# 欢迎使用

## 一、概述

本模块为方便大家更便捷使用Python来操作Excel文件而编写，基于openpyxl进行编写，所以在使用本模块前请确保您的计算机中已安装有Python环境以及openpyxl库。

- 作者：北极星光
- E-mail：light22@126.com
- Pypi官方库地址：https://pypi.org/project/excel-operate-light22
- Github地址：https://github.com/18513233125/Excel-operate

## 二、功能介绍

- 简化openpyxl库的导入流程，无需分别导入Workbook和load_workbook来创建和打开工作薄，只需要使用ExcelOperate类实例化对象时，指定file_path参数即打开已有的Excel文件，不指定即为新建工作薄。同时实例化打开EXcel文件时可以直接指定sheet_name参数来快速打开工作表，不指定则打开当前激活工作表。
- 方便快捷插入和删除行（列）。解决直接使用openpyxl中工作表对象的insert_rows/insert_cols/delete_rows/delete_cols方法时，插入或删除的行（列）的位置后方行（列）的行高（列宽）以及如果存在已合并单元格的情况下，已合并的单元格信息等格式不会因插入或删除的行（列）而下（后）移或上（前）移，从而导致表格的整体格式甚至数据受到影响。通过本模块的方法插入或删除行（列）后的效果与在Excel程序中插入或删除行（列）的效果可以做到近乎完全相同。同时为insert_rows和insert_cols方法增加height和width参数，方便插入行（列）时快捷设定行高和列宽。
- 工作表复制，解决openpyxl只能在工作薄内复制工作表，无法跨工作薄复制工作表的问题

- 可以使用openpyxl库中的所有操作。
- 其它功能敬请期待

# 三、使用方法

- 导入本模块方法：
  - 在需要导入本模块的代码中写入导入语句：*from excel_operate import ExcelOperate*。
- 对象实例化方法：
  - 通过语句 `excel = ExcelOperate(file_path=None,sheet_name=None)`，获得一个名为 `excel`的对象。
    - *参数 `file_path`为文件路径，指定后打开文件路径对应的**Excel**文件（相当于`openpyxl`中的`load_workbook`方法）不指定则会新建一个空白**Excel**文件（相当于`openpyxl`中的`Workbook`类的实例化）；*
    - *参数`sheet_name`为工作表名，指定后打开对应的工作表(如果打开的工作薄中不存在指定名称的工作表则以指定的工作表为名创建一个工作表)，不指定则打开当前激活的工作表（相当于工作薄对象的`active`方法）。*
- 插入或删除行（列）：
  - 插入行 `excel.insert_rows(idx,amount=1,height='before')`;
  - 删除行 `excel.delete_rows(idx,amount=1)`;
  - 插入列 `excel.insert_cols(idx,amount=1,width='before')`;
  - 删除列 `excel.delete_cols(idx,amount=1)`。
    - *参数`idx`为插入或删除的起始行（列）*
    - *参数`amount`*为插入或删除的行（列）数，不指定的情况下默认插入或删除1行（列）
    - 参数`height`和`width`仅在插入行和列时使用，可指定为整数或者浮点数来设置插入的行高或列宽，不指定的情况下默认参数为`'before'`，该参数可以让插入的行或列继承插入前上一行（列）的行高或列宽。另外插入行时，`height`参数也可指定为`None`，此参数可以设置插入行的行高为自动行高。
- 工作表复制：
  - 首先通过语句`from excel_operate import SheetCopy`导入模块
  - 然后通过实例化对象`copyer = SheetCopy(src_file_path, tag_file_path=None, sheet_name=None, column_adjust=0)`来获得实例化对象`copyer`
    - 参数`src_file_path`为源文件路径；

        - 参数 `tag_file_path`为目标文件路径，不指定的话将新建一个工作薄；
        - 参数 `sheet_name`为被复制的工作表名称，不指定的话为源文件的当前工作表；
        - 参数 `column_adjust`为列宽修正系数，原因为 `openpyxl`中设置的列宽与实际列宽存在误差，一般为0~0.9之间。如果复制的工作表与源工作表列宽不相同的话可以修改此参数从而使复制的工作表与源工作表列宽相等。`)` ，
    - 然后通过 `tag_file = copyer.copy_sheet()`来获得一个 `ExcelOperate`对象 `tag_file`。
- 其它 `openpyxl`操作：
    - 使用 `wb = excel.wb`可以*获得 `openpyxl` 中的工作薄对象 `wb`，从而进行* `openpyxl`中关于工作薄对象的一切操作；
    - 使用 `ws = excel.ws` *可以获得 `openpyxl` 中的工作表对象 `ws`，从而进行* `openpyxl`中关于工作表对象的一切操作。
- Excel文件的保存：
    - 可直接使用 `excel.save (file_path=None)` *的方法来保存。*
        - 参数 `file_path`为文件保存路径，不指定的情况下为覆盖原文件保存，指定后可以另存为新的路径（如果之前实例化 `ExcelOperate`对象时没有指定 `file_path`，即创建空白Excel文件的情况下，保存时不指定路径的话则会报错。）

# 四、版本历史

- V 1.0.3
    - 优化代码逻辑，增加在ExcelOperate类的实例化中如果打开的Excel工作薄中没有sheet_name参数指定的工作表则会以sheet_name参数为名，创建一个工作表。
- V 1.0.2
    - 优化代码结构，增加函数参数注释。
- V 1.0.1
    - 正式版本，新增工作表复制模块SheetCopy，优化README文档结构。
- V 0.2.0
    - 测试版本，优化README描述，正式在pypi.org官方上线版本。
- V 0.1.0
    - 初始版本、测试版本

---