# 高效办公O1复习资料

恭喜完成oa高效办公第一阶段课程呀。

现在我们先来回顾一下这7天里我们都学习了什么内容。

因为第1关是导学课，主要是引导大家如何更好的学习高效办公这门课程，并没有包含什么 python 知识点，所以我们直接从第二关开始复习~

## ——第2关——

这一关我们接触了 openpyxl 库，这个库是oa高效办公的重点，可以说我们的自动化很大一部分是建立在openpyxl库上的。

在回顾openpyxl库前，我们先回顾一下三个概念👇

关于Excel的三个概念：工作簿（workbook），工作表（worksheet）和单元格（cell）。

工作簿像一本小册子。

而工作表像这本册子中一页一页的内容。

单元格则是表格中的方块。

在写代码前需要先导入模块，这样才能调用相应的方法。

通过from...import...语句可以直接导入模块里的函数、类或变量。

```
# 同时导入openpyxl模块中的load_workbook和Workbook方法
from openpyxl import load_workbook, Workbook
```

使用openpyxl.load_workbook()可以打开已有工作簿，使用openpyxl.Workbook()创建新工作簿。

下面我们先尝试打开已有工作簿，并定位到工作表中。

```
from openpyxl import load_workbook

# 打开工作簿
wb = load_workbook('财务报表.xlsx')

```

```
# 用 wb.active 打开工作表
ws = wb.active
# 接下面代码👇
```

在获取表头时，我们需要定位到某一行、某一列、某一单元格，方法如下：

获取**某行**语法：ws[行值]，例如：获取第一行：ws[1]；行值是整数，从 1 开始。

获取**某列**语法：ws[列值]，例如：获取第一列：ws['A']，或ws['a']。

获取**某单元格**语法：ws[单元格坐标]，单元格坐标是行列的组合，类型是字符串。
例如：获取第一行第一列的'A1'单元格，语法是ws['A1']，或ws['a1']。

```
# 接上面代码👆
# 获取第一行
header = ws[1]
# 打印查看
print('表格第一行：')
print(header)

# 获取第一列
column_a = ws['A']
# 打印查看
print('表格第一列：')
print(column_a)

# 获取单元格C2
cell_c2 = ws['C2']
# 打印查看
print('单元格C2:')
print(cell_c2)
```

通过定位获得的单元格只是个对象，即Cell 对象。需要通过**cell.value**来进一步获取单元格里面的值。

```
# 获取单元格C2
cell_c2 = ws['C2']
# 打印查看单元格的值
print('查看单元格C2的值:')
print(cell_c2.value)
```

除了单独定位的方法，也有取多行多列的方法。

如果我们需要取出，除了第一行（表头）以外的数据，也就是从第二行开始取，可以使用iter_rows()方法。

iter_rows() 是工作表对象的一个方法，其功能是：通过行列序号指定遍历范围。

ws.iter_rows()可以与 for 循环结合使用，指定行列区域，按行遍历工作表，常用的参数包括：

**ws.iter_rows()的参数**

| 参数名 | 含义 | 数据类型 | 取值 |
| --- | --- | --- | --- |
| min_row | 起始行 | 默认参数None；<br>传入参数为整数int | 这四个参数取默认值时，<br>表示取到头；<br>min_row/min_col<br>的最小值是1，代表第一行 |
| max_row | 终止行 | | |
| min_col | 起始列 | | |
| max_col | 终止列 | | |
| values_only | 是否只取值 | 布尔类型 | False，默认值，表示取单元格对象；<br>True，表示只取单元格对象的值 |



min_row，max_row，min_col，max_col四个参数可以顾名思义，限定了行列的范围；values_only决定了我们是获取单元格对象、还是单元格的值。最终以元组的形式返回。

```
# 遍历从第2行开始的每行内容，values_only=True可以直接获取单元格的值
for row in ws.iter_rows(min_row=2, values_only=True):
    # 遍历每一行内的单元格
    for r in row:
        print(r)
```

在得到想要的内容后，就得写入新的表格之中。工作表对象的 **append()** 方法 能在表格的末尾**追加一行**数据，语法是：ws.append()，参数可以是一个元组或列表。

ws.append()会将元组或列表中的每个元素，从左向右，依次填入各个单元格中。

```
from openpyxl import load_workbook

# 打开工作簿
wb = load_workbook('财务报表.xlsx')
# 用 wb.active 打开工作表
ws = wb.active

new_list = [1, 2, 3]
# 将new_list添加如工作表中
ws.append(new_list)
```


成功写入后，剩下的是修改工作表的名字以及保存工作簿了，修改的工作表需要工作表对象.title=’更改后的名字’

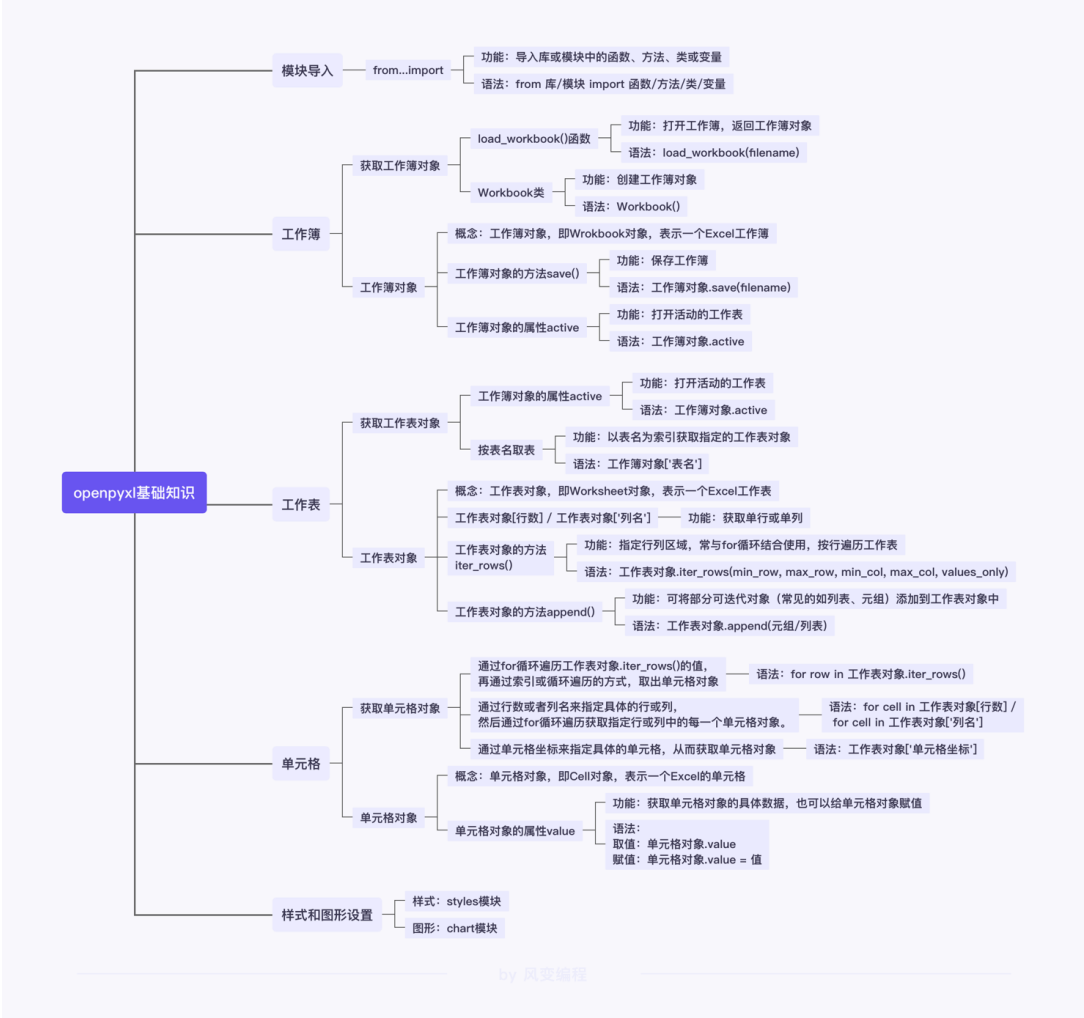
```
# 打印工作表名
print(ws.title)

# 修改工作表标题
ws.title = '工作表1'
```

而利用openpyxl库对表格进行修改后，要记得使用save()方法进行保存，否则修改的excel表格是不会自动保存。

```
# 保存工作簿
wb.save('拆分表格.xlsx')
```

总结：




## ——第3、4关——

进入第3关的学习，这一关没有新的知识点，通过四个小案例、对第二关所学的知识点进行巩固、解析。

这四个小案例分别是获取个人工资信息、生成前十行绩效信息表、计算并打印奖金信息、创建薪资信息字典

根据四个小案例，我们对“单元格读写”、“按行读写”、“按行取数计算”、“按行取数存为字典”等不同的模式进行熟悉。

我们看看这四个小案例的目的分解表格👇

## 案例问题分解表格

| 案例问题 | 获取哪些数据 | 如何使用数据 | 数据输出什么结果 | 模式 |
| --- | --- | --- | --- | --- |
| 案例一 | 一个工作表中个别单元格（已知坐标） | 原样不变 | 写入其他已有工作表并保存 | 单元格读写 |
| 案例二 | 一个工作表前10行数据 | 原样不变 | 写入其他已有工作表并保存 | 按行读写 |
| 案例三 | 一个工作表中除表头外的数据 | 取出4个单元格的值，并计算奖金信息 | 打印格式化的字符串 | 按行取数计算 |
| 案例四 | 一个工作表中除表头外的数据 | 分别取出各个单元格，存入字典 | 打印字典 | 按行取数存为字典（按行读取为字典） |

by 风变编程

通过分析上述案例，我们清晰认识到要解决现实中某些重复性工作，少不了这三步。

第一步👇，确认目的。明确自己的目前所遇到的问题是什么，并进行拆解，例如四个小案例的目的：获取个人工资信息、取前十行绩效信息表、计算奖金信息...等 你应该也有属于的目的，那么请先思索一下。

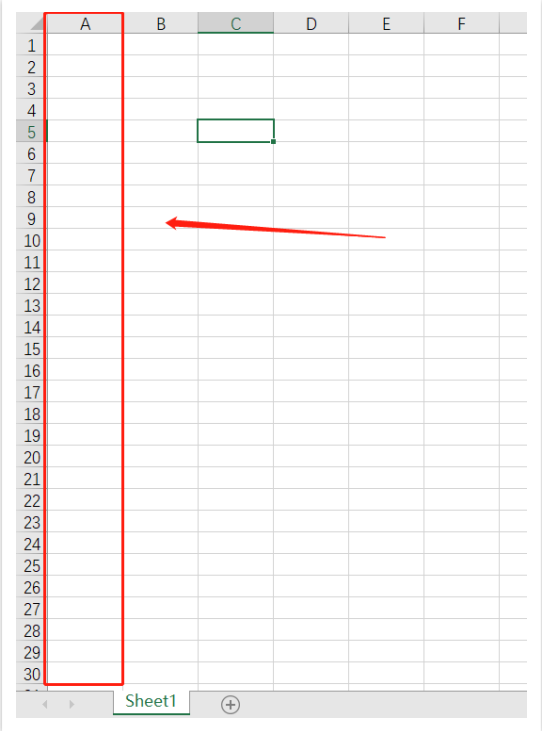
在确认目的的基础上，才能进行下一步思考。

第二步👇，你所需要是哪部分数据，利用openpyxl库处理excel表格，对单元格的处理主要有两种方法。

| 取值 | 方法 |
| --- | --- |
| 某些坐标里的值 | 工作表对象【单元格坐标】 |
| 某个几行的数据 | iter_rows（）方法 |

在处理数据时，需要我们去斟酌，自己想要实现的结果是什么？根据想要的结果来选择如何去写代码。

例如：我想要把工资表中的坐标A1~A30单元格数据读取来。




那我们想要的结果既然已经知道了，就可以根据前面所学的知识点，利用iter_rows进行取值。

```
for item in 工作表对象.iter_rows(min_row=1,max_col=1,max_row=30,values_only=True):
    print(item)
```

再复杂一些，把坐标为A1~A30单元格数据读取出来，同时不打印坐标为A11单元的内容

举两种实现这种结果的代码👇

①可以加入if语句进行判断。

```
for item in 工作表对象.iter_rows(min_row=1,max_col=1,max_row=30,values_only=True):
    for items in item:
        if items == 工作表对象['A11'].value:
            print("不打印")
```

②也可以将表格分为两部分进行读取,先读取坐标为A1~A10单元里的内容，再读取A11~A30的内容。

```
for item in 工作表对象.iter_rows(min_row=1,max_col=1,max_row=10,values_only=True):
    print(item)

for items in 工作表对象.iter_rows(min_row=12,max_col=1,max_row=30,values_only=True):
    print(items)
```

自动化办公的难点也就在于此，你通过什么代码，把自己的想要数据取出来并满足原先设想的结果。

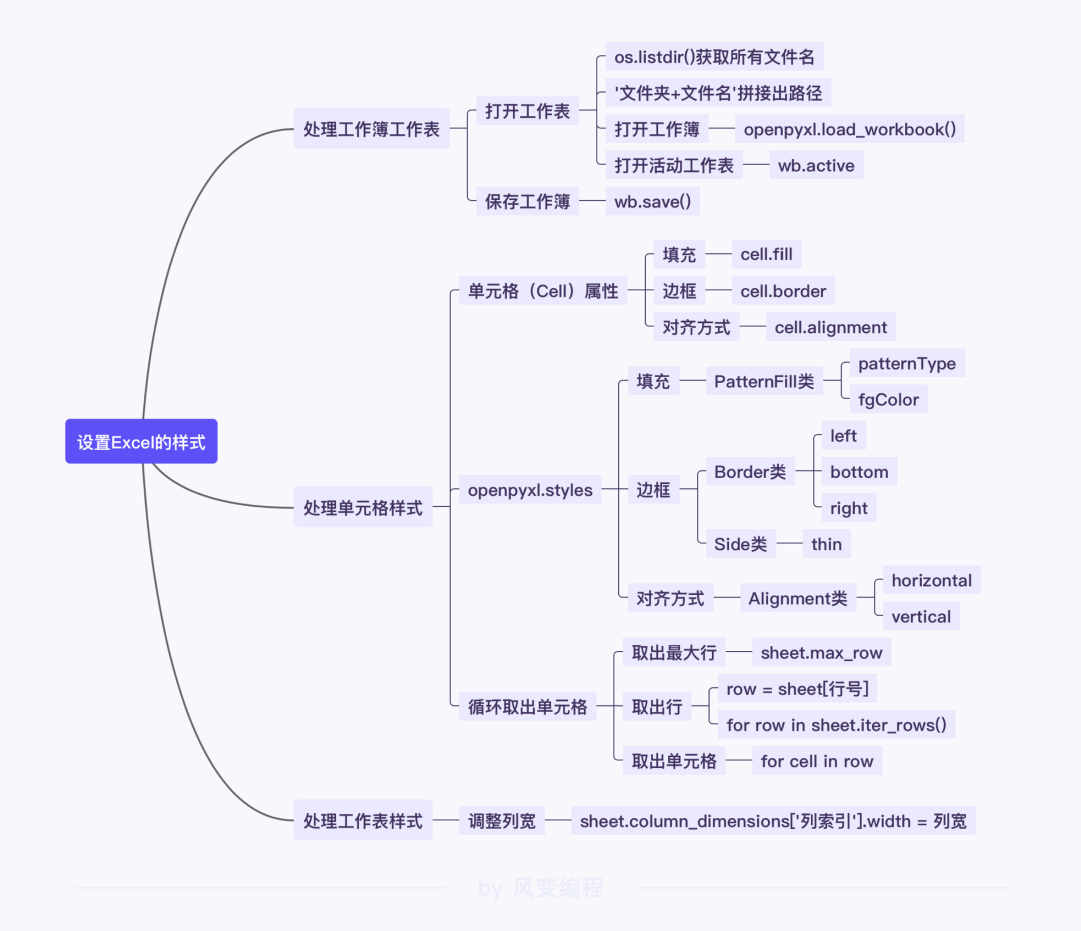
第三步👇处理好的数据要放到哪里？ 第三步关系到目的了，若目的是将一个表格中的数据，读取出来处理、筛选后，写入到另一个表格中。

那在构写代码时，就需要考虑的是新建一个工作簿，而不是直接通过print()函数，打印在终端出来。

根据自己的需求要写代码，改代码。

## ——第5关——

在第5关中我们则进一步学习如何修改Excel的样式。




首先我们还是需要导入模块，我们使用的方法都在openpyxl.styles模块下。

```
from openpyxl.styles import PatternFill, Alignment, Side, Border
```

然后就可以通过openpyxl修改表格的列宽、单元格的颜色、对齐方式和边框。

使用openpyxl修改Excel表格样式的步骤是：
1、选择样式属性；
2、定义该属性的样式值 ；
3、赋值修改（具体各个）单元格的样式值。

调整工作表的列宽，需要用到Sheet.column_dimensions[’列位置’].width。这条语句可以确定列位置，并用 width 属性，对该列的列宽进行修改。

比如说我想让第1列的列宽为20个单位，那么我就可以用 ws.column_dimensions[’A’] 先确定找到第1列。然后使用ws.column_dimensions[’A’].width = 20进行赋值。

除了列宽，其实也可以设置行高。

ws.column_dimensions['列名'].width = 数值类型，设置列宽。
ws.row_dimensions['行数'].height = 数值类型，设置行高。

```
# 打开工作簿
wb = load_workbook(file_path)
# 打开工作表
sheet = wb.active

# 调整列宽
sheet.column_dimensions['A'].width = 10
# 调整行高
sheet.row_dimensions[1].height = 30
```

刚刚提到openpyxl修改Excel表格样式步骤的第1点是——选择样式属性。即是选择关于单元格的样式属性。

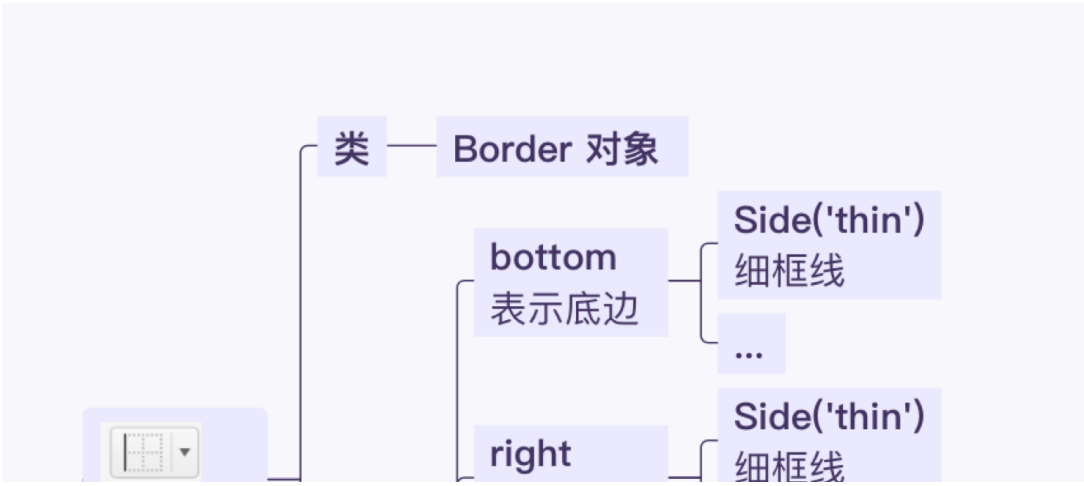
cell单元格的常见用法有：cell.value、cell.fill、cell.alignment、cell.border

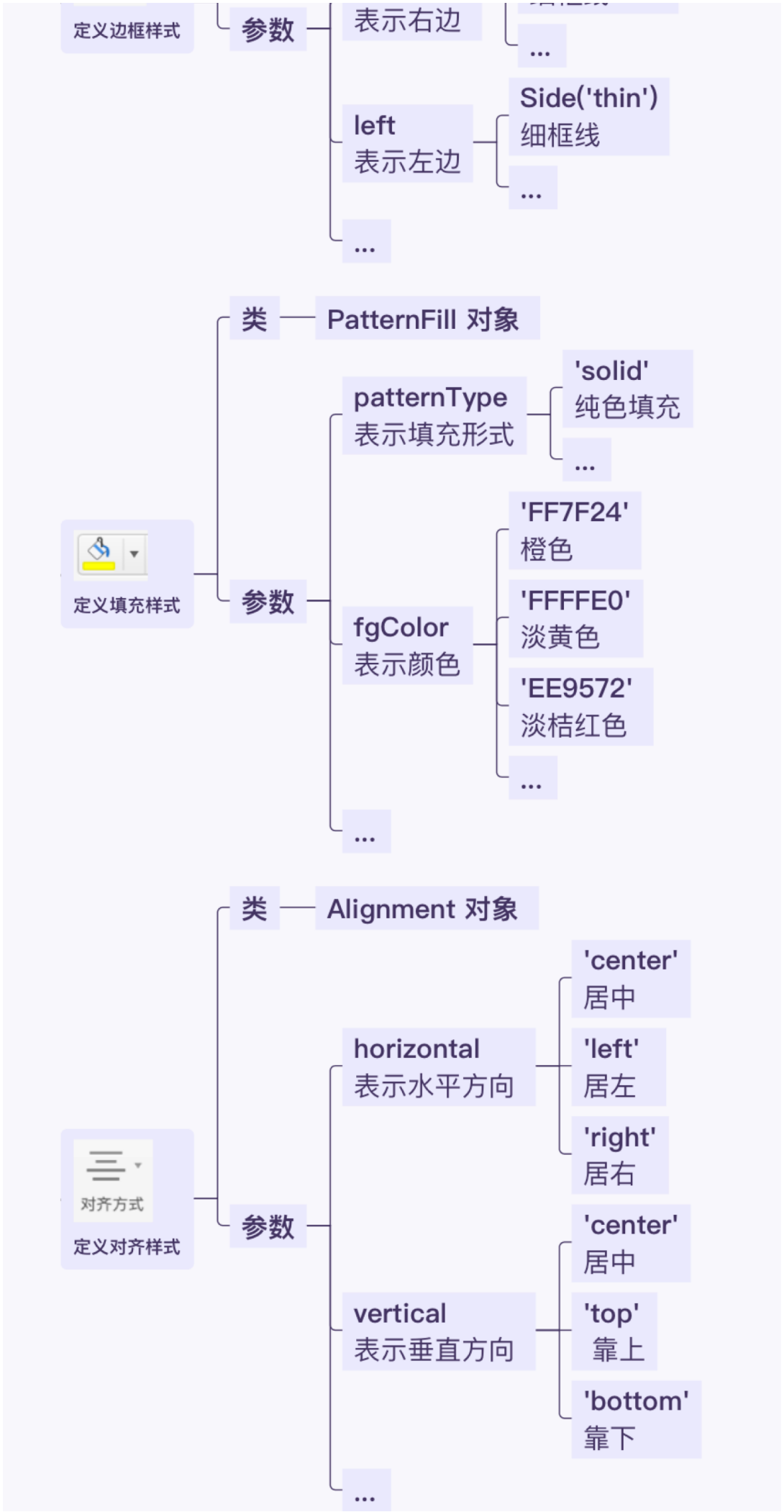
- cell.value：获取单元格内的值；
- cell.fill：设置单元格内的填充颜色；
- cell.alignment：设置单元格内的对齐方式；
- cell.border：设置单元格内的边框样式。

代码中的表示是：

```
# 定位到工作表的第1行，遍历里面的所有单元格
for cell in sheet['1']:
    # 设置单元格填充颜色
    cell.fill = # 使用定义好的样式
    # 设置单元格对齐方式
    cell.alignment = # 使用定义好的样式
    # 设置单元格边框
    cell.border = # 使用定义好的样式
```

大基调定下后，就需要处理小细节，比如样式是怎么定义的。先看思维导图。




定义边框样式
参数
表示右边
...
left
表示左边
Side('thin')
细框线
...
...
定义填充样式
类
PatternFill 对象
参数
patternType
表示填充形式
'solid'
纯色填充
...
fgColor
表示颜色
'FF7F24'
橙色
'FFFFE0'
淡黄色
'EE9572'
淡桔红色
...
...
对齐方式
定义对齐样式
类
Alignment 对象
参数
horizontal
表示水平方向
'center'
居中
'left'
居左
'right'
居右
vertical
表示垂直方向
'center'
居中
'top'
靠上
'bottom'
靠下
...

by 风变编程

样式分为边框设置、颜色填充、对齐方式，分别对应Border对象、PatternFill对象、Alignment对象。

---

Border()方法可以给单元格设置边框，可以同时设置上下左右四个方向。而设置的样式则需要通过Side()方法来进行。

语法：Border(top=Side(style= , color= ) , bottom=Side(style= , color= ) , left=Side(style= , color= ) , right=Side(style= , color= ))

- style参数需要加入样式类型：thin（细条）、medium（中等）、double（双重）等等。
- color参数需要加入十六进制颜色码。
- top、bottom、lef、right是单元格的位置，后面接样式。

---

PatternFill()类其实就是对表格颜色的一个填充。

语法：PatternFill(patternType='', fgColor='')

- patternType参数表示填充形式，一般为'solid'纯色填充
- fgColor参数需要传入一个十六进制的颜色码，可在以下链接查询
https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6%E9%A2%9C%E8%89%B2%E7%A0%81/10894232?fr=aladdin

---

Alignment()类可以实现自动换行及字符串对齐方式修改，然后应用到指定的cell上。

语法：Alignment(horizontal='', vertical='')

- horizontal代表水平方向，可以左对齐left，还有居中center和右对齐right，等等。
- vertical代表垂直方向，可以居中center，还可以靠上top，靠下bottom，等等。

---

代码整合起来的话，如下所示：

```
# 定义表头颜色样式为橙色
header_fill = PatternFill(patternType='solid', fgColor='FF7F24')
# 定义数据部分颜色样式为淡黄色
content_fill = PatternFill(patternType='solid', fgColor='FFFFE0')
# 定义表尾颜色样式为淡桔红色
bottom_fill = PatternFill(patternType='solid', fgColor='EE9572')

# 定义边样式为细条
side = Side('thin')
# 定义表头边框样式，有底边和右边
```

```
header_border = Border(bottom=side, right=side)
# 定义数据部分边框样式，有左边
content_border = Border(left=side)

# 循环第一行单元格，调整表头样式
for cell in sheet['1']:
    # 用定义好的样式，去设置单元格填充颜色
    cell.fill = header_fill
    # 设置单元格对齐方式
    cell.alignment = align
    # 设置单元格边框
    cell.border = header_bord
```

可能有部分同学看到这么多内容、这么多行代码就会头疼，就会被吓唬到。其实不用担心的，代码不需要你背诵，只需要先留个印象，等到需要使用到的时候，再去查找，随查随用就可以了。

至于代码，每一行代码的作用都很清楚，根据openpyxl修改Excel表格样式的步骤来，先确定好哪个部分需要修改样式，再确定什么样式，最后根据需求把代码直接填上去就可以了。