任何一个面向对象的编程语言都会有构造函数的概念，Kotlin 中也有，但是 Kotlin 将构造函数分成了两种：主构造函数和次构造函数。

主构造函数将会是你最常用的构造函数，每个类默认都会有一个不带参数的主构造函数，当然你也可以显式地给它指明参数。主构造函数的特点是没有函数体，直接定义在类名的后面即可。比如下面这种写法：

```
class Student(val sno: String, val grade: Int) : Person() {
}
```

这里我们将学号和年级这两个字段都放到了主构造函数当中，这就表明在对 Student 类进行实例化的时候，必须传入构造函数中要求的所有参数。比如：

```
val student = Student("a123", 5)
```

这样我们就创建了一个 Student 的对象，同时指定该学生的学号是 a123，年级是 5。另外，由于构造函数中的参数是在创建实例的时候传入的，不像之前的写法那样还得重新赋值，因此我们可以将参数全部声明成 val。

你可能会问，主构造函数没有函数体，如果我想在主构造函数中编写一些逻辑，该怎么办呢？Kotlin 给我们提供了一个 init 结构体，所有主构造函数中的逻辑都可以写在里面：

```
class Student(val sno: String, val grade: Int) : Person() {
    init {
        println("sno is " + sno)
        println("grade is " + grade)
    }
}
```

这里我只是简单打印了一下学号和年级的值，现在如果你再去创建一个 Student 类的实例，一定会将构造函数中传入的值打印出来。

到这里为止都还挺好理解的吧？但是这和那对括号又有什么关系呢？这就涉及了 Java 继承特性中的一个规定，子类中的构造函数必须调用父类中的构造函数，这个规定在 Kotlin 中也要遵守。

那么回头看一下 Student 类，现在我们声明了一个主构造函数，根据继承特性的规定，子类的构造函数必须调用父类的构造函数，可是主构造函数并没有函数体，我们怎样去调用父类的构造函数呢？你可能会说，在 init 结构体中去调用不就好了。这或许是一种办法，但绝对不是一种好办法，因为在绝大多数的场景下，我们是不需要编写 init 结构体的。

Kotlin 当然没有采用这种设计，而是用了另外一种简单但是可能不太好理解的设计方式：括号。子类的主构造函数调用父类中的哪个构造函数，在继承的时候通过括号来指定。因此再来看一遍这段代码，你应该就能理解了吧。

```
class Student(val sno: String, val grade: Int) : Person() {
}
```