

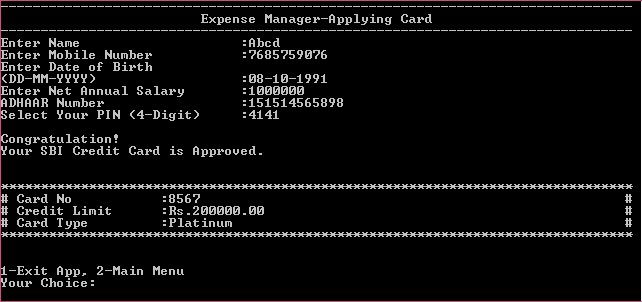
Solution:

Software Requirement Specifications

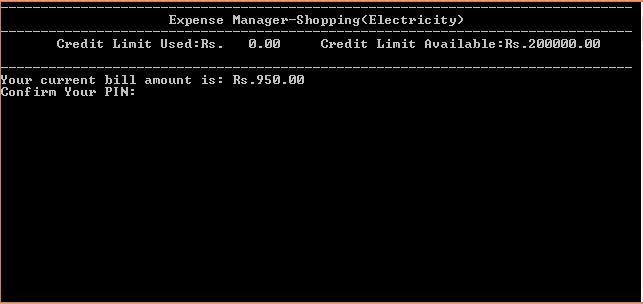
Expense Manager

**Steps of Execution of Code File:**

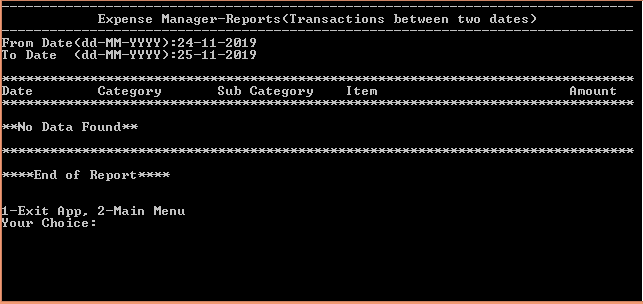
1. **1** Run the mainApp.py file with either command prompt or using Python Idle.
2. Before you ahead with Shopping or to view your transaction, you need to have a registered credit card or you must apply for a credit card.
3. Apply for a credit card:
   1. Enter Name: \*\*\*\*\*\*\*\*
   2. Enter Mobile Number: \*\*\*\*\*\*\*\*\*\* (10-Digits)
   3. Enter Date of Birth: \*\*-\*\*-\*\*\*\* (DD-MM-YYYY)
   4. Enter Net Annual Salary: \*\*\*\*\*\*\*
   5. Enter ADHAAR Number: \*\*\*\*\*\*\*\*\*\*\*\*(12-Digit)
   6. Select Your PIN(4-Digit): \*\*\*\*



1. Go back to main menu by giving your choice as 2 or exit the app by giving choice as 1.
2. Once you go back to main menu, you can go ahead with 2nd option as Shopping or you can view your transaction using the 3rd option.
3. Let’s do some shopping
   1. Enter your card number: \*\*\*\*(4-Digit)
   2. Enter PIN
   3. Once validation of card and pin is done, go ahead and select an option from the 4 categories defined:
      1. Bills
      2. Vehicle
      3. Entertainment
      4. Shopping
   4. Let’s select 1 i.e. Bills
   5. Select an option from the given 4 Sub-Categories:
      1. Electricity
      2. Mobile
      3. Internet
      4. Water
   6. Let’s select Electricity .i.e. 11
   7. Your current bill amount will be shown and you need to confirm your credit card PIN to pay the bill, once pin is entered, bill will be paid automatically



1. Let’s check our transactions
   1. Go to main menu and enter option 3
   2. Enter your card number: \*\*\*\*
   3. Enter your PIN: \*\*\*\*
   4. Select a transaction type from the given list to view:
      1. 1-All transactions done between two dates
      2. 2-All transactions that fall within a specified amount range
      3. 3-All transactions done on a category
      4. 4-Total amount spent on a category
      5. 5-Transactions stored based on amount
   5. Let’s select the first option
   6. Enter From Date(DD-MM-YYYY):\*\*\*\*
   7. Enter To Date(DD-MM-YYYY):\*\*\*\*



*The frontend will feed the* ***registration information*** *into the* ***backend*** *where the registration information is stored and also can extract the data from the backend as per the user request after validating a user.*

***The following are the class diagram for the code used:***

* itemId
* category
* subCategory
* itemName
* price
* itemType
* priceRange
* msg
* \_\_init\_\_(itemId, category, subCategory, itemName, price, itemType, priceRange, msg)

**ItemData**

* catId
* category
* \_\_init\_\_(catId, category)

**CategoryData**

* subCatId
* subCategory
* catId
* \_\_init\_\_(subCatId, subCategory, catId)

**SubCategoryData**

**TranData**

* tranId
* cardId
* cardNo
* cardHolderName
* tranDate
* itemId
* price
* tranItem
* isTranItem
* \_\_init\_\_(tranId, cardId, cardNo, cardHolderName, tranDate, itemId, price)
* Category()
* SubCategory()
* ItemName()
* subCatId
* subCategory
* catId
* \_\_init\_\_(subCatId, subCategory, catId)

**SubCategoryData**

* dbItem
* dbCat
* dbSubCat
* dbCard
* dbTran
* IsLoggedIn
* logInCard
* \_\_init\_\_()
* loadItem(IsMsg=True)
* loadCard (IsMsg=True)
* loadTran (valueCard, IsMsg=True)
* loadTranByItem (valueCard, IsMsg=True)
* loadItemCard (IsMsg=True)

**dBase**

**Class Diagram -1**

**Data Objects of Data Base**

**CardData**

* cardId
* name
* mobileNo
* dob
* netSalary
* cardNo
* cardType
* creditLimit
* creditUsed
* passCode
* adharNo
* rewardPoint
* rewardUsed
* \_\_init\_\_(name, mobileNo, dob, netSalary, adharNo)
* creditBal()
* clear()
* dateToStr(valueDate,formatStr = '%d-%m-%Y')
* strToDate(strDate,formatStr = '%d-%m-%Y %H:%M:%S')
* now()
* dateAddDays(valueDate,valueDays)
* age(dob)
* isValidDate(strDate,formatStr='%d-%m-%Y')
* isValidDigit(strNo,isExactLen=True,noOfDigits=10)
* numberInput(prompt,isExactLen=False,noOfDigits=5)
* sleep(sec)
* randNo(startNo, endNo, stepValue, isExcluded=False)
* parseItemMsg(valueItem,passCode,creditBal)
* mainMenu()
* startApp()
* Login(fnToCall,tranLoader,pageTitle='Login')
* centerText(text)
* linePrint(text='-',spaces=79)
* title(screenName)
* priceToStr(price)
* creditLine()
* printCaptionData(caption, data, colonAt=40)

**UtilGp**

**Class Diagram - 2**

**Utility Class (General Purpose)**

* show ()

**Home**

**Class Diagram - 3**

**View Classes**

* show ()

**ApplyCard**

* show ()
* showFailure()
* showCat()
* showSubCat(valueCat)
* showItem(valueSubCat)
* savingItem(resItem)
* saveItem(valueCard, valueItem, price)

**Shop**

* show ()
* showDateRange()
* showBySortedAmount()
* showAmountRange()
* showByCat()
* showCatTot()
* reportMenu()

**Report**

* Code for app start

**Module mailApp**

**Class Diagram - 4**

**Classes for Data Access Methods**

**Data Access Objects (DAO)**

* addToCard(valueCard)
* initDbCards()
* updateCards(dbCard,argCard)
* readCards()
* checkCardNoExist(cardNo, dbCard)
* countCardByAdhaar(dbCard,adhaarNo)
* genCardNo(dbCard)
* getCardObject(dbCard, cardNo)

**DbCardData**

* initItems ()
* addToItem(valueItem)
* updateItems(dbItem)
* readItems()
* getDbCategory(dbItem)
* getDbSubCategory(dbItem)
* getItemById(dbItem, itemId)

**DbItemData**

* getFileName(cardHolderName,cardNo)
* addToTran(valueTran)
* readTrans(valueCard,HasItem=False,dbItem=None)
* queryByDateRange(dbTran,fromDate,toDate)
* queryAll(dbTran)
* queryByAmount(dbTran,fromAmount,toAmount)
* getItemById(dbItem, itemId)
* queryByCategory(dbTran,categoryId)
* printTran(dbTran)

**DbTranData**

***The following are the explanation for the usage of each of the classes:***

Class: **dBase**

|  |  |
| --- | --- |
| Class dBase is the model class for all data base objects. “ndb” is the dBase object. This is used throughout the program.  1. ndb. loadItem() is called to load all items into ndb.dbItem list. It is needed for shopping. Refer “class Item” for attributes of class Item. Using items list dbItem we can construct dbCat and dbSubCat lists for category list and sub category list.  2. ndb. loadCard () is called to load all user cards. Refer “class Card” for attributes of Card class. For user login, we use Card object of the particular user.  3. ndb.loadTran () is called to load all particular user transactions into ndb.dbTran for given user’s Card object . Refer “class Tran” for attributes of Tran class. For user login, we use Card object of the particular user.  4. ndb.loadTranByItem() loads transactions in ndb.dbTran. And each transaction Item object is set in every tran object. This function is used in reports.  5. ndb.loadItemCard() loads both ndb.dbItem and ndb.dbCard. Inside this function, we call ndb. loadItem() and ndb. loadCard ().  In short, dBase has ndb object throughout the app. dbItem, dbCard are loaded at app start. For Shopping, we load dbTran after login. For Reports, after login, dbTran is loaded where item object is found and set inside each tran object. | |
| **Attributes** | |
| dbItem | List of Item Data |
| dbCat | List of Category Data |
| dbSubCat | List of Sub Category Data |
| dbCard | List of Users Card Data |
| dbTran | List of Transactions for logged In User |
| IsLoggedIn | If user is logged in, this attribute becomes True. Else False. |
| logInCard | Logged In User’s Card object |
| **Methods** | |
| \_\_init\_\_() | Constructor |
| loadItem(IsMsg=True) |  |
| loadCard (IsMsg=True) |  |
| loadTran (valueCard, IsMsg=True) |  |
| loadTranByItem (valueCard, IsMsg=True) |  |
| loadItemCard (IsMsg=True) |  |

Class: **Item**

|  |  |
| --- | --- |
| Class **ItemData** is the model class for items needed for shopping. Items are under sub category. Sub categories are under Category.  There are 23 items.  We have four categories Bills, Vehicle, Entertainment, and Shopping.  Category “Bills” has sub categories Electricity, Mobile, Internet, and Water. This category cannot have Items. Bill amount will be automatically calculated and displayed for payment.  Category “Vehicle” has two sub categories Fuel, Servicing, and Accessories.  “Fuel” sub category has items Diesel or Petrol. After selection of diesel or petrol user has to enter amount for which fuel has to be filled.  “Servicing” sub category has items Full Car Service, Water Wash Service. For these items we are displaying price.  “Accessories” sub category has items Floor Mate, Seat Cover, Tyre. For these items we are displaying price.  Category “Entertainment” has sub categories “Movies” and “Dining”.  Sub Category “Movies” has items Hindi Movie, Telugu Movie, and English Movie. These items have price.  Sub Category “Dining” has items Music Restaurant (for 2 pax) and Pub with DJ(for 2 pax). These items have price.  Category “Shopping” has sub categories “Clothes” and “Food”.  Sub Category “Clothes” has items Formal Trouser, Formal Shirt, Casual Trouser, and Casual Shirt. They have price.  Sub Category “Food” has items South Indian Thali, North Indian Thali, and Chinese Thali. They have price. | |
| **Attributes** | |
| itemId | itemid is the id for each item. Based on id we can calculate category id and sub category id. First (1000 place) digit is the category id. First two digits(1000 and 100 place digits) is the sub category id. |
| category | Category name |
| subCategory | Sub category name |
| itemName | Shopping Item name |
| price | Price is nothing but amount going to spend in shopping |
| itemType | itemType can take values of 1, 2, and 3. For case 1: amount in shopping will be calculated automatically. For case 2: amount is entered by user to refill fuel. For case 3: the price from item master is the amount during shopping. |
| priceRange | priceRange is set only for itemType=1. |
| msg | msg is the series of messages needed for shopping. After item picked up, There is Shopping View displayed based on message in msg attribute. This message is parsed to form Shopping View based on item type. |
| **Methods** | |
| \_\_init\_\_(itemId, category, subCategory, itemName, price, itemType, priceRange, msg) | Constructor |

Class: **SubCategoryData**

|  |  |
| --- | --- |
| Class **SubCategoryData** is definition for shopping item’s sub category. Each sub category can have items. | |
| **Attributes** | |
| subCatId | Sub Category ID for internal purpose. The first two digits of each item itemId is the Sub Category ID. In subCatId first digit is the category ID. |
| Subcategory | Sub Category Name |
| **Methods** | |
| \_\_init\_\_(subCatId, subCategory, catId) | Constructor to define sub category object |

Class: **CategoryData**

|  |  |
| --- | --- |
| Class **CategoryData** is definition for shopping item’s category. Each category can have sub categories. | |
| **Attributes** | |
| catId | Category ID for internal purpose. The first digit of each item itemId is the Category ID. |
| category | Category Name |
| **Methods** | |
| \_\_init\_\_(catId, category) | Constructor to define category object |

Class: **CardData**

|  |  |
| --- | --- |
| Class “CardData” represent a particular user Card Details such as Card No, PIN Number, Card Holder Name, etc. We use this object to login. We record Credit Limit Used during shopping. We display Credit Limit Used, Credit Limit Available throughout shopping. After shopping a particular item, we display these two along Reward Earned for transaction and User’s Total Reward Earned. | |
| **Attributes** | |
| cardId | Data Base internal ID for user card |
| name | Card Holder Name |
| mobileNo | 10 digit Mobile Number |
| dob | Date of Birth of the user. Age should be 18 or above to apply card. |
| netSalary | Annual Net Salary of the user to apply for card. Rs3,00,000 should be minimum annual salary. |
| cardNo | Unique card number. It is used to login for shopping and viewing reports. |
| cardType | It has value of Gold or Platinum. The gold card is issued to card holder when user credit limit is less than Rs.1,00,000. If user credit limit is Rs.1,00,000 or above then platinum card will be issued. |
| creditLimit | 20% of Card Holder Net Salary is Credit Limit |
| creditUsed | Every transaction amount is added here |
| passCode | 4-digit Number for authentication and completing shopping the particular item. |
| adharNo | 12-digit Number for uniqueness of each user |
| rewardPoint | Initial value is 0, For gold card holder every Rs100 purchase 1 point will be added. For platinum card holder every Rs200 purchase 2points will be added. |
| rewardUsed | Initial value is 0. It is used for project future purpose at the time of redeem reward. |
| **Methods** |  |
| \_\_init\_\_(name, mobileNo, dob, netSalary, adharNo) | Constructor to define user card object |
| creditBal() | Returns creditLimit – creditUsed ie Credit Limit Available |

Class: **TranData**

|  |  |
| --- | --- |
| Class “**TranData**” is the record which will be saved in data base after each shopping. | |
| **Attributes** | |
| tranId | Each transaction internal data base ID |
| cardId | The cardId of person who is doing shopping |
| cardNo | cardNo of logged in user. This is used only for creating / accessing user Transaction Data Base file. |
| cardHolderName | cardHolderName of user. This is used only for creating / accessing user Transaction Data Base file. |
| tranDate | tranDate is datetime object of datetime module |
| itemId | itemId is shopping item id |
| price | Amount spent on shopping trasaction |
| tranItem | This is stored at the time of report generation. It is “None” value during shopping. |
| isTranItem | False at report generation. It is False value during shopping. |
| **Methods** |  |
| \_\_init\_\_(tranId, cardId, cardNo, cardHolderName, tranDate, itemId, price) | Constructor to define shopping transaction |
| Category() | Return Category Name using tranItem. If “isTranItem” is False, empty string is returned. |
| SubCategory() | Return Sub Category Name using tranItem. If “isTranItem” is False, empty string is returned. |
| ItemName() | Return Item Name using tranItem. If “isTranItem” is False, empty string is returned. |

Class: **UtilGp**

|  |  |
| --- | --- |
| Class “**UtilGp**” is a general purpose class. Common functions defined here. It is a collection of static classes. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| clear() | This function helps to clear the screen. In windows, we have to use command prompt to check this functionality. |
| dateToStr(valueDate,formatStr = '%d-%m-%Y') | It returns String format of given valueDate (datetime object) argument and optional formatStr argument |
| strToDate(strDate,formatStr = '%d-%m-%Y %H:%M:%S') | It returns datetime object of given strDate argument and optional formatStr argument. |
| now() | It returns datetime object for system date time |
| dateAddDays(valueDate,valueDays) | It adds “valueDays” number of days for given datetime object valueDate |
| age(dob) | It finds no of years between datetime “dob” and “now()” |
| isValidDate(strDate,formatStr='%d-%m-%Y') | It returns True if strDate (string date) is of formatStr (given format) |
| isValidDigit(strNo,isExactLen=True,noOfDigits=10) | If given strNo (given data) is of digits. If isExactLen is True, given data has to be “noOfDigits” length. |
| numberInput(prompt,isExactLen=False,noOfDigits=5) | This will ask user input for sequence of digits. It can check user input is of length “noOfDigits” when isExactLen is True. |
| sleep(sec) | It sleeps for sec-1 seconds |
| randNo(startNo, endNo, stepValue, isExcluded=False) | It generates random number between startNo and endNo with modulus value of stepValue. It includes startNo and endNo if isExluded is False. |
| parseItemMsg(valueItem,passCode,creditBal) | It is parser for each shopping item and it is “Shopping Completion Sub View”. It takes msg attribute of valueItem. Each msg lines of text are parsed. Here, ” :price”, “:tranMonYear”, “ : inputPrice”, “ :item” “:tranNextDate”, :tranDate”, “:pinConfirm” are parameter variables in “msg” attribute. |
| **Notes:**  ” :price” is replaced with valueItem.price for valueItem.itemType=3. For valueItem.itemType=2, “:inputPrice” is replaced with user input to get amount to be filled for fuel. For valueItem.itemType=3, random price based on valueItem.priceRange (for example “700-1000,50”, here 700-1000 is random nos range with modulus of 50) is replaced with user input to get amount to be filled for fuel. Technically we use regular expression here to split three components.  “:tranMonYear” (example “Oct-2019”) is replaced with MMM-YYYY format of today date.  “:tranDate” (example “10-10-2019”) is replaced with dd-MM-YYYY format of today date.  “:tranNextDate” (example “10-10-2019”) is replaced with dd-MM-YYYY format of tomorrow date.  “ :item” is replaced with valueItem.itemName attribute.  “:pinConfirm” It asks “Confirm PIN:” to complete shopping. | |
| mainMenu() | It display main menus |
| startApp() | It loads item and card data from data base and then invokes mainApp() method. |
| Login(fnToCall,tranLoader,pageTitle='Login') | It is called in two places: 1) Before shopping and 2) Report generation. fnToCall is function argument, being invoked after login. tranLoader is function argument to load ndb.dbTran list after login. |
| centerText(text) | It centers the text for total 79 chars |
| linePrint(text='-',spaces=79) | It prints sequence of “spaces” number given char (example, ‘-‘ or ‘\*’) |
| title(screenName) | It prints for Title of each Page View |
| priceToStr(price) | It converst given Price to String |
| creditLine() | It prints Credit Limit Used and Credit Limit Available during every shopping Page View. |
| printCaptionData(caption, data, colonAt=40) | It prints lines of “caption” : “data” where “:” is printed at position 40 and all print text are inside ‘\*’ed box. Here caption and data are tuple data. |

Class: **Home**

|  |  |
| --- | --- |
| Class “**Home**” is a View Class for main menus page. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| show () | Displays main menus. At app start,it loads database from file into app memory. |

Class: **Shop**

|  |  |
| --- | --- |
| Class “**Shop**” is a View Class for category page, sub category page, item page, Shopping Completion sub view. It contains also special method for failure handling code after invalid user input data. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| show () | Displays Login Page for shopping. After Login It calls showCat() method to display Category Page. If login fails, it calls showFailure() method to display options to exit or to go to main menu. |
| showFailure() | Displays options 1. Exit, 2-Go To Main Menu |
| showCat() | Displays Category Page. After selection of Category, it calls showSubCat() method where we pass selected category object as argument. If invalid category is selected, it calls showFailure() method. |
| showSubCat(valueCat) | Displays Sub Category Page. After selection of Sub Category, we display “Shopping Completion Sub View” and Items Page for Bills Sub Category selection and Other Category respectively. |
| showItem(valueSubCat) | Displays Items Page |
| savingItem(resItem) | If Bills Sub Category is selected or Item for other than Bills Sub Category is selected, we will prepare to save the shopping transaction. |
| saveItem(valueCard, valueItem, price) | It saves shopping transaction into database and displays the following: 1. Reward Earned for the transaction 2. Credit Limit Used 3. Credit Limit Available 4. Total Rewards Earned. |

Class: **ApplyCard**

|  |  |
| --- | --- |
| Class “**ApplyCard**” is a View Class for Applying Card. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| show () | Displays Applying Card Page. It asks user inputs for Name, Mobile Number, Annual Net Salary, Date of Birth, ADHAAR number, and PIN number. After valid data, It displays generated Card Number, Credit Limit, and Card Type. Then saves User Card data to database. |

Class: **Report**

|  |  |
| --- | --- |
| Class “**Report**” is a View Class for Reports Menu Page and for each report specific page. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| show () | Displays Reports Main Menu Page(reporMenu() method) after successful login using card no and PIN. After picking report option it takes specific report page. |
| showDateRange() | It asks from date and to date. For given date range it constructs report from ndb.dbTran list. |
| showBySortedAmount() | It arranges all transactions based on amount in ascending order. |
| showAmountRange() | It asks from amount and to amount. For within amount range, it constructs report using ndb.dbTran list. |
| showByCat() | It displays Category Page. Use has to pick category. For given category, it displays transactions. |
| showCatTot() | It calculates total amount spent for given category and displays the same. |
| reportMenu() | Displays Reports Menu Page. |

Class: **DbItemData**

|  |  |
| --- | --- |
| Class “**DbItemData**” is a Data Base Access / Manipulation Class for items data. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| initItems () | This returns list of items (hard coded objects) needed for shopping. We can save this item into Item DataBase Binary File. |
| addToItem(valueItem) | It saves “valueItem” item object into Item DataBase File. |
| updateItems(dbItem) | It updates dbItem list by replacing old content in Item DataBase file. |
| readItems() | It reads dbItem list from Item DataBase File and returns the same. |
| getDbCategory(dbItem) | It constructs dbCat list using ndb.dbItem data list. |
| getDbSubCategory(dbItem) | It constructs dbSubCat list using ndb.dbItem data list. |
| getItemById(dbItem, itemId) | For given item id, we find item object in dbItem data list. |

Class: **DbCardData**

|  |  |
| --- | --- |
| Class “**DbCardData**” is a Data Base Access / Manipulation Class for user cards data. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| addToCard(valueCard) | It saves “valueCard” user card object into Card DataBase Binary File. |
| initDbCards() | Creates Card DataBase Empty Binary File |
| updateCards(dbCard,argCard) | For given dbCard card data list and argCard card object, we update the argCard object into Card DataBase File. |
| readCards() | It returns card data list from Card DataBase File. |
| checkCardNoExist(cardNo, dbCard) | Validation Method to check cardNo exists in dbCard data list. |
| countCardByAdhaar(dbCard,adhaarNo) | It returns number of cards for given ADHAAR number. |
| genCardNo(dbCard) | It generates and returns card number between 8001 to 8999. |
| getCardObject(dbCard, cardNo) | It returns card object for given card number(“cardNo”). |

Class: **DbTranData**

|  |  |
| --- | --- |
| Class “**DbTranData**” is a Data Base Access / Manipulation Class for shopping transactions data. | |
| **Attributes** | |
| No Attributes |  |
| **Methods** | Collection of Static Methods. |
| getFileName(cardHolderName,cardNo) | It return transaction binary file name for given card holder name and card number. |
| addToTran(valueTran) | It saves “valueTran” shopping transaction object into Transaction DataBase Binary File. |
| updateCards(dbCard,argCard) | For given dbCard card data list and argCard card object, we update the argCard object into Card DataBase File. |
| readTrans(valueCard,HasItem=False,dbItem=None) | It returns transaction data list from Transaction DataBase File. |
| queryByDateRange(dbTran,fromDate,toDate) | Constructs report for given data range |
| queryAll(dbTran) | Duplicates dbTran transaction data list into another list and returns the same. |
| queryByAmount(dbTran,fromAmount,toAmount) | Constructs report for given amount range |
| queryByCategory(dbTran,categoryId) | Constructs report for given category |
| printTran(dbTran) | Prints dbTran transaction data list in the console app report form such as header, data display and line printing. |