databricks Scala basic Notebook 2023-08-07 11:12:22

(https://databricks.com)

```
var num= List(1,2,3,4)
```

```
num: List[Int] = List(1, 2, 3, 4)
```

```
num.head
```

```
res0: Int = 1
```

```
// first two element are show
num.take(2)
```

```
res3: List[Int] = List(1, 2)
```

```
num.sum
```

```
res4: Int = 10
```

```
var dev = List(1,1,1,2,3,4,2,2)
```

```
dev: List[Int] = List(1, 1, 1, 2, 3, 4, 2, 2)
```

```
// this element show only value to not same like 1,1,2,2,3,4,4 == 1,2,3,4
dev.distinct
```

```
res7: List[Int] = List(1, 2, 3, 4)
```

```
num(0)
```

```
res8: Int = 1
```

```
num(1)= 10
```

```
command-1298605597657410:1: error: value update is not a member of List[Int]
num(1)= 10
^
```

```
dev.size
```

```
res10: Int = 8
```

```
dev.reverse
```

```
res11: List[Int] = List(2, 2, 4, 3, 2, 1, 1, 1)
```

```
dev.min
```

```
res12: Int = 1
```

```
dev.max
```

```
res13: Int = 4
```

```
dev.isEmpty
```

```
res14: Boolean = false
```

```
var dev1 = List()
dev1.isEmpty
```

```
dev1: List[Nothing] = List()
res16: Boolean = true
```

```
var number = Array(1,2,3,4,5,6,7)
```

```
number: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7)
```

```
var lang = Array("Scall", "Python","Spark")
```

```
lang: Array[String] = Array(Scall, Python, Spark)
```

```
lang.head
```

```
res17: String = Scall
```

```
lang.tail
```

```
res18: Array[String] = Array(Python, Spark)
```

```
number(1)=10
```

```
number
```

```
res20: Array[Int] = Array(1, 10, 3, 4, 5, 6, 7)
```

```
import scala.collection.mutable.ArrayBuffer
```

```
import scala.collection.mutable.ArrayBuffer
```

```
var cars = new ArrayBuffer[String]()
```

```
cars: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer()
```

```
cars +="BMW"
```

```
res23: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW)
```

```
cars +="Jaguar"
cars +="Jaguar"
```

```
res24: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar)
```

```
cars +="TATA"
```

```
res25: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar, TATA)
```

```
cars.length
```

```
res26: Int = 4
```

```
cars.trimEnd(1)
```

```
cars
```

```
res28: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar)
```

```
cars.insert(2,"Bentley")
```

```
cars
```

```
res30: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Bentley, Jaguar)
```

```
cars.insert(4,"Bugatti")
```

```
cars
```

```
res33: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Bentley, Jaguar, Bugatti)
```

```
val num= List(1,2,3,4,5)
```

```
num: List[Int] = List(1, 2, 3, 4, 5)
```

```
num.map(x => x*x)
```

```
res34: List[Int] = List(1, 4, 9, 16, 25)
```

```
num.map(y => y*y)
```

```
res35: List[Int] = List(1, 4, 9, 16, 25)
```

```
  val a = num.map(x => x+1)
```

```
a: List[Int] = List(2, 3, 4, 5, 6)
```

```
  // Nested map
  val b = a.map(x => x*x)
```

```
b: List[Int] = List(4, 9, 16, 25, 36)
```

```
  val c = b.map(x => x-1)
```

```
c: List[Int] = List(3, 8, 15, 24, 35)
```

```
  val d = c.map(x => -x)
```

```
d: List[Int] = List(-3, -8, -15, -24, -35)
```

```
  num.map(x => x+1).map(x => x*x).map(x => x-1).map(x => -x)
```

```
res36: List[Int] = List(-3, -8, -15, -24, -35)
```

```
  val fruits = List("orange", "banana", "Apple")
```

```
fruits: List[String] = List(orange, banana, Apple)
```

```
  // Length of the words
  fruits.map(x => (x,x.length))
```

```
res39: List[(String, Int)] = List((orange,6), (banana,6), (Apple,5))
```

```
  // how many corrector of the words of their like banana word is geater than 5.
  fruits.filter(x => x.length > 5)
```

```
res40: List[String] = List(orange, banana)
```

```
  val ratings=List(2.4,5.6,7.8,9.5)
```

```
ratings: List[Double] = List(2.4, 5.6, 7.8, 9.5)
```

```
  fruits.filter(x => x.length == 5)
```

```
res42: List[String] = List(Apple)
```

```
  fruits.filter(x => x.length != 5)
```

```
res43: List[String] = List(orange, banana)
```

```
  val marks=ratings.map(x => x+10)
```

```
marks: List[Double] = List(12.4, 15.6, 17.8, 19.5)
```

```
  val marks=ratings.map(x => x*10)
```

```
marks: List[Double] = List(24.0, 56.0, 78.0, 95.0)
```

```
  val marks1=ratings.filter(x=> x>=60 && x<=74)
```

```
marks1: List[Double] = List()
```

```
  // Funtion in Scala
```

```
  def add(a:Double = 100 , b:Double = 200) : Double =
  {
    var sum : Double = 0
    sum = a+b
    return sum
  }
  println("Sum:" + add(a = 10, b=20))
```

```
Sum:30.0
add: (a: Double, b: Double)Double
```

```
  // if value is not given in add funcation return add funcation of a = 100 and b = 200
  println("Sum1:" + add())
```

```
Sum1:300.0
```

```
  var x = 1
    val y = if(x<3)
      println("value of x is less than 3")
    else
      println("value of x is greater than or equal to 3")
```

```
value of x is less than 3
x: Int = 1
y: Unit = ()
```

```
  var marks = 75
    if(marks>=75){
      println("Distinction")
    }else if(marks >= 60 && marks <70){
      println("First class")
    }else if(marks >= 50 && marks <60){
      println("Seocnd class")
    }else if(marks >= 40 && marks <50){
      println("Pass class")
    }else println("fail")
```

```
Distinction
marks: Int = 75
```

```
  // 3 funcations are calls and returns 3 times
  def square(x:Double) :Double = {
    return x*x
  }
  def sumsquare(x:Double, y:Double) :Double = {
    return square(x) + square(x)
  }
  println("Sum of squares:"+ sumsquare(4,5))
```

```
Sum of squares:32.0
square: (x: Double)Double
sumsquare: (x: Double, y: Double)Double
```

```
  def time() :Long = {
    println("Inside the funcations")
    return System.nanoTime()
  }
  def exect(t:Long) :Long = {
    println("Inside the Exect funcations")
    println("Time :"+t)
    println("Exiting From exect funcations")
    return t
  }
  println("Main Functions:"+ exect(time()))
```

```
Inside the funcations
Inside the Exect funcations
Time :4627384344930
Exiting From exect funcations
Main Functions:4627384344930
time: ()Long
exect: (t: Long)Long
```

```
  var i =10
    while(i>0){
      println("hello :"+i)
      i = i-1
    }
```

```
hello :10
hello :9
hello :8
hello :7
hello :6
hello :5
hello :4
hello :3
hello :2
```

```
hello :1
i: Int = 0
```

```
  var a =2
    do{
      println(a)
      a = a +2
    }while(a <= 10)
```

```
2
4
6
8
10
a: Int = 12
```

```
  ;// how to create object and how to create a class

  object classEg{
    def main(arg:Array[String]){
      var obj = new NewClass("Hello World")
      obj.sayHi()
    }
  }

  class NewClass(mssg:String){
    def sayHi()= println(mssg)
  }
```

```
defined object classEg
defined class NewClass
```

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

```
1
2
3
4
5
6
7
8
9
10
```

```
  // factorial number

  var Factorial =1
    var num =5
    while(num>0){
      Factorial = Factorial * num
      num = num-1
    }
    println(Factorial)
```

```
120
Factorial: Int = 120
num: Int = 0
```

```
  def isPrime(i: Int): Boolean =
      if (i <= 1)
          false
      else if (i == 2)
          true
      else
          !(2 until i).exists(n => i % n == 0)
```

```
isPrime: (i: Int)Boolean
```