Done by Chinese-broccoli (Mason Chang, Yueyang Liu & Ruikang Luo)

Daily.py

import frontEnd as fn

import backEnd as bn

import sys

import os

class daily:

    def \_\_init\_\_(self, day):

        self.day = "day" + str(day)                                             # processing day corresponding to a directory

        self.rootDir = os.path.join(os.path.dirname(\_\_file\_\_), self.day)        # all files read and written within the directory

        self.merge\_file\_from = os.path.join(self.rootDir, "frontend\_out")       # merge files from frontend output

        self.merge\_file\_to = os.path.join(self.rootDir, "backend\_in")           # merged file as backend input

        name = "mergedSummaryFile.txt"                                          # the merged file name

        self.open\_frontend()

        self.write\_file(name)

        self.pass\_backend(name)

    # run the fronted end

    def open\_frontend(self):

        write\_to = os.path.join(self.merge\_file\_from, "transaction.txt")        # frontend ouput file write to

        trans = os.path.join(self.rootDir, "frontend\_in/trans.txt")             # user input come from the file

        account\_list = os.path.join(self.rootDir, "frontend\_in/account.txt")    # same day, the same account list

        for i in range(1,4):                                                    # 1, 2, 3 sessions

            write\_to\_temp = os.path.splitext(write\_to)[0] + str(i) + ".txt"

            trans\_temp = os.path.splitext(trans)[0] + str(i) + ".txt"

            sys.argv = ["frontEnd.py", account\_list, write\_to\_temp]

            f= open(trans\_temp, "r")

            sys.stdin = f

            fn.main()

            f.close()

    # pass backend

    def pass\_backend(self, name):

        backend\_inputDir = self.merge\_file\_to

        masterAcc = os.path.join(backend\_inputDir, "masterAccount.txt")

        mergeTrans = os.path.join(backend\_inputDir, name)

        sys.argv = ['backEnd.py', mergeTrans, masterAcc]

        bn.main()

    # write the merged file

    def write\_file(self, name):

        mergered\_file = self.read\_merge\_file(self.merge\_file\_from)

        targetFile = os.path.join(self.merge\_file\_to, name)

        if not os.path.exists(self.merge\_file\_to):

            os.makedirs(self.merge\_file\_to)

        with open(targetFile, "w") as f:

            f.writelines(mergered\_file)

    # Read files and merge them

    def read\_merge\_file(self, targetDir):

        flist = []

        for filename in os.listdir(targetDir):

            if filename.endswith('.txt'):

                with open(os.path.join(targetDir,filename)) as f:

                    flist += f.readlines()[:-1]

        flist.append("EOS")

        return flist

def main():

    daily(1)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Weekly.py

import Daily as dy

import os

class Weekly:

    def \_\_init\_\_(self):

        for i in range(1,6):

            print('Welcome to DAY ' + str(i))

            dy.daily(i)

            if i != 5:

                self.updateFile(i)

                print('\n' \* 5)

    # prepare next day's account list and master account

    def updateFile(self, d):

        f1 = open("day"+ str(d) +"/backend\_out/masterAccount.txt", 'r')     # today's back end output is next

        f2 = open("day"+ str(d+1) + "/backend\_in/masterAccount.txt", 'w')   # day back end input

        m = f1.readlines()

        for item in m:

            f2.write(item)

        f1.close()

        f2.close()

        f1 = open("day"+ str(d) +"/backend\_out/accountList.txt", 'r')

        f2 = open("day"+ str(d+1) + "/frontend\_in/account.txt", 'w')

        m = f1.readlines()

        for item in m:

            f2.write(item)

        f1.close()

        f2.close()

def main():

    Weekly()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**printout of each front end:**

|  |  |  |
| --- | --- | --- |
|  | Transaction input | Transaction output |
| Front end #1 | login  agent  createacct  1000327  newAcc01  createacct  1000330  newAcc04  logout | NEW 1000327 000 0000000 newAcc01  NEW 1000330 000 0000000 newAcc04  EOS |
| Front end #2 | login  agent  createacct  1000328  newAcc02  createacct  1000332  newAcc06  logout | NEW 1000328 000 0000000 newAcc02  NEW 1000332 000 0000000 newAcc06  EOS |
| Front end #3 | login  agent  createacct  1000329  newAcc03  createacct  1000331  newAcc05  logout | NEW 1000329 000 0000000 newAcc03  NEW 1000331 000 0000000 newAcc05  EOS |

**Merged transaction:**

NEW 1000327 000 0000000 newAcc01

NEW 1000330 000 0000000 newAcc04

NEW 1000328 000 0000000 newAcc02

NEW 1000332 000 0000000 newAcc06

NEW 1000329 000 0000000 newAcc03

NEW 1000331 000 0000000 newAcc05

EOS

**Printout of master account:**

|  |  |  |
| --- | --- | --- |
| DAY | in | out |
| Day1 | Empty | 1000332 000 newAcc06  1000331 000 newAcc05  1000330 000 newAcc04  1000329 000 newAcc03  1000328 000 newAcc02  1000327 000 newAcc01 |
| Day2 | 1000332 000 newAcc06  1000331 000 newAcc05  1000330 000 newAcc04  1000329 000 newAcc03  1000328 000 newAcc02  1000327 000 newAcc01 | 1000332 000 newAcc06  1000331 50000 newAcc05  1000330 50000 newAcc04  1000329 50000 newAcc03  1000328 50000 newAcc02  1000327 50000 newAcc01 |
| Day3 | 1000332 000 newAcc06  1000331 50000 newAcc05  1000330 50000 newAcc04  1000329 50000 newAcc03  1000328 50000 newAcc02  1000327 50000 newAcc01 | 1000331 35000 newAcc05  1000330 650000 newAcc04  1000329 50000 newAcc03  1000328 75000 newAcc02  1000327 25000 newAcc01  1000326 000 newAcc07 |
| Day4 | 1000331 35000 newAcc05  1000330 650000 newAcc04  1000329 50000 newAcc03  1000328 75000 newAcc02  1000327 25000 newAcc01  1000326 000 newAcc07 | 1000338 000 newAcc038  1000331 585000 newAcc05  1000330 100000 newAcc04  1000329 50000 newAcc03  1000328 65000 newAcc02  1000327 5025000 newAcc01  1000326 94999999 newAcc07 |
| Day5 | 1000338 000 newAcc038  1000331 585000 newAcc05  1000330 100000 newAcc04  1000329 50000 newAcc03  1000328 65000 newAcc02  1000327 5025000 newAcc01  1000326 94999999 newAcc07 | 1000338 4999999 newAcc038  1000331 775000 newAcc05  1000329 150000 newAcc03  1000328 65000 newAcc02  1000327 5000000 newAcc01  1000326 90000000 newAcc07 |

|  |  |  |  |
| --- | --- | --- | --- |
| Defect | Cause | Fixed | Comment |
| Can not find exist account in account list | In atm.py | | The order of account list was not matching. Now change it to descending order |
| Write file failed when log out | In atm.py | | Check the writing directory if exist. If not create the directory first |
| Writing directory was not expected | In backend.py | | Change the backend output to “backend\_out”. More logically make senses |
| Output masterlist had wrong format | In backend.py | | Add a new line character to the update info |
| Program would output “2” when int(“001”) +int(“001”) | def convert\_to\_cent(self, num):          amount\_in\_cent = str(num)          if (num < 10):              amount\_in\_cent = "00" + amount\_in\_cent          elif (num < 100):              amount\_in\_cent = "0" + amount\_in\_cent          return amount\_in\_cent | | Add “0” to the result if it necessary. |
| Failed to deposit more than $2,000 in agent mode | In agent.py | | Overwrite the verifyDepositAmount function  (Omission when design test case) |
| Failed to withdraw more than $1,000 in agent mode | In agent.py | | Overwrite the verifyWithdrawAmount function  (Omission when design test case) |
| Failed to transfer more than $10,000 in agent mode | In agent.py | | Overwrite the verifyTransferAmount function in  (Omission when design test case) |