1. 项目概述

本项目为实验数据采集与处理软件，旨在为化学实验过程中的数据分析提供便捷的工具。通过图形化界面，用户可以导入实验数据、进行数据处理与清洗、检测异常值、执行拟合分析，并生成各类图表。软件支持多种实验类型，包括但不限于精馏实验、干燥实验、流体实验等。最终，所有处理结果都可以导出为图表文件和报告，便于实验人员进一步分析与分享。

该项目采用Python编写，结合Tkinter构建图形化用户界面，使用Matplotlib、NumPy等库进行数据处理与可视化。

文件结构如下

```
|-- 文件: .gitattributes
|-- 文件: .gitignore
|-- 文件夹: gui
|   |-- 文件: app.py                # 主应用程序文件，负责界面的初始化和管理
|   |-- 文件夹: screens             # 存放不同实验屏幕模块
|   |   |-- 文件夹: calculators     # 存放计算器模块，每个实验对应一个计算器
|   |   |-- 文件夹: common_screens  # 存放通用的屏幕类，如基本界面布局
|   |   |-- 文件夹: common_widgets  # 存放通用的界面组件，如表格、按钮等
|   |   |-- 文件: distillation_screen.py # 精馏实验界面
|   |   |-- 文件: drying_screen.py  # 干燥实验界面
|   |   |-- 文件: extraction_screen.py # 萃取实验界面
|   |   |-- 文件: filteration_screen.py # 过滤实验界面
|   |   |-- 文件: fluid_flow_screen.py # 流体流动实验界面
|   |   |-- 文件: heat_transfer_screen.py # 传热实验界面
|   |   |-- 文件: oxygen_desorption_screen.py # 氧解吸实验界面
|   |   |-- 文件夹: processors      # 各实验数据处理模块
|   |   |-- 文件夹: utils           # 工具类，如配置文件、串口处理等
|   |   |-- 文件夹: widgets         # 存放自定义的控件，如图表显示、输入框等
|-- 文件: LICENSE                   # 开源许可证文件
|-- 文件: README.md                 # 项目说明文件
|-- 文件: requirements.txt          # 项目依赖包
|-- 文件: main.py                   # 程序入口，启动应用
|-- 文件夹: logos                   # 存放应用程序图标
|-- 文件夹: manual                  # 文档文件夹（为空）
```

1.1. gui 文件夹

- **app.py**：应用程序的主逻辑和界面管理。
- **screens**：不同实验的界面模块。
  - **calculators**：包含各实验的数据计算器。
  - **common_screens**：包含界面布局和通用组件。

- **common_widgets**：界面中使用的各种小部件，例如表格、按钮等。
- **processors**：每种实验对应的数据处理模块。
- **widgets**：定制化的控件，用于显示和交互。
- **distillation_screen.py**：精馏实验界面。
- **drying_screen.py**：干燥实验界面。
- **extraction_screen.py**：萃取实验界面。
- **filteration_screen.py**：过滤实验界面。
- **fluid_flow_screen.py**：流体流动实验界面。
- **heat_transfer_screen.py**：传热实验界面。
- **oxygen_desorption_screen.py**：氧解吸实验界面。

1.1.1. app.py 函数说明

`__init__`

```
__init__(
    self,
    dx: float = 0.1,
    time_interval: int = 500,
    plot_max_points: int = 500,
    port_timeout: float = 0.25,
    std_limit: float = 0.005,
    time_lower_limit: int = 30,
    time_upper_limit: int = 40,
    width_height_inches: tuple = (10, 7),
    dpi: int = 600,
    py_path: str = os.path.dirname(os.path.abspath(__file__))
)
```

初始化应用程序，设置数据配置，创建界面窗口，加载所有实验屏幕并初始化当前屏幕。此函数还根据屏幕尺寸调整窗口大小，确保窗口显示在屏幕左上角，并启动主事件循环。

- 参数:
  - `dx`: 积分、绘图步长（默认值：0.1）
  - `time_interval`: 记录数据的时间间隔（单位：毫秒，默认值：500）
  - `plot_max_points`: 绘图最大点数（默认值：500）
  - `port_timeout`: 串口超时时间（单位：秒，默认值：0.25）
  - `std_limit`: 自动寻找平台期的标准差阈值（默认值：0.005）
  - `time_lower_limit`: 自动寻找平台期的最小时间窗口（默认值：30）

- `time_upper_limit`: 自动寻找平台期的最大时间窗口（默认值：40）
- `width_height_inches`: 保存图片的尺寸（单位：英寸，默认值：(10, 7)）
- `dpi`: 图片的DPI（默认值：600）
- `py_path`: 当前脚本的路径（默认值：自动获取）

`_create_mode_menu`

```
_create_mode_menu(self)
```

创建模式切换菜单，允许用户选择不同的实验模式进行切换。为每个实验模式添加对应的菜单项并绑定显示屏幕的功能。

`show_screen`

```
show_screen(self, screen_name)
```

根据传入的屏幕名称显示对应的实验界面。如果当前已有屏幕，则先隐藏当前屏幕。该方法允许在多个实验界面之间进行切换。

- **参数**:
  - `screen_name`: 要显示的屏幕名称（字符串）。

`change_mode`

```
change_mode(self, *args)
```

根据当前选择的模式切换实验界面。此方法用于处理模式切换的事件，例如切换到不同的实验模式屏幕。

- **参数**:
  - `args`: 通过参数传入的模式值，根据该值切换不同的屏幕。

`data_changed`

```
data_changed(self)
```

数据修改响应函数，作为预留接口，方便未来扩展与数据变化的响应处理。当前版本未实现具体功能。

1.1.2. screens 文件夹

包含所有实验界面模块，采用模块化设计实现功能复用

模块名称	功能说明
calculators/	各实验专用计算器（数据处理核心）
common_screens/	基础界面框架和通用功能
common_widgets/	可复用界面组件库
processors/	实验数据后处理模块
utils/	通用工具类（配置/串口等）
widgets/	定制化UI组件

1.1.2.1. 实验界面文件

1. **distillation_screen.py**
   精馏实验可视化界面
2. **drying_screen.py**
   干燥曲线实时显示界面
3. **extraction_screen.py**
   萃取实验操作界面
4. **filteration_screen.py**
   过滤实验监控界面
5. **fluid_flow_screen.py**
   流体阻力与泵特性双实验集成界面
6. **heat_transfer_screen.py**
   传热界面
7. **oxygen_desorption_screen.py**
   氧解吸传质过程界面

1.1.2.2. 设计特点

1. 统一继承`Base_Screen`基础框架
2. 计算器与界面分离架构
3. 采用响应式布局设计
4. 支持实验数据持久化
5. 内置串口通信管理

1.1.2.3. distillation_screen.py

`__init__(window)`
初始化精馏实验界面，继承基类`Base_Screen`，设置文件路径、处理器列表和图像路径初始状态，调用布局调整方法。

`_init_parameter_input()`
完全覆盖基类参数输入配置，创建精馏实验专用参数组件，包含回流比(R)、αm、F(mL/min)、tS(℃)、tF(℃)等参数输入框，并设置验证规则。

`_init_control_buttons()`
保留串口控制按钮并添加精馏实验专用按钮组（导入CSV、处理数据、绘制图形），调整按钮文本和状态。

`_adjust_layout()`
配置表格初始列和列宽，结果表格预设列宽为[80,100,120,120,100]，支持后续动态列更新。

`start_data_acquisition()`
实现采集启动功能，检查串口连接状态后开始采集，显示状态提示信息。

`stop_data_acquisition()`
实现停止采集功能，更新状态显示并提示用户。

`load_data()`
加载CSV实验数据文件，动态生成表格列（序号+CSV列头），更新原始数据表格显示。

`process_data()`
执行数据处理流程，根据界面参数创建处理器，计算理论塔板数等结果并更新结果表格。

`plot_graph()`
生成McCabe-Thiele图等精馏实验图表，更新绘图框架显示，启用分页控件。

`_create_processors()`
根据界面参数动态创建处理器实例，包含常规回流比(R=4)和全回流(R=∞)两种情况。

`_process_all_cases()`
批量处理所有回流比情况，调用处理器进行计算但不立即显示图表。

`_update_raw_table(df)`
将DataFrame数据格式化后填充原始数据表格，自动添加行号列。

`_update_result_table()`
格式化显示处理结果，包含回流比、理论/实际塔板数（保留2位小数）和分离效率。

`_generate_plots()`
生成McCabe-Thiele图并保存到指定路径，收集所有生成的图像文件路径。

`_get_plot_paths()`
验证并返回实际存在的图表文件路径列表，确保只显示有效图像。

1.1.2.4. drying_screen.py

`__init__(window)`
初始化干燥实验界面，继承基类`Base_Screen`，设置文件路径、处理器和结果数据初始状态，调用组件调整和按钮状态更新方法。

`_adjust_base_components()`
调整界面布局，隐藏不需要的参数输入组件，初始化专用表格配置（原始数据表列宽[120,120,140,140]，结果表列宽[180,220]）。

`_init_drying_tables()`
创建优化后的表格布局，原始数据表显示15行数据，结果表显示8行数据，使用pack布局管理器排列。

`_update_button_states()`
根据串口连接状态动态更新按钮可用性，实验控制按钮在串口连接后启用，数据处理按钮在加载文件后启用。

`_find_named_frame(frame_name)`
递归查找指定名称的框架组件，支持在多级子组件中定位目标框架。

`_open_serial(port, baudrate)`
重写串口打开方法，连接成功后更新按钮状态并显示提示信息，异常时调用统一错误处理。

`_close_serial()`
重写串口关闭方法，断开连接后更新按钮状态并显示提示信息。

`start_data_acquisition()`
实现数据采集启动逻辑，需先验证串口连接状态，显示处理中提示框，预留具体采集逻辑实现位置。

`stop_data_acquisition()`
停止数据采集，关闭处理中提示框并显示操作提示。

`load_data()`
加载干燥实验CSV数据文件，验证必须包含"原始数据1"和"原始数据2"两个文件，读取数据后更新原始表格。

`_validate_files(file_paths)`
验证文件命名规范，确保加载的文件包含实验所需的全部数据类型，缺失时抛出明确错误。

`_update_raw_table(data)`
格式化显示原始实验数据，数值类型保留2位小数，支持混合数据类型显示。

`process_data()`
执行数据处理流程，计算恒定干燥速率、传热系数等关键参数，生成图表文件路径，更新结果表格。

`_update_result_table()`
格式化显示处理结果，关键参数显示4-6位小数精度，包括恒定干燥速率、传热系数和初始体积流量。

`plot_graph()`
显示生成的干燥曲线图表，支持多图切换查看，需先完成数据处理流程。

`_handle_error(context, error)`
统一错误处理方法，关闭处理中提示框，显示错误弹窗并记录详细错误日志到文件。

1.1.2.5. extraction_screen.py

`__init__(window)`
初始化萃取实验界面，继承基类`Base_Screen`的初始化方法，创建文件字典用于存储原始数据和分配曲线文件路径，调用组件调整和按钮状态更新方法。

`_adjust_components()`
调整界面组件布局，重新创建原始数据表格（15行高度）和结果表格（8行高度），调整列宽配置，使用`pack`布局管理器进行组件排列。

`_update_button_states()`
更新按钮可用状态，检查文件字典中是否同时存在原始数据和分配曲线文件，根据检查结果设置数据处理按钮的启用/禁用状态。

`_find_button(frame_name, column)`
查找指定位置的按钮组件，根据容器框架名称和列号定位按钮对象，若组件不存在会引发`tkinter.TclError`。

`load_data()`
加载双CSV文件数据，弹出文件选择对话框，通过智能分类和验证后读取原始数据CSV并更新表格，捕获文件读取错误并记录日志。

`_classify_files(paths)`
智能分类CSV文件，根据文件名中的关键词（原始数据/origin/*m，分配曲线/distribution*/d）将文件分类到origin/distribution键中。

`_validate_files()`
验证文件完整性，检查文件字典是否存在空值，若存在缺失文件则抛出包含明确提示信息的`ValueError`。

`process_data()`
执行数据处理流程，显示处理中提示框后初始化处理器并运行计算，更新结果表格和状态提示，需确保已完整加载必需文件。

`_update_result_table()`
更新结果表格数据，显示分配系数（保留4位小数）、操作线斜率（保留2位小数）和积分结果（科学计数法格式）。

`plot_graph()`
显示处理生成的图表，检查处理器输出目录并加载所有PNG文件，调用基类方法显示图片，自动触发首张图片显示。

`_safe_close()`
安全释放资源，删除处理器实例并调用基类清理方法，确保程序退出时正确释放所有资源。

1.1.2.6. filteration_screen.py

`__init__(window)`
初始化过滤实验界面，继承基类`Base_Screen`，设置CSV文件路径、处理数据和处理器为初始状态，调用实验按钮配置方法。

`_configure_experiment_buttons()`
配置实验专用按钮状态，保留但禁用串口功能按钮，保持界面布局完整性。

`_disable_unimplemented_buttons()`
禁用未实现功能的按钮组件，特别是串口相关按钮，通过遍历按钮框架子组件实现。

`load_data()`
加载CSV实验数据文件，使用pandas读取并预处理数值数据，更新原始数据表格显示，包含错误处理和状态提示。

`process_data()`
执行数据处理流程，初始化处理器进行斜率/截距计算，将结果更新至结果表格，数值格式化为4位小数和2位小数。

`plot_graph()`
生成实验图表并更新显示，调用处理器的绘图方法获取图像路径，更新绘图框架内容，包含处理状态提示。

`_safe_close()`
安全释放资源，清理处理器实例并调用基类关闭方法，确保程序退出时资源正确释放。

1.1.2.7. fluid_flow_screen.py

`__init__(window)`
初始化流体流动实验界面，继承基类`Base_Screen`，设置处理器、文件路径和图像路径初始状态，调用组件调整和按钮状态更新方法。

`_adjust_components()`
调整界面表格布局，重新配置原始数据表格（15行高度，列宽[60,120,120,120]）和结果表格（8行高度，列宽[60,120,120,120,120,120]），设置图像路径更新方法。

`_update_button_states()`
根据文件加载状态更新数据处理按钮可用性，当CSV文件路径列表非空时启用处理按钮。

`_find_button(frame_name, column)`
在指定框架中按列号定位按钮组件，用于动态控制按钮状态。

`load_data()`
加载流体阻力和离心泵实验数据文件，通过智能分类验证文件后，读取并显示流体阻力数据到原始表格，包含完整的错误处理流程。

`_classify_files(paths)`
根据文件名关键词（流体阻力/fluid，离心泵/pump）自动分类实验数据文件，返回排序后的文件路径列表。

`_validate_files()`
验证是否同时包含两类实验文件，缺失时抛出包含明确命名提示的ValueError异常。

`process_data()`
执行完整的数据处理流程，包括流体阻力计算和离心泵特性分析，通过处理器生成计算结果并更新结果表格。

`_update_result_table()`
格式化显示处理结果，流体阻力部分显示平均雷诺数和摩擦系数，离心泵部分显示平均扬程、功率和效率，均保留2位小数。

`plot_graph()`
生成并显示实验图表，精确匹配三种标准图表文件（阻力系数拟合和泵特性曲线），初始化绘图框架并显示首张图片。

`_safe_close()`
安全释放资源，确保处理窗口关闭后清理处理器实例，最后调用基类关闭方法完成界面清理。

1.1.2.8. heat_transfer_screen.py

`__init__(window)`
初始化传热实验界面，继承优化后的基类版本，设置文件路径和处理器初始状态，调用增强初始化方法配置界面。

`_enhance_init()`
执行额外的初始化配置，创建参数输入组件，设置初始状态提示，初始化日志记录器。

`_create_parameter_widget()`
创建专用的参数输入组件模板，使用StringEntriesWidget实现参数输入区域，支持后续扩展。

`load_data()`
加载传热实验CSV文件，支持多文件选择，自动分类不同实验条件的文件，验证完整性后加载数据到原始表格。

`_classify_files(paths)`
智能分类传入的文件路径，识别无/有强化套管数据文件及其预处理文件，缺失必要文件时抛出明确错误。

`process_data()`
执行传热实验数据处理流程，分步骤计算和存储结果，更新结果表格显示关键传热参数。

`_update_result_table()`
格式化显示处理结果，包括雷诺数(Re)、普朗特数(Pr)和Nu/Pr^0.4等参数，均保留2位小数精度。

`plot_graph()`
生成并显示传热性能图表，包括拟合曲线和性能对比图，自动压缩结果文件节省空间。

`_update_table(table, data)`
通用表格更新方法，支持数值格式化（保留2位小数）和类型安全转换，自动添加行号列。

`_handle_error(title, error)`
统一错误处理机制，记录详细错误日志到文件，显示用户友好提示，确保资源正确释放。

`start_data_acquisition()`
空实现采集方法（本实验不需要实时采集功能），保持接口兼容性。

`stop_data_acquisition()`
空实现停止方法（本实验不需要实时采集功能），保持接口兼容性。

1.1.2.9. oxygen_desorption_screen.py

`__init__(window)`
初始化氧解吸实验界面，继承基类`Base_Screen`，设置文件路径字典和图像路径列表，调用自定义组件初始化方法。

`_init_custom_components()`
配置实验特有界面元素，包括更新表格列宽（原始表各列100px，结果表各列150px），初始化参数输入区域，绑定功能按钮。

`_bind_buttons()`
动态查找并绑定基类创建的"处理数据"和"绘制图形"按钮实例，便于后续状态控制。

`_init_parameter_input()`
初始化参数输入框架，当前版本不包含具体参数输入（空配置），保留参数变更事件绑定结构。

`load_data()`
重写基类数据加载方法，调用文件导入逻辑，支持多文件选择。

`_import_all_files()`
实现多文件导入功能，根据文件名关键词（干填料/湿填料/水流量一定/空气流量一定）自动分类文件路径。

`_check_files_complete()`
验证实验必需的4类文件是否齐全，更新处理按钮状态，缺失时显示明确提示。

`_load_data_preview(path)`
加载干填料文件的预览数据到原始表格，显示水流量、空气流量等关键实验参数。

`process_data()`
执行完整数据处理流程，初始化实验处理器并运行所有计算，更新结果表格，启用绘图按钮。

`_update_results()`
格式化显示计算结果，包括填料塔性能拟合类型和氧解吸传质系数（保留4位小数）。

`plot_graph()`
动态获取最新生成的图表路径（填料塔性能对比图和氧解吸传质关联图），更新绘图区域显示。

`_on_parameter_change(event)`
参数变更响应方法（当前版本未实际使用，保留结构）。

`_safe_close()`
安全关闭方法，确保资源正确释放，调用基类关闭逻辑。

1.1.3. base_screen 文件夹

1.1.3.1. base_screen.py

GUI基础屏幕模块，提供实验界面通用框架和核心功能组件

`Base_Screen(ttk.Frame)`: 实验屏幕基类，集成界面布局、串口通信、数据展示等核心功能
`__init__(self, window)`: 初始化框架并绑定安全关闭事件
`_init_status_bar(self)`: 创建底部状态栏显示实时状态
`_init_components(self)`: 构建左右面板基础布局
`_init_parameter_input(self)`: 创建空参数输入容器供子类扩展
`_init_control_buttons(self)`: 初始化实验控制和数据处理按钮组
`_init_data_tables(self)`: 创建原始/结果双表格组件
`_toggle_serial(self)`: 切换串口连接状态
`start_data_acquisition(self)`: 数据采集启动接口（需子类实现）
`show_processing(self)`: 显示带进度条的模态对话框
`update_table(self)`: 通用表格数据更新方法
`_safe_close(self)`: 安全关闭窗口并释放资源

1.1.3.2. init.py

(空文件，无函数或类需要说明)

1.1.3.3. calculators 文件夹

1.1.3.3.1. distillation_calculator.py

Distillation_Calculator: 精馏塔计算器类
__init__: 初始化并执行完整计算流程
set_constants: 设置乙醇-水物性常数
calculate_feed_parameters: 计算进料热状态参数q值
calculate_x_ethanol: 体积分数转摩尔分数
calculate_compositions: 计算关键组分组成
solve_material_balance: 求解物料平衡方程
y_e: 气液平衡方程计算
x_e: 反平衡方程计算
y_np1: 精馏段操作线方程
y_mp1: 提馏段操作线方程
calculate_stages: 逐板法计算理论塔板数
save_results: 保存关键计算结果

1.1.3.3.2. drying_calculator.py

Drying_Calculator: 干燥实验计算器
__init__: 初始化并加载双数据文件
load_data: 解析静态参数和时间序列数据
preprocess_data: 核心预处理计算干基含水量
further_calculations: 高级计算传热系数和流量
run_full_calculation: 执行完整计算流程

1.1.3.3.3. extraction_calculator.py

Extraction_Calculator: 萃取实验计算器
__init__: 初始化主数据和分配曲线文件
load_data: 加载流量和滴定数据
preprocess_data: 计算苯甲酸浓度和物料平衡
load_distribution_curve_data: 加载分配曲线数据集
fit_distribution_curve: 三次多项式拟合分配曲线
calculate_operating_lines: 计算操作线参数
perform_graphical_integration: 图解积分计算传质单元数
run_calculations: 执行完整计算流程

1.1.3.3.4. filteration_calculator.py

Filteration_Calculator: 过滤实验计算器
__init__: 初始化过滤面积参数
load_csv_data: 加载并清洗CSV数据
perform_linear_fit: 执行线性回归拟合
detect_outliers: Z-score异常值检测
refit_data_after_outlier_removal: 异常值移除后重新拟合
process_single_group_data: 处理单组实验数据
process_all_groups: 处理全部三组数据

1.1.3.3.5. fluid_flow_calculator.py

Fluid_Flow_Calculator: 流体阻力分析器
__init__: 初始化管径和物性参数
process: 计算雷诺数和摩擦系数
Centrifugal_Pump_Characteristics_Calculator: 离心泵分析器
quadratic: 二次拟合函数
process: 计算扬程和效率参数

1.1.3.3.6. heat_transfer_calculator.py

Heat_Transfer_Calculator: 传热分析器
__init__: 分类强化/非强化套管数据
_categorize_files: 根据文件名自动分类
preprocess_data: 预处理物性参数和传热系数
fit_func: 通用对数坐标拟合函数
process_data: 执行完整传热分析

1.1.3.3.7. oxygen_desorption_calculator.py

Experiment_Data_Loader: 实验数据加载器
__init__: 初始化四类数据文件路径
get_file: 通过标识符获取文件路径
Packed_Tower_Calculator: 填料塔分析器
calc_fluid_dynamics: 流体力学参数计算
Oxygen_Desorption_Calculator: 氧解吸分析器
oxygen_solubility: 温度修正溶解度公式
analyze_file: 计算传质系数和推动力

1.1.3.3.8. init.py

(空文件，无函数或类需要说明)

1.1.3.4. plotters 文件夹

1.1.3.4.1. distillation_plotter.py:

`Distillation_Plotter` 精馏塔可视化绘图类，生成McCabe-Thiele图及理论塔板计算结果
`_generate_plot_data` 生成绘图所需数据
`plot_mccabe_thiele` 绘制McCabe-Thiele图

1.1.3.4.2. drying_plotter.py:

`Drying_Plotter` 干燥实验数据可视化类
`plot_drying_curve` 绘制物料干基含水量随时间变化曲线
`plot_drying_rate_curve` 绘制干燥速率曲线
`integrate_images` 生成组合对比图
`compress_results` 打包结果文件
`serialize_results` 序列化实验结果

1.1.3.4.3. extraction_plotter.py:

`Extraction_Plotter` 萃取实验数据可视化类
`plot_origin_curves` 绘制分配曲线与操作线分析图
`plot_integration_curves` 绘制图解积分曲线
`package_results` 打包分析结果

1.1.3.4.4. filteration_plotter.py:

`Filteration_Plotter` 过滤实验数据可视化类
`create_initial_fit_figure` 创建初拟合图表
`create_refit_figure` 创建重新拟合图表
`generate_comparison_figures` 生成对比分析图
`integrate_figures` 合并所有图表

1.1.3.4.5. fluid_flow_plotter.py:

`Fluid_Flow_Plotter` 流体阻力数据可视化类
`plot` 绘制雷诺数与阻力系数双对数拟合图
`Centrifugal_Pump_Characteristics_Plotter` 离心泵特性曲线可视化类
`PlotManager` 绘图管理类，协调多个绘图任务

1.1.3.4.6. heat_transfer_plotter.py:

`Heat_Transfer_Plotter` 传热实验数据可视化类
`plot_fit` 执行曲线拟合并绘制结果
`generate_comparison_plot` 生成传热性能对比图

1.1.3.4.7. oxygen_desorption_plotter.py:

`Packed_Tower_Plotter` 填料塔流体力学性能可视化类
`plot_comparison` 绘制干/湿填料性能对比图
`Oxygen_Desorption_Plotter` 氧解吸实验数据可视化类
`plot_correlation` 绘制传质系数关联图

1.1.3.4.8. init.py:

(空文件，无函数或类需要说明)

1.1.3.5. processors 文件夹

1.1.3.5.1. distillation_experiment_processor.py

`Distillation_Experiment_Processor` 精馏实验流程处理器，自动化执行完整实验流程并管理结果文件
`_prepare_directory` 创建结构化输出目录
`process_experiment` 执行完整实验处理流程
`_save_text_results` 保存文本计算结果
`_generate_plots` 生成可视化图表
`_create_archive` 创建ZIP打包文件

1.1.3.5.2. drying_experiment_processor.py

`Drying_Experiment_Processor` 干燥实验数据处理类，继承自Drying_Calculator
`process_experiment` 处理整个干燥实验过程
`get_results` 获取完整的计算结果
`get_plots` 获取绘图结果

1.1.3.5.3. extraction_experiment_processor.py

`ExtractionExperimentProcessor` 萃取实验数据处理类
`validate_files` 验证输入文件有效性
`setup_components` 初始化各处理组件
`process_data` 执行完整数据处理流程
`print_summary` 输出处理结果摘要
`run` 主执行流程

1.1.3.5.4. filteration_experiment_processor.py

`Filteration_Experiment_Processor` 过滤实验数据处理类
`calculate` 使用计算类处理数据
`store` 存储处理后的数据
`plot` 生成所有所需图形
`compress_results` 压缩结果图像文件

1.1.3.5.5. fluid_flow_experiment_processor.py

`Fluid_Flow_Expriment_Processor` 流体流动实验处理类
`process_fluid_flow` 处理流体阻力实验数据
`process_pump_characteristics` 处理离心泵特性实验数据
`generate_all_plots` 生成所有分析图表
`get_fluid_flow_results` 获取流体阻力实验结果
`get_pump_characteristics_results` 获取离心泵特性实验结果

1.1.3.5.6. heat_transfer_experiment_processor.py

`Heat_Transfer_Experiment_Processor` 传热实验数据处理类
`calculate` 处理传热实验数据
`store` 存储处理后的数据
`plot` 生成所有图形
`compress_results` 压缩结果文件
`fit_data_summary` 生成实验拟合数据报告

1.1.3.5.7. oxygen_desorption_experiment_processor.py

`Result_Compressor` 结果文件压缩工具类
`Oxygen_Desorption_Experiment_Processor` 氧解吸实验处理类
`run_all_calculations` 执行完整计算流程

1.1.3.5.8. init.py

(空文件，无函数或类需要说明)

1.1.3.6. common_widgets 文件夹

1.1.3.6.1. plot_widget.py

`PlotWidget`类：基于Matplotlib的Tkinter绘图控件，支持折线图、散点图和图像显示，带分页功能。

`_set_plot_style`函数：配置图表样式（隐藏刻度线并保留边框）。

`clear`函数：清除当前图表内容。

`plot`函数：绘制折线图。

scatter函数：绘制散点图。

_adjust_plot_limits函数：自动调整坐标范围使图形填满绘图区。

show_current_image函数：显示当前图像（填满整个绘图区）。

resize_image函数：响应窗口大小变化重新绘制图像。

_create_pagination_controls函数：创建居中分页控制组件。

set_images_paths函数：设置图像路径列表并显示第一张图。

_update_page_controls函数：更新分页控制组件的状态。

prev_page函数：切换到上一页图像。

next_page函数：切换到下一页图像。

1.1.3.6.2. string_entries_widget.py

StringEntriesWidget类：统一输入框宽度的输入组件，支持验证和参数变更事件。

update_entries函数：根据配置更新输入框布局和验证规则。

_validate_input函数：使用正则表达式验证输入内容。

get_values函数：获取所有输入框的值。

set_values函数：批量设置输入框的值。

clear函数：清空所有输入框。

validate_all函数：验证所有输入框内容是否符合规则。

1.1.3.6.3. table_widget.py

TableWidget类：通用表格数据显示组件，支持动态列配置和多线程安全操作。

_create_table函数：初始化表格核心组件和列配置。

_create_scrollbar函数：构建表格的垂直滚动条系统。

update_columns函数：动态更新表格列配置。

append函数：添加数据行并支持自动滚动。

append_thread_safe函数：线程安全方式添加数据行。

_process_buffer函数：处理缓冲区中的待添加数据。

clear函数：清空表格所有数据。

set_cell_value函数：设置指定单元格的值。

get_cell_value函数：获取指定单元格的值。

sort_column函数：按列排序表格数据。

set_font函数：设置表格字体样式。

set_foreground函数：设置表格前景色。

set_background函数：设置表格背景色。

set_row_colors函数：设置行的交替颜色。

auto_resize_columns函数：自动调整列宽以适应内容。

1.1.3.6.4. init.py

(空文件，无函数或类需要说明)

1.1.3.7. utils 文件夹

1.1.3.7.1. config.py

SHORTCUT_CODE 快捷键对应event.state的数值字典
DATA_CONFIG 应用数据配置字典，包含窗口、模式等设置
SCREEN_CONFIG 屏幕显示配置，包含边框宽度和样式
MAIN_FRAME_CONFIG 主框架显示配置
RAISED_SUBFRAME_CONFIG 凸起子框架显示配置
FLAT_SUBFRAME_CONFIG 扁平子框架显示配置
ENTRY_LABEL_CONFIG 输入框标签配置
PLOT_CONFIG 绘图样式配置字典
DEFAULT_DATA_VALUE 默认数据值字典

1.1.3.7.2. expserial.py

Experiment_Serial 实验序列化类(当前为空实现)

1.1.3.7.3. init.py

(空文件，无函数或类需要说明)

1.1.3.8. maths 文件夹

1.1.3.8.1. common_maths.py

`Class`类：示例类（当前为空实现）。
`func`函数：示例函数（当前为空实现）。

1.1.3.8.2. init.py

(空文件，无函数或类需要说明)

1.1.3.9. widgets 文件夹

1.1.3.9.1. init.py

(空文件，无函数或类需要说明)

1.1.4. logos 文件夹

```
目录结构:
|-- 文件: ce.icns
|-- 文件: ce.ico
|-- 文件: ce.png
```

1.1.5. manual

(空文件夹)

1.2. init.py

（空文件，无函数或类需要说明）

1.3. LICENSE

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. https://fsf.org/
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

1.4. main.py

这个 `main.py` 文件是一个 Python 程序的入口文件，负责初始化配置、路径处理，主要用于启动一个 GUI 应用程序。以下是对其功能的详细解释：

1.4.1. 导入模块

- **内置模块**：`sys`, `os`, `shutil` 用于处理系统路径、文件操作等。
- **动态路径设置**：通过 `sys.path.insert` 将项目根目录添加到 Python 路径中，确保可以导入项目内的其他模块。
- **自定义模块**：从 `gui.app` 导入 `App` 类，这是程序的主要 GUI 应用类。

1.4.2. 可调参数

定义了一系列可配置的参数，用于控制程序的行为：

- `dx`：积分和绘图的步长。
- `time_interval`：数据记录的时间间隔（毫秒）。
- `plot_max_points`：绘图时显示的最大数据点数。
- `port_timeout`：串口通信的超时时间（秒）。
- `std_limit`：自动寻找平台期的标准差阈值。
- `time_lower_limit` 和 `time_upper_limit`：平台期的最小和最大时间窗口。
- `width_height_inches` 和 `dpi`：保存图片的尺寸和分辨率。

1.4.3. 主程序逻辑

- **路径处理**：
  - 检查程序是否被打包（如 PyInstaller 打包的 exe 或 macOS 的 app）。
  - 如果是打包的程序，获取可执行文件的路径；如果是 macOS 的 app，需要向上回溯 3 层目录。
  - 如果是直接运行的 Python 脚本，获取脚本所在的目录。
- **启动应用**：
  - 创建 `App` 类的实例，传入所有可调参数和路径。
- **清理缓存**：
  - 删除 `__pycache__` 目录（如果存在），避免缓存文件干扰。

1.4.4. 功能总结

- 这是一个物理化学实验数据采集和分析程序的入口，主要用于：
  1. 配置程序运行参数（如绘图、串口通信等）。
  2. 处理路径问题，确保程序在打包或直接运行时都能正确找到资源。
  3. 启动 GUI 应用（`App` 类）。
  4. 清理临时文件。

1.4.5. 备注

- 程序参考了北京大学的物理化学实验项目设计，是一个非盈利软件。
- 支持 Windows 和 macOS 平台，通过 `main-win.exe` 或 `main-mac.app` 启动。
- 如果运行异常，可以通过 `requirements-win.exe` 或 `requirements-mac.app` 安装依赖。

1.5. README.md

说明文件

1.6. requirements.txt

依赖文件