```
            println(fruit)
        }
    }
```

这里先使用 mutableListOf()函数创建一个可变的集合，然后向集合中添加了一个新的水果，最后再使用 for-in 循环对集合进行遍历。现在重新运行一下代码，结果如图 2.24 所示。

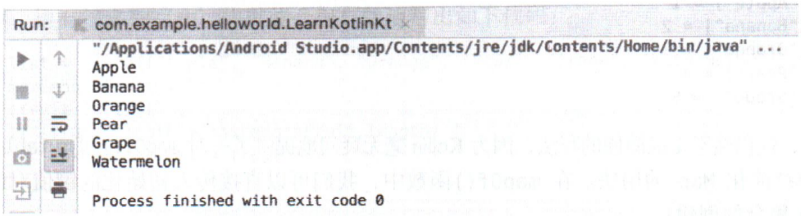

图 2.24 对可变集合进行遍历

可以看到，新添加到集合中的水果已经被成功打印出来了。

前面我们介绍的都是 List 集合的用法，实际上 Set 集合的用法几乎与此一模一样，只是将创建集合的方式换成了 setOf()和 mutableSetOf()函数而已。大致代码如下：

```
val set = setOf("Apple", "Banana", "Orange", "Pear", "Grape")
for (fruit in set) {
    println(fruit)
}
```

需要注意，Set 集合中是不可以存放重复元素的，如果存放了多个相同的元素，只会保留其中一份，这是和 List 集合最大的不同之处。当然这部分知识属于数据结构相关的内容，这里就不展开讨论了。

最后再来看一下 Map 集合的用法。Map 是一种键值对形式的数据结构，因此在用法上和 List、Set 集合有较大的不同。传统的 Map 用法是先创建一个 HashMap 的实例，然后将一个个键值对数据添加到 Map 中。比如这里我们给每种水果设置一个对应的编号，就可以这样写：

```
val map = HashMap<String, Int>()
map.put("Apple", 1)
map.put("Banana", 2)
map.put("Orange", 3)
map.put("Pear", 4)
map.put("Grape", 5)
```

我之所以先用这种写法，是因为这种写法和 Java 语法是最相似的，因此可能最好理解。但其实在 Kotlin 中并不建议使用 put()和 get()方法来对 Map 进行添加和读取数据操作，而是更加推荐使用一种类似于数组下标的语法结构，比如向 Map 中添加一条数据就可以这么写：

```
map["Apple"] = 1
```