

```

1 Receipt.py
2     from helper_functions import *
3     from Product import *
4     from Customer import *
5
6     class Receipt:
7         def __init__(self):
8             self.dict = {}
9
10        def __updateLineItem(self, receiptlineItemList):
11            receiptItemList = []
12            total = 0
13            for lineItem in receiptlineItemList:
14                receiptlineItemDict = {}
15                receiptlineItemDict["Invoice No"] = lineItem["Invoice No"]
16                receiptlineItemDict["Amount Paid Here"] = lineItem["Amount Paid Here"]
17                total += lineItem["Amount Paid Here"]
18                receiptItemList.append(receiptlineItemDict)
19            return receiptItemList, total
20
21    ① def create(self, receiptNo, receiptDate, customerCode, paymentMethod, paymentReference, remark, receiptlineItemList):
22        if receiptNo in self.dict:
23            return {'Is Error': True, 'Error Message': "Receipt No '{}' already exists. Cannot Create.".format(receiptNo)}
24        else:
25            receiptlineItemList, total = self.__updateLineItem(receiptlineItemList)
26
27            self.dict[receiptNo] = {"Date": receiptDate, "Customer Code": customerCode, "Payment Method": paymentMethod, "Payment Reference": paymentReference}
28            return {'Is Error': False, 'Error Message': ""}
29
30        def read(self, receiptNo):
31            if receiptNo in self.dict:
32                retreceipt = self.dict[receiptNo]
33            else:
34                return {'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Read.".format(receiptNo), ()}
35
36            return {'Is Error': False, 'Error Message': ""}, retreceipt
37
38    ② def update(self, receiptNo, newReceiptDate, newCustomerCode, newPaymentMethod, newPaymentReference, newRemark, newReceiptlineItemList):
39        if receiptNo in self.dict:
40            self.dict[receiptNo]["Date"] = newReceiptDate
41            self.dict[receiptNo]["Customer Code"] = newCustomerCode
42            self.dict[receiptNo]["Payment Method"] = newPaymentMethod
43            self.dict[receiptNo]["Payment Reference"] = newPaymentReference
44            self.dict[receiptNo]["Remarks"] = newRemark
45            receiptlineItemList, total = self.__updateLineItem(newReceiptlineItemList)
46
47            self.dict[receiptNo]["Total Received"] = total
48            self.dict[receiptNo]["Receipt Line Item"] = newReceiptlineItemList
49        else:
50            return {'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Update.".format(receiptNo)}
51
52        return {'Is Error': False, 'Error Message': ""}
53
54    ③ def delete(self, receiptNo):
55        if receiptNo in self.dict:
56            del self.dict[receiptNo]
57        else:
58            return {'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Delete.".format(receiptNo)}
59
60        return {'Is Error': False, 'Error Message': ""}
61
62        def dump(self):
63            # will dump all products data by returning 1 dictionary as output.
64            return (self.dict)
65
66    ④ def update_receipt_line(self, receiptNo, invoiceNo, amountPaid):
67        if receiptNo in self.dict:
68            receiptlineItemList = []
69            bUpdated = False
70            for lineItem in self.dict[receiptNo]["Receipt Line Item"]:
71                invoiceLineItem = {}
72                if lineItem["Invoice No"] == invoiceNo:
73                    invoiceLineItem["Invoice No"] = invoiceNo
74                    invoiceLineItem["Amount Paid Here"] = amountPaid
75
76                    receiptlineItemList.append(invoiceLineItem)
77                    bUpdated = True
78                else:
79                    receiptlineItemList.append(lineItem)
80
81            if bUpdated:
82                receiptlineItemList, total = self.__updateLineItem(receiptlineItemList)
83                self.dict[receiptNo]["Receipt Line Item"] = receiptlineItemList
84                self.dict[receiptNo]["Total Received"] = total
85            else:
86                return {'Is Error': True, 'Error Message': "Receipt Code '{}' not found in Invoice No '{}' . Cannot Update.".format(receiptNo, invoiceNo)}
87
88        return {'Is Error': False, 'Error Message': ""}
89
90    ⑤ def delete_receipt_line(self, receiptNo, invoiceNo):
91        # The line item of this invoice number is updated to delete this product code.
92        # Note that all the related data in the invoice must be updated such as Total, VAT, and Amount Due.
93        # Returns dictionary ('Is Error': ___, 'Error Message': ____)
94        if receiptNo in self.dict:
95            total = 0
96            receiptlineItemList = []
97            bDeleted = False
98            for lineItem in self.dict[receiptNo]["Receipt Line Item"]:
99                if lineItem["Invoice No"] == invoiceNo:
100                    bDeleted = True
101                else:
102                    receiptlineItemList.append(lineItem)
103
104            if bDeleted:
105                receiptlineItemList, total = self.__updateLineItem(receiptlineItemList)
106                self.dict[receiptNo]["Receipt Line Item"] = receiptlineItemList
107                self.dict[receiptNo]["Total Received"] = total
108
109            else:
110                return {'Is Error': True, 'Error Message': "Receipt Code '{}' not found in Invoice No '{}' . Cannot Delete.".format(receiptNo, invoiceNo)}
111
112        return {'Is Error': False, 'Error Message': ""}
113
114        return {'Is Error': False, 'Error Message': ""}

```

987 API.py

```
# APIpy
from helper_functions import *
#this file will contain all API functions calls exposed to outside world for users to use

# function about payment method
def create_payment_method(paymentMethod, code, name):
    result = paymentMethod.create(code, name)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Payment Method Success.')
    return result #send result for caller program to use

def read_payment_method(paymentMethod, code):
    result = paymentMethod.read(code) #returns tuple of (error dict, data dict)
    if result[0][0]: #in case error
        print(result[0][0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_payment_method(paymentMethod, code, newName):
    result = paymentMethod.update(code, newName) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Payment Method Success.')
    return result #send result for caller program to use

def delete_payment_method(paymentMethod, code):
    result = paymentMethod.delete(code)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Delete Success.')
    return result #send result for caller program to use

def report_list_payment_method(paymentMethod):
    result = paymentMethod.dump()
    #printDictInCSVFormat(result, ('Code'), ('Name', 'Units'))
    print(result)
    return result #send result for caller program to use

# function about Product
def create_product(products, code, name, units):
    result = products.create(code, name, units)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Create Success.')
    return result #send result for caller program to use

def read_product(products, code):
    result = products.read(code) #returns tuple of (error dict, data dict)
    if result[0][0]: #in case error
        print(result[0][0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_product(products, code, newName, newUnits):
    result = products.update(code, newName, newUnits) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Update Success.')
    return result #send result for caller program to use

def delete_product(products, code):
    result = products.delete(code)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Delete Success.')
    return result #send result for caller program to use

def report_list_products(products):
    result = products.dump()
    #printDictInCSVFormat(result, ('Code'), ('Name', 'Units'))
    print(result)
    return result #send result for caller program to use

# Function about Customer
def create_customer(customers, customerCode, customerName, address, creditlimit, country):
    result = customers.create(customerCode, customerName, address, creditlimit, country)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Create Success.')
    return result #send result for caller program to use

def read_customer(customers, customerCode):
    result = customers.read(customerCode) #returns tuple of (error dict, data dict)
    if result[0][0]: #in case error
        print(result[0][0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_customer(customers, customerCode, newCustomerName, newAddress, newCreditLimit, newCountry):
    result = customers.update(customerCode, newCustomerName, newAddress, newCreditLimit, newCountry) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Update Success.')
    return result #send result for caller program to use

def delete_customer(customers, customerCode):
    result = customers.delete(customerCode)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Delete Success.')
    return result #send result for caller program to use

def report_list_all_customers(customers):
    result = customers.dump()
    printDictInCSVFormat(result, ('Customer Code'), ('Name', 'Address','Credit Limit', 'Country'))
    return result #send result for caller program to use

# function about Invoice
def create_invoice(invoices, invoiceNo, invoiceDate, customerCode, dueDate, invoiceLineTuplesList):
    result = invoices.create(invoiceNo, invoiceDate, customerCode, dueDate, invoiceLineTuplesList) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Create Success.')
    return result #send result for caller program to use

def read_invoice(invoices, invoiceNo):
    result = invoices.read(invoiceNo) #returns tuple of (error dict, data dict)
    if result[0][0]: #in case error
        print(result[0][0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_invoice(invoices, invoiceNo, newInvoiceDate, newCustomerCode, newDueDate, newInvoiceLineTuplesList):
    result = invoices.update(invoiceNo, newInvoiceDate, newCustomerCode, newDueDate, newInvoiceLineTuplesList) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Update Success.')
    return result #send result for caller program to use

def delete_invoice(invoices, invoiceNo):
    result = invoices.delete(invoiceNo)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Delete Success.')
    return result #send result for caller program to use
```

payment - method

Product

Customer

Invoice

↑(mbo)

```

153 def update_invoice_line(invoices, invoiceNo, productCode, newQuantity, newUnitPrice):
154     result = invoices.update_invoice_line(invoiceNo, productCode, newQuantity, newUnitPrice) #returns error dictionary
155     if result['Is Error']: #if error
156         print(result['Error Message'])
157     else:
158         print('Invoice Line Item Update Success.')
159     return result #send result for caller program to use
160
161 def delete_invoice_line(invoices, invoiceNo, productCode):
162     result = invoices.delete_invoice_line(invoiceNo, productCode) #returns error dictionary
163     if result['Is Error']: #if error
164         print(result['Error Message'])
165     else:
166         print('Invoice Line Item Delete Success.')
167     return result #send result for caller program to use
168
169 def report_list_all_invoices(invoices, customers, products):
170     # Will dump all invoices data and return 1 dictionary as a result (with header and line item joined).
171     # Please show the customer name and product name also.
172     # A helper function such as def print_tabular_dictionary(tabularDictionary) can then be called to print this in a tabular (table-like) form with column headings and data.
173
174     allInvoice = invoices.dump()
175     # re-format result by add Customer Name in object and Product Name in line item
176     result = {}
177     for invoiceNo, invoiceDetail in allInvoice.items():
178         newValue = {}
179         for invoiceColumn, invoiceData in invoiceDetail.items():
180             if invoiceColumn == "Customer Code":
181                 newValue[invoiceColumn] = invoiceData
182                 customer = customers.read(invoiceDetail["Customer Code"])
183                 if customer[0]['Is Error']:
184                     newValue["Customer Name"] = ""
185                 else:
186                     newValue["Customer Name"] = customer[1]["Name"]
187             elif invoiceColumn == "Items List":
188                 newListItemList = []
189                 for lineItem in invoiceDetail['Items List']:
190                     newItem = {}
191                     for lineItemColumn, lineItemData in lineItem.items():
192                         if lineItemColumn == "Product Code":
193                             newItem[lineItemColumn] = lineItemData
194
195                             product = products.read(lineItem["Product Code"])
196                             if product[0]['Is Error']:
197                                 newItem["Product Name"] = ""
198                             else:
199                                 newItem["Product Name"] = product[1]["Name"]
200
201                             newItem[lineItemColumn] = lineItemData
202                             newListItemList.append(newItem)
203                             newValue[invoiceColumn] = newListItemList
204
205             else:
206                 newValue[invoiceColumn] = invoiceData
207         result[invoiceNo] = newValue
208     #print (result)
209     printDictInCSVFormat(result, ('Invoice No','Customer Code','Due Date','Total','VAT','Amount Due','Items List'))
210     return result #send result for caller program to use
211
212 def report_products_sold(invoices, products, dateStart, dateEnd):
213     # Will return 2 dictionaries:
214     # 1) a dictionary as list of products sold in the given date range in tabular format of: Product Code, Product Name, Total Quantity Sold, Total Value Sold. Here, (product code) is primary key.
215     # And 2) a second dictionary of the footer will also be returned containing: t the end also show the sum of Total Value Sold.
216     allInvoice = invoices.dump()
217     result = {}
218     result2 = {}
219     sumTotalValueSold = 0
220     for key, value in allInvoice.items():
221         if (dateStringInDateRange(value['Date'], dateStart, dateEnd)):
222             for lineItem in value['Items List']:
223                 primaryKey = lineItem['Product Code']
224                 if primaryKey in result:
225                     result[primaryKey]['Total Quantity Sold'] += lineItem['Quantity']
226                     result[primaryKey]['Total Value Sold'] += lineItem['Extended Price']
227                 else:
228                     product = products.read(lineItem["Product Code"])
229                     if product[0]['Is Error']:
230                         productName = ""
231                     else:
232                         productName = product[1]["Name"]
233                     result[primaryKey] = {'Product Code':lineItem['Product Code'], 'Product Name':productName,'Total Quantity Sold':lineItem['Quantity'],'Total Value Sold':lineItem['Extended Price']}
234
235         sumTotalValueSold += lineItem['Extended Price']
236
237 result2['Sum of'] = {'Total Value Sold':sumTotalValueSold}
238 printDictInCSVFormat(result, (None, ('Product Code','Product Name', 'Total Quantity Sold', 'Total Value Sold')))
239 printDictInCSVFormat(result2, (None, ('Total Value Sold',)))
240     return result.values(), result2
241
242 def report_customer_products_sold_list(invoices, products, customers, dateStart, dateEnd):
243     # Will return 2 dictionaries:
244     # 1) a dictionary as list of customers and list the products sold to them in the given date range in this format: Customer Code, Customer Name, Product Code, Product Name, Invoice No, Total Quantity Sold, Total Value Sold.
245     # And 2) a second footer dictionary showing: At the end also show the sum of Quantity Sold and sum of Value Sold.
246     allInvoice = invoices.dump()
247     result = {}
248     result2 = {}
249     sumTotalQuantitySold = 0
250     sumTotalValueSold = 0
251     for key, value in allInvoice.items():
252         invoiceNo = key
253         if (dateStringInDateRange(value['Date'], dateStart, dateEnd)):
254             for lineItem in value['Items List']:
255                 primaryKey = value['Customer Code'] + lineItem['Product Code'] + invoiceNo
256                 if primaryKey in result:
257                     result[primaryKey]['Quantity Sold'] += lineItem['Quantity']
258                     result[primaryKey]['Value Sold'] += lineItem['Extended Price']
259                 else:
260                     product = products.read(lineItem["Product Code"])
261                     if product[0]['Is Error']:
262                         productName = ""
263                     else:
264                         productName = product[1]["Name"]
265
266                     customer = customers.read(value['Customer Code'])
267                     if customer[0]['Is Error']:
268                         customerName = ""
269                     else:
270                         customerName = customer[1]["Name"]
271                     result[primaryKey] = {'Customer Code':value['Customer Code'], 'Customer Name':customerName,'Product Code':lineItem['Product Code'], 'Product Name':productName,'Invoice No':invoiceNo,'Total Quantity Sold':lineItem['Quantity'],'Total Value Sold':lineItem['Extended Price']}
272
273         sumTotalQuantitySold += lineItem['Quantity']
274         sumTotalValueSold += lineItem['Extended Price']
275
276 result2['Sum of'] = {'Quantity Sold':sumTotalQuantitySold, 'Value Sold':sumTotalValueSold}
277 printDictInCSVFormat(result, (None, ('Customer Code','Customer Name', 'Product Code', 'Product Name', 'Invoice No', 'Quantity Sold', 'Value Sold')))
278 printDictInCSVFormat(result2, (None, ('Quantity Sold', 'Value Sold')))
279     return result.values(), result2
280
281 def report_customer_products_sold_total(invoices, products, customers, dateStart, dateEnd):
282     allInvoice = invoices.dump()
283     result = {}
284     result2 = {}
285     sumTotalQuantitySold = 0
286     sumTotalValueSold = 0
287     for key, value in allInvoice.items():
288         invoiceNo = key
289         if (dateStringInDateRange(value['Date'], dateStart, dateEnd)):
290             for lineItem in value['Items List']:
291                 primaryKey = value['Customer Code'] + lineItem['Product Code']
292                 if primaryKey in result:
293                     result[primaryKey]['Total Quantity Sold'] += lineItem['Quantity']
294                     result[primaryKey]['Total Value Sold'] += lineItem['Extended Price']
295                 else:
296                     product = products.read(lineItem["Product Code"])
297                     if product[0]['Is Error']:
298                         productName = ""
299                     else:
300                         productName = product[1]["Name"]
301
302                     customer = customers.read(value['Customer Code'])
303                     if customer[0]['Is Error']:
304                         customerName = ""
305                     else:
306                         customerName = customer[1]["Name"]
307                     result[primaryKey] = {'Customer Code':value['Customer Code'], 'Customer Name':customerName,'Product Code':lineItem['Product Code'], 'Product Name':productName,'Total Quantity Sold':lineItem['Quantity'],'Total Value Sold':lineItem['Extended Price']}
308
309         sumTotalQuantitySold += lineItem['Quantity']
310         sumTotalValueSold += lineItem['Extended Price']
311
312 result2['Sum of'] = {'Total Quantity Sold':sumTotalQuantitySold,'Total Value Sold':sumTotalValueSold}
313 printDictInCSVFormat(result, (None, ('Customer Code','Customer Name', 'Product Code', 'Product Name', 'Total Quantity Sold', 'Total Value Sold')))
314 printDictInCSVFormat(result2, (None, ('Total Quantity Sold', 'Total Value Sold')))
315     return result.values(), result2

```

Invoice

```

313 def report_unpaid_invoices(invoices,customers,receipts):
314     total = 0
315     allreceipt = receipts.dump()
316     result = {}
317     for receiptKey, receiptCol in allreceipt.items():
318         for receiptLine in receiptCol['Receipt Line Item']:
319             if receiptLine['Invoice No'] not in result: # ດັລ້ວມາ Invoice ດ້ວຍເຫຼືອ
320                 newValue = {}
321                 newValue['Invoice Amount Received'] = receiptLine['Amount Paid Here']
322                 invoice = invoices.read(receiptLine['Invoice No'])
323                 if invoice[0]['Is Error']:
324                     newValue['Customer Name'] = ""
325                     newValue['Invoice Amount Due'] = ""
326                     newValue['Invoice Amount Not Paid'] = ""
327                 else:
328                     newValue['Invoice Date'] = invoice[1]['Invoice Date']
329                     newValue['Invoice Amount Due'] = invoice[1]['Amount Due']
330                     newValue['Invoice Amount Not Paid'] = newValue['Invoice Amount Due'] - newValue['Invoice Amount Received']
331                     customer = customers.read(invoice[1]['Customer Code'])
332                     if customer[0]['Is Error']:
333                         newValue['Customer Name'] = ""
334                     else:
335                         newValue['Customer Name'] = customer[1]['Name']
336                     result[receiptLine['Invoice No']] = newValue
337             else:
338                 result[receiptLine['Invoice No']]['Invoice Amount Received'] += receiptLine['Amount Paid Here']
339                 result[receiptLine['Invoice No']]['Invoice Amount Due'] = result[receiptLine['Invoice No']]['Invoice Amount Due'] - result[receiptLine['Invoice No']]['Invoice Amount Received']
340                 result[receiptLine['Invoice No']]['Invoice Amount Not Paid'] = result[receiptLine['Invoice No']]['Invoice Amount Due'] - result[receiptLine['Invoice No']]['Invoice Amount Received']
341                 printDictToCSVFormat(result, ('Invoice No,'), ('Invoice Date', 'Customer Name', 'Invoice Amount Due', 'Invoice Amount Received', 'Invoice Amount Not Paid'))
342
343     for line in result:
344         total += result[line]['Invoice Amount Not Paid']
345
346     print("Total Debt: ",total)
347
348
349
350
351     # Function about Receipt
352     def create_receipt(receipt, receiptNo, receiptDate, customerCode, paymentMethod, paymentReference, remark, receiptLineItemList):
353         result = receipt.create(receiptNo, receiptDate, customerCode, paymentMethod, paymentReference, remark, receiptLineItemList)#returns error dictionary
354         if result[0]['Is Error']: #if error
355             print(result[0]['Error Message'])
356         else:
357             print('Receipt Create Success.')
358         return result #send result for caller program to use
359
360     def read_receipt(receiptNo):
361         result = receipt.read(receiptNo) #returns tuple of (error dict, data dict)
362         if result[0]['Is Error']: #in case error
363             print(result[0]['Error Message'])
364         else:
365             print(result[1])
366         return result #send result for caller program to use
367
368     def update_receipt(receipt, receiptNo, newReceiptDate, newCustomerCode, newPaymentMethod, newPaymentReference, newRemark, newReceiptLineItemList):
369         result = receipt.update(receiptNo, newReceiptDate, newCustomerCode, newPaymentMethod, newPaymentReference, newRemark, newReceiptLineItemList) #returns error dictionary
370         if result[0]['Is Error']: #if error
371             print(result[0]['Error Message'])
372         else:
373             print('Receipt Update Success.')
374         return result #send result for caller program to use
375
376     def delete_receipt(receiptNo):
377         result = receipt.delete(receiptNo)#returns error dictionary
378         if result[0]['Is Error']: #if error
379             print(result[0]['Error Message'])
380         else:
381             print('Receipt Delete Success.')
382         return result #send result for caller program to use
383
384     def update_receipt_line(receipt, receiptNo, invoiceNo, amountPaid):
385         result = receipt.update_receipt_line(receiptNo, invoiceNo, amountPaid) #returns error dictionary
386         if result[0]['Is Error']: #if error
387             print(result[0]['Error Message'])
388         else:
389             print('Invoice Line Item Update Success.')
390         return result #send result for caller program to use
391
392     def delete_receipt_line(receipt, receiptNo, invoiceNo):
393         result = receipt.delete_receipt_line(receiptNo, invoiceNo) #returns error dictionary
394         if result[0]['Is Error']: #if error
395             print(result[0]['Error Message'])
396         else:
397             print('Receipt Line Item Delete Success.')
398         return result #send result for caller program to use
399
400     def report_list_all_receipt(receipt, invoices, customers):
401         allreceipt = receipt.dump()
402         # re-format result by add Customer Name in object and Product Name in line item
403         result = {}
404         for receiptNo, receiptDetail in allreceipt.items():
405             newValue = {}
406             for receiptColumn, receiptData in receiptDetail.items():
407                 if receiptColumn == 'Customer Code':
408                     newValue[receiptColumn] = receiptData
409                     customer = customers.read(receiptDetail['Customer Code'])
410                     if customer[0]['Is Error']:
411                         newValue['Customer Name'] = ""
412                     else:
413                         newValue['Customer Name'] = customer[1]['Name']
414
415                 elif receiptColumn == "Receipt Line Item":
416                     newlineItemList = []
417                     for lineitem in receiptDetail['Receipt Line Item']:
418                         newlineItem = {}
419                         for lineitemColumn, lineitemData in lineitem.items():
420                             if lineitemColumn == "Invoice No":
421                                 newlineItem[lineitemColumn] = lineitemData
422                                 invoice = invoices.read(lineitem["Invoice No"]) #join
423
424                                 if invoice[0]['Is Error']:
425                                     newlineItem[1]['Invoice Date'] = ""
426
427                                 else:
428                                     newlineItem['Invoice Date'] = invoice[1]['Invoice Date']
429
430                         newlineItemList.append(newlineItem)
431                         newValue[receiptColumn] = newlineItemList
432
433                     newValue[receiptColumn] = receiptData
434             result[receiptNo] = newValue
435         printDictToCSVFormat(result, ('Receipt No,'), ('Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remarks', 'Total Received', 'Receipt Line Item'))
436
437         return result #send result for caller program to use
438

```

unpaid

receipt

v 98% 1 lab2_mainfunction.py

```

1 from helper.functions import *
2 from Invoice import *
3 from Receipt import *
4 from Customer import *
5 from PaymentMethod import *
6 from API import *
7
8
9 #replace these lines with main function
10 def main():
11     #main function begins here
12     try:
13
14         products = Product() # create object products from class Product. Starts as blank dict.
15
16         #HDD01: {'Name': 'Seagate HDD 80 GB', 'Units': 'PCS'},
17         #HDD02: {'Name': 'IBM HDD 60 GB', 'Units': 'PCS'},
18         #INT99: {'Name': 'Intel Pentium IV 3.6 GHz', 'Units': 'PCS'}
19
20         C create_product(products, "HDD01", "Seagate HDD 80 GB", "PCS")
21         C create_product(products, "HDD02", "IBM HDD 60 GB", "PCS")
22         C create_product(products, "INT99", "Intel Pentium IV 3.6 GHz", "PCS")
23         C create_product(products, "INT99", "Intel Pentium V 4.2 GHz", "PCS")
24         report_list_products(products)
25         waitForKeyPress("Above are results for creating 4 products.")
26
27 R read_product(products, "HDD01")
28 R read_product(products, "HDD099") #Error
29     #correct the spelling error of "Intle"
30     update_product(products, "HDD01", "Intel Pentium IV 3.6 GHz", newunits = "PCS",
31     newCode = "INT99")
32     U update_product(products, "INT99", "Intel Pentium V 4.2 GHz", "PCS")
33     report_list_products(products)
34     waitForKeyPress("Results after 2 reads, 2 updates to correct Intle spelling.")
35
36 D delete_product(products, "INT33") #Error
37 D delete_product(products, "INT99")
38     report_list_products(products)
39     waitForKeyPress("Results after deleting INT33 (not exist error) and INT99.")
40
41     # pretty print a dictionary (in helper functions):
42     printDictDetail(products.dict)
43     waitForKeyPress("Above is dictionary printed in better format.")
44
45     # pretty print in column format a dictionary (in helper functions):
46     printDicttInCSFormat(products.dict, ("Code",), ("Name", "Units"))
47     waitForKeyPress("Above is dictionary printed in csv format for copy/paste to excel.")
48
49     ...
50
51     #Shows in case of untrapped exception:
52     result = products.dict["HDD05"]
53     waitForKeyPress("There will be error and exit before you see this.")
54
55
56     customers = Customer()
57     C create_customer(customers, "Sam", "Sam Co., Ltd.", "122 Bangkok, 500000, Thailand")
58     C create_customer(customers, "CP", "Thaivon Pokaphan", "14 Sukhumvit, Bangkok, 2000000, Thailand")
59     report_list_all_customers(customers)
60     waitForKeyPress("Above are results for creating 2 customers.")
61     print(" ")
62
63 R read_customer(customers, "IT City")#Error
64 read_customer(customers, "Sam")
65 U update_customer(customers, newCustomerName = "CPL", newAddress = "123 Bangkok", newCreditLimit = 100000, newCountry = "Theiland", customerCode = "CP")
66 D delete_customer(customers, "CP1")#not found
67     update_customer(customers, "CP")
68     report_list_all_customers(customers)
69     waitForKeyPress("Results after read, update and delete customer CP")
70     print(" ")
71
72     I invoices, invoices()
73     C create_invoice(invoices, "INT100/20", invoiceDate="2020-01-02", customerCode="Sam", dueDate=None, invoiceLineTuplesList=[{"Product Code": "HDD01", "Quantity": 2, "Unit Price": 3000}, {"Product Code": "HDD02", "Quantity": 1, "Unit Price": 2000}])
74     C create_invoice(invoices, "INT100/20", "2020-01-04", "CP", None, [{"Product Code": "HDD02", "Quantity": 1, "Unit Price": 2000}])
75     report_list_all_invoices(invoices, customers, products)
76     waitForKeyPress("Above are results for creating 2 invoices and line item.")
77     print(" ")
78
79     R read_invoice(invoices, "INT100/20")
80     update_invoice(invoices, "INT100/20", newInvoiceNo="INT100/20", newCustomerCode="Sam", newDueDate=None, newInvoiceLineTuplesList=[{"Product Code": "HDD01", "Quantity": 2, "Unit Price": 3000}, {"Product Code": "HDD02", "Quantity": 1, "Unit Price": 2000}])
81     report_list_all_invoices(invoices, customers, products)
82     waitForKeyPress("Results after read, update and delete Invoice Line Item")
83     print(" ")
84
85     update_invoice_line(invoices, "INT100/20", "HDD02")
86     report_list_all_invoices(invoices, customers, products)
87     waitForKeyPress("Results after delete Invoice Line Item")
88     print(" ")
89
90     D delete_invoice_line(invoices, "INT100/20", "HDD02")
91     report_list_all_invoices(invoices, customers, products)
92     waitForKeyPress("Results after delete Invoice Line Item")
93     print(" ")
94
95     #Test receipt functions
96     receipts = Receipt()
97     receipts.read_receipts(customers)
98     print("****receipts: ", receipts)
99     print("****receipts.read_receipts(customers): ", receipts.read_receipts(customers))
100    print("****receipts.read_receipts(customers): ", receipts.read_receipts(customers))
101    waitForKeyPress("Results of creating 3 receipts: RCT1001/20, RCT1002/20, and RCT1003/20")
102
103    #Read receipt
104    read_receipt(receipts, "RCT1001/20")
105    read_receipt(receipts, "RCT1003/20")
106    read_receipt(receipts, "RCT1002/20")
107    report_list_all_receipts(receipts, invoices, customers)
108    waitForKeyPress("Results of reading 3 receipts: RCT1001/20 (successfully), RCT1003/20 (successfully), and RCT1002/20 (unsuccessfully)")
109    print(" ")
110
111    #Update receipt
112    print("****50. Update Receipt", "***50)
113    update_receipt(receipts, "RCT1001/20", "2020-02-05", "Sam", "Debit Card", "VISA card", "Partially paid on INT100/20 and INT101/20", ["Invoice No": "INT100/20", "Amount Paid Here":100], ["Invoice No": "INT101/20", "Amount Paid Here":200])
114    create_receipt(receipts, "RCT1001/20", "2020-02-06", "CP", "Credit Card", "Master Card, Citibank", "Partially paid on INT100/20", ["Invoice No": "INT100/20", "Amount Paid Here":100], ["Invoice No": "INT100/20", "Amount Paid Here":100])
115    report_list_all_receipts(receipts, invoices, customers)
116    waitForKeyPress("Results of updating 3 receipts: RCT1001/20 (successfully), RCT1002/20 (successfully), and RCT1003/20 (unsuccessfully)")
117    print(" ")
118
119    #Delete receipt
120    print("****50. Delete Receipt ", "***50)
121    print("****")
122    delete_receipt(receipts, "RCT1003/20")
123    delete_receipt(receipts, "RCT1002/20")
124    report_list_all_receipts(receipts, invoices, customers)
125    waitForKeyPress("Results of deleting 2 receipts: RCT1002/20 (successfully) and RCT1004/20 (unsuccessfully)")
126    print(" ")
127
128    #Delete receipt
129    print("****50. Delete Receipt line ", "***50)
130    print("****")
131    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 2000)
132    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 5000)
133    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 5000)
134    report_list_all_receipts(receipts, invoices, customers)
135    waitForKeyPress("Result of updating 3 receipt line, first successfully, others unsuccessfully")
136    print(" ")
137
138    #Delete receipt line
139    print("****50. Update receipt line ", "***50)
140    print("****")
141    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 2000)
142    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 5000)
143    update_receipt_line(receipts, "RCT1001/20", "INT100/20", 5000)
144    report_list_all_receipts(receipts, invoices, customers)
145    waitForKeyPress("Result of deleting 3 receipt line, first successfully, others unsuccessfully")
146    print(" ")
147
148    #Print receipt report
149    print("****50. Delete receipt line ", "***50)
150    print("****")
151    delete_receipt_line(receipts, "RCT1001/20", "INT101/20")
152    delete_receipt_line(receipts, "RCT1003/20", "INT100/20")
153    delete_receipt_line(receipts, "RCT1002/20", "INT103/20")
154    report_list_all_receipts(receipts, invoices, customers)
155    waitForKeyPress("Result of deleting 3 receipt line, first successfully, others unsuccessfully")
156    print(" ")
157
158    #Print ("Unfile Report")
159    report_products_sold(invoices, products, "2020-01-01", "2020-01-31")
160    report_customer_products_sold_list(invoices, products, customers, "2020-01-01", "2020-01-31")
161    report_customer_products_sold_total(invoices, products, customers, "2020-01-01", "2020-01-31")
162    report_unpaid_invoices(invoices, customers, receipts)
163    print(" ")
164
165
166    except: #this traps for unexpected system errors
167        print ("Unexpected error:", sys.exc_info()[0])
168        raise #this line can be erased. It is here to raise another error so you can see which line to debug.
169    else:
170        print("Normal Termination. Goodbye!")
171
172 #main function ends
173
174
175
176
177 if __name__ == "__main__":
178     main()

```

- ჟერენერირებული CRUD მქონე კრეიტ რეცეპტის დამზადება *** Create receipt ***

ເລືອກຫົດ

Product Create Success.
{'H001': {'Name': 'Seagate HDD 80 GB', 'Units': 'PCS'}, 'H002': {'Name': 'IBM HDD 60 GB', 'Units': 'PCS'}, 'INT01': {'Name': 'Intel Pentium IV 3.6 GHz', 'Units': 'PCS'}, 'INT99': {'Name': 'Intel Pentium V 4.2 GHz', 'Units': 'PCS'}}
Above are results for creating 4 products. (Press a key to continue).
{'Name': 'IBM HDD 60 GB', 'Units': 'PCS'}
Product Code 'H003' not found. Cannot Read.
Product Update Success.
{'H001': {'Name': 'Seagate HDD 80 GB', 'Units': 'PCS'}, 'H002': {'Name': 'IBM HDD 60 GB', 'Units': 'PCS'}, 'INT01': {'Name': 'Intel Pentium IV 3.6 GHz', 'Units': 'PCS'}, 'INT99': {'Name': 'Intel Pentium V 4.2 GHz', 'Units': 'PCS'}}
Results after 2 reads, 2 updates to correct Intel spelling. (Press a key to continue).
Product Code 'INT33' not found. Cannot Read.
Product Delete Success.
{'H001': {'Name': 'Seagate HDD 80 GB', 'Units': 'PCS'}, 'H002': {'Name': 'IBM HDD 60 GB', 'Units': 'PCS'}, 'INT01': {'Name': 'Intel Pentium IV 3.6 GHz', 'Units': 'PCS'}}
Results after deleting INT33 (not exist error) and INT99. (Press a key to continue).

H001, Seagate HDD 80 GB, PCS
H002, IBM HDD 60 GB, PCS
INT01, Intel Pentium IV 3.6 GHz, PCS
Above is dictionary printed in csv format for copy/paste to excel. (Press a key to continue).
Customer Create Success.
Customer Create Success.
Customer Code, Name, Address, Credit Limit, Country
Sam Co., Ltd., 122 Bangkok, 500000, Thailand
(P) CP, Sam Co., Ltd., 122 Bangkok, 100000, Thailand
Above are results for creating 2 customers. (Press a key to continue).

Customer Code 'IT' City not found. Cannot Read.
{'Name': 'Sam Co., Ltd.', 'Address': '122 Bangkok', 'Credit Limit': 500000, 'Country': 'Thailand'}
Customer Update Success.
Customer Code 'CP' not found. Cannot Delete.

Customer Code, Name, Address, Credit Limit, Country
Sam, Sam Co., Ltd., 122 Bangkok, 500000, Thailand
CP, CPALL, 123 Bangkok, 100000, Thailand
Results after 2 reads, and update and delete customer CP (Press a key to continue).

Invoice Create Success.
Invoice Create Success.

Invoice No, Invoice Date, Customer Code, Due Date, Total, VAT, Amount Due, Items List
INT100/20, 2020-01-03, Sam, None, 8000, 560.0, 8560.0, [{"Product Code": "H001", "Product Name": "Seagate HDD 80 GB", "Quantity": 2, "Unit Price": 3000, "Extended Price": 6000}, {"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}]
INT101/20, 2020-01-04, CP, None, 2000, 140.0, 2140.0, [{"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}]
Above are results for creating 2 invoices and line item. (Press a key to continue).

{'Invoice Date': '2020-01-02', 'Customer Code': 'Sam', 'Due Date': None, 'Total': 8000, 'VAT': 560.0, 'Amount Due': 8560.0, 'Items List': [{"Product Code": "H001", "Quantity": 2, "Unit Price": 3000, "Extended Price": 6000}, {"Product Code": "H002", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}], 'Extended Price': 2000}
Invoice Update Success.

Invoice No, Invoice Date, Customer Code, Due Date, Total, VAT, Amount Due, Items List
INT100/20, 2020-01-03, Sam, None, 8000, 560.0, 8560.0, [{"Product Code": "H001", "Product Name": "Seagate HDD 80 GB", "Quantity": 2, "Unit Price": 3000, "Extended Price": 6000}, {"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}]
INT101/20, 2020-01-04, CP, None, 2000, 140.0, 2140.0, [{"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}]
Results after read, update and delete invoice (Press a key to continue).

Invoice Line Item Update Success.

Invoice No, Invoice Date, Customer Code, Due Date, Total, VAT, Amount Due, Items List
INT100/20, 2020-01-03, Sam, None, 14000, 980.0, 14980.0, [{"Product Code": "H001", "Product Name": "Seagate HDD 80 GB", "Quantity": 2, "Unit Price": 3000, "Extended Price": 6000}, {"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 8, "Unit Price": 1000, "Extended Price": 8000}], 'Extended Price': 8000]
INT101/20, 2020-01-04, CP, None, 2000, 140.0, 2140.0, [{"Product Code": "H002", "Product Name": "IBM HDD 60 GB", "Quantity": 1, "Unit Price": 2000, "Extended Price": 2000}]
Results after update Invoice Line Item (Press a key to continue).
***** Create Receipt *****
Receipt Create Success.
Receipt Create Success.
Receipt Create Success.
Receipt No, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 300, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 100}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-05, Sam, Sam Co., Ltd., Credit Card, Master Card, Citibank, Partially paid on INT100/20, 10000, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8560}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14400}]
RCT100/20, 2020-02-06, CP, CPALL, Debit Card, Debit Card, This will later be deleted, 30, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 10}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 20}]]
Results of creating 3 receipts: RCT100/20, RCT100/20, and RCT100/20 (Press a key to continue).
{'Date': '2020-02-04', 'Customer Code': 'CP', 'Payment Method': 'Cash', 'Payment Reference': 'Nothing', 'Total Received': 30, 'Remarks': 'Paid all invoices partially', 'Receipt Line Item': [{"Invoice No": "INT100/20", "Amount Paid Here": 100}, {"Invoice No": "INT101/20", "Amount Paid Here": 200}], 'Extended Price': 200}
{'Date': '2020-02-05', 'Customer Code': 'CP', 'Payment Method': 'Debit Card', 'Payment Reference': 'Debit Card', 'Total Received': 10, 'Remarks': 'This will later be deleted', 'Receipt Line Item': [{"Invoice No": "INT100/20", "Amount Paid Here": 8560}, {"Invoice No": "INT101/20", "Amount Paid Here": 14400}], 'Extended Price': 14400}
{'Date': '2020-02-06', 'Customer Code': 'CP', 'Payment Method': 'Debit Card', 'Payment Reference': 'Debit Card', 'Total Received': 30, 'Remarks': 'This will later be deleted', 'Receipt Line Item': [{"Invoice No": "INT100/20", "Amount Paid Here": 10}, {"Invoice No": "INT101/20", "Amount Paid Here": 20}], 'Extended Price': 20}
Receipt No 'RCT100/20' not found. Cannot Read.

Receipt No, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 300, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 100}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-05, Sam, Sam Co., Ltd., Credit Card, Master Card, Citibank, Partially paid on INT100/20, 10000, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8560}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14400}]
RCT100/20, 2020-02-06, CP, CPALL, Debit Card, Debit Card, This will later be deleted, 30, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 10}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 20}]]
Results of reading 3 receipts: RCT100/20 (successfully), RCT100/20 (successfully), and RCT100/20 (unsuccessfully) (Press a key to continue).
***** Update Receipt *****
Receipt Update Success.
Receipt No 'RCT100/20' not found. Cannot Update.

Receipt No, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 300, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 100}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-05, Sam, Sam Co., Ltd., Credit Card, Master Card, VISA card, Partially paid on INT100/20 and INT101/20, 9480, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8000}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14000}]
Receipts of updating 2 receipts: RCT100/20 (successfully) and RCT100/20 (unsuccessfully) (Press a key to continue).
***** Delete Receipt *****
Receipt Delete Success.
Receipt No 'RCT100/20' not found. Cannot Delete.

Receipt No, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 300, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 100}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-05, Sam, Sam Co., Ltd., Credit Card, VISA card, Partially paid on INT100/20 and INT101/20, 9480, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8000}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14000}]
Receipts of deleting 2 receipts: RCT100/20 (successfully) and RCT100/20 (unsuccessfully) (Press a key to continue).

Invoice No, Invoice Date, Customer Name, Invoice Amount Due, Invoice Amount Received, Invoice Amount Not Paid
INT100/20, 2020-01-03, Sam Co., Ltd., 14000.0, 8100, 6800.0
INT101/20, 2020-01-04, CPALL, 2140.0, 1000, 540.0
Total Debt: 7500.0
Report unpaid invoice 3 (Press a key to continue).
***** Update receipt line *****
Receipt Line Item Update Success.
Receipt No 'INT100/20' not found. Cannot Update.
Receipt Code 'RCT100/20' not found in Invoice No 'INT100/20'. Cannot Update.

Receipt No, Date, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 400, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 200}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-06, Sam, Sam Co., Ltd., Debit Card, VISA card, Partially paid on INT100/20 and INT101/20, 9480, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8000}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14000}]
Result of updating 3 receipt line, first successfully, others unsuccessfully (Press a key to continue).
***** Delete receipt line *****
Receipt Line Item Delete Success.
Receipt No 'RCT100/20' not found. Cannot Delete.
Receipt Code 'RCT100/20' not found in Invoice No 'INT100/20'. Cannot Delete.

Receipt No, Date, Customer Code, Customer Name, Payment Method, Payment Reference, Remarks, Total Received, Receipt Line Item
RCT100/20, 2020-02-04, CP, CPALL, Cash, Nothing, Paid all invoices partially, 400, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 200}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 200}]
RCT100/20, 2020-02-06, Sam, Sam Co., Ltd., Debit Card, VISA card, Partially paid on INT100/20 and INT101/20, 9480, [{"Invoice No": "INT100/20", "Invoice Date": "2020-01-03", "Amount Paid Here": 8000}, {"Invoice No": "INT101/20", "Invoice Date": "2020-01-04", "Amount Paid Here": 14000}]
Result of deleting 3 receipt line, first successfully, others unsuccessfully (Press a key to continue).

Invoice No, Invoice Date, Customer Name, Invoice Amount Due, Invoice Amount Received, Invoice Amount Not Paid
INT100/20, 2020-01-03, Sam Co., Ltd., 14000.0, 8200, 6700.0
INT101/20, 2020-01-04, CPALL, 2140.0, 1000, 740.0
Total Debt: 7500.0
Normal Termination. Goodbye!