A blue and black logo

Description automatically generated

**Deep Comparison of information 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

As a sequel of [Information and Data Collation using Python with Excel](https://github.com/yaolongt/smua), Catering and Logs (C&L) team frequently generates new data as part of their workflow. They need an efficient way to visualise the differences between the data generated, making sure that the changes reflected will allow them to make the necessary changes.

## 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 2 exact Excel (not CSV) files with certain naming standardised (both with “CDL” at the start of the file name, and “.xlsx” at the end, which also means Excel file extension) present in the **parent** directory *(i.e. a folder before this current folder)* of the Python script. Otherwise, an error will be thrown.
* For easier visualisation and compilation, the program only allow compilation of 2 files.
* Both files must be of the same format, *i.e. both have the same number of columns and column names.*
* External libraries have been used to help with the comparison, hence, installation of external packages/libraries is needed. *(I have included it in “requirements.txt” and manual installation is NOT needed. Simply press the “run.bat” (Windows)/ “run.sh” (MacOS) to install the various libraries and run the code.)*

# 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 189-190: Check if there are exactly 2 files, otherwise, it will show the message.
* Line 192: Read data from the relevant Excel files.
* Line 194-197: Comparing the 2 files’ data in Dictionary. “!=” means “Not equal”, *i.e. first file is not equal second file*. This code will execute when there are changes between the files compared.

Note: Line 195 is to check the differences between the 2 files data. Line 196 is to structure the data for output.

* Line 198: This part of the code executes when both file data are the exact same.
* Line 200-202: Update the output data columns with relevant values. *i.e. Column “Changes From” will have “Not modified” in each row.*

## 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.

A screen shot of a computer

Description automatically generated

### Reading From Files

This function is to read data from the Excel files provided.

A screen shot of a computer code

Description automatically generated

* Line 37-39: Get the file name one by one, read in the data using “read\_excel()”. Use the specific column names to specify what we need with “usecols=headers”. “converters” is to change the data type of column “Total Pax” to be an integer.
* Line 41-42: For the 2 files in the list, we use another list to store the Dictionary type of the data *(i.e. “files\_dict”)*. The data was initially DataFrame type (*i.e. “files\_df”, which stands for files in DataFrame, is a name I gave for readability*), and we have converted it by using “to\_dict()” function.

Note: We have used the “Course No.” as the key for each column, so that we can easily retrieve value if we have the course number because each row is unique.

### Get Files from Parent Directory

This part of the code allows the program to identify the files needed for it to run the code. It dynamically finds for file name starting with “CDL” and ends with “.xlsx” (Excel file). Since the location of Excel files are in the parent directory, we would need to let the program know that.

A computer code with text

Description automatically generated

* Line 19: Get the exact path of this Python script.
* Line 20: Get the parent path of this Python script.
* Line 25: In the parent directory, search for files with “CDL” and “.xlsx”, and then add into a list. Sort the list accordingly because the order determines who is the latest file

## Check Differences Between Files

In this function, the program utilised an external library to aid in the comparison between the files. *(Thankfully :’))*

A computer screen shot of a program code

Description automatically generated

* Line 81-82: Using Dictionaries to store the modified row and new row so that we can format the data cells in the later part of the code.
* Line 84: Use the library to perform a deep comparison between values. “ignore\_order=True” means that the ordering of the data (*i.e. rows*) does not matter, it only compares between the values.
* Line 86-89: If there are new rows added, the course name will be reflected. Hence, the data will be added into “new\_row” dictionary.
* Line 91-101: If there are modified rows, add the corresponding old and new value to a list, using the key to identify. Inside the list, there will be 2 (string) values, where the first value is to store data for “Changed From” column, and second value is to store data for “Changed To” column.

## Structure Data

For easier visualisation of what has changed and what has added, a new file will be generated with the changes reflected accordingly. For the new file that will be generated, it will have 3 new columns, “Last Updated”, “Changes From”, “Changes To”. We will need to add new data for these columns, hence, it is done in this part of the code.

A screen shot of a computer program

Description automatically generated

* Line 108: The 2 parameters have a default value of empty Dictionaries, just in case if there are no values given as the input.
* Line 113: Loops through all the data in the newer version of the Excel file.
* Line 115: Using the headers’ name to add corresponding value in, one by one. *i.e. “Pillar”, “Course No.” will get its corresponding data from the Dictionary we have converted before.*
* Line 116-117: If the current header is “Course No.”, then the value we want is the key to the data, hence, it will be the key to the Dictionary[[1]](#footnote-1).
* Line 118-119: If the current header is NOT “Course No.”, then use corresponding values
* Line 120-125: This is to fill up the new columns with the relevant values.
* Line 126: Add into a list so that it can be prepared for output into Excel.

## Exporting Data to New File

This part of the code of “export\_to\_file()” function is to export the data, with the relevant changes reflected, to a new Excel sheet. The new Excel file name will be **“Combined\_CDL\_<first-file-name>-<second-file-name>.xlsx”.**

A screen shot of a computer program

Description automatically generated

* Line 135: Change the data from Dictionary type into Pandas DataFrame type.
* Line 136: Sort the values by the column of “Start Date” and “Course No.” incrementally.
* Line 139: Get the file names of the 2 Excel files.
* Line 142: Format the file name according to the 2 Excel files provided.
* Line 145: Prepare the directory to export with some configuration.
* Line 146: Export the data to Excel file using the configuration made earlier.

### Formatting the Data

This part of the code is to format the style as the same as the data source Excel files.

A screen shot of a computer program

Description automatically generated

### Colouring New Rows

As there are multiple issues with using the same library to format the data and having to colour the cells of the new rows, we sought to use another library that could help us keep the original formatting of the Excel sheet, while being able to colour the new rows.

A screen shot of a computer code

Description automatically generated

* Line 172: Open the exported Excel sheet using the path directory.
* Line 173: Use the correct sheet with “Sheet1”.
* Line 174: Configuration for the formatting of the cells.
* Line 175: Loops through the Excel sheet row by row
* Line 176: Get the value of the “Course No.”, which is the second column. (*i.e. 1 in Python because the counting starts from 0 instead of 1.*)
* Line 179-181: Check if the course number is a newly added record. If it is, then we will colour the row of it. Otherwise, the code will just ignore.
* Line 182: Save the changes to the output of Excel file.

## Update Sources Excel Files with New Timestamp

This part of the code is to update the sources Excel files (*i.e. the 2 Excel files that was provided in the parent directory*) with a new timestamp, indicating a comparison has been done at a certain time. The bottom part of the code is to style the Excel file as the same as before, keeping the original formatting without overriding them with the new values.

A screen shot of a computer program

Description automatically generated

A black text on a white background

Description automatically generated

*An example of how the “Last Updated” column will look like for each row.*

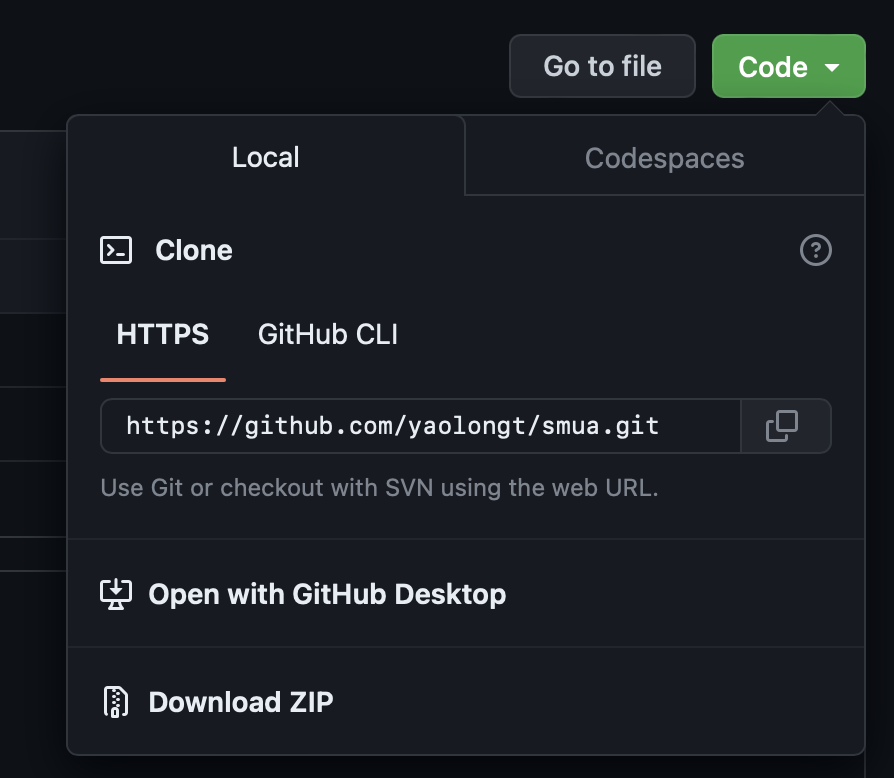
# Execution of the Code

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

You may double click “run.bat” (Windows)/ “run.sh” (Mac) to run the installation of the libraries and run the code sequentially. These files consists of another language called Bash, which is a lower level than Python.

Note: Excel files must be placed in the **parent** directory of the Python script with the names starting with “CDL” and ending with “.xlsx” (File extension of the Excel sheets).

# 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 “deep-comparison” directory.

2

1

1. To see how Python Dictionary works, please visit [here](https://www.programiz.com/python-programming/dictionary). [↑](#footnote-ref-1)