```python
#IMPORTS
import datetime


#GLOBAL VARIABLES
current_id = -1;
failed_attempts = 0;

#SUB PROCESSES
#
#
def userdetails():
    file = eval(open("userdetails.txt","r").read()) # Opens file containing all user data, converts string to list
    return file; # returns the file
#
#

#
#
def find_account(username): # Searches for the account in array and returns its position.
    found = -1;
    for(logindetails)in(range(0,len(userdetails()))): # Goes through the files array calling the function "userdetails()" and recieving an array
        try:
            if(userdetails()[logindetails][0]==username): # Checks if the arrays first value is equal to the username
                found = logindetails;break; # Labels found as anything 0+, found is changed from -1 to the index of the accounts nested array
        except:print();
    return(found); # Returns the value of found, if equal to -1 then no account was found
#
#


#
#
def login(username,password): # Attempts to login to an account finding the username first and matching the password in the array.
    found = find_account(username); # found is defined from the return of the function "find_account"
    if((not found>-1)or(userdetails()[found][1]!=password)):
        print("ERROR! Incorrect username/or pin...");
        return -1; # If the account details are different to the account it has found, then it will return -1 as no account was found
    print("Logging into account '"+username+"'...");return found; # Shows logging in message, returns the variable found which was defined earlier
#
#


#
#
```

```python
def check_balance(id): # Checks the balance of an account via its position in the main array.
    try: # Try, except incase the id is incorrect
        print("Current balance: £"+"{:,}".format(round(userdetails()[id][2],2))); # Uses .format() to place commas inbetween thousands
    except:
        print("ERROR! Invalid account.") # The only error that could occur is that the file is corrupted or most commonly the id presented is incorrect
#
#


#
#
def deposit_cash(id): # Adds a value onto the previous balance as long as the value is above 0.
    amount = float(input("Enter amount to deposit: ")); # Asks the user how much they would like to deposit
    old = userdetails(); # Defines the variable "old" as a clone of the "userdetails.txt" array
    if(amount<0.01):print("ERROR! Invalid deposit amount.");return;
    old[id][3].append(["ATM","ATM",amount,"deposit"]); # Displays transaction infromation on the users nested array "userdetails.txt"
    old[id][2] += amount;open("userdetails.txt","w").write(str(old)); # Rewrites the the file with the modified array
#
#


#
#
def withdraw_cash(id): # Removes a value from the previous balance as long as the value is above 0 and below the total balance.
    amount = float(input("Enter amount to withdraw: ")); # Asks the user how much they would like to withdraw
    old = userdetails(); # Defines the variable "old" as a clone of the "userdetails.txt" array
    if((amount<0.01)or(amount>old[id][2])):print("ERROR! Invalid withdraw amount.");return; # Returns nothing, quickly exits the function, if the amount of cash is above the account holders balance or below 0
    old[id][3].append(["ATM","ATM",-amount,"withdraw"]); # Displays transaction infromation on the users nested array "userdetails.txt"
    old[id][2] -= amount;open("userdetails.txt","w").write(str(old)); # Rewrites the the file with the modified array
#
#


#
#
def logout(): # Resets the global id variable telling the code it has logged out of the account.
    global current_id;
    current_id = -1; # Sets current_id globally to -1, meaning the while loop in the MAIN CODE section will think it has not been logged into or has
```

```python
already logged out
    print("Successfully logged out of your account.");
#
#


#
#
def logged_in(): # Checks whether the account is logged in by checking if the id value is above -1.
    return(current_id!=-1) # Returns a TRUE or FALSE boolean value depending on whether the current_id variable is set to -1 or not (-1 meaning it has been logged out, anythign above means it has been logged into)
#
#


#
#
def transfer_to(): # Opens a menu to transfer cash from current account to the next account via inputs.s
    account_name = input("Enter the name of the account you would like to send money to:\n"); # Asks the user which account they would like to send their money to
    new_acc = find_account(account_name); # Pases the given name through the find_account() function, if -1 is returned no account has been found otherwise it will return a number 0+
    if((new_acc==-1)or(new_acc==current_id)):print("ERROR! Invalid account name.\n");return; # Returns the function if new_acc is equal to -1 (Meaning no account has been found)
    amount_of_cash = float(input("How much money would you like to send?\n")); # Asks the user how much cash they would like to transfer to said account via a float datatype
    if((amount_of_cash<0.01)or(amount_of_cash>userdetails()[current_id][2])):print("ERROR! Invalid amount of money.\n");return;
    message_to_account = input("Enter payment reason for payment, (payment message):\n");
    print("Payment details:\n     £"+str(amount_of_cash)+"\n     recipient '"+account_name+"'\n     message: "+message_to_account);
    if({'y':True}.get(input("Enter 'y' to confirm payment."))):
        old = userdetails(); # Defines "old" as a clone of the array
        old[current_id][2]-=amount_of_cash; # Removes the cash from the sender account
        old[new_acc][2]+=amount_of_cash; # Adds the cash the the receving account
        old[current_id][3].append([account_name,message_to_account,-amount_of_cash,"to"]); # Saves the transaction infromation to the nested array for the receiving end for the payment
        old[new_acc][3].append([old[current_id][0],message_to_account,amount_of_cash,"from"]); # Saves the transaction infromation to the nested array for the account thats paying for the payment
        open("userdetails.txt","w").write(str(old)); # Rewrites the the file with the modified array
    else:
        print("Payment cancelled..\n");return; # Checks whether the user will
```

```
still want to send the money
#
#


#
#
def change_pin(id,new): # Confirms the pin from second variable and changes it according to the given id.
    if(input("Confirm new pin: ")!=new):print("ERROR! Pin does not match.\n\n");return; # Confirms the new pin, if its incorrect an ERROR statement will be pased an the function will be returned
    old = userdetails(); # Creates a copy of the userdetails.txt file
    old[id][1] = new; # Modifies the copy of userdetails.txt
    open("userdetails.txt","w").write(str(old)); # Rewrites the userdetails.txt file with its cloned counterpart
    print("Successfully changed pin!\n");
#
#


#
#
def view_transaction_history(id): # Goes through item 3 of each users array to develop a transaction menu.
    if(id=="ERROR!"):print("ERROR! Invalid username.");return; # If the account entered is invalid it will send the id through as "ERROR!"
    print("----------------------------------------| TRANSACTIONS |----------------------------------------");
    for(item)in(userdetails()[id][3]):
        print("     "+item[3]+": "+str(item[0])+", message: "+str(item[1])+", amount: "+str(item[2]));
      # Loops around showing each transaction, saved from the file
    print("------------------------------------------------------------------------------------------------\n\n");
#
#


#
#
def show_menu(): # Prints out a 2d interactable interface to the user.
    print("\n\n") # Creates a gap between previous print statements to spread out the command line
    print("----------------|",userdetails()[current_id][0].upper(),"|----------------");
    print("Current time   :",datetime.datetime.now().replace(microsecond=0)); # Displays the date and time, uses microsecond set to 0 stopping a mass amount of decimals after the current second
    check_balance(current_id); # Sends the ID through the check_blance procedure that prints out "Current balance:"etc..
    print("What would you like to do?\n      Enter:")
    print("         - 'd' to Deposit cash");
    print("         - 'w' to Withdraw cash");
    print("         - 's' to send cash to another account");
```

```python
    print("        - 't' to view transaction history");
    print("        - 'p' to change pin");
    print("        - 'l' to logout of account");
    admin = False;
    try:
        admin = userdetails()[current_id][4]; # Checks whether the userdetails array whether the account is admin or not, this is sent through try/except due to some accounts not having this option
        print("ADMIN MENU");
        print("        - 'at' to view another accounts transactions.");
        print("        - 'ab' to check another accounts balance.");
    except:admin=admin; # Sets admin to admin and leaves the except statement empty with minimal clutter
    while(True):
        id = {"d":1,"w":2,"l":3,"s":4,"t":5,"p":6,"at":7,"ab":8}.get(input()) or "ERROR! Invalid input.";#Asks the user to input a series of letters corresponding to their task, these letters are matched in a dictionary and converted to numbers 1-8
        #The 'id' is then sent through a series of questions allowing it to send the user to their destination
        if(id==1):deposit_cash(current_id);break;#SEND TO DEPOSIT MENU       deposit_cash(id):
        elif(id==2):withdraw_cash(current_id);break;#SEND TO WITHDRAW MENU       withdraw_cash(id):
        elif(id==3):return("log");#LOGOUT OF ACCOUNT              logout():
        elif(id==4):transfer_to();break;#SEND TO TRANSFER MENU         transfer_to():
        elif(id==5):view_transaction_history(current_id);break;#SEND TO TRANSACTION HISTORY    view_transaction_history(id):
        elif(id==6):change_pin(current_id,input("Enter new pin: "));break;#SEND TO PIN CHANGE             change_pin(id,new):
        if(admin): # Checks whether the try/except statement went through and set admin to True allowing for this statement to pass, if not it will display an error message as no conditions above have been filled
            if(id==7):view_transaction_history(find_account(input("Enter the name of the account you would like to view: ")) or"ERROR!");break;#SEND TO TRANSACTION HISTORY OF GIVEN ACCOUNT.
            elif(id==8):check_balance(find_account(input("Enter the name of the account you would like to check the balance of: ")) or"ERROR!");break;#SEND TO SHOW BALANCE FUNCTION WITH DIFFERENT ID.
        print("ERROR! Invalid input.") # Each line breaks the loop stopping this print statement from appearing unless all the conditions are false and they have inputted an incorrect letter
#
#


#MAIN CODE
while(True): # Loops the menu option, also allows for the user to exit the program.
    if(failed_attempts==3):
        print("You have entered an incorrect username/or pin 3 times, quitting program."); # Displays quit message after first layer of security is alarmed
    if(logged_in()): # Checks whether 'current_id' is equal to -1 which means it is not logged in, calls the function logged_in()
        if(show_menu()=="log"):logout();
        continue; # Continues the loop restarting it and stopping the exit statement from appearing
    else:
        current_id = login(input("Enter username: "),input("Enter pin: ")); # Asks the user for login details and sends them into the login() function
        if(current_id!=-1):
            failed_attempts = 0; # Resets the variable failed_attempts back to 0 to allow for the password security to be reset
            continue;
        failed_attempts+=1;
        if(failed_attempts!=3): # Makes sure the code does not say "0 more times" as the console will look terrible
            print("You have entered an incorrect username/ or pin", ["once","twice"][failed_attempts-1]+",",3-failed_attempts,"more",["times","time"][failed_attempts-1],"and the program will quit.");
    if({"y":True}.get(input("Enter 'y' to exit or nothing to login."))):break; # Goes through an if statement that matches the input through a dictionary, if the input is equal to 'y' then it will return True allowing the if statement to pass and the loop to break
```