Quinterac - Back End

The back end of our Quinterac program is run from the terminal using the following command:

```
python -m back_end
```

With the following parameters:

1. Current master account text file
2. Name for merged transaction text file to be created
3. Name for new valid account list text file to be created
4. Name for new master account text file to be created
5. As many transaction summary text files as necessary

ie. python -m back_end master_account.txt merged_trans.txt new_valid_accounts.txt new_master_account.txt transactions_1.txt transactions_2.txt etc.

Implementation Overview

Our implementation is based around a single file run.py that contains the core functionality of the program but also uses specific groups of functionality from three files in the objects folder. The specific functions and attributes of the classes and files contained in the program are described in the tables below as well as how they deal with failures or invalid transactions.

run.py

Contains global reference to an instance of a MasterAccount object that holds information on the current state of all accounts. Instantiated above the main function

Method	Desc	Parameters	Output
main()	Gathers references that are passed from the terminal command Creates merged transaction file, iterates through transactions on that file Creates new valid account file and master account file	All txt files: master accounts, merged transaction file name, new valid accounts list file, new master account file, any transaction summaries to be processed	Merged transaction.txt New valid account list.txt New master account list.txt
handle_transactions()	Takes in transaction strings and send them to appropriate function to be handled	Transaction string defining the given transaction	Changes master account object Outputs error messages to terminal in regards to failed or invalid transactions
handle_new_account()	Adds valid accounts to master account object and produces error if account created already exists	to_account: account number to be created Name: name of account holder	Adds account to master account object Or Outputs failed attempt to duplicate account
handle_del_account()	Checks if account exists and is able to be deleted if so deletes from master account object	to_account: account number to be deleted name: name of account holder	Deletes account from master account object Or Error if delete transaction invalid
handle_transfer()	Checks if both accounts exist and have valid balances for transfer to work	to_account: account to receive transfer from_account: account to give transfer amount: amount to be transfered	Changes account balances in master account object Or Error if insufficient funds in giver account or too much funds in receivers account or if one or both accounts do not exist
handle_deposit()	Checks if account exists and if it does not have too large a balance to handle deposit	To_account: account to receive deposit amount: amount of deposit	Changes balance of given account in master account object Or Produces error if account does not exist or has too large a balance
handle_withdraw()	Checks if account exists and has enough money to handle withdraw	to_account: account withdrawn from amount: amount to be withdrawn	Changes balance of given account in master account object Or Produces error if account does not exist or has insufficient funds
generate_merge_transaction()	Creates merged transaction file and validates transactions in transaction summaries producing fatal error if format incorrect	Merged_file_name: name for the merged transaction file Transaction Files: list of references to transaction files to be processed	Merged transaction file containing concatenation of all transactions from summaries Or Fatal error if there is an incorrectly formatted transaction summary

field_validator.py

File containing utility functions for objects. Validation functions for all fields which require a validity check.

Class: SessionHandler

Method	Desc	Parameters	Output
number_is_valid()	Insures that the account number of a supplied account is valid. If account number is less than 9999999 and greater than 1000000 then it is valid.	Account_num = Account Number	True/False
amount_is_valid()	Insures that the amount or balance of the supplied account is valid. If the amount is greater than 0 and less than 1 million.	amount	True/False
name_is_valid()	Insures that the account name is no shorter then 3 letters and no longer then 30 (including spaces).	account_name = Account Name	True/False
transaction_is_valid()	Insures that the transaction is valid by utilizing all the other functions mentioned above.	transaction = transaction summary line.	True/False

Master_accounts.py

Object which maintains the dictionary of valid accounts generated from the master accounts file. Supports the creation and deletion of accounts from the master account dictionary and produces the valid_account.txt file and the new master_accounts.txt file.

Class: SessionHandler

Method	Desc	Parameters	Output
__init__()	On instantiation declares a new dictionary which is equal to that which is returned by the from_file() function.	DIrectory of the master accounts	NA
from_file()	Opens master account file and reads in all accounts from file saving them in a dictionary with a key of the account number and value of an account object.	Directory of the master accounts file.	Returns the master_account dictionary.
add()	.Adds a new account object to the master account dictionary.	New accounts number, new accounts name	NA
remove()	Removes an account from the master account dictionary..	Account number to be removed.	NA
create_valid_account_list ()	Creates a new valid account list file	Directory of valid account list.	valid_account_list.txt
create_new_master_account_file()	Creates a new master_account file.	DIrectory of current master_account file.	master_account.txt

account.py

File containing account object that holds information regarding the accounts that are held in the master account object's dictionary of accounts

Attribute/Method	Desc	Parameters	Output
Number (str)	Account number for given account	NA	NA
Name (str)	Name of account holder		
Balance (int)	Balance of account		
Valid (bool)	Indicates whether the account has been involved in any invalid transactions. If so indicated as invalid and subsequently not added to valid account list		
setInvalid()	Sets the account valid to False	None	Accounts valid property to false