A blue and black logo

Description automatically generated

**Information and Data Collation using Python with Excel**

**Functionality & Code Documentation**

**Dated: December 2023**

**For: Catering & Logs Team**

Table of Contents

[Agenda & Purpose 3](#_Toc152256040)

[Agenda 3](#_Toc152256041)

[Background 3](#_Toc152256042)

[Solution 3](#_Toc152256043)

[Constraints & Assumptions 3](#_Toc152256044)

[Functionalities & Code 4](#_Toc152256045)

[Main Function (Starting Point) 4](#_Toc152256046)

[Reading Data 5](#_Toc152256047)

[Defining Columns Needed 5](#_Toc152256048)

[Reading From Files 5](#_Toc152256049)

[Get School Buildings 6](#_Toc152256050)

[Conversion of Data 6](#_Toc152256051)

[Mapping Sessions to Schedule 7](#_Toc152256052)

[Combine Value Session Date-Time & Session Venue 7](#_Toc152256053)

[Combine Multiple Sessions to Schedule 8](#_Toc152256054)

[Getting Course Audience 9](#_Toc152256055)

[Structuring Data for Output 9](#_Toc152256056)

[Verifying Schedule Record Validity 10](#_Toc152256057)

[Values for Output 11](#_Toc152256058)

[Find Venue Type 12](#_Toc152256059)

[Change Venue Names 13](#_Toc152256060)

[Adding Total Pax 13](#_Toc152256061)

[Styling of Excel 14](#_Toc152256062)

[Execution of the Code 15](#_Toc152256063)

[Downloading of the Code 15](#_Toc152256064)

# Agenda & Purpose

## Agenda

This documentation serves as a purpose to document down the functionalities of the code and explanation. Some knowledge of Python programming is beneficial as there will be code snippets in this documentation to demonstrate the functionalities.

## Background

Catering and Logs (C&L) team needs an efficient and reliable way to compile an overview of the courses that is being provided by SMU Academy. The C&L team is responsible to order catering services whenever there are courses that is being offered. Navigating between multiple windows of different Excel sheets is a challenge to effectively carry out the task.

## Solution

The solution created to tackle the problem is a Python program script that can generate the desired output of overview consistently. Within clicks of the Python script, it should be able to compile Excel sheets, manipulate data, and generate an output Excel sheet with overview.

## Constraints & Assumptions

Before coming up with the Python program, there are certain assumptions and constraints that were made to scoping the requirement:

* There will be 3 exact Excel (not CSV) files with certain naming standardised (“gvSession.xlsx”, “Manage Schedule.xlsx”, “Enrolment Summary.xlsx”) present in the same directory with the Python script. Otherwise, an error will be thrown.
* The headers’ name (a.k.a. the first row of Excel) does not change often.
* All the “manageSchedule.xlsx” *(“Schedule”)* records will have more than or equals to 0 records of “gvSession.xlsx” *(“Session”)* records. If there is a Session and its Schedule is not reflected, this means that the Session does not have a Schedule.
* Schedule will be the primary file that collates the data and information. There will be a key “Schedule #” *(“Schedule Number”)* that maps everything together.

# Functionalities & Code

**Note: The exact numbering of the code and the code snippets may differ slightly, but the naming of the functionality will be roughly the same. Refer accordingly.**

## Main Function (Starting Point)

This is the entry/starting point of the code. This will make sure that the code runs sequentially and executes. This part of the code is usually located the most bottom part of the code because Python executes from top-down, *i.e. executing A then B allow B to ‘use’ A, but not the other way round.*



Under the main function, it consists of various functions that executes step-by-step, which it is readable by its naming convention. In this part, I will try to summarise the code’s functionality, if need be, more elaboration can be found in the code functions explanation.A screen shot of a computer program

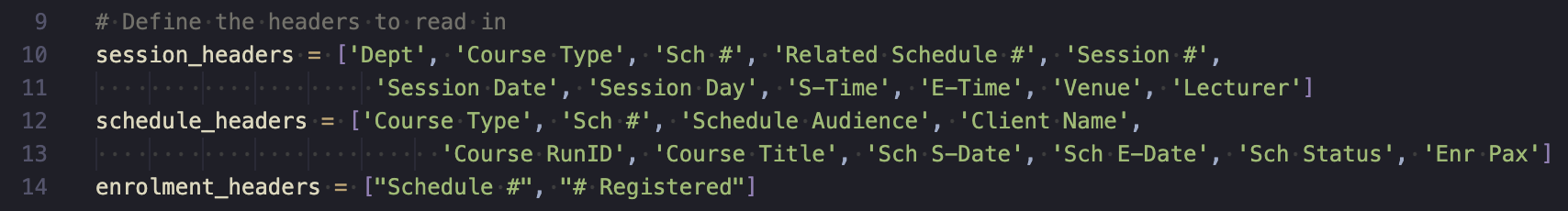
Description automatically generated

* Line 309-312: This is to check if there exist 3 exact files together with the same directory as the source code. The name of the files is stated in the code.
* Line 314: Read data from the relevant Excel files.
* Line 315: Get relevant inputs (that may be changed regularly) from a JSON[[1]](#footnote-1) file.
* Line 317: Convert the data that is being read into Dictionary­­­­[[2]](#footnote-2) so that it is easier to use
* Line 318: Get relevant details like pillar and compile all the different Sessions to a Schedule using a Schedule Number as the key.
* Line 319: Get course audience and map it to relevant Schedule Number.
* Line 321: Structure the data so that it is ready to export it out to an Excel sheet.
* Line 323-324: Get the current datetime in a certain format and use it as part of the filename when exporting the data.
* Line 326: Converts the Dictionary data into a DataFrame[[3]](#footnote-3) data so that it is easier to export as it has built-in functions to do so.
* Line 327: Sort the data by “Start Date” column first, then “Course No.” column. The order will be increasing.
* Line 329: Prepare the Excel file and the type of “exporter” to export the raw data into an Excel sheet with the name.

## Reading Data

### Defining Columns Needed

In this code, the column names of the relevant files have been specified to read in the relevant columns. Although there is a way to read all the data into the program and filter, defining it at the start of the code will allow easier maintainability and reference in the future without opening the Excel sheet. If there is a column that has a change of name, you may define it here.



* Line 10: “session\_headers” is for Session records.
* Line 12: “schedule\_headers” is for Schedule records.
* Line 14: “enrolment\_headers” is for Enrolment Summary records.

### Reading From Files

This function is to read data from the Excel files provided, as you can see the filename specified. In this code snippet, the Python code primarily deals with Python DataFrame type of data, hence, in this format, it may not be the most optimal to deal with when trying to do some specific data manipulation. *(We will convert it later ;D)*

A screen shot of a computer program

Description automatically generated

* Line 52-54: Read in data from Excel, using the column names that is being specified. “fillna()” is a function to replace cell values that are empty to “-“.
* Line 56: Pass back the relevant data to the Main function.

### Get Regularly Change Data

This function is to get data that (may) regularly change. Thus, in order to enhance the user’s flexibility on giving input while not having to interact with the code that often, user can input the data into external file called “data.json”.

Data that is being read in and used are “days” (to flag out course Start-End date that is more than a certain number of days), and “schools” (for the venue types).

**Note: Changes to the school building names and number of days to be flagged out, please use the “data.json” file instead.**

A screen shot of a computer program

Description automatically generated

An example of how the JSON file looks like

A computer screen with text on it

Description automatically generated

* Line 29: Open a file called “data.json” and read in the data.
* Line 30: Using a library called “json”, load in the file data as JSON type.
* Line 31: With the data read in, get the “schools” data (which is a list of school building names), and then convert it into a set (to prevent duplication).
* Line 32: Get the “days” data.
* Line 34: Return the relevant data back to the main code

## Conversion of Data

As the data is in Python DataFrame, it may be easy to deal with when doing certain operations, *i.e. mapping multiple Sessions to a Schedule by using the Schedule Number*. We will convert it into a Dictionary type as it can be accessible using the column name later.

A screen shot of a computer program

Description automatically generated

* Line 34: This is a function with parameters as the function needs the 3 different data that has been read in to do data conversion. If nothing is provided, the code will throw error.
* Line 38-40: Convert Python DataFrame data type to Python Dictionary.

Note: For Line 38, there may be multiple Sessions with the same Schedule Number, hence, I cannot use Schedule Number as the key. For Line 39-40, “set\_index()” function allows the column specified to be the key for the rest of the other values in the same row. All these will result into a Dictionary.

* Line 42: Return the dictionary back to the main code.

## Mapping Sessions to Schedule

### Combine Value Session Date-Time & Session Venue

This function is to combine and format multiple columns like session’s date, time, number, and type into a single value. In this function, we are not using the Dictionary yet.

A screen shot of a computer program

Description automatically generated

* Line 69-72: Format each row to have session date & time. *i.e. “2023-11-01 M7 : Wednesday 09:00 AM to 12:45 PM”.*
* Line 75-77: Format each row to have session venue. *i.e. “2023-11-01 M7 - Venue: Cancelled”.*

### Combine Multiple Sessions to Schedule

As there may be multiple Session record, both normal and assessment Sessions, with the same Schedule Number, we need all the Sessions’ details like date, time, and venue, etc. This function helps to collate all the relevant data and map it to a single Schedule Number. Additionally, the course pillar is stored inside each individual Session record, hence, course pillar will be extracted to the map.

A screen shot of a computer program

Description automatically generated

* Line 108: Call the function to combine and format Session’s date, time, and venue.
* Line 110-113: A dictionary initialised so that the pillar names can be converted to a short form.
* Line 115-127: A dictionary initialised, using the Schedule Number as key, so that all the Sessions details can be put inside the map for easier access.
* Line 129: Goes through all the Session records.
* Line 130: Get the Schedule Number of the Session.
* Line 132-137: This condition is to check if there is pillar exists inside the map, otherwise, pillar will be assigned.

Note: Line 133 is to convert the pillar name into short form, if the pillar is not found in “name\_dict”, it will just assign “No dept”.

* Line 139-149: This code assigns the relevant Session data, checking if the Session is a normal or assessment Session.

Note: If a particular Session is an “Assessment” Session, it will use alternative column “Related Schedule #”, which is to identify the original Schedule Number.

## Getting Course Audience

As the name of the function suggests, it is to get and format the relevant course audience.

A screen shot of a computer program

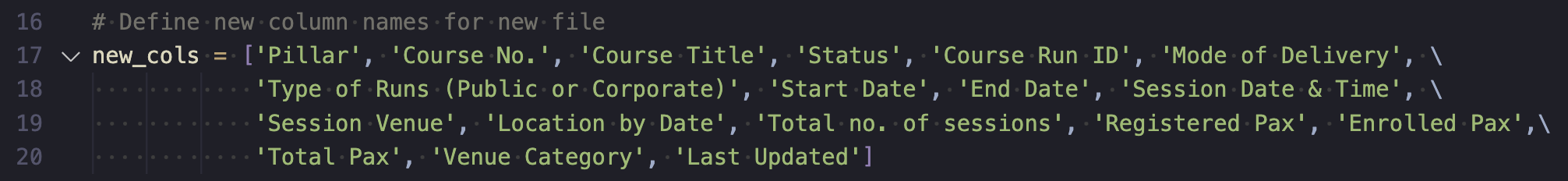
Description automatically generated

* Line 86: An empty dictionary initialised.
* Line 87: Goes through all the Schedule records to find out the course audiences.
* Line 90-97: Format the data such that if the Schedule record’s audience is empty, it will be just “-”.

Note: Line 90 checks if the audience is an empty value in Excel. *(Data in the code has been changed by “fillna()” function earlier, hence, the condition checks for ‘-‘ instead)* Line 93 checks if there is a specific client name if it is for a company, otherwise, the audience will just be the company’s name.

## Structuring Data for Output

* **Refer to “structure\_data()” function.**

This part of the code is critical for the output as the structuring of data to output is done here. For reference, we have initialised “new\_cols” at the top of the code to structure how we want the columns of the Excel sheet to look like.

The code is quite long (not a good practice) hence, the code will be broken down into smaller parts for clarity and understandability.

### Verifying Schedule Record Validity

This part of the code is to verify if a record of Schedule is valid or not. If it is not valid, we do not need to waste time to extract relevant details and display it for the user.

A screen shot of a computer code

Description automatically generated

* Line 225: Check if a Schedule is an assessment. If it is, then skip this record. All the relevant data have already added via the “map\_sessions()” function.
* Line 228: Check if there are Sessions for this Schedule. If no, skip this Schedule because it is irrelevant to show no Sessions (*i.e. 0 sessions*).
* Line 232-235: Get the pillar of the Schedule and check if it exists. If no pillar exists, then skip this Schedule as well.

### Values for Output

This part of the code is to get the relevant values for output. Each value may need to undergo some formatting or manipulating; hence, I have chosen to use variables to store them.

A screen shot of a computer program

Description automatically generated

* Line 237-239, 262, 267: Directly retrieving values from data sources.
* Line 242-243: Sort the lists of Session date and time incrementally for both normal and assessment Sessions, and combine them together. *i.e. Normal sessions first then Assessment Sessions.*
* Line 244, 260: As each value is now an item in a list, we will need to join the values into a single string for output. ““ \n”.join()” is to indicate that all the values will be starting a new line.

A white background with black text

Description automatically generated

Normal Session (M)

Assessment Session (A)

*An example of how the combined session date and time will look like.*

* Line 247-248: Sort the lists of Session venue incrementally for both normal and assessment Sessions, and combine them together.
* Line 253-257: For each venue, we need to find out the type of venue, *i.e. “Onsite”, “Offsite”, or “Online”*, for better visualisation of the data. This is done with the function “find\_venue\_type()”, which will be elaborated further in the latter. Additionally, each venue may be of short forms, *i.e. “SR”, “CR”, etc,* hence, we will need to convert the short forms to its original name. This is done with the function “change\_venue\_names()”, which will be elaborated further in the latter.
* Line 259: Grouping up of session venue by the day of session.

**Assumption made:** If there are multiple Session for this Schedule in a day, it will be the same venue.

* Line 261: Specifying the mode of delivery by checking if there is an online Session in the combined string of venue (combination is done earlier).
* Line 263-264: Format the datetime into “HHHH-MM-DD”.
* Line 265-266: Calculates the number of assessment and normal Sessions.
* Line 268: Check for registered pax (*i.e. not confirmed participants*).
* Line 269: Combine the total number of participants *(i.e. Enrolled Pax + Registered Pax)*

### Find Venue Type

This function is to find the type of venue by providing the actual venue name.

A screen shot of a computer code

Description automatically generated

* If a venue is empty (*in the code, it will be “-“ due to conversion)* or the Session is “Cancelled”, value will be “-“.
* If a venue is “Online”, then the value will be “Online”.
* If a venue consists of the school building names (read in from a file earlier), then it will be “Onsite”. Otherwise, “Offsite” will be the default value.

### Change Venue Names

As the name of the function suggests, it changes the venue name whereby the program searches the venue string, and then change if there is such phrase found inside the string.

A computer screen shot of a program

Description automatically generated

* “SR” is found, then it will be changed to “ Seminar Room”. (Space in front, otherwise words will stick together after conversion.
* “CR” is found, then it will be changed to “ Classroom”.
* “ SMU “ is found, then remove it because it is not needed.

### Adding Total Pax

This helper function is to add Enrolled Pax and Registered Pax together. It checks if registered pax or enrolled pax is empty, otherwise, it will add together.

A screen shot of a computer code

Description automatically generated

## Styling of Excel

This part of the code functionality is under the Main code. It executes after the extraction, manipulation, and formatting of the data has been done.

A computer screen shot of a program code

Description automatically generated

* Line 336: “book” is a Python Pandas’ (library) function that refers to the Excel sheet with the given name specified earlier at “writer” variable.
* Line 339 & 348: “sheets[<sheetname>]” is the specific spreadsheet you want to format.
* Line 340 & 349: Calls the “format\_cells()” function and pass in the relevant Excel sheets/tabs for formatting.

For reusability (reusing of code without copy-pasting it to do the same thing), a function called “format\_cells()” have been created to format the cells accordingly.

A screen shot of a computer program

Description automatically generated

* Line 291-293: Formatting for the cells with various attributes.
* Line 297-298: Format the first row (header) by bolding and enlarging the texts.
* Line 300-308: Customised attributes for each row. The function is specified as “**set\_column(<rows>, <row\_width>, <custom\_attributes>)**”.

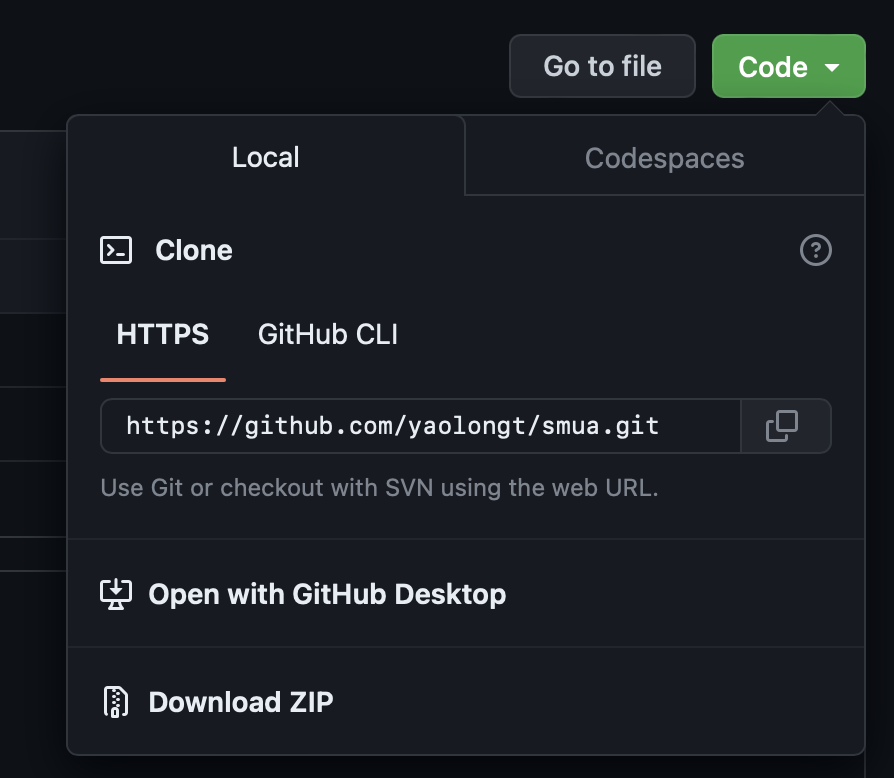
# Execution of the Code

Python needs to be installed into your device’s environment, and make sure that it is correctly configured.

You may double click or right click at the Python script and press “Run” to execute the script.

Note: Excel files must be placed in the same directory as the Python script with the name of “gvSession.xlsx”, “Manage Schedule.xlsx”, and “Enrolment Summary.xlsx”.

# Downloading of the Code

The code can be found online, <https://github.com/yaolongt/smua>. You may zip and download it by pressing “Code” with the drop-down box for the latest version. The code is in “data-collation” directory.

2

1

1. JSON, a.k.a. JavaScript Object Notation, is a file type that uses “key-value” pair. It is easy to read and fast. To understand more about JSON, click [here](https://www.w3schools.com/whatis/whatis_json.asp). [↑](#footnote-ref-1)
2. Dictionary is a type of data in Python. For more information about Python Dictionary, click [here](https://www.programiz.com/python-programming/dictionary). [↑](#footnote-ref-2)
3. DataFrame is type of data by a popular library by Pandas that is good for analyzing data, but it is not built-in Python data types. For more information about DataFrame, click [here](https://www.w3schools.com/python/pandas/pandas_dataframes.asp). [↑](#footnote-ref-3)