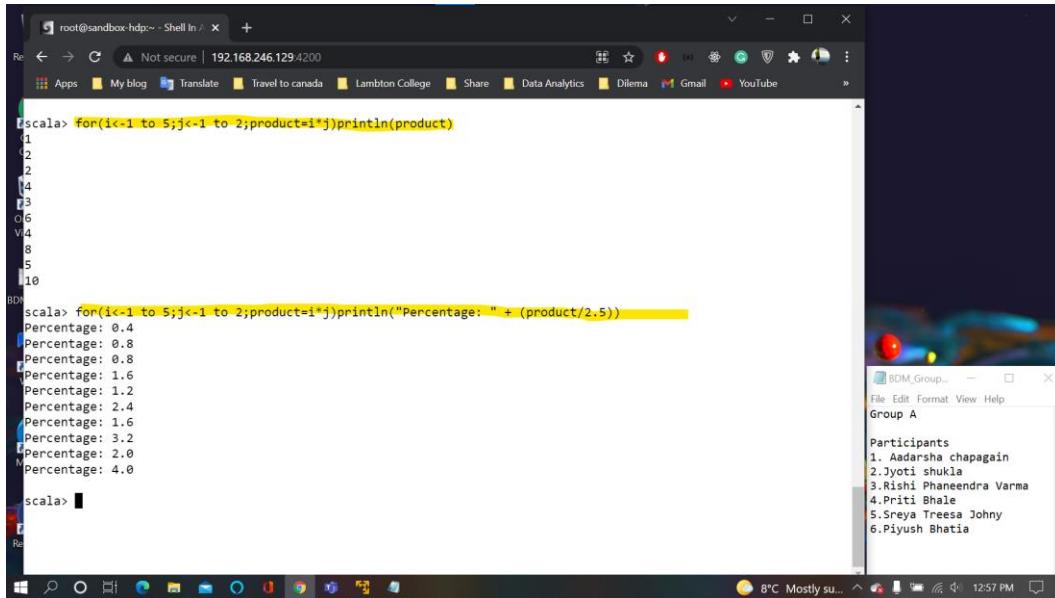
Below the terminal, a messaging application window titled "BDM_Group..." is open, showing a list of participants in "Group A":

- 1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 42:

```
for(i<-1 to 5;j<-1 to 2;product=i*j)println(product)
for(i<-1 to 5;j<-1 to 2;product=i*j)println("Percentage: " + (product/2.5))
```

Within println we can concatenate two expression to form one single expression
Dynamic value is product/2.5

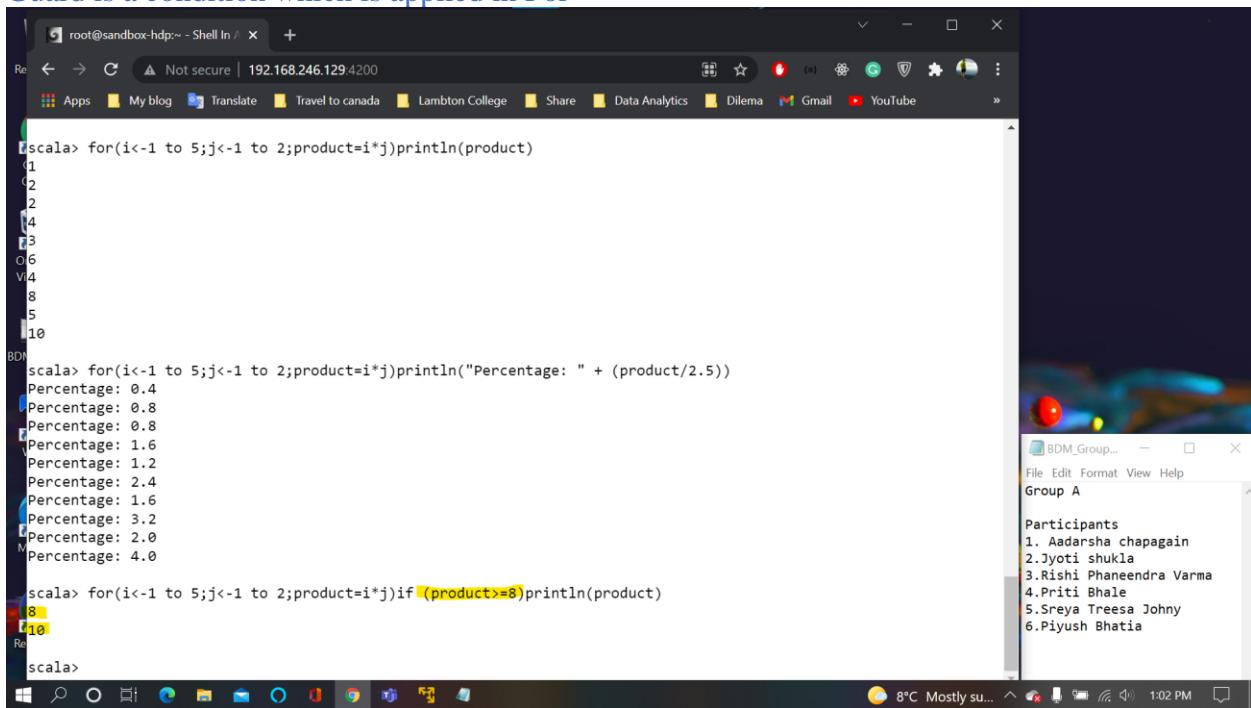


A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code being run. The code consists of two for loops. The first loop prints the product of i and j. The second loop prints the percentage of the product divided by 2.5. The output shows the products 1 through 10, followed by their respective percentages: 0.4, 0.8, 0.8, 1.6, 1.2, 2.4, 1.6, 3.2, 3.2, 2.0, and 4.0. On the right, a file explorer window titled 'BDM_Group...' is open, showing a list of participants for 'Group A': 1. Adarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treesa Johny, and 6. Piyush Bhatia.

```
scala> for(i<-1 to 5;j<-1 to 2;product=i*j)println(product)
1
2
3
4
5
6
7
8
9
10
BDN
scala> for(i<-1 to 5;j<-1 to 2;product=i*j)println("Percentage: " + (product/2.5))
Percentage: 0.4
Percentage: 0.8
Percentage: 0.8
Percentage: 1.6
Percentage: 1.2
Percentage: 2.4
Percentage: 1.6
Percentage: 3.2
Percentage: 3.2
Percentage: 2.0
Percentage: 4.0
scala>
```

Slide 43:

Guard is a condition which is applied in For



A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code being run. The code includes three for loops. The first two loops are identical to the ones in the previous slide. The third loop adds a guard condition: it only prints the product if the product is greater than or equal to 8. The output shows the products 1 through 10, with the last two products (8 and 10) being printed because they meet the guard condition. On the right, a file explorer window titled 'BDM_Group...' is open, showing a list of participants for 'Group A': 1. Adarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treesa Johny, and 6. Piyush Bhatia.

```
scala> for(i<-1 to 5;j<-1 to 2;product=i*j)println(product)
1
2
3
4
5
6
7
8
9
10
BDN
scala> for(i<-1 to 5;j<-1 to 2;product=i*j)println("Percentage: " + (product/2.5))
Percentage: 0.4
Percentage: 0.8
Percentage: 0.8
Percentage: 1.6
Percentage: 1.2
Percentage: 2.4
Percentage: 1.6
Percentage: 3.2
Percentage: 3.2
Percentage: 2.0
Percentage: 4.0
scala> for(i<-1 to 5;j<-1 to 2;product=i*j)if (product>=8)println(product)
8
10
scala>
```

Slide 45:

Yield is used for compression. If we want to do something on individual value for loop `is` used and if we want all the individual values to be combined `yield` is used

root@sandbox-hdp:~ Shell In A +

Not secure | 192.168.246.129:4200

Apps My blog Translate Travel to canada Lambton College Share Data Analytics Dilema Gmail YouTube

```
scala> for(i<-1 to 5;j<-1 to 2;product=i*j) yield product
res29: scala.collection.immutable.IndexedSeq[Int] = Vector(1, 2, 2, 4, 3, 6, 4, 8, 5, 10)
scala> for(i<-1 to 5;j<-1 to 2;product=i*j) println (product)
1
2
2
4
3
6
4
8
5
10

scala>
```

BDM_Group... File Edit Format View Help Group A

Participants

- 1. Aadarschapagain
- 2. Jyoti shukla
- 3. Rishi Phaneendra Varma
- 4. Priti Bhale
- 5. Sreya Treesa Johny
- 6. Piyush Bhatia

Slide 46:

ASCCI Value

The first one will print ASCII value of every character in Hello

The Second one will print ASCII + value given by I from the loop

For H it will go like $72+1(i=1)$, $72+2(i=2)$ and so on for every character.

The screenshot shows a Windows desktop environment. In the foreground, a terminal window titled "root@sandbox-hdp:~ - Shell /" is open at port 192.168.246.129:4200. The terminal displays Scala code being run, with line numbers 115 through 113 visible on the left. The code includes two for-loops that print the characters of the string "Hello" and then print each character followed by its index from 0 to 4. In the background, a file viewer window titled "BDM_Group..." is open, showing a list of participants: Aadarsha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhale, Sreya Treesa Johnny, and Piyush Bhatia.

```
115
116
scala> for(ch<- "Hello") println(ch+0)
72
101
188
188
O 111
V
BDN
111
110
M 111
112
113
109
F 110
Re 111
112
113
```

```
115
116
scala> for(ch<- "Hello";i<- 1 to 5 ) println(ch+i)
73
74
75
76
77
102
103
104
105
106
109
110
M 111
112
113
109
F 110
Re 111
112
113
```

File Edit Format View Help
Group A

Participants

1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhale
- 5.Sreya Treesa Johnny
- 6.Piyush Bhatia

Slide 47:

If the expression of yield has two different types than the output of the yield will have the datatype of first generator that is integer here

A screenshot of a Windows desktop environment. On the left, there is a terminal window titled 'root@sandbox-hdp:~ - Shell In / x'. It contains Scala code and its output:

```
scala> for(i<-1 to 5; ch<-"Hello") yield i+ch
res35: scala.collection.immutable.IndexedSeq[Int] = Vector(73, 102, 109, 109, 112, 74, 103, 110, 110, 113, 75,
104, 111, 111, 114, 76, 105, 112, 112, 115, 77, 106, 113, 113, 116)

scala>
```

On the right, there is a file explorer window titled 'Group A' which lists participants:

Participants

1. Aadarsha chapagain
2. Jyoti shukla
3. Rishi Phaneendra Varma
4. Priti Bhale
5. Sreya Treesa Johnny
6. Piyush Bhatia

Slide 53:

Function in scala

A screenshot of a Windows desktop environment. On the left, there is a terminal window titled 'root@sandbox-hdp:~ - Shell In / x'. It contains Scala code and its output:

```
scala> for(i<- 1 to 5; ch<-"Hello") yield i+ch
res35: scala.collection.immutable.IndexedSeq[Int] = Vector(73, 102, 109, 109, 112, 74, 103, 110, 110, 113, 75,
104, 111, 111, 114, 76, 105, 112, 112, 115, 77, 106, 113, 113, 116)

scala> def product (a:Int, b:Int)={
|   | a*b
|   |
| }
product: (a: Int, b: Int)Int

scala> def product (a:Int, b:Double)={
|   | a*b
|   |
| }
product: (a: Int, b: Double)Double

scala> def product (a:Int, b:Double)=a*b
product: (a: Int, b: Double)Double

scala> def product (a:String, b:Double)=a+b
product: (a: String, b: Double)String

scala> product("Hello",5)
res36: String = Hello5.0

scala>
```

On the right, there is a file explorer window titled 'Group A' which lists participants:

Participants

1. Aadarsha chapagain
2. Jyoti shukla
3. Rishi Phaneendra Varma
4. Priti Bhale
5. Sreya Treesa Johnny
6. Piyush Bhatia

Slide 54:

Multiple Line function

A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In /" shows a Scala session. The user defines several functions: a simple product function, a product function that takes a Double, a product function that takes a String, and an area function that calculates the area of a circle given its radius. On the right, a Notepad window titled "BDM_GroupA" displays a list of participants.

```
| a*b
| }
product: (a: Int, b: Int)Int
scala> def product (a:Int, b:Double)={
| a*b
| }
product: (a: Int, b: Double)Double
scala> def product (a:Int, b:Double)=a*b
product: (a: Int, b: Double)Double
BDN scala> def product (a:String, b:Double)=a+b
product: (a: String, b: Double)String
scala> product("Hello",5)
res36: String = Hello5.0
scala> def area(radius:Int)={
| val PI=3.142
| PI*radius*radius
| }
area: (radius: Int)Double
scala> area(5)
res37: Double = 78.55
scala> var x=area(5)
x: Double = 78.55
```

Participants
1. Aadarsha chapagain
2.Jyoti shukla
3.Rishi Phaneendra Varma
4.Priti Bhale
5.Sreya Treesa Johny
6.Piyush Bhatia

Slide 56:

A procedure is a function that does not return value

A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In /" shows a Scala session. The user defines a procedure named "concatandprint" which concatenates two strings and prints the result. When called with "Hello" and "GroupA", it outputs "Hello GroupA". On the right, a Notepad window titled "BDM_GroupA" displays a list of participants.

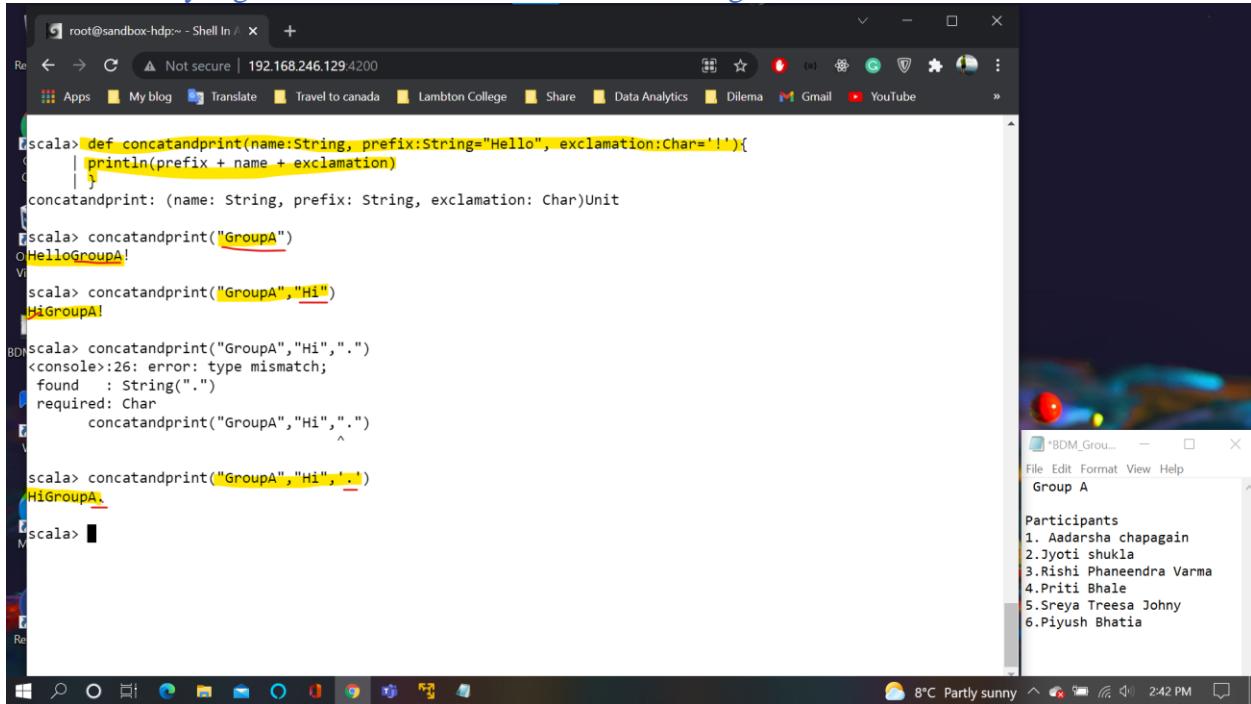
```
scala> def concatandprint(str1:String, str2:String){
| | println(str1+ " " + str2)
| }
concatandprint: (str1: String, str2: String)Unit
scala> concatandprint("Hello", "GroupA")
Hello GroupA
scala>
```

Participants
1. Aadarsha chapagain
2.Jyoti shukla
3.Rishi Phaneendra Varma
4.Priti Bhale
5.Sreya Treesa Johny
6.Piyush Bhatia

Slide 58:

Default argument

The mandatory argument should be in front of default arguments.



A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code being run. The code defines a function 'concatandprint' with a default argument 'prefix="Hello"'. It then calls this function with 'GroupA' and 'Hi', resulting in 'HelloGroupA!' and 'HiGroupA!'. It also attempts to call it with 'Hi', which fails due to a type mismatch. On the right, a file explorer window titled 'Group A' lists participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treesa Johnny, and 6. Piyush Bhatia. The desktop taskbar at the bottom shows various icons for apps like File Explorer, Edge, and Mail.

```
scala> def concatandprint(name:String, prefix:String="Hello", exclamation:Char='!'){
    |   println(prefix + name + exclamation)
    |
}
concatandprint: (name: String, prefix: String, exclamation: Char)Unit

scala> concatandprint("GroupA")
HelloGroupA!

scala> concatandprint("GroupA","Hi")
HiGroupA!

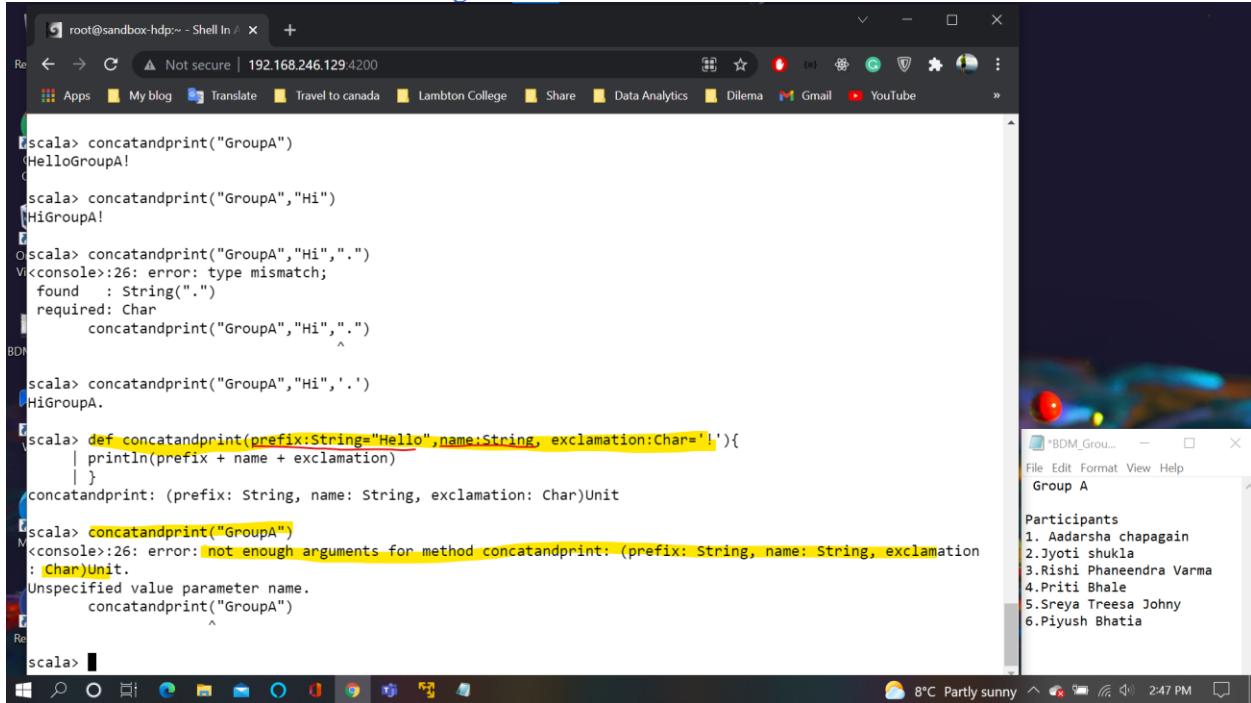
scala> concatandprint("GroupA","Hi",".")
<console>:26: error: type mismatch;
 found   : String(".")
 required: Char
         concatandprint("GroupA","Hi",".")

scala> concatandprint("GroupA","Hi",'!')
HiGroupA.

scala>
```

Slide 59:

The order and the name of the argument must match in the slide 58 both were matched but they are not matched here so order of argument must match



A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code being run. The code defines a function 'concatandprint' with parameters 'prefix', 'name', and 'exclamation'. It then calls this function with 'GroupA' and 'Hi', resulting in 'HelloGroupA!', 'HiGroupA!', and 'HiGroupA.'. It also attempts to call it with 'Hi', which fails due to a type mismatch. On the right, a file explorer window titled 'Group A' lists participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treesa Johnny, and 6. Piyush Bhatia. The desktop taskbar at the bottom shows various icons for apps like File Explorer, Edge, and Mail.

```
scala> concatandprint("GroupA")
HelloGroupA!

scala> concatandprint("GroupA", "Hi")
HiGroupA!

scala> concatandprint("GroupA", "Hi", ".")
<console>:26: error: type mismatch;
 found   : String(".")
 required: Char
         concatandprint("GroupA", "Hi", ".")

scala> concatandprint("GroupA", "Hi", '.')
HiGroupA.

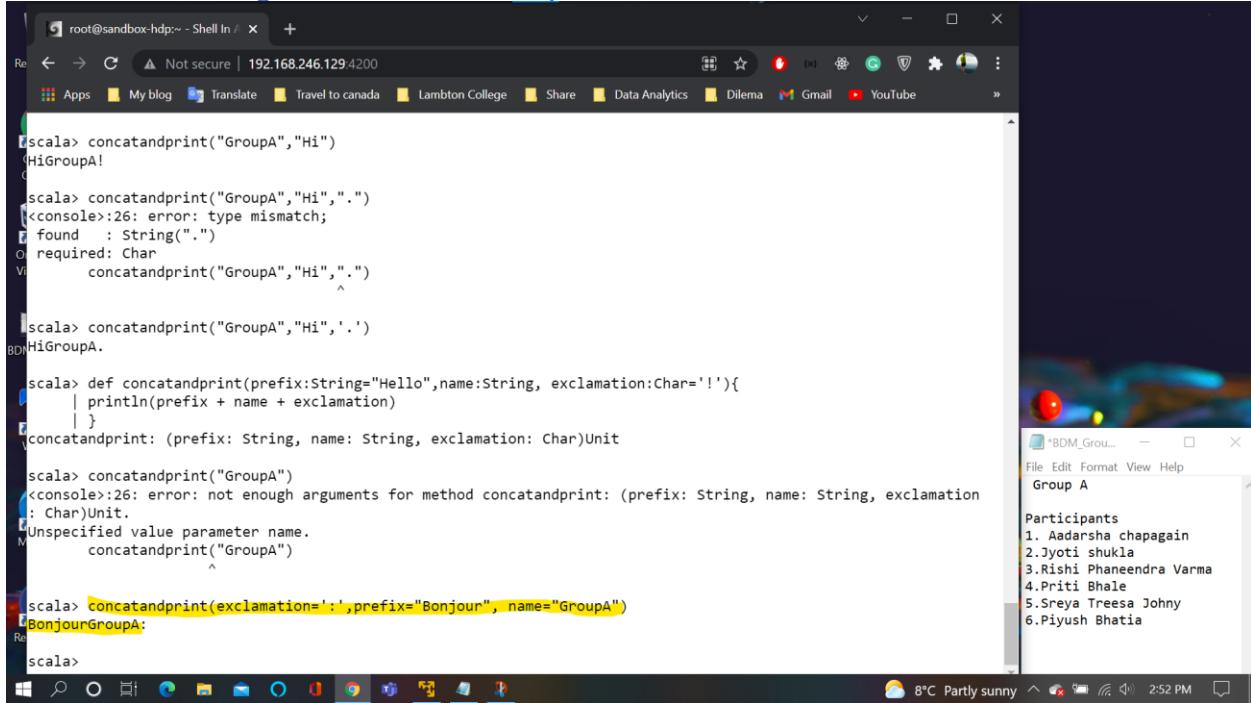
scala> def concatandprint(prefix:String="Hello",name:String, exclamation:Char='!'){
    |   println(prefix + name + exclamation)
    |
}
concatandprint: (prefix: String, name: String, exclamation: Char)Unit

scala> concatandprint("GroupA")
<console>:26: error: not enough arguments for method concatandprint: (prefix: String, name: String, exclamation: Char)Unit.
Unspecified value parameter name.
        concatandprint("GroupA")
          ^

scala>
```

Slide 60:

In case of named argument order is not important



A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In" shows Scala code being run. On the right, a Notepad window titled "BDM_Grou..." displays a list of participants for "Group A".

```
scala> concatandprint("GroupA","Hi")
HiGroupA!
 concatandprint("GroupA","Hi",".")
<console>:26: error: type mismatch;
 found   : String(".")
 required: Char
V       concatandprint("GroupA","Hi",".")
          ^
scala> concatandprint("GroupA","Hi",'.')
HiGroupA.

scala> def concatandprint(prefix:String="Hello",name:String, exclamation:Char='!'){
    | println(prefix + name + exclamation)
    |
concatandprint: (prefix: String, name: String, exclamation: Char)Unit

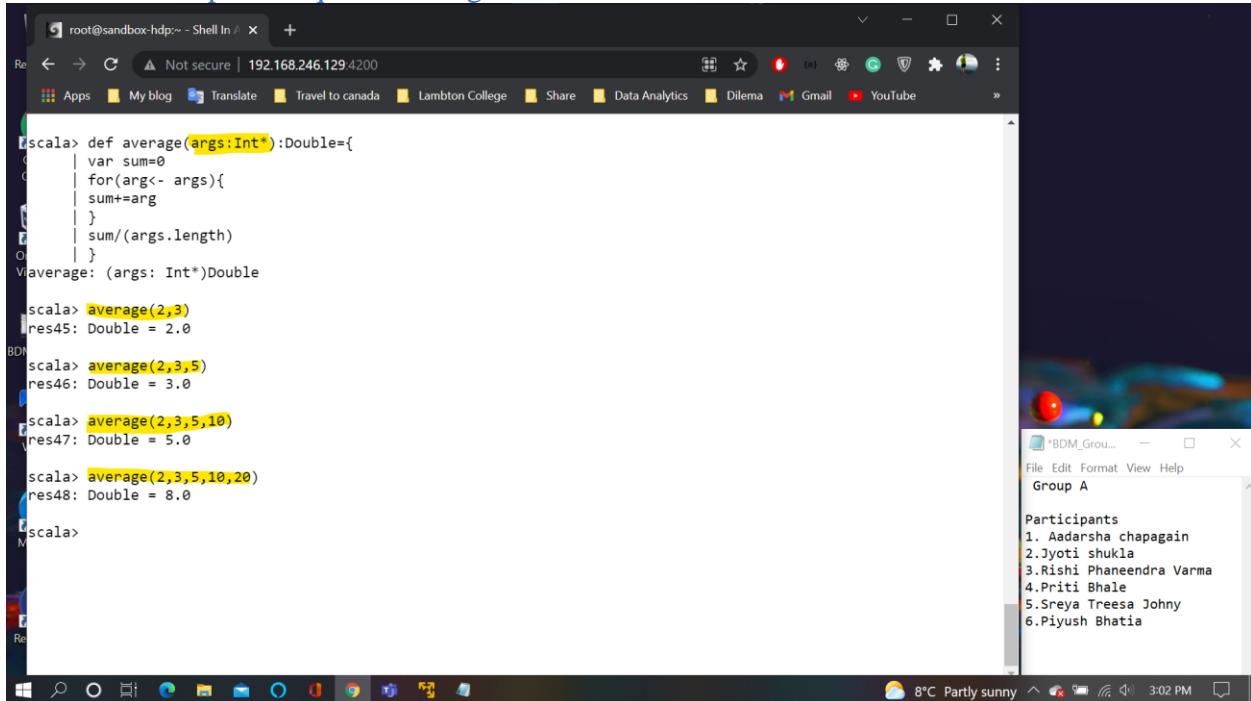
scala> concatandprint("GroupA")
<console>:26: error: not enough arguments for method concatandprint: (prefix: String, name: String, exclamation: Char)Unit.
Unspecified value parameter name.
M       concatandprint("GroupA")
          ^
scala> concatandprint(exclamation=':',prefix="Bonjour", name="GroupA")
BonjourGroupA;
scala>
```

Notepad content (Group A Participants):

- Participants
- 1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhole
- 5.Sreya Treessa Johnny
- 6.Piyush Bhatia

Slide62:

The function expects sequence as arguments



A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In" shows Scala code being run. On the right, a Notepad window titled "BDM_Grou..." displays a list of participants for "Group A".

```
scala> def average(args:Int):Double={
    | var sum=0
    | for(arg<- args){
    | sum+=arg
    | }
    | sum/(args.length)
    |
average: (args: Int*)Double

scala> average(2,3)
res45: Double = 2.0
BDM
scala> average(2,3,5)
res46: Double = 3.0

scala> average(2,3,5,10)
res47: Double = 5.0

scala> average(2,3,5,10,20)
res48: Double = 8.0

scala>
```

Notepad content (Group A Participants):

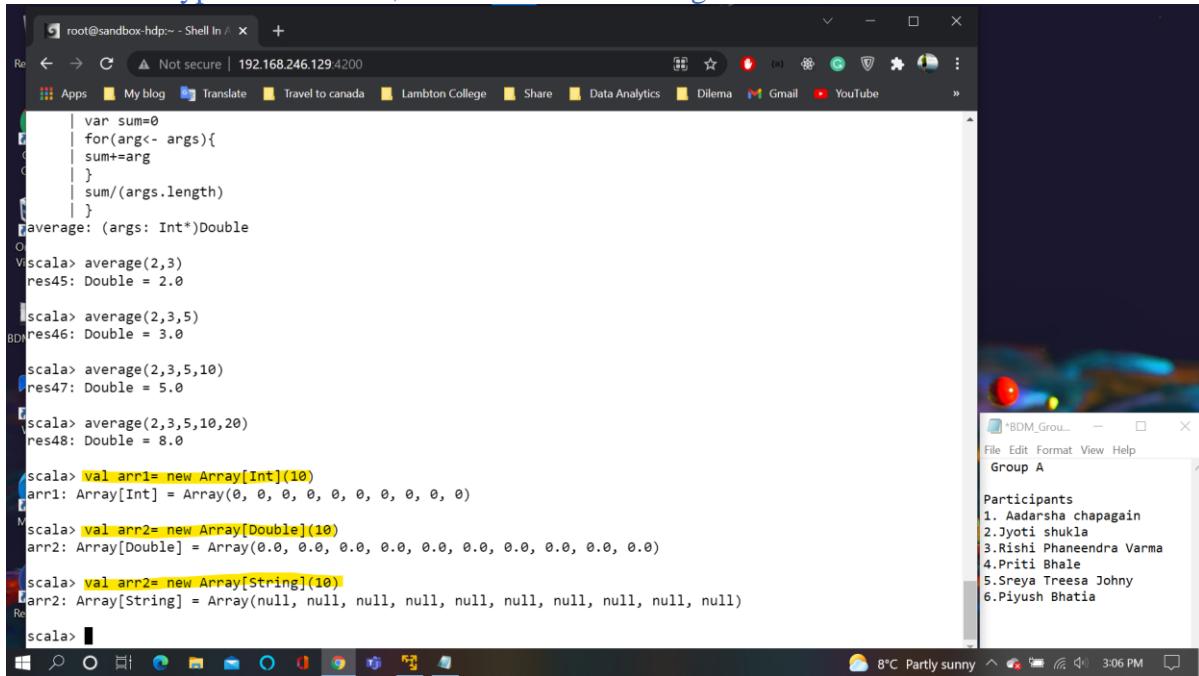
- Participants
- 1. Aadarsha chapagain
- 2.Jyoti shukla
- 3.Rishi Phaneendra Varma
- 4.Priti Bhole
- 5.Sreya Treessa Johnny
- 6.Piyush Bhatia

Slide64:

Internally, Scala arrays are converted into a Java array of the corresponding type

Array is a class

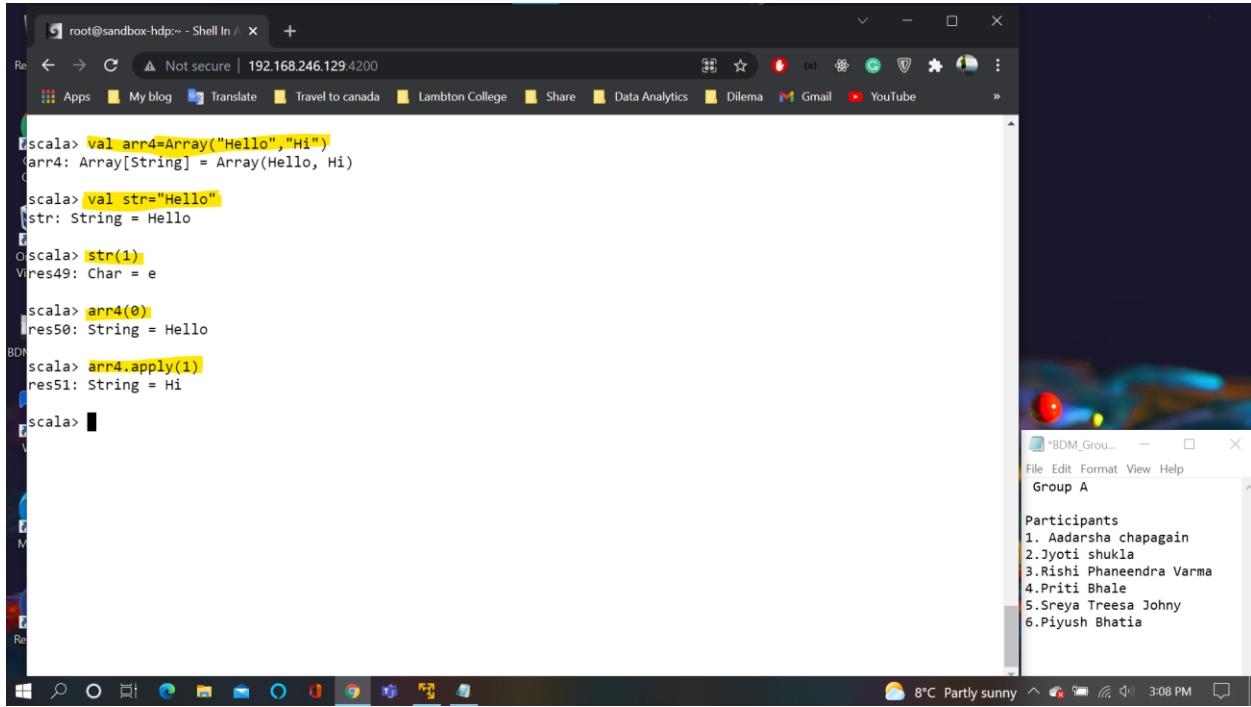
Default data type for inter is 0, double is 0.0 and string is null.



A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code being run. The code includes defining a function 'average' that takes an array of integers and returns a double, and then calling it with various arguments. It also creates three arrays: arr1 (Int), arr2 (Double), and arr3 (String). On the right, a file explorer window titled 'BDM_Grou...' shows a list of participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhave, 5. Sreya Treesa Johny, and 6. Piyush Bhatia. The desktop taskbar at the bottom shows icons for various applications like File Explorer, Task View, and Start.

```
| var sum=0  
| for(arg<- args){  
| sum+=arg  
| }  
| sum/(args.length)  
| }  
average: (args: Int*)Double  
scala> average(2,3)  
res45: Double = 2.0  
  
scala> average(2,3,5)  
res46: Double = 3.0  
  
scala> average(2,3,5,10)  
res47: Double = 5.0  
  
scala> average(2,3,5,10,20)  
res48: Double = 8.0  
  
scala> val arr1= new Array[Int](10)  
arr1: Array[Int] = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)  
  
scala> val arr2= new Array[Double](10)  
arr2: Array[Double] = Array(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)  
  
scala> val arr3= new Array[String](10)  
arr3: Array[String] = Array(null, null, null, null, null, null, null, null, null, null)
```

Slide65:

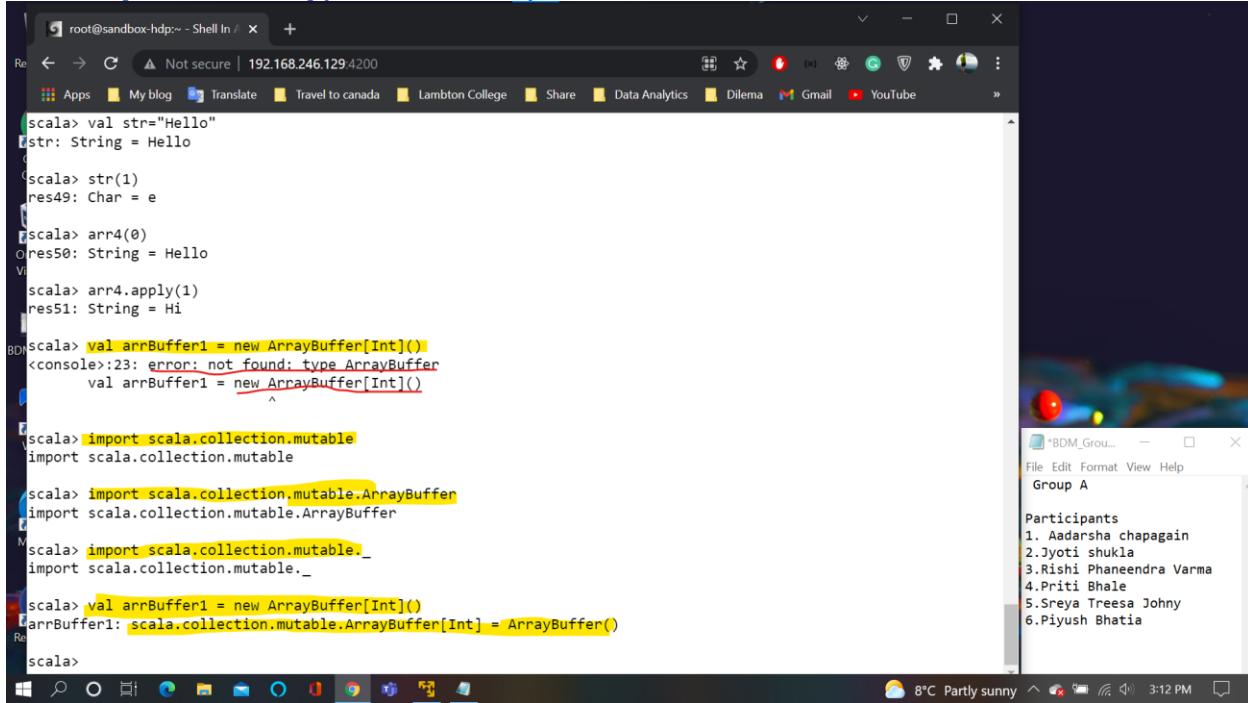


A screenshot of a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code demonstrating array operations. It creates an array arr4 with two elements ("Hello" and "Hi"), prints its first element, prints its second element, and then uses the apply method to print the second element again. On the right, a file explorer window titled 'BDM_Grou...' shows the same list of participants as in the previous slide. The desktop taskbar at the bottom shows icons for various applications.

```
scala> val arr4=Array("Hello","Hi")  
arr4: Array[String] = Array(Hello, Hi)  
  
scala> val str="Hello"  
str: String = Hello  
  
scala> str(1)  
res49: Char = e  
  
scala> arr4(0)  
res50: String = Hello  
  
scala> arr4.apply(1)  
res51: String = Hi  
  
scala>
```

Slide67:

The ArrayBuffer Data type should be imported from scala.collection.mutable

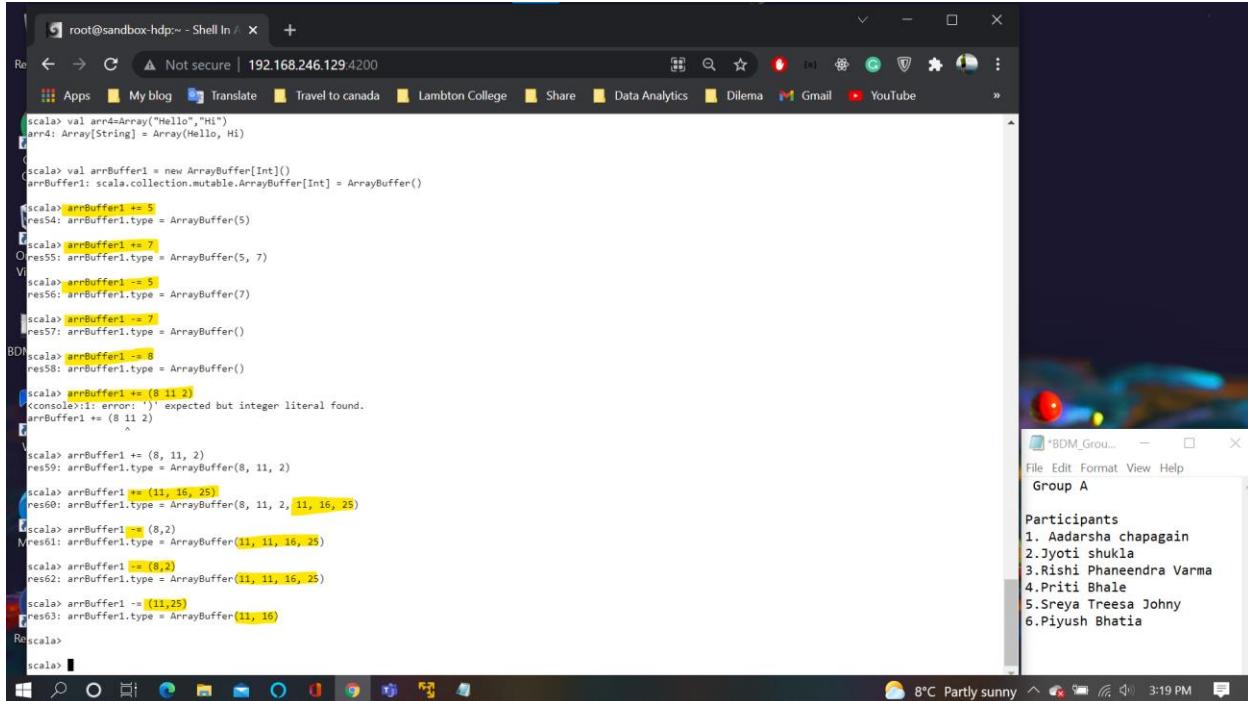


A screenshot of a Linux desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' shows Scala code. The code attempts to create an ArrayBuffer without importing it, resulting in an error. It then imports the mutable collection package and creates a successful ArrayBuffer. On the right, a file explorer window titled 'Group A' lists participants: Adarsha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhole, Sreya Treesa Johnny, and Piyush Bhatia.

```
scala> val str="Hello"
str: String = Hello
|
| scala> str(1)
res49: Char = e
|
| scala> arr4(0)
res50: String = Hello
|
| scala> arr4.apply(1)
res51: String = Hi
|
| scala> val arrBuffer1 = new ArrayBuffer[Int]()
<console>:23: error: not found: type ArrayBuffer
      val arrBuffer1 = new ArrayBuffer[Int]()
                                         ^
|
| scala> import scala.collection.mutable
import scala.collection.mutable
|
| scala> import scala.collection.mutable.ArrayBuffer
import scala.collection.mutable.ArrayBuffer
|
| scala> import scala.collection.mutable._
import scala.collection.mutable._

scala> val arrBuffer1 = new ArrayBuffer[Int]()
arrBuffer1: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer()
```

Slide68:



A screenshot of a Linux desktop environment, similar to the previous slide. The terminal window shows Scala code demonstrating various ArrayBuffer operations like concatenation (+=), copying (:=), and range creation (->). The file explorer window 'Group A' shows the same list of participants: Adarsha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhole, Sreya Treesa Johnny, and Piyush Bhatia.

```
scala> val arr4Array = "Hello","Hi"
arr4: Array[String] = Array(Hello, Hi)
|
| scala> val arrBuffer1 = new ArrayBuffer[Int]()
arrBuffer1: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer()
|
| scala> arrBuffer1 += 3
res4: arrBuffer1.type = ArrayBuffer(3)
|
| scala> arrBuffer1 += 7
res5: arrBuffer1.type = ArrayBuffer(3, 7)
|
| scala> arrBuffer1 -- 3
res6: arrBuffer1.type = ArrayBuffer(7)
|
| scala> arrBuffer1 -- 7
res7: arrBuffer1.type = ArrayBuffer()
|
| scala> arrBuffer1 += 8
res8: arrBuffer1.type = ArrayBuffer()
|
| scala> arrBuffer1 += (8 11 2)
<console>:11: error: ')' expected but integer literal found.
arrBuffer1 += (8 11 2)
                                         ^
|
| scala> arrBuffer1 += (8, 11, 2)
res9: arrBuffer1.type = ArrayBuffer(8, 11, 2)
|
| scala> arrBuffer1 := (11, 16, 25)
res10: arrBuffer1.type = ArrayBuffer(8, 11, 2, 11, 16, 25)
|
| scala> arrBuffer1 -= (8,2)
res11: arrBuffer1.type = ArrayBuffer(11, 11, 16, 25)
|
| scala> arrBuffer1 -- (11,25)
res12: arrBuffer1.type = ArrayBuffer(11, 11, 16)
```

Slide69:

Operations on slide (addition and removal)

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "Not secure | 192.168.246.129:4200". The terminal is running a Scala session. The user has defined three arrays: `aar1` (an array of integers), `aar2` (an array of doubles), and `aar2` (an array of strings). They then define an ArrayBuffer `arrBuffer1` and copy its contents into `aar1`. They perform element-wise addition of `aar1` and `aar2` into `arrBuffer1` using the `+=` operator. They also perform element-wise subtraction using the `-=` operator. Finally, they print the value at index 2 of `aar1`, which is 8. In the background, there is a file explorer window titled "r1Group A" showing a list of participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treessa Johnny, and 6. Piyush Bhatia. The desktop taskbar at the bottom shows various icons for apps like File Explorer, Mail, and Google Chrome.

```
scala> val aar1 = new Array[Int](10)
aar1: Array[Int] = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
scala> val aar2 = new Array[Double](10)
aar2: Array[Double] = Array(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
scala> val aar2 = new Array[String](10)
aar2: Array[String] = Array(null, null, null, null, null, null, null, null, null, null)
scala> arrBuffer1
res66: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 16)
scala> aar1
res67: Array[Int] = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
scala> arrBuffer1 += aar1
res68: arrBuffer1.type = ArrayBuffer(11, 16, 0, 0, 0, 0, 0, 0, 0, 0)
scala> arrBuffer1 -= aar1
res69: arrBuffer1.type = ArrayBuffer(11, 16)
scala> aar1(2)=8
scala>
```

Slide71:

Different operation on array

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "Not secure | 192.168.246.129:4200". The terminal is running a Scala session. The user defines an ArrayBuffer `arrBuffer1` and adds elements 10 and 10 to it using the `+=` and `+=` operators. They then add elements 10 and 17 to the array using the `+=` and `+=` operators. They insert elements 1, 10, 16, 17, 10, 16 into the array using the `insert` method. They also insert elements 1, 5, 6, 7, 8 into the array using the `insert` method. They remove element 7 from the array using the `-=` operator. Finally, they remove element 1 from the array using the `remove` method. In the background, there is a file explorer window titled "r1Group A" showing a list of participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treessa Johnny, and 6. Piyush Bhatia. The desktop taskbar at the bottom shows various icons for apps like File Explorer, Mail, and Google Chrome.

```
scala> var arrbuffer = new ArrayBuffer[Int]
arrbuffer: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer()
scala> arrBuffer1
res71: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 16)
scala> arrBuffer1 += 10
res72: arrBuffer1.type = ArrayBuffer(11, 16, 10)
scala> arrBuffer1 -= 10
res73: arrBuffer1.type = ArrayBuffer(11, 16)
scala> arrBuffer1.insert(1,10)
res74: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 10, 16)
scala> arrBuffer1.insert(1,17)
res75: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 17, 10, 16)
scala> arrBuffer1.insert(1,5,6,7,8)
res76: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 5, 6, 7, 8, 17, 10, 16)
scala> arrBuffer1 -= 7
res78: arrBuffer1.type = ArrayBuffer(11, 5, 6, 8, 17, 10, 16)
scala> arrBuffer1.remove(1)
res81: Int = 5
```

Slide74:

More Operations on Array

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "Not secure | 192.168.246.129:4200". The terminal is running a Scala session where various operations on an ArrayBuffer are demonstrated. In the background, a file explorer window titled "r1Group A" is open, displaying a list of participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treessa Johnny, and 6. Piyush Bhatia.

```
scala> arrBuffer1
res88: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(11, 6, 8, 17, 10, 16)
scala> arrBuffer1.sorted
res89: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(6, 8, 10, 11, 16, 17)
scala> arrBuffer1.mkString
res90: String = 1168171016
V
scala> arrBuffer1.mkString(",")
res91: String = 11,6,8,17,10,16
scala> arrBuffer1.mkString(" ")
res92: String = 11 6 8 17 10 16
scala> arrBuffer1.mkString(">")
res93: String = 11>6>8>17>10>16
scala> arrBuffer1.mkString("[", ">", "]")
res94: String = [11>6>8>17>10>16]
scala> arrBuffer1.mkString("*****", ">", "*****")
res95: String = *****11>6>8>17>10>16*****
```

```
Mscala> arrBuffer1.sum
res96: Int = 68
scala>
```

Slide76:

On clicking tab all the available function wil be displayed

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "Not secure | 192.168.246.129:4200". The terminal is running a Scala session where the user types "arrBuffer1." followed by a tab key. This triggers an auto-completion feature that lists numerous methods available for the ArrayBuffer type. In the background, a file explorer window titled "r1Group A" is open, displaying a list of participants: 1. Aadarsha chapagain, 2. Jyoti shukla, 3. Rishi Phaneendra Varma, 4. Priti Bhole, 5. Sreya Treessa Johnny, and 6. Piyush Bhatia.

```
scala> arrBuffer1.
          find Tab
  !=      asInstanceOf      find
  ###    canEqual           flatMap
  +      clear              flatten
  ++     clone              fold
  +++=   collect             foldLeft
  O++=   collectFirst       foldRight
  Vi+   combinations        forall
  companion         foreach
  +=     compose            formatted
  +==   contains           genericBuilder
  -     containsSlice       getClass
  --    copy               groupBy
  --=   copyToArray         grouped
  -=   copyToBuffer        hasDefiniteSize
  >    corresponds       hashCode
  /:    count              head
  :+    diff               headOption
  :\    distinct           indexOf
  <<   drop               indexOfSlice
  ==   dropRight          indexWhere
  WithFilter dropWhile       indices
  addString endsWith       init
  aggregate ensuring       inits
  MandThen eq              insert
  append equals           insertAll
  appendAll exists          intersect
  apply filter            isDefinedAt
  applyOrElse filterNot    isEmpty
```

```
isInstanceOf      prefixLength      seq      toList
isTraversableAgain  prepend      size      toMap
iterator           prependAll    sizeHint  toParArray
lastIndex          product      sizeHintBounded toSeq
lastIndexOf        productArity  slice      toSet
lastIndexOfSlice  productElement  slice    toStream
lastIndexWhere     productIterator sortBy    toString
lastOption         productPrefix  sortWith  toTraversable
length            readOnly      sorted    toVector
lengthCompare     reduce      span      transform
lift              reduceLeft    splitAt  transpose
map               reduceLeftOption startsWith trimEnd
mapResult          reduceOption  stringPrefix trimStart
max               reduceRight  sum      union
maxBy             reduceRightOption synchronized unzip
min               reduceToSize  tail      unzip3
minBy             remove      tails    update
ne                result      take    view
nonEmpty          reverse      takeRight  wait
notify            notifyAll    takeWhile withFilter
notifyAll         reverseMap  toArray    zip
notifyAll         runwith     toBuffer  zipAll
orElse            padTo      sameElements  zipWithIndex
partition          par      scan      toDS
patch             patch      scanLeft  toIndexedSeq
permutations       permutations  scanRight toIterable
segmentLength     segmentLength  toIterator
```

```
Rescala> arrBuffer1.
```

Slide78:

Fixed and variable length Array

The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' is open, displaying Scala code related to arrays. On the right, a Microsoft Teams chat window titled 'r1Group A' is visible, showing a list of participants: Adarsha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhole, Sreya Treesa Johnny, and Piyush Bhatia. The desktop taskbar at the bottom shows various icons for applications like File Explorer, Task View, and the Start button.

```
scala> var multiDimArr = Array.ofDim[Int](4,3)
multiDimArr: Array[Array[Int]] = Array(Array(0, 0, 0), Array(0, 0, 0), Array(0, 0, 0), Array(0, 0, 0))
scala> multiDimArr
res97: Array[Array[Int]] = Array(Array(0, 0, 0), Array(0, 0, 0), Array(0, 0, 0), Array(0, 0, 0))
scala> multiDimArr(1)
res98: Int = 0
scala> multiDimArr(1)(1)=25
scala> multiDimArr(1)(1)
res100: Int = 25
BDM
scala> val varLengthArr= new Array[Array[Int]](10)
varLengthArr: Array[Array[Int]] = Array(null, null, null, null, null, null, null, null, null, null)
scala> varLengthArr(0)=Array(1,2,4)
scala> varLengthArr
res102: Array[Array[Int]] = Array(Array(1, 2, 4), null, null, null, null, null, null, null, null, null)
scala> varLengthArr(5)=Array(10,20,4)
Mscala> varLengthArr
res104: Array[Array[Int]] = Array(Array(1, 2, 4), null, null, null, null, Array(10, 20, 4), null, null, null, null)
scala> 
```

Slide80:

The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'root@sandbox-hdp:~ - Shell In' is open, displaying Scala code related to maps. On the right, a Microsoft Teams chat window titled 'r1Group A' is visible, showing a list of participants: Adarsha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhole, Sreya Treesa Johnny, and Piyush Bhatia. The desktop taskbar at the bottom shows various icons for applications like File Explorer, Task View, and the Start button.

```
scala> var mutMap = Map(1-> "Hello", 2->"Hi", 3->"Welcome")
mutMap: scala.collection.mutable.Map[Int,String] = Map(2 -> Hi, 1 -> Hello, 3 -> Welcome)
scala> var map = Map(1-> "Hello", 2->"Hi", 3->"Welcome")
map: scala.collection.mutable.Map[Int,String] = Map(2 -> Hi, 1 -> Hello, 3 -> Welcome)
scala> map(1)
res105: String = Hello
scala> map.apply(1)
res106: String = Hello
scala> map(3)
res107: String = Welcome
scala> map.apply(3)
res108: String = Welcome
scala> map.apply(4)
java.util.NoSuchElementException: key not found: 4
  at scala.collection.MapLike$class.default(MapLike.scala:228)
  at scala.collection.AbstractMap.default(Map.scala:59)
  at scala.collection.mutable.HashMap.apply(HashMap.scala:65)
  ... 51 elided
Mscala> map(1) = "Bonjour"
scala> map
res111: scala.collection.mutable.Map[Int,String] = Map(2 -> Hi, 1 -> Bonjour, 3 -> Welcome)
scala> 
```

Slide81:

`mapOrElseGet`: If the value is present get the value else get the specified default value

A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In" shows Scala code demonstrating the `mapOrElseGet` method. On the right, a Notepad window titled "BDM_Group_A" displays a list of participants.

```
import scala.collection.immutable.Map

scala> map.
  + collectFirst foldLeft keys reduceLeftOption stringPrefix transpose
  ++ companion foldRight keyIterator reduceOption sum
  ++: compose forall last reduceRight tail
  ++= contains foreach lastOption reduceRightOption tails
  - copyToArray genericBuilder lift remove take
  - copyToBuffer get map repr takeRight updated
  O-- count getOrElse mapResult retain to valuesIterator
  V-- drop groupBy max runWith toArray
  /: dropRight grouped maxBy sameElements toBuffer withDefault
  \: dropWhile hasDefiniteSize min scan toIndexedSeq withDefaultValue
  WithFilter empty hashCode head mkString scanLeft toIterable
  addString equals headOption nonEmpty seq toIterator zip
  aggregate exists headOption orElse size toList zipAll
  andThen filter init partition sizeHintBounded toMap zipWithIndex
  apply filterKeys inits par sizeHint toSeq
  applyOrElse filterNot isDefinedAt product slice toStream
  canEqual find isEmpty put reduce span toTraversable
  clear flatMap isTraversableAgain put reduceLeft splitAt toVector
  clone flatten iterator reduceLeft
  collect fold keySet reduceLeft

scala> map.get
get getClass getOrElse getOrElseUpdate

scala> map.getOrElse("Default Value")
res112: String = Default Value
M
scala> map.getOrElse(3, "Default Value")
res113: String = Welcome

scala> map(4)
java.util.NoSuchElementException: key not found: 4
  at scala.collection.MapLike$class.default(MapLike.scala:228)
Re  at scala.collection.AbstractMap.default(Map.scala:59)
  at scala.collection.mutable.HashMap.apply(HashMap.scala:65)
... 52 elided
```

Participants

1. Adarsha chapagain
2. Jyoti shukla
3. Rishi Phaneendra Varma
4. Priti Bhole
5. Sreya Treesa Johnny
6. Piyush Bhatia

Slide82:

A screenshot of a Windows desktop environment. On the left, a terminal window titled "root@sandbox-hdp:~ - Shell In" shows Scala code demonstrating various string representation methods for maps. On the right, a Notepad window titled "BDM_Group_A" displays a list of participants.

```
scala> map.mkString
res116: String = 2 -> Hi1 -> Bonjour3 -> Welcome
|
scala> map.mkString(" ")
res117: String = 2 -> Hi 1 -> Bonjour 3 -> Welcome
|
scala> map.mkString(": ")
res118: String = 2 -> Hi: 1 -> Bonjour: 3 -> Welcome

scala> map.keySet
<console>:34: error: value KeySet is not a member of scala.collection.mutable.Map[Int,String]
      map.KeySet
               ^

scala> map.keySet
res120: scala.collection.Set[Int] = Set(2, 1, 3)

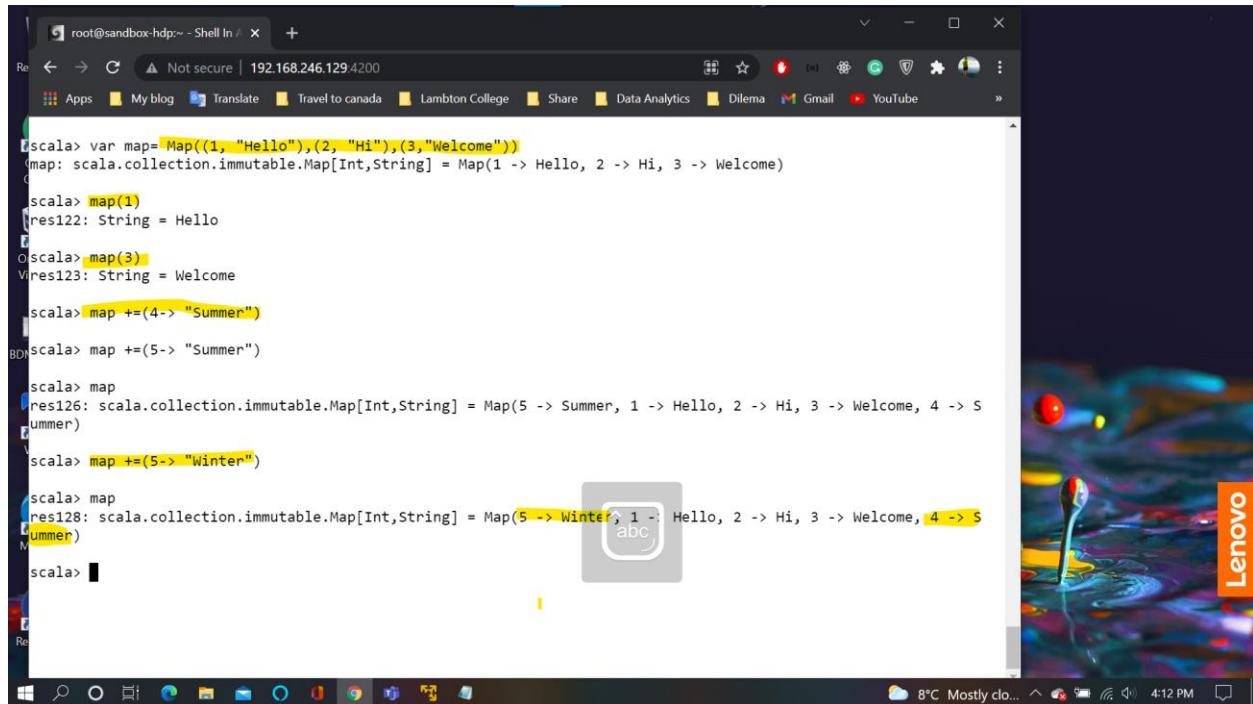
scala> for(key <- map.keySet)
    | {println("key:" + key + ",Value:" + map(key))
    | }
key:2,Value:Hi
key:1,Value:Bonjour
Mkey:3,Value:Welcome

scala>
```

Participants

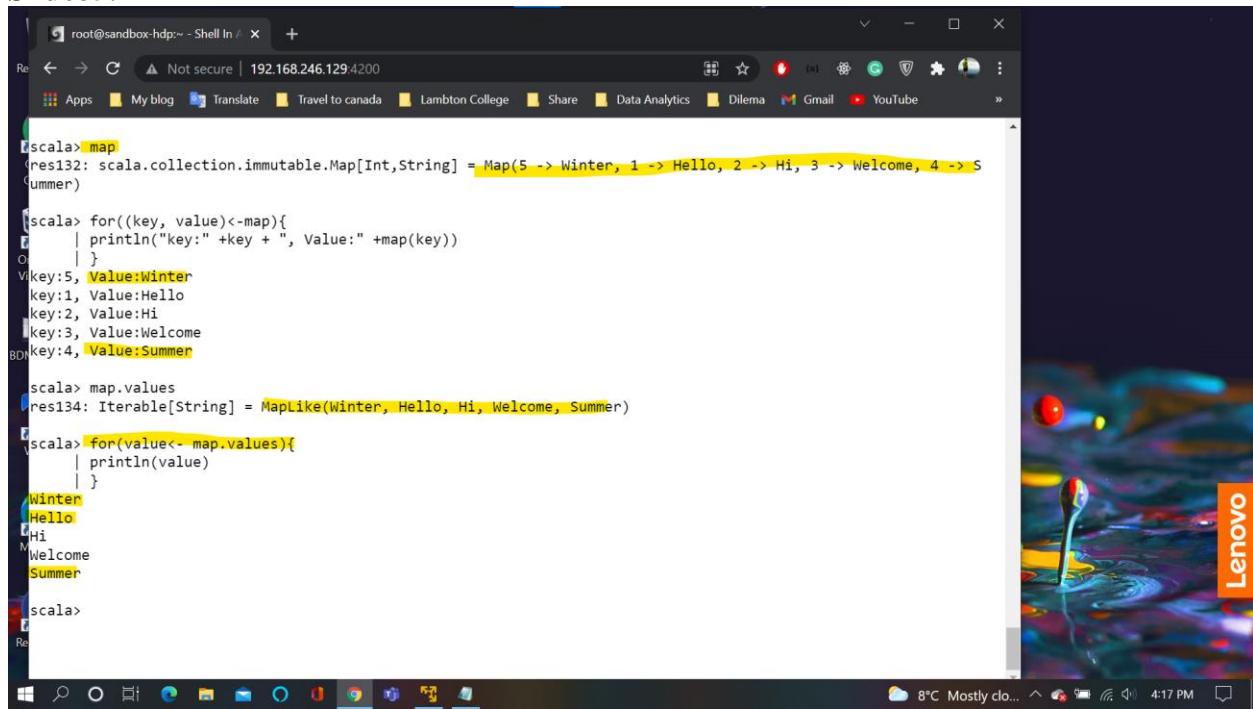
1. Adarsha chapagain
2. Jyoti shukla
3. Rishi Phaneendra Varma
4. Priti Bhole
5. Sreya Treesa Johnny
6. Piyush Bhatia

Slide84:



```
scala> var map=Map((1,"Hello"),(2,"Hi"),(3,"Welcome"))
map: scala.collection.immutable.Map[Int,String] = Map(1 -> Hello, 2 -> Hi, 3 -> Welcome)
scala> map(1)
res122: String = Hello
scala> map(3)
res123: String = Welcome
scala> map +=(4-> "Summer")
scala> map +=(5-> "Summer")
scala> map
res126: scala.collection.immutable.Map[Int,String] = Map(5 -> Summer, 1 -> Hello, 2 -> Hi, 3 -> Welcome, 4 -> Summer)
scala> map +=(5-> "Winter")
scala> map
res128: scala.collection.immutable.Map[Int,String] = Map(5 -> Winter, 1 -> Hello, 2 -> Hi, 3 -> Welcome, 4 -> Summer)
```

Slide85:

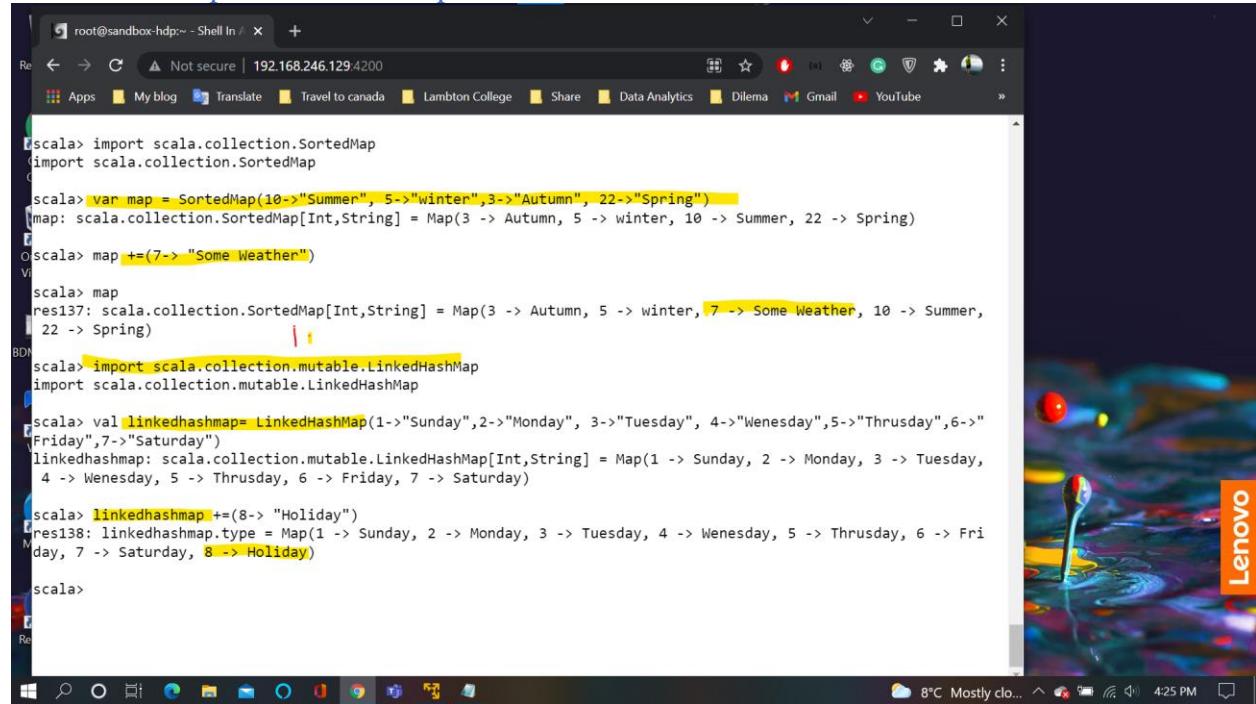


```
scala> map
res132: scala.collection.immutable.Map[Int,String] = Map(5 -> Winter, 1 -> Hello, 2 -> Hi, 3 -> Welcome, 4 -> Summer)
scala> for((key, value)<-map){
    |   println("key:" +key + ", Value:" +map(key))
    | }
Value:Winter
key:1, Value:Hello
key:2, Value:Hi
key:3, Value:Welcome
key:4, Value:Summer
scala> map.values
res134: Iterable[String] = MapLike(Winter, Hello, Hi, Welcome, Summer)
scala> for(value<- map.values){
    |   println(value)
    | }
Winter
Hello
Hi
Welcome
Summer
```

Slide87:

The map entries would be sorted based on key

LinkedHashMap will create a map that will retain the insertion order of element



root@sandbox-hdp:~ - Shell In +

Re ← → C Not secure | 192.168.246.129:4200

Apps My blog Translate Travel to canada Lambton College Share Data Analytics Dilema Gmail YouTube

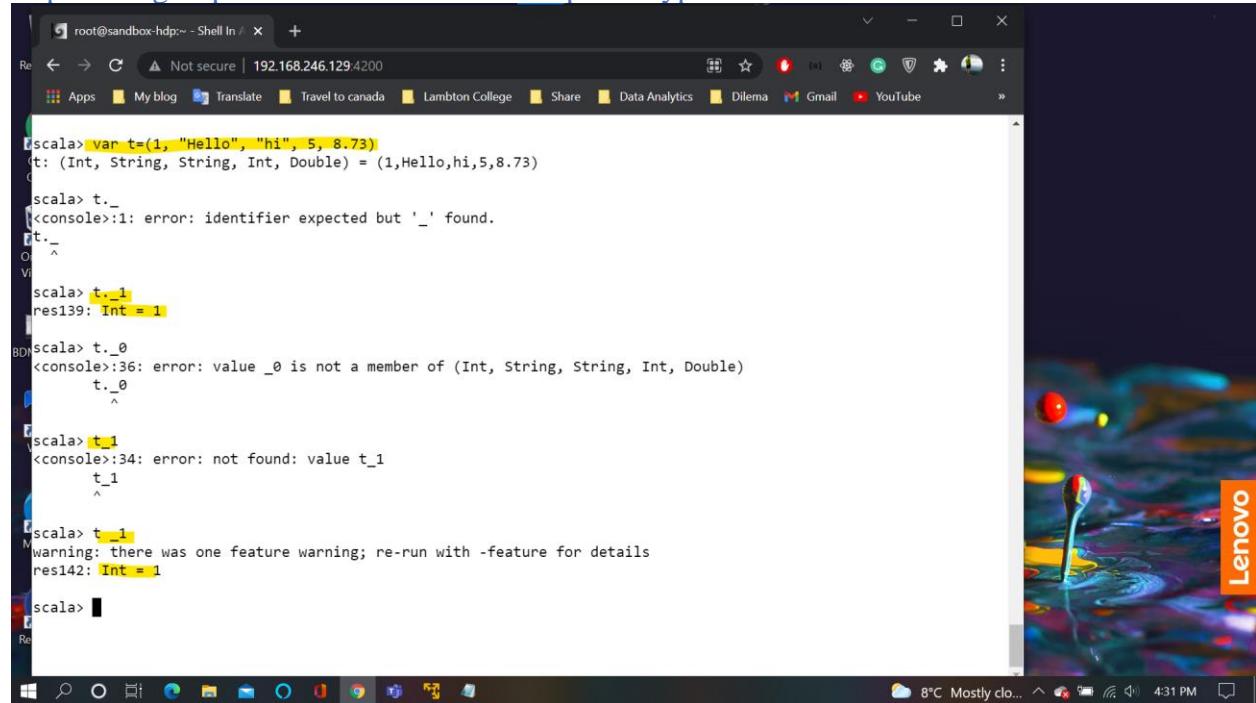
```
scala> import scala.collection.SortedMap
import scala.collection.SortedMap
scala> var map = SortedMap(10->"Summer", 5->"winter",3->"Autumn", 22->"Spring")
map: scala.collection.SortedMap[Int, String] = Map(3 -> Autumn, 5 -> winter, 10 -> Summer, 22 -> Spring)
scala> map +=(7-> "Some Weather")
res136: scala.collection.SortedMap[Int, String] = Map(3 -> Autumn, 5 -> winter, 7 -> Some Weather, 10 -> Summer, 22 -> Spring)
BDR
scala> import scala.collection.mutable.LinkedHashMap
import scala.collection.mutable.LinkedHashMap
scala> val linkedhashmap= LinkedHashMap(1->"Sunday",2->"Monday", 3->"Tuesday", 4->"Wednesday",5->"Thursday",6->"Friday",7->"Saturday")
linkedhashmap: scala.collection.mutable.LinkedHashMap[Int, String] = Map(1 -> Sunday, 2 -> Monday, 3 -> Tuesday, 4 -> Wednesday, 5 -> Thursday, 6 -> Friday, 7 -> Saturday)
scala> linkedhashmap.+(8-> "Holiday")
res138: linkedhashmap.type = Map(1 -> Sunday, 2 -> Monday, 3 -> Tuesday, 4 -> Wednesday, 5 -> Thursday, 6 -> Friday, 7 -> Saturday, 8 -> Holiday)
scala>
```

Lenovo

8°C Mostly clo... 4:25 PM

Slide88:

Tuples are group of value that can be of disparate type



root@sandbox-hdp:~ - Shell In +

Re ← → C Not secure | 192.168.246.129:4200

Apps My blog Translate Travel to canada Lambton College Share Data Analytics Dilema Gmail YouTube

```
scala> var t=(1, "Hello", "hi", 5, 8.73)
t: (Int, String, String, Int, Double) = (1,Hello,hi,5,8.73)
scala> t._1
<console>:1: error: identifier expected but '_' found.
          t._1
          ^
V
scala> t._1
res139: Int = 1
scala> t._0
<console>:36: error: value _0 is not a member of (Int, String, String, Int, Double)
          t._0
          ^
V
scala> t._1
<console>:34: error: not found: value t_1
          t_1
          ^
V
scala> t._1
Warning: there was one feature warning; re-run with -feature for details
res142: Int = 1
scala>
```

Lenovo

8°C Mostly clo... 4:31 PM