Documentation:

*\*Disclaimer\**

*The documentation was done in the time of coding. Some codes might have changed in the final version. Refer the link below for the final version of the code*

[*https://github.com/Nayan-Chimariya/excel-with-python/blob/master/main.py*](https://github.com/Nayan-Chimariya/excel-with-python/blob/master/main.py)

The project was done to get concept on how we can handle excel sheets using python. The program was initially written to transfer balance from one account to another from an excel sheet. Later some other features were added.

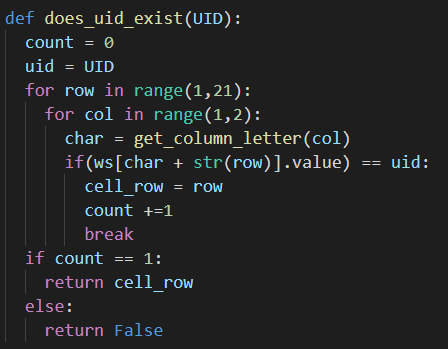
A library **openpyxl** was used to handle excel sheets

*The program is divided into two sections based on the user type. the user can be either admin or general user both have specific functions.*

**General User Section**

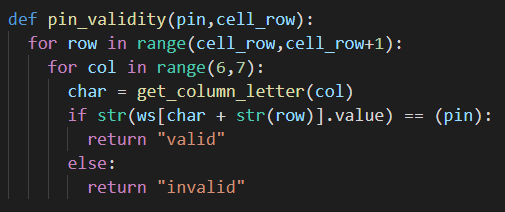
The main program starts by asking user for their user id or uid in short.

A function to check the existence of uid is created that return the cell row of the uid if present in the excel sheet else it returns false



If the returned value form the function **does\_uid\_exist(UID)** is not false, the programs ask for the user to enter their pin number.

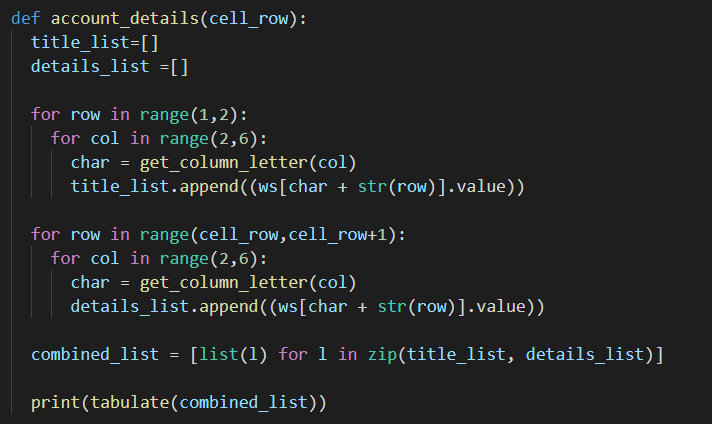
The check for the corresponding pin to the valid uid is done providing the **pin\_validity()** function with the argument of pin and cell row of the corresponding uid which was obtained from the **does\_uid\_exist(UID)** function.



The above function returns string values "valid" or "invalid" if the pin matches or does not match respectively.

If the user entered pin matches to the corresponding uid then the user is provided with his bank account information. This information is provided by the function

**account\_details(cell\_row)**

****

Here a function **tabulate()** is used that is imported from the library **tabulate**. This function takes a nested list as argument and prints the nested list in a tabulated form. Two empty list namely **title\_list[]** and **details\_list[]** are created that store the values of the excel file.

For example :

["Account Number", "User Name", "Account Type", "Balance"]

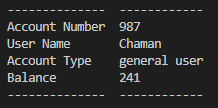
The above list can be a title list

["1234", "Nayan", "Admin", "203"]

The above list can be the details list

As said earlier, the **tabulated()** function takes a nested list. So, a function zip is used to nest **details\_list** within the **title\_list** and the new list is named **combined\_list**

The output of the tabulated function would be something like this:



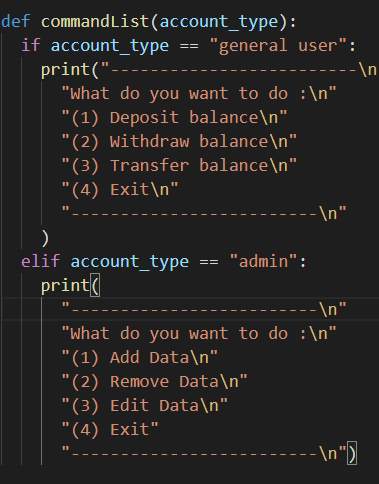
After the account details have been shown to the user, the program calls the function **available\_function()** this function takes 2 parameters. One is the account\_type and the other is cell\_row. The value for account\_type is returned by the **account\_details()** function.

\*\* *function was made to return value in the later section of programming* \*\*

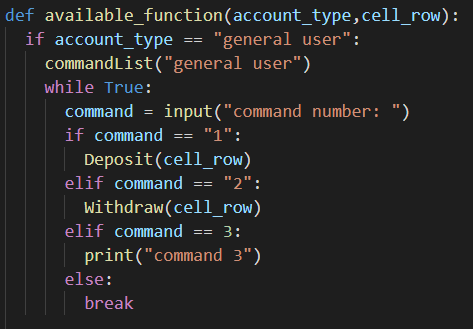
The user type and their available function is shown in the table below:

|  |  |
| --- | --- |
| Account Type | Functions available |
| 1. Admin | (1) Add Data  (2) Remove Data  (3) Edit Data |
| 2. General User | (1) Deposit balance  (2) Withdraw balance  (3) Transfer balance  (4) See Status |

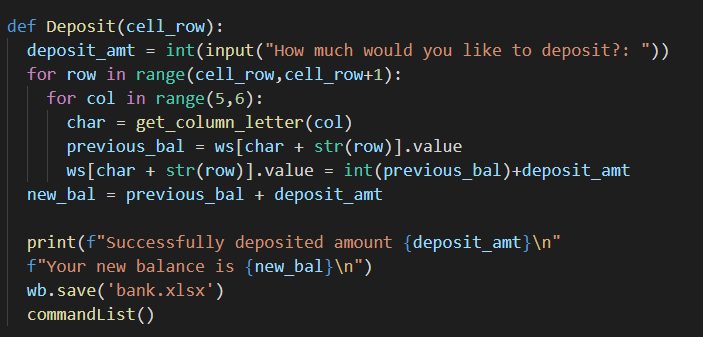
Command numbers and their meaning was frequently being repeated so they were kept in a function just to print.



The **available\_function()** calls functions based on the user's choice which is set to infinite while loop until user wishes to exit

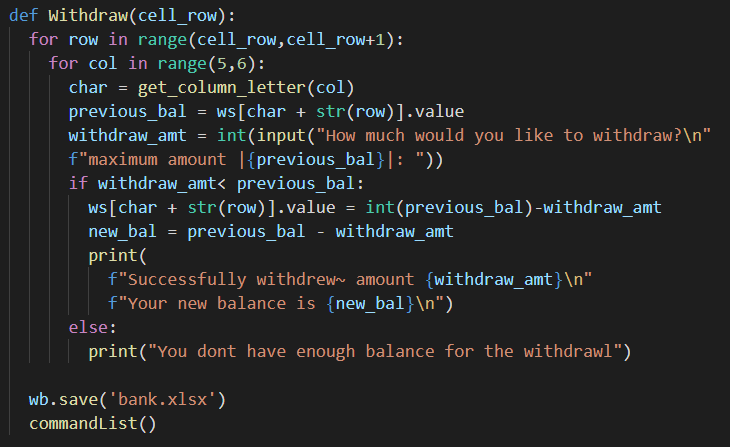


Providing "1" to the **command** variable invokes the deposit function



This function just updates the cell value that contains the balance of the client whose **cell\_row** was provided. The new value is the incremented value of previous value with respect to the deposit amount.

In similar way there is the withdraw function that is called by providing "2" to the **command** variable.



Here first the withdrawal amount is compared to the balance the user has. If the user has sufficient balance for the withdrawal request, then the request is acknowledged. Else a message is thrown saying that there is insufficient balance for the desired withdrawal.

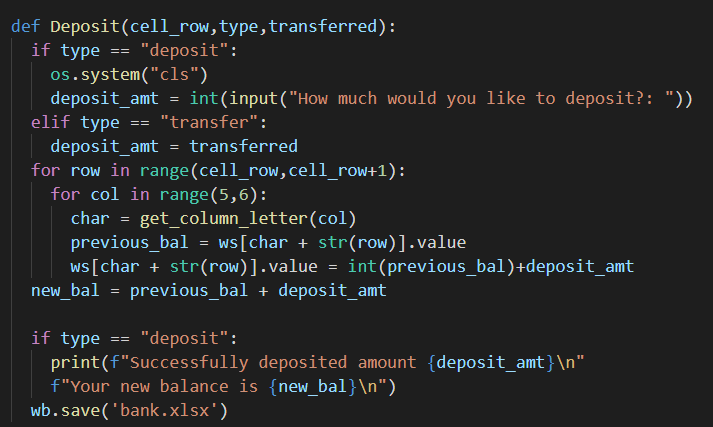
In a success withdrawal scenario, the new balance would be the value after subtracting the withdrawal amount from the old balance.

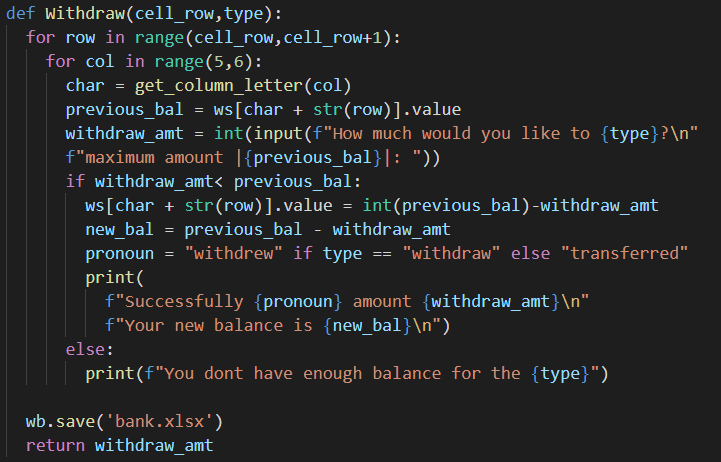
\*\**Side note: The changes in the excel sheet is visible only if the sheet is closed and reopened*\*\*

Command number "3" calls the **transfer()** functionwhich is the combined function of withdrawal and deposit function.

Some changes were made in the **withdraw()** and **deposit()** function so that it can be used to transfer balance as well as act as individual function. This was done by giving parameters and conditional statements within the function.

Modified functions are as follows

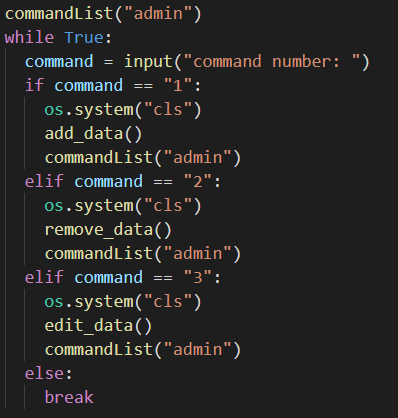




And finally, the command "4" calls the **account\_details()** . This command prints the user details in tabulated form.

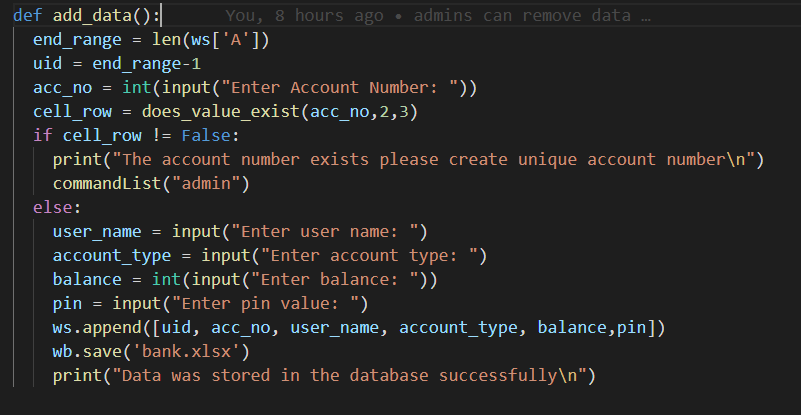
This completes the general user section.

**Admin User Section**

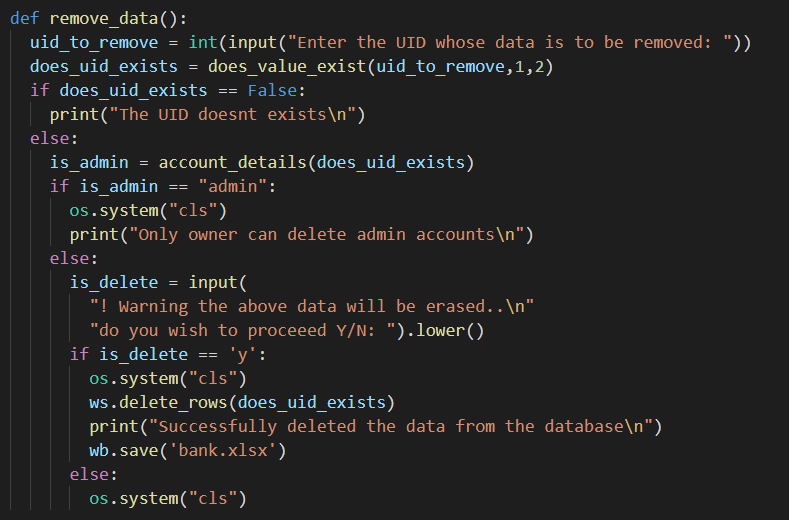
****

The admins have commands that allows them to add, modify or remove the data from the spreadsheet.

Calling the command "1" will call the function **add\_data()**.



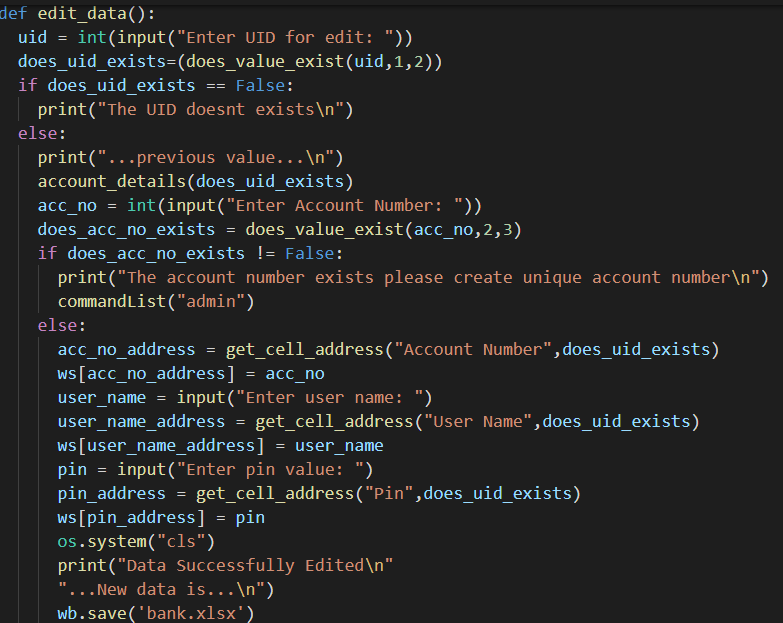
The **len()** function is provided with argument ws['A'] which is the column value of the active worksheet. This function returns the last empty row of the column and the value is stored in end\_range variable. Since the UID has to be unique and continuous in the sheet. Instead of admin entering the uid value. The program auto assigns the value as the 1 decrement from end\_range. Other necessary values are asked from the user and stored in their respective variables and later is appended to the work sheet.

Calling the command "2" will call the function **remove\_data()**.

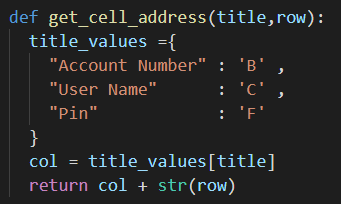
This function just asks for the uid of the user whose data is to be removed and checks if the uid exists or not. The process continues only if the uid exists. The account details is shown of the entered uid using the function **account\_details()**. This function prints the account details as well as return the user type. If the type is admin, then the data can't be erased. If it’s a general user's account then the admin is prompted if he wishes to proceed or not.

Proceeding will delete the general user's data from the worksheet.

Calling the command "3" will call the function **remove\_data()**.



The function asks admin to enter uid of the user whose data is to be modified. Existence for uid and account number is checked. The program continues if the uid exists and the new account number does not. To every new value asked, a function called **get\_cell\_address()** is used to get the cell address for the old value. This is done by providing the function with the parameter of title of the value and the row number.



A dictionary is used that gives the column value for provided title and the function finally return the concatenation of column and row to the variable the calls the function. Using this address, we can directly assign modified values to the old values in the worksheet.

This completes the admin user section.

*Documentation ends here, I may or may not have made the program modular. Got lazy XD*

*~* [*https://github.com/Nayan-Chimariya*](https://github.com/Nayan-Chimariya)