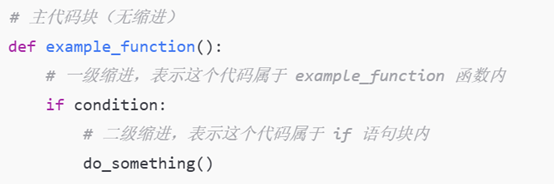
（自动解析）**IS-CFG（缩进敏感上下文无关文法）**是一种专门处理Python缩进问题的方法。它通过一些语法规则把Python的缩进要求形式化，可以开发出解析器，来确保代码的缩进符合标准。简单来说，IS-CFG会用NEWLINE来分隔语句，用INDENT和DEDENT来表示缩进的增加和减少，还可以用=、>、≥等符号来定义缩进关系。

IS-CFG的**优点**在于它能非常精准地表达Python的缩进规则，这样可以避免代码中因为缩进产生的歧义。通过IS-CFG，我们可以开发出高效的解析器来自动检查并解析代码，确保代码的缩进符合规范。此外，统一的缩进规则还能提升代码的可读性和维护性，避免因为缩进问题而引发的语法错误。IS-CFG还适合用来开发代码格式化工具，自动调整代码的缩进，让代码一键符合规范要求。这对于编写和维护Python代码都非常实用。

不过，IS-CFG也有一些**缺点**。首先，它有一定的学习门槛，理解和掌握这些规则需要一些理论基础，对初学者可能会有点难。此外，基于IS-CFG开发解析器或代码格式化工具需要一定的技术投入，开发成本不低。

****（直观检测）**立体视觉算法**是另一种方法，它通过将Python代码导入电子表格中并用不同视觉方式来展示缩进级别，帮助发现缩进问题。具体操作是先把代码转换成电子表格格式，每行代码对应电子表格中的一行，单元格中显示该行代码的**前导空格数**（一行代码或文本开头的连续空格数量）。接下来通过“左眼视图”和“右眼视图”两种方式展示缩进级别。在“左眼视图”中，程序将空格数转化为缩进级别：缩进块数等于当前代码行前面的空格数除以默认的缩进空格数（比如8个空格为一级缩进），如果有余数，则再加一个缩进块。而在“右眼视图”中，缩进块的数量直接等于代码行前的空格数除以默认的缩进空格数，不加余数，显示更直观。如果左右视图中同一行的缩进块数量不同，程序员就能快速发现缩进错误，便于后续调整。

这个方法的**优点**是直观，缩进问题在电子表格中一目了然，而且两种视图对比使得错误位置更清晰易懂，并且自动修正小缩进错误的功能还能节省时间。

不过，它的**缺点**是需要手动把代码导出到电子表格里，对于大型项目或频繁编辑的代码可能不太方便。此外，它只能修正一些小范围内的缩进错误，对于较大的问题还是需要手动调整。