Email Management

created by Ethan Hunter – version 2020-10-28

https://github.com/instrangeland/EmailManagement

# Intro

This script offers tools for working with lists of emails and names in Excel files. To run the command, copy the files you wish to manipulate, the files “main.py” and “functs.py”, and the config.txt file into a folder (hereafter called your Workspace.). Double click the file main.py to execute the macro.

I use bold to highlight the instructions given to each command, and italics to highlight the actual commands. You should not use italics in the actual commands.

Contents

[Intro 1](#_heading=h.gjdgxs)

[Writing Macros 2](#_heading=h.30j0zll)

[Intro to “config.txt” 2](#_heading=h.1fob9te)

[The Commands 2](#_heading=h.3znysh7)

[new\_table 2](#_heading=h.2et92p0)

[delete 2](#_heading=h.tyjcwt)

[open 2](#_heading=h.3dy6vkm)

[save\_as 2](#_heading=h.1t3h5sf)

[copy\_over 2](#_heading=h.4d34og8)

[append\_to 2](#_heading=h.2s8eyo1)

[remove\_duplicates 2](#_heading=h.17dp8vu)

[keep\_only\_duplicates 3](#_heading=h.3rdcrjn)

[remove\_rows\_matching 3](#_heading=h.26in1rg)

[Examples 3](#_heading=h.lnxbz9)

[Removing Duplicates in a File 3](#_heading=h.35nkun2)

[Exist in Both Files (AND) 3](#_heading=h.1ksv4uv)

[Unique in Both Files (OR) 3](#_heading=h.44sinio)

[Appending Two Files (ADD) 3](#_heading=h.2jxsxqh)

[Subtracting Two Files (SUB) 4](#_heading=h.z337ya)

[Installing Python 4](#_heading=h.3j2qqm3)

# Writing Macros

|  |
| --- |
|  |
| Example of “config.txt” |

## Intro to the Macro Language

Macros are stored as text files. The macro language operates on opening excel files as tables, modifying them, and then saving them as new excel files. You must have Python installed. If you do not, see the “Installing Python” section.

To use the macros, type the commands you want the script to do in order, from top to bottom, a text file.

## The Commands

### new\_table

New table creates a new, empty, table in memory. The syntax is “*nameOfTable new\_table*”. For instance, the command “*allMyStuff new\_table*” would create a new table named **allMyStuff**.

### delete

The delete command has two usages. First, you can use it to delete a table in memory. The syntax for this is “*nameOfTable delete.*” For instance, to remove our old table “**allMyStuff**”, we would type “*allMyStuff delete*”.

It can also delete an excel file. This one is slightly more dangerous in terms of your data, so make sure to back up all excel sheets in your workspace using if you plan to make use of this command. The syntax is *“excelFileToDelete.xlsx delete*”. For instance, “*allMyData.xlsx delete*” would delete the excel spreadsheet titled “**allMyData.xlsx**”.

### open

The open command creates a new table in memory if one doesn’t already exist with the name you give it, and then opens an excel spreadsheet into it. The syntax is “*nameOfTable open nameOfSpreadsheet.xlsx”*. For instance, entering *“allMyStuff open allMyData.xlsx”*  
would create a new table called “**allMyStuff”** (if it doesn’t exist) and put the data from “**allMyData.xlsx**” into it.

### save\_as

The save\_as command saves a table into your workspace. The syntax is “*nameOfTableToSave save\_as excelSpreadsheet.xlsx”*. For instance, to save the table “**allMyStuff**” as **“newData.xlsx**”, you would type “*allMyStuff save\_as newData.xlsx”*.

### copy\_over

The copy\_over command copies the data from one table and pastes it over another table. The syntax for this is *“tableToCopyFrom copy\_over tableToCopyTo*”. For instance, to copy the data from “**allMyStuff**” to “**copyOfStuff**”, you’d type “*allMyStuff copy\_over copyOfStuff*”.

Only the table the data is copied from must exist, the table the data is pasted to will be created if it doesn’t.

### append\_to

This command copies data from one table and pastes it at the end of another. The syntax is *“fileToCopyFrom append\_to fileToAppendTo*”. For instance, to append “**allMyStuff**” to “**copyOfStuff**”, you’d type “*allMyStuff append\_to copyOfStuff*”. This command only modifies the table being pasted to.

### remove\_duplicates

This command takes a table and removes all the duplicates from it. The syntax is “*fileToRemoveDuplicates remove\_duplicates*”. For instance, to remove all duplicates from “**allMyStuff**”, you’d type “*allMyStuff remove\_duplicates*”.

### keep\_only\_duplicates

This command does the opposite of remove\_duplicates. Rather than removing any duplicate rows, it removes all rows that aren’t duplicated, and then keeps only one copy of each duplicate line. For instance, to keep only the duplicates, you’d type “*allMyStuff keep\_only\_duplicates*”.

### remove\_lines\_matching

This command takes two tables. It goes through the first table line by line, and if the second table has the same row, removes it from the first table. For example, to remove any rows that are in both “**allMyStuff”** and “**copyOfStuff**” from “**allMyStuff**”, you’d type “*allMyStuff remove\_rows\_matching copyOfStuff*”.

### copy\_column

This command takes a table, and two names of columns. It takes the second column, and overwrites it with data from the first. For example, to copy “**col1”** to “**col2**” from “**allMyStuff**”, you’d type “*allMyStuff copy\_column col1 to col2*”.

### note

This command tells the program to ignore the rest of the line. It is specifically to leave notes and comments for future programmers. Note that this does *not* apply during python mode operation.

### remove\_blank\_in

This command takes a table, and a column name. For each row in the table, if the column is empty, the row is removed. For instance, to remove all rows where **“col1”** is blank from **“allMyStuff”**, you’d type “*allMyStuff remove\_blank\_in col1*”.

### python\_begin

This command tells the interpreter to begin viewing all further lines as python code, until the command python\_end. Note that this command is not for the faint of heart! It requires an intimate knowledge of the interpreter internals and pandas. However, it can vastly expand the usage of this macro language.

### python\_end

This command tells the interpreter stop viewing all lines as python code: all future commands will be viewed as normal macro code.

### swap

This swaps two columns in memory. For instance, to swap “**col1”** and “**col2**” from “**allMyStuff**”, you’d type *“allMyStuff swap col1 and col2.”*

## Examples

|  |
| --- |
|  |
| Example of an email list |

### Removing Duplicates in a File

Assuming your list (single excel tabbed file) is similar to “testemails\_Duplicates.xlsx”, which has several sets of duplicate emails. To remove duplicates and save the duplicate free list to “testemails\_DuplicateFree.xlsx”, use the “remove\_duplicates” command as follows:

mainFile open testemails\_Duplicates.xlsx  
mainFile remove\_duplicates  
mainFile save\_as testemails\_DuplicateFree.xlsx

### Exist in Both Files (AND)

To create a list of emails that exist in two different lists, ignoring the emails that only exist in one list, you will use the following. First, open the two files. Take the second list, and *append\_to* the first list. Then, run *keep\_only\_duplicates* on the first file, and save it. Note that this also has the effect of removing any duplicates that already existed in the first list.

mainFile open emailsForTest.xlsx  
toAppend open testDuplicates.xlsx  
toAppend append\_to mainFile

mainFile keep\_only\_duplicates  
mainFile save\_as appendedFile.xlsx

### Keep Only Unique in Two Files, With Merging (XOR)

To merge the lists, and keep only unique emails, i.e. ignoring the emails that exist in both lists, you will use the following. First, open both files. Take the second list, and *append\_to* the first list. Then, run remove\_duplicates on the first list. Finally, save the list.

uniqueList1 open emailsForTest.xlsx  
uniqueList2 open testDuplicates.xlsx

uniqueList2 append\_to uniqueList1

uniqueList1 remove\_duplicates  
uniqueList1 save\_as uniqueList1.xlsx

### Keep Only Unique in Two Files, Without Merging (XOR)

To make the lists keep only emails unique to them, i.e. ignoring the emails that exist in both lists, you will use the following. First, open both files. Second, make a copy of the first list using *copy\_over*. Then, run *remove\_rows\_matching* on the first list, with the second list as an argument. Then, run *remove\_rows\_matching* on the second list, using the copy of the first list as an argument. Finally, save the two unique lists.

uniqueList1 open emailsForTest.xlsx  
uniqueList2 open testDuplicates.xlsx

uniqueList1 copy\_over file1Copy  
uniqueList1 remove\_rows\_matching uniqueList2

uniqueList2 remove\_rows\_matching file1Copy  
uniqueList1 save\_as uniqueList1.xlsx

uniqueList2 save\_as uniqueList2.xlsx

### Appending Two Files (ADD)

To append two files lists and remove the duplicates, assume you have two files: file1 and file2. Then two commands are listed in sequence, first the “toAppend” command.

mainFile open file1.xlsx  
toAppend open file2.xlsx  
toAppend append\_to mainFile  
mainFile save\_as appendedFile.xlsx

### Subtracting Two Files (SUB)

To remove all emails from one list that are also in a second list, assume you have two files, file1 and file2. Open them as lists. Take the first list, and *remove\_rows\_matching* the second list. Finally, you can save the first list.

mainFile open file1.xlsx  
duplicates open file2.xlsx  
mainFile remove\_lines\_matching duplicates  
mainFile save\_as appendedFile.xlsx

# Installing Python

Navigate to python.org, then downloads ([Download Python | Python.org](https://www.python.org/downloads/)). Download the latest version, for Windows x86-64. After opening the installer, make sure to click the checkbox reading “add Python 3.9 to PATH.” Then select install now.

After installation, there are a few final steps. Open the command prompt. First, type “pip install numpy==1.19.3” (without the quotes, and hit enter at the end). When it completes, type “pip install pandas.” Finally, type “pip install openpyxl.”