Scala Notes

用var或val声明变量

* val声明的变量必须在声明时就要进行初始化，并且以后不能被重新赋值
* var更像是java里的正常变量声明，可以背重新赋值
* scala的理念中，所有的变量都应该用val声明

函数定义：

* def+函数名+圆括号括起来的参数列表+等号+函数体（如果多行要使用花括号）
* 匿名函数：(name: String) => “Hello” + name，可以把这个函数赋值给一个变量，然后通过这个变量调用这个函数
* 可以传递一个函数给另外一个参数有类型要求的函数：
  + List(“Rob”, “Jane”, “Frieddie”).map((name)=> “Hello”+name).foreach(println)
* 函数的返回类型是由函数最后一行语句的执行结果值决定的

元组：

* scala不支持多个返回值，但是scala支持能提供相似能力的元组语法，从Tuple2到Tuple22。可以通过成员变量\_1、\_2来访问元组中的元素。
  + val t=Tuple2(“Rod”,3) 或 val t=(“Rod”,3)
  + println(t.\_1 + “has” + t.\_2 + “coconuts”)
  + Rod has 3 coconuts

类：

* class myClass (val name: String, id: Integer=0){
  + def make Message= “Hi, I’m a” + name + “with id” + id
* }
* val y1=new myClass(“cat”,3) #name为cat，id为3
* val y2=new myClass(“dog”) #name为dog，id为0

Map和Reduce：

* map()函数将传入的一个转换函数作为参数，把一个集合转变成另外一个集合。
* reduce()函数传入一个两两合并的聚集函数作为参数((Int,Int)=>Int)，把一个集合聚集为单个值

class fruitCount(val name: String, val num: Int)

val groceries: List[fruitCount]=List(

new fruitCount(“banana”,1),

new fruitCount(“apple”,2),

new fruitCount(“orange”,7),

new fruitCount(“pear”,8)

)

groceries.map(f=>f.num).reduce((a: Int, b: Int)=>a+b)

* 用下划线避免对匿名函数的参数命名
  + groceries.map(\_.num)
  + \_+\_ 用法
    - 下划线在scala中时通配符
    - 第一个下划线表示第一个参数
    - 第二个下划线表示第二个参数
    - 这是(a,b)=>(a+b)的 简写

if/else:

* scala里的if/else返回一个结果值，与java中的三元操作符“?”类似:
  + val s = if (2.3>2.2) “Bigger” else “Smaller”

模式匹配：

* 模式匹配会返回一个结果值，采用的是中缀表达式。

class parserState

case class mantissaState() extends parserState

case class fractionalState() extends parserState

case class exponentState() extends parserState

def stateMantissaConsume(c:Char)=c match {

case ‘.’ => fractionalState

case ‘E’ => exponentState

case \_ => mantissaState

}

函数式编程语言中两种天然的迭代方式时map()和递归。