尚硅谷大数据技术之 Scala
基础语法

官网：www.atguigu.com

ShangGuigu Technologies Co., Ltd.

尚硅谷技术有限公司

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

一　基础语法

Scala 语言是基于 Java 虚拟机运行的，所以基本的语法和 Java 是没有区别的。但是为了简化 Java 的开发以及融合其他的编程语言的优点和特性，Scala 在基本的语法上做了大量的修改和优化，让程序员开发起来更简单，方便，灵活。

1.1 标识符

Scala 中的标识符声明，基本和 Java 是一致的，但是细节上会有所变化。

- 首字符为字母，后续字符任意字母和数字，美元符号，可后接下划线_
- **首字符为操作符，后续字符为任意操作符**
- 用反引号`....`包括的任意字符串，即使是关键字也可以
- Scala 中有些保留字，不能用作标识符，但是反引号括起除外，如 yield 在 Scala 中就是保留字

1.2 变量

Scala 声明变量有两种方式，一个用 val，一个用 var。

val / var 变量名 ： 变量类型 = 变量值。

val 定义的值是不可变的，它不是一个常量，是不可变量，或称之为只读变量。

val 示例：

```
scala> val a1 = 10
scala> a1 = 20（X）
```

var 示例：

```
scala> var a2 = 10
scala> a2 = 20（OK）
```

1.3 常用类型

Scala 语言是完全面向对象的语言，所以并不区分基本类型和引用类型，这些类型都是对象，我们称之为常用类型。

Scala 常用类型中包含有 7 种数值类型：Byte、Char、Short、Int、Long、Float、Double

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

及 Boolean 类型，还有 String 类型。

Boolean	true 或者 false
Byte	8 位，有符号
Short	16 位，有符号
Int	32 位，有符号
Long	64 位，有符号
Char	16 位，无符号
Float	32 位，单精度浮点数
Double	64 位，双精度浮点数
String	其实就是由 Char 数组组成

注：既然 Java 的基本类型数据在 Scala 中都是对象，那么也就意味着 Java 中基本类型的数据在 Scala 中都是可以调用对象方法的。

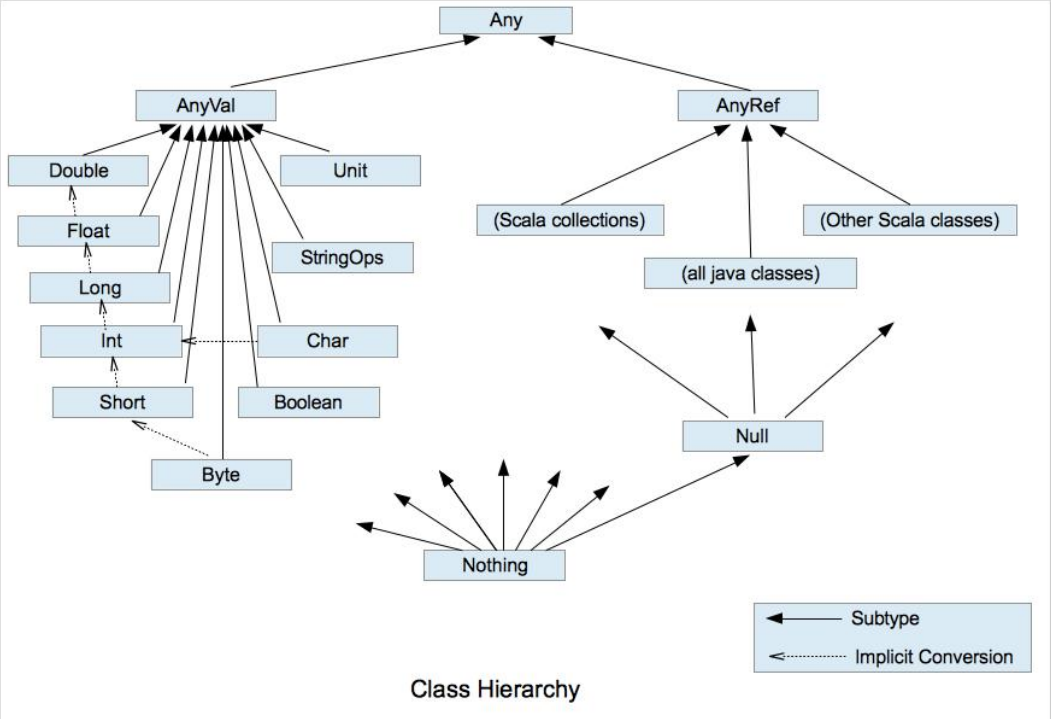
```
scala> 1.toString
```

相对于 java 的类型系统，scala 无疑要复杂的多!也正是这复杂多变的类型系统才让面向对象编程和函数式编程完美的融合在了一起!

Scala 中，所有的值都是类对象，而所有的类，包括值类型，都最终继承自一个统一的根类型 **Any**。统一类型，是 Scala 的又一大特点。更特别的是，Scala 中还定义了几个底层类（Bottom Class），比如 **Null** 和 **Nothing**。

1) Null 是所有引用类型的子类型，而 Nothing 是所有类型的子类型。Null 类只有一个实例对象，null，类似于 Java 中的 null 引用。null 可以赋值给任意**引用类型**，但是不能赋值给**值类型**。
2) Nothing，可以作为没有正常返回值的方法的返回类型，非常直观的告诉你这个方法不会正常返回，而且由于 Nothing 是其他任意类型的子类，他还能跟要求返回值的方法兼容。
3) Unit 类型用来标识过程，也就是没有明确返回值的函数。 由此可见，Unit 类似于 Java 里的 void。Unit 只有一个实例，()，这个实例也没有实质的意义。

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】



1.4 算数操作符重载

+-*/%可以完成和 Java 中相同的工作，但是有一点区别，Scala 是完全面向对象语言，所以所有的运算符都是对象的运算行为，也就是所谓的对象方法。

举例：

```
scala> 1 + 2
等同于：
scala> 1.+(2)
```

注：Scala 中没有++、--操作符，需要通过+=、-=来实现同样的效果。

1.5 调用函数和方法

在 scala 中，一般情况下我们不会刻意的去区分函数与方法的区别，但是他们确实是不同的东西。

1) 调用函数，求方根

```
scala> import scala.math._
scala> sqrt(100)
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

2) 调用方法，静态方法（scala 中没有静态方法这个概念，需要通过类的伴生对象来实现）

```
scala> BigInt.probablePrime(16, scala.util.Random)
```

3) 调用方法，非静态方法，使用对象调用

```
scala> "HelloWorld".distinct
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

二　控制结构

Scala 语言中控制结构和 Java 语言中的控制结构基本相同，在不考虑特殊应用场景的情况下，代码书写方式以及理解方式都没有太大的区别

2.1 if else 表达式

```
var sumVal = 0
if ( sumVal == 0 ) {
    println("true")
} else {
    println("false")
}
```

注：如果大括号内的逻辑代码只有一行，大括号可以省略

注：Scala 中任意表达式都是有返回值的，也就意味着 if else 表达式其实是有返回结果的，具体返回结果的值取决于满足条件的代码体的最后一行内容

```
var result = if ( sumVal == 0 ) {
    println("true") //  此处为满足条件逻辑的最后一行内容，打印语句是没有返回值的
} else {
    "false" //此处为不满足条件逻辑的最后一行内容，此处返回值为字符串 String
}
```

因为 Scala 在编译期间无法知道程序的逻辑是满足或不满足，所以无法推断出 result 变量的具体类型，所以就设定为 Any 类型（所有类型的公共超类型）

如果缺少一个判断，什么都没有返回，但是 **Scala 认为任何表达式都会有值**，对于空值，使用 Unit 类，叫做无用占位符，相当于 java 中的 void，打印结果为()，

```
val result =
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

```
  if(sumVal > 20){
    "结果大于 20"
  }
println(result)
```

Java 中表达式是没有值的，所以为了弥补这样的缺陷，就出现了三元运算符，但是 Scala 中是没有的，因为根本不需要。

```
// Java
int result = flg ? 1 : 0
```

```
// Scala
val result = if (flg) 1 else 0
```

2.2 while 表达式

1) while 循环

Scala 提供和 Java 一样的 while 和 do 循环，与 If 语句不同，While 语句本身没有值，即整个 While 语句的结果是 Unit 类型的()。

```
var n = 1;
val while1 = while(n <= 10){
  n += 1
}
println(while1)
println(n)
```

注：因为 while 和 do…while 中没有返回值,所以当要用该语句来计算并返回结果时,就不可避免的使用变量 ，而变量需要声明在 while 循环的外部，那么就等同于循环的内部对外部的变量造成了影响， 也就违背了函数式编程的重要思想，所以不推荐使用。

2) while 循环的中断

Scala 内置控制结构特地去掉了 break 和 continue，是为了更好的适应函数化编程，推荐使用函数式的风格解决 break 和 contine 的功能，而不是一个关键字。

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

```
import util.control.Breaks._
breakable{
   while(n <= 20){
      n += 1;
      if(n == 18){
         break()
      }
   }
}
```

2.3 for 表达式

Scala 也为 for 循环这一常见的控制结构提供了非常多的特性，这些 for 循环的特性被称为 for 推导式（for comprehension）或 for 表达式（for expression）。

1) 范围数据循环 1：to 左右两边为前闭后闭的访问

```
for(i <- 1 to 3){
   print(i + " ")
}
println()
```

2) 范围数据循环 2：until 左右两边为前闭后开的访问

```
for(i <- 1 until 3) {
   print(i + " ")
}
println()
```

3) 循环守卫：引入循环保护式（也称条件判断式，守卫）。保护式为 true 则进入循环体内部，为 false 则跳过，类似于 continue

```
for(i <- 1 to 3 if i != 2) {
   print(i + " ")
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

```
}
println()
```

4) 引入变量：

```
for(i <- 1 to 3; j = 4 - i) {
    print(j + " ")
}
```

5) 嵌套循环：

```
for(i <- 1 to 3; j <- 1 to 3) {
    print(i*j + " ")
}
```

6) 循环返回值：（将遍历过程中处理的结果返回到一个新集合中，使用 yield 关键字）

```
val for5 = for(i <- 1 to 10) yield i
println(for5)
```

7) 使用花括号{}代替小括号()：

```
for{
    i <- 1 to 3
    j = 4 - i}
    print(i * j + " ")
```

注：{}和()对于 for 表达式来说都可以。for 推导式有一个不成文的约定：当 for 推导式仅包含单一表达式时使用原括号，当其包含多个表达式时使用大括号。值得注意的是，使用原括号时，早前版本的 Scala 要求表达式之间必须使用分号。

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

三　函数

scala 定义函数的标准格式为：

```
def 函数名(参数名 1: 参数类型 1, 参数名 2: 参数类型 2) : 返回类型 = {函数体}
```

1) 函数有返回值：

```
def f1() : String= {
    return "f1"
}
```

注：Scala 中的函数可以根据函数体最后一行代码自行推断函数返回值类型。那么在这种情况下，return 关键字可以省略，既然 Scala 可以自行推断，所以在省略 return 关键字的场合，返回值类型也可以省略。

如果函数明确使用 return 关键字，那么函数无法省略返回值类型。

2) 函数无返回值

```
def f2() ： Unit = {
    println("f2")
}
```

注：如果函数明确声明无返回值（声明 Unit），那么函数体中即使使用 return 关键字也不会有返回值。

如果明确函数无返回值或不确定返回值类型，那么返回值类型可以省略

```
def f3(s: String) = {
    if(s.length >= 3)
        s + "123"
    else
        3
}
```

3)函数有参数：

```
def f4(p: String) = {
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

```
    println(p)
}
```

注：函数如果参数列表不为空，那么在调用时需要传递参数，不能省略，这和 JavaScript 语法不一样，在 JavaScript 中调用有参函数时，如果没有传递参数，那么参数会自动赋值为 undefined， Scala 也可以实现类似的功能，就是在声明参数时，直接赋初始值。

```
def f5(p:String = "f5") {
    println(p);
}
```

调用函数时，如果不传递参数，就采用默认的初始值，如果传递参数，传递的参数会覆盖初始化。

```
f5() // 不传参，打印 f5
f5("function") // 传参，覆盖默认值，打印 function
```

如果函数存在多个参数，每一个参数都可以设定默认值，那么这个时候，传递的参数到底是覆盖默认值，还是赋值给没有默认值的参数，就不确定了(默认按照声明顺序)。在这种情况下，可以采用**带名参数**

```
def f6 ( p1 : String = "v1", p2 : String ) {
    println(p1 + p2);
}
f6("v2" )   // (X)
f6(p2="v2") // (OK)
```

4)变长参数（不确定个数参数，类似 Java 的...）

```
def f7(args: Int*) = {
  var result = 0
  for(arg <- args)
    result += arg
  result
}
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

5) 递归函数

```
// 递归函数未执行之前是无法推断出来结果类型，在使用时必须有明确的返回值类型
def f8(n: Int): Int = {
  if(n <= 0)
    1
  else
    n * f8(n - 1)
}
```

6) 过程：将函数的返回类型为 Unit 的函数称之为过程。如果明确函数没有返回值，那么等号可以省略

```
def f9(content: String) = {
  println("f7")
}
```

```
def f9(content: String) { // 明确无返回值
  println(content)
}
```

注：开发工具的自动代码补全功能，虽然会自动加上 Unit，但是考虑到 Scala 语言的简单，灵活，能不加最好不加

7) 惰性函数：当函数返回值被声明为 lazy 时，函数的执行将被推迟，直到我们首次对此取值。这种函数我们称之为惰性函数，在 Java 的某些框架代码中称之为懒加载（延迟加载）

```
  def f10(): String = {
    println("f10 方法执行")
  }
  lazy val msg = f10()
  println("f10 方法没有执行")
  println(msg)
```

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

四　异常

Scala 提供 try 和 catch 块来处理异常。try 块用于包含可疑代码。catch 块用于处理 try 块中发生的异常。可以根据需要在程序中有任意数量的 try...catch 块。

语法处理上和 Java 类似，但是又不尽相同。

```
// Java
try {
    // 可疑代码
    int I = 0;
    int b = 10;
    int c = b / i; // 执行代码时，会抛出 ArithmeticException 异常
} catch(Exception e)  {
    e.printStackTrace();
}finally {
    // 最终要执行的代码
}
```

```
// Scala
try {
      val r = 10 / 0
} catch {
      case ex: ArithmeticException=> println("捕获了除数为零的算数异常")
      case ex: Exception => println("捕获了异常")
} finally {
      // 最终要执行的代码
}
```

1) 我们将可疑代码封装在 try 块中。 在 try 块之后使用了一个 catch 处理程序来捕获异常。

【更多 Java、HTML5、Android、Python、大数据 资料下载，可访问尚硅谷（中国）官网 www.atguigu.com 下载区】

如果发生任何异常，catch 处理程序将处理它，程序将不会异常终止。

2) Scala 的异常的工作机制和 Java 一样，但是 Scala 没有“checked”异常，你不需要声明说函数或者方法可能会抛出某种异常。受检异常在编译器被检查，java 必须声明方法所会抛出的异常类型。

3) 用 throw 关键字，抛出一个异常对象。所有异常都是 Throwable 的子类型。throw 表达式是有类型的，就是 Nothing，因为 Nothing 是所有类型的子类型，所以 throw 表达式可以用在需要类型的地方。

4) 在 Scala 里，借用了模式匹配的思想来做异常的匹配，因此，在 catch 的代码里，是一系列 case 子句。

5) 异常捕捉的机制与其他语言中一样，如果有异常发生，catch 字句是按次序捕捉的。因此，在 catch 字句中，越具体的异常越要靠前，越普遍的异常越靠后。 如果抛出的异常不在 catch 字句中，该异常则无法处理，会被升级到调用者处。

6) finally 字句用于执行不管是正常处理还是有异常发生时都需要执行的步骤，一般用于对象的清理工作。

7) Scala 提供了 throws 关键字来声明异常。可以使用方法定义声明异常。 它向调用者函数提供了此方法可能引发此异常的信息。 它有助于调用函数处理并将该代码包含在 try-catch 块中，以避免程序异常终止。在 scala 中，可以使用 throws 注释来声明异常。

```
@throws(classOf[NumberFormatException])
def f11()  = {
      "abc".toInt
}
```