Kotlin 语言声明一个变量使用关键字var，声明一个常量使用val，声明时 Kotlin 语言是可以自动推测出字段类型的，例如上面代码中的var quantity = 5会被认为是Int类型，但如果你希望它是一个Double类型，则需要显示声明类型，例如var quantity: Double = 5。

**in关键字的使用**

判断一个对象是否在某一个区间内，可以使用in关键字

//如果存在于区间(1,Y-1)，则打印OK

if (x in 1..y-1)

print("OK")

//如果x不存在于array中，则输出Out

if (x !in 0..array.lastIndex)

print("Out")

//打印1到5

for (x in 1..5)

print(x)

//遍历集合(类似于Java中的for(String name : names))

for (name in names)

println(name)

//如果names集合中包含text对象则打印yes

if (text in names)

print("yes")

when表达式

类似于 Java 中的 switch，但是 Kotlin 更加智能，可以自动判断参数的类型并转换为响应的匹配值。

fun cases(obj: Any) {

when (obj) {

1 -> print("第一项")

"hello" -> print("这个是字符串hello")

is Long -> print("这是一个Long类型数据")

!is String -> print("这不是String类型的数据")

else -> print("else类似于Java中的default")

}

}

**智能类型推测**

判断一个对象是否为一个类的实例，可以使用is关键字

fun getStringLength(obj: Any): Int? { if (obj is String) { // 做过类型判断以后，obj会被系统自动转换为String类型 return obj.length } //同时还可以使用!is，来取反 if (obj !is String){ } // 代码块外部的obj仍然是Any类型的引用 return null }

**空值检测**

Kotlin  是空指针安全的，也就意味着你不会再看到那恼人的空指针异常。  
例如这句代码 println(files?.size)，只会在files不为空时执行。  
以及，你可以这样写

**函数的声明**

函数使用关键字fun声明

**函数的默认参数**

你也可以使用默认参数来实现重载类似的功能

fun say(str: String = "hello"): String = str

**扩展函数**

你可以给父类添加一个方法，这个方法将可以在所有子类中使用  
例如，在 Android 开发中，我们常常使用这样的扩展函数：

fun Activity.toast(message: CharSequence, duration: Int = Toast.LENGTH\_SHORT) {

Toast.makeText(this, message, duration).show()

}

这样，我们就可以在每一个Activity中直接使用toast()函数了。

**将函数作为参数**

Kotlin 中，可以将一个函数作为参数传递给另一个函数

kotlin fun lock<T>(lock: Lock, body: () -> T ) : T { lock.lock() try { return body() } finally { lock.unlock() } } 上面的代码中，我们传入了一个无参的 body() 作为 lock() 的参数，