Semana 6

1. Función para calcular desviación estándar de un grupo de números

val nros = List(727.7, 1086.5, 1361.3, 1490.5, 1956.1)

def promedio(valores : List[Double]) : Double = valores.sum / valores.size

def devEst(valores : List[Double]) : Double = {

val avg = promedio(valores)

def varianza(valores: List[Double]) : Double =

valores.map(x => Math.pow(x - avg, 2)).sum \* (1.0/ valores.size)

Math.sqrt(varianza(valores))

}

def clasifica(valores : List[Double]): Unit ={

val s = devEst(valores)

val avg = promedio(valores)

(1 to 3).foreach(i =>

println(valores.filter(x =>

x >= (avg - i \* s) && x <= (avg + i \* s)).size))

}

1. Tuplas

Definir una Tupla

val par = (1, "Abad Ana")

Llamar a diferentes datos de la Tupla

println(par.\_1)

println(par.\_2)

val student = (1, "Abad Ana")

val datos = ("jorgaf", 'M', 45, 95.2, true)

1. Algunas funciones con Tuplas

val student2 = (1, "Abad Ana")

val (edad, nombre) = student2

print(edad)

print(nombre)

student2.swap

student2.canEqual((1, "Abad Ana"))

Estructuras de datos funcionales

1. Listas

val values = List(1, 2, 3, 4)

val names = List("jorge", "Pepe")

val values2: List[Int] = List(1, 2, 3, 4)

val names2: List[String] = List("jorge", "Pepe")

1. Diferentes Operaciones de las listas

val myList = List(10, 20, 30, 40, 50, 60)

myList.isEmpty

myList.drop(2)

myList.dropWhile(\_ < 25)

myList.slice(2, 4)

myList.tail

myList.take(3)

myList.takeWhile(\_ < 30)

myList.sorted

1. Otras operaciones con listas

Va desde la izquierda hasta la derecha

myList.foldLeft(0)((x, y) => x + y)

x almacena el valor y Y es el valor actual de la lista

myList.foldLeft(0)(\_ + \_)

myList.foldRight(0)((x, y) => x + y)

myList.foldRight(0)(\_ + \_)

1. Operaciones con foldLeft()

def average(valores : Seq[Double]): Double = {

val t = valores.foldLeft((0.0, 0))((acc, currVal) => (acc.\_1 + currVal, acc.\_2 + 1))

t.\_1 / t.\_2

}