OPENPYXL

1. 安装openpyxl

在开始学习openpyxl之前，我们需要先安装openpyxl模块。可以使用pip命令进行安装：

```
pip install openpyxl
```

2. 创建Excel文件

在openpyxl中，我们可以使用Workbook类来创建一个Excel文件。下面是一个简单的例子：

```
from openpyxl import Workbook

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置单元格的值
ws['A1'] = 'Hello'
ws['B1'] = 'World'

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1和B1的值，并将文件保存为example.xlsx。

3. 打开Excel文件

如果我们想要打开一个已经存在的Excel文件，可以使用openpyxl.load_workbook()函数。下面是一个例子：

```
from openpyxl import load_workbook

# 打开文件
wb = load_workbook('example.xlsx')

# 获取默认的工作表
ws = wb.active

# 获取单元格的值
print(ws['A1'].value)
print(ws['B1'].value)
```

我们使用load_workbook()函数打开了example.xlsx文件，并获取了默认的工作表。接着，我们分别输出了单元格A1和B1的值。

4. 创建工作表

在openpyxl中，我们可以使用Workbook.create_sheet()方法来创建一个工作表。下面是一个例子：

```
from openpyxl import Workbook

# 创建一个Workbook对象
wb = Workbook()

# 创建一个名为Sheet1的工作表
ws1 = wb.create_sheet('Sheet1')

# 创建一个名为Sheet2的工作表
ws2 = wb.create_sheet('Sheet2')

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象。然后，我们使用Workbook.create_sheet()方法创建了两个工作表，分别命名为Sheet1和Sheet2。最后，我们将文件保存为example.xlsx。

5. 选择工作表

在openpyxl中，我们可以使用Workbook.active属性来获取当前选中的工作表。我们也可以使用Workbook.get_sheet_by_name()方法来获取指定名称的工作表。下面是一个例子：

```
from openpyxl import load_workbook

# 打开文件
wb = load_workbook('example.xlsx')

# 获取默认的工作表
ws1 = wb.active

# 获取名为Sheet2的工作表
ws2 = wb.get_sheet_by_name('Sheet2')

# 输出工作表的名称
print(ws1.title)
print(ws2.title)
```

我们使用load_workbook()函数打开了example.xlsx文件，并获取了默认的工作表ws1和名为Sheet2的工作表ws2。接着，我们分别输出了这两个工作表的名称。

6. 写入数据

在openpyxl中，我们可以使用Worksheet.cell()方法来设置单元格的值。下面是一个例子：

```
from openpyxl import Workbook

# 创建一个Workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置单元格的值
```

```
ws.cell(row=1, column=1, value='Hello')
ws.cell(row=1, column=2, value='World')

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们使用Worksheet.cell()方法设置了单元格A1和B1的值，并将文件保存为example.xlsx。

7. 读取数据

在openpyxl中，我们可以使用Worksheet.cell()方法来获取单元格的值。下面是一个例子：

```
from openpyxl import load_workbook

# 打开文件
wb = load_workbook('example.xlsx')

# 获取默认的工作表
ws = wb.active

# 获取单元格的值
print(ws.cell(row=1, column=1).value)
print(ws.cell(row=1, column=2).value)
```

我们使用load_workbook()函数打开了example.xlsx文件，并获取了默认的工作表。接着，我们使用Worksheet.cell()方法分别获取了单元格A1和B1的值，并将它们输出到控制台。

8. 设置单元格样式

在openpyxl中，我们可以使用openpyxl.styles模块来设置单元格的样式。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.styles import Font, Alignment
```

```
# 创建一个Workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置单元格的值
ws['A1'] = 'Hello'
ws['B1'] = 'World'

# 设置单元格的样式
font = Font(name='Arial', size=16, bold=True)
alignment = Alignment(horizontal='center',
vertical='center')
ws['A1'].font = font
ws['B1'].font = font
ws['A1'].alignment = alignment
ws['B1'].alignment = alignment

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1和B1的值，并使用openpyxl.styles模块设置了它们的样式。最后，我们将文件保存为example.xlsx。

9. 处理日期和时间

在openpyxl中，我们可以使用datetime模块来处理日期和时间。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.utils import get_column_letter
from openpyxl.styles import Font, Alignment
from datetime import datetime

# 创建一个Workbook对象
wb = Workbook()

# 获取默认的工作表
```

```
ws = wb.active

# 设置表头
ws['A1'] = 'Date'
ws['B1'] = 'Time'

# 设置表头的样式
font = Font(name='Arial', size=16, bold=True)
alignment = Alignment(horizontal='center',
vertical='center')
for col in range(1, 3):
    cell = ws.cell(row=1, column=col)
    cell.font = font
    cell.alignment = alignment

# 设置数据
for row in range(2, 10):
    date = datetime(2023, 6, row)
    time = datetime(2023, 6, row, 14, 30)
    ws.cell(row=row, column=1, value=date)
    ws.cell(row=row, column=2, value=time)

# 设置数据的格式
for col in range(1, 3):
    column_letter = get_column_letter(col)
    for row in range(2, 10):
        cell = ws[column_letter + str(row)]
        cell.number_format = 'yyyy-mm-dd hh:mm:ss'

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了表头，并使用openpyxl.styles模块设置了它们的样式。然后，我们使用datetime模块设置了数据，并使用Worksheet.cell()方法将它们写入到工作表中。最后，我们使用Worksheet.cell().number_format属性设置了数据的格式，并将文件保存为example.xlsx。

10. 处理公式

在openpyxl中，我们可以使用Worksheet.cell().value属性来设置单元格的值，并使用Worksheet.cell().formula属性来设置单元格的公式。下面是一个例子：

```
from openpyxl import Workbook

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置单元格的值和公式
ws['A1'] = 10
ws['B1'] = 20
ws['C1'] = '=A1+B1'

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1和B1的值，并使用Worksheet.cell().formula属性设置了单元格C1的公式。最后，我们将文件保存为example.xlsx。

11. 处理图表

在openpyxl中，我们可以使用openpyxl.chart模块来处理图表。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.chart import BarChart, Reference

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active
```

```
# 设置数据
for row in range(1, 6):
    ws.cell(row=row, column=1, value=row)
    ws.cell(row=row, column=2, value=row * 10)

# 创建一个柱状图
chart = BarChart()

# 设置图表的数据
data = Reference(ws, min_col=2, min_row=1, max_col=2,
max_row=5)
categories = Reference(ws, min_col=1, min_row=1, max_row=5)
chart.add_data(data)
chart.set_categories(categories)

# 将图表插入到工作表中
ws.add_chart(chart, 'D1')

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了数据，并使用openpyxl.chart模块创建了一个柱状图。然后，我们使用Reference类来设置图表的数据和分类。最后，我们使用Worksheet.add_chart()方法将图表插入到工作表中，并将文件保存为example.xlsx。

12. 处理样式

在openpyxl中，我们可以使用openpyxl.styles模块来处理样式。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.styles import Font, Fill, PatternFill,
Border, Side, Alignment

# 创建一个workbook对象
wb = Workbook()
```

```
# 获取默认的工作表
ws = wb.active

# 设置单元格的值
ws['A1'] = 'Hello'
ws['B1'] = 'World'

# 设置单元格的样式
font = Font(name='Arial', size=16, bold=True, italic=True,
underline='single', color='FF0000')
fill = PatternFill(patternType='solid', fgColor='FFFF00')
border = Border(left=Side(style='thin', color='000000'),
right=Side(style='thin', color='000000'),
top=Side(style='thin', color='000000'),
bottom=Side(style='thin', color='000000'))
alignment = Alignment(horizontal='center',
vertical='center', wrapText=True)
ws['A1'].font = font
ws['B1'].font = font
ws['A1'].fill = fill
ws['B1'].fill = fill
ws['A1'].border = border
ws['B1'].border = border
ws['A1'].alignment = alignment
ws['B1'].alignment = alignment

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1和B1的值，并使用openpyxl.styles模块设置了它们的样式。最后，我们将文件保存为example.xlsx。

13. 处理合并单元格

在openpyxl中，我们可以使用Worksheet.merge_cells()方法来合并单元格。下面是一个例子：

```
from openpyxl import Workbook
```

```
from openpyxl.styles import Alignment

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置单元格的值
ws['A1'] = 'Hello'
ws['B1'] = 'World'

# 合并单元格
ws.merge_cells('A1:B1')

# 设置合并单元格的样式
alignment = Alignment(horizontal='center',
vertical='center')
ws['A1'].alignment = alignment

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1和B1的值，并使用Worksheet.merge_cells()方法合并了它们。最后，我们使用Alignment类设置了合并单元格的样式，并将文件保存为example.xlsx。

14. 处理行和列

在openpyxl中，我们可以使用Worksheet.row_dimensions和Worksheet.column_dimensions属性来设置行和列的属性。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.styles import Font

# 创建一个workbook对象
wb = Workbook()
```

```
# 获取默认的工作表
ws = wb.active

# 设置行和列的属性
for row in range(1, 6):
    ws.row_dimensions[row].height = 30
    for col in range(1, 6):
        ws.column_dimensions[chr(col + 64)].width = 20
        ws.cell(row=row, column=col, value=row * col)

# 设置单元格的样式
font = Font(name='Arial', size=16, bold=True)
for row in range(1, 6):
    for col in range(1, 6):
        ws.cell(row=row, column=col).font = font

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们使用Worksheet.row_dimensions和Worksheet.column_dimensions属性设置了行和列的属性。然后，我们使用Worksheet.cell()方法设置了单元格的值，并使用Font类设置了它们的样式。最后，我们将文件保存为example.xlsx。

15. 处理超链接

在openpyxl中，我们可以使用Worksheet.cell().hyperlink属性来设置单元格的超链接。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.styles import Font
from openpyxl.utils import quote_sheetname

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
```

```
ws = wb.active

# 设置单元格的值和超链接
ws['A1'] = 'Google'
ws['A1'].font = Font(underline='single', color='0000FF')
ws['A1'].hyperlink = 'https://www.google.com/'

# 设置单元格的值和超链接
ws['A2'] = 'Baidu'
ws['A2'].font = Font(underline='single', color='0000FF')
ws['A2'].hyperlink = 'https://www.baidu.com/'

# 设置单元格的值和超链接
ws['A3'] = 'Microsoft'
ws['A3'].font = Font(underline='single', color='0000FF')
ws['A3'].hyperlink = 'https://www.microsoft.com/'

# 设置单元格的值和超链接
ws['A4'] = 'Yahoo'
ws['A4'].font = Font(underline='single', color='0000FF')
ws['A4'].hyperlink = 'https://www.yahoo.com/'

# 设置单元格的值和超链接
ws['A5'] = 'Amazon'
ws['A5'].font = Font(underline='single', color='0000FF')
ws['A5'].hyperlink = 'https://www.amazon.com/'

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们设置了单元格A1到A5的值，并使用Font类设置了它们的样式。然后，我们使用Worksheet.cell().hyperlink属性设置了它们的超链接。最后，我们将文件保存为example.xlsx。

16. 处理数据验证

在openpyxl中，我们可以使用openpyxl.worksheet.datavalidation.DataValidation类来设置数据验证。下面是一个例子：

```
from openpyxl import Workbook
from openpyxl.worksheet.datavalidation import
DataValidation

# 创建一个workbook对象
wb = Workbook()

# 获取默认的工作表
ws = wb.active

# 设置数据验证
dv = DataValidation(type='list', formula1='"Yes,No"',
allow_blank=True)
ws.add_data_validation(dv)
dv.add('A1')

# 保存文件
wb.save('example.xlsx')
```

我们首先创建了一个Workbook对象，然后获取了默认的工作表。接着，我们使用openpyxl.worksheet.dat