CLASS	METHODS	DESCRIPTION
Accounts Class contains methods concerning gathering information about accounts, as well as checking if account parameters are legitimate.		
	accNumDecimalOnly(accNum)	Check to make sure parameter accNum contains decimal numbers only. Used to confirm that account numbers and amounts only contain decimal characters.
	accNumValid(accNum, error)	Check to make sure parameter accNum is a valid account number (i.e. length must be 7, can't begin with 0, and contains decimal characters).
	accNumExists(accNum, accountList)	Check to see if paramater accNum corresponds to an account number that already exists by checking the ACCOUNTS_FILE (vaf) that is inputted into the program.
	getAccBalance(accNum,transactionType, backendDict)	Proposed back-end method to see the balance in the account corresponding to parameter accNum. Not currently used, but required to check to make sure transactions do not exceed limits.
	updateDailyAmount(accNum, amount, transactionType, backendDict)	Update amount in account after a transaction, as well as updating the daily amounts
	accNameAlphaNum(string)	Check to make sure parameter string contains alphanumeric characters only, used to confirm validity of an account name.
	accNameValid(accName, error)	Check to make sure parameter accName is valid and meets all restrictions (i.e. 3>length>30, starts and ends with acceptable characters, and contains only alphanumeric characters).
Error Class contains methods to print error messages		
	invalidInputA()	Prints error message: 'Invalid Input'
	invalidInputB(inputAction)	Prints error message: 'Invalid Input' followed by the parameter inputAction
	errorMsg(error)	Used to print error messages: prints whatever parameter is passed (error)
Actions Handles actions specified by the user, i.e. login, logout, create/delete account.		
	handleKeyboardInput()	Allows user to input their desired action, save input and pass to actionHandler to proceed.
	login()	Called when user enters login at an acceptable time: changes global variable 'status' to 'login' and loads the inputted accounts file (vaf).
	logout()	Called when user enters logout at an acceptable time: changes global variable 'status' to 'logout' writes action to the TSF file.
	setatm()	Sets global variable 'status' to ATM.
	setagent()	Sets global variable 'status' to AGENT.
	createAccount(action)	input command is valid (contains createacc, number, name), checks to make sure the account number and name are valid, that the number doesn't already exist. If it meets all these requirements, write to VAF file as well as save command to the TSF file.
	deleteAccount(action)	Deletes an existing account when user enters a valid deleteacct command. Checks to make sure the input command is valid (contains deleteacc, number, name), checks to make sure the account number exists. If it meets all these requirements, write to VAF file as well as save command to the TSF file.
	deposit(action)	the input command is valid (contains deposit, accNumber, amount), checks to make sure the account number exists and is valid, that the amount is valid and doesn't exceed any limits. If it meets all these
	withdraw(action)	sure the input command is valid (contains withdraw, accNumber, amount), checks to make sure the account number exists and is valid, that the amount is valid and doesn't exceed any limits. If it meets all
	transfer(action)	make sure the input command is valid (contains transfer, accNumberFrom, accNumberTo, amount), checks to make sure that both account numbers exist, are valid, and are not the same, that the amount is
	actionHandler(action)	Checks to see if inputted command is valid: (ie. can't login when already logged in), and then calls the appropriate method (createAccount, deleteAccount etc.)