```
val list = ArrayList<String>()
list.add("Apple")
list.add("Banana")
list.add("Orange")
list.add("Pear")
list.add("Grape")
```

但是这种初始化集合的方式比较烦琐，为此 Kotlin 专门提供了一个内置的 `listOf()`函数来简化初始化集合的写法，如下所示：

```
val list = listOf("Apple", "Banana", "Orange", "Pear", "Grape")
```

可以看到，这里仅用一行代码就完成了集合的初始化操作。

还记得我们在学习循环语句时提到过的吗？`for-in` 循环不仅可以用来遍历区间，还可以用来遍历集合。现在我们就尝试一下使用 `for-in` 循环来遍历这个水果集合，在 `main()`函数中编写如下代码：

```
fun main() {
    val list = listOf("Apple", "Banana", "Orange", "Pear", "Grape")
    for (fruit in list) {
        println(fruit)
    }
}
```

运行一下代码，结果如图 2.23 所示。

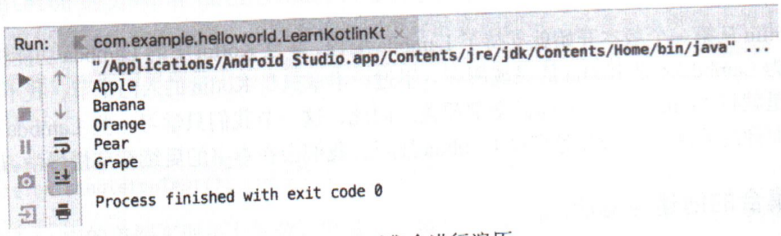

图 2.23　对集合进行遍历

不过需要注意的是，`listOf()`函数创建的是一个不可变的集合。你也许不太能理解什么叫作不可变的集合，因为在 Java 中这个概念不太常见。不可变的集合指的就是该集合只能用于读取，我们无法对集合进行添加、修改或删除操作。

至于这么设计的理由，和 `val` 关键字、类默认不可继承的设计初衷是类似的，可见 Kotlin 在不可变性方面控制得极其严格。那如果我们确实需要创建一个可变的集合呢？也很简单，使用 `mutableListOf()`函数就可以了，示例如下：

```
fun main() {
    val list = mutableListOf("Apple", "Banana", "Orange", "Pear", "Grape")
    list.add("Watermelon")
    for (fruit in list) {
```