Python缩进规则

在 Python 中，对于类定义、函数定义、流程控制语句、异常处理语句等，行尾的冒号和下一行的缩进，表示下一个代码块的开始，而缩进的结束则表示此代码块的结束。

注意，Python 中实现对代码的缩进，可以使用空格或者 Tab 键实现。但无论是手动敲空格，还是使用 Tab 键，通常情况下都是采用 4 个空格长度作为一个缩进量（默认情况下，一个 Tab 键就表示 4 个空格）

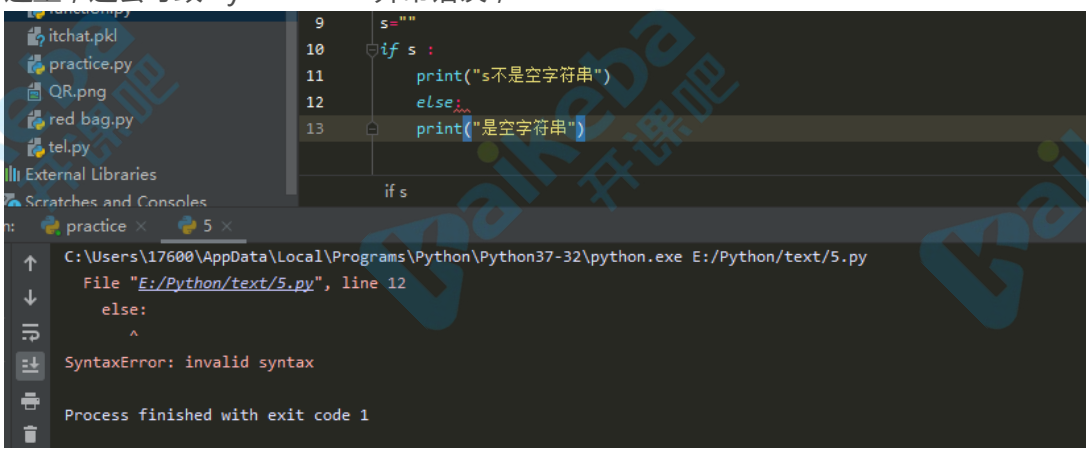
下面通过一段代码来体现缩进规则：

```
s = ” ”
if s :
    print(’s不是空字符串’)
else:
    print(’s是空字符串’)
```

Python 对代码的缩进要求非常严格，同一个级别代码块的缩进量必须一样，否则解释器会报 SyntaxError 异常错误。例如，对上面代码做错误改动，将位于同一作用域中的 2 行代码，它们的缩进量分别设置为 4 个空格，如下所示：

```
s=” ”
if s :
    print(”s不是空字符串”)
    else:
    print(”是空字符串”)
```

可以看到，第二行代码和第四行代码本来属于同一作用域，但我们手动修改了各自的缩进量，这会导致 SyntaxError 异常错误，



对于 Python 缩进规则，初学者可以这样理解，Python 要求属于同一作用域中的各行代码，它们的缩进量必须一致，但具体缩进量为多少，并不做硬性规定

通常缩进错误有三种：
1、代码前后缩进量不一致
2、代码前后缩进符号不一致
3、tab与space混用

这时候就会很奇怪了，明明看着缩进很规范，但程序就是会报这个错误，例如pycharm，一旦出现缩进错误，pycharm在出错的那一行的空白缩进处出现黄色区域

```
def download_pictures(folder_name, picture_address):
    file_path = r'D:\Beautiful\{0}'.format(folder_name)
    if not os.path.exists(file_path):
        # 新建一个文件夹
```


如图，缩进虽然表面上看着没有问题，其实时tab和space混用了。
解决办法：重新缩进；点击小灯泡，出来下拉框，其中有两项

```
def download_pictures(folder_name, picture_address):
    file_path = r'D:\Beautiful\{0}'.format(folder_name)
    if not os.path.exists(file_path):
        Convert indents to spaces
        Convert indents to tabs
        in(r'G:\Beautiful', folder_name))
        # 下载图片保存到新建文件夹
```


随便选一个后，pycharm会自动调整错误缩进。