```
In [ ]:                                 Python Project

                                  Grocery Store Checkout App

                     Your friend operates a grocery store and sells the following items:
                         1. Beverages: Chocolate,drinks,coffee,tea,soy drinks,pop and soda
                         2. Phone accessories: Carrying case,earpieces,screen guards
                         3. Toiletries: Toilet paper,Body soap,Scrubs,Body creme,shampoo
                         4. Pastry: Pizza,Burgers,Donuts,Muffins,Cheesecakes
                         5. cosmetics: Perfumes,Vanishes,Nail Polish,Dedorants,Facial Scrubs.

         Your friend wants to be able to record each sale and automatically compute total sale for a customer at
         check out and generate a receipt for the customer.

                     As part of the requirements.

                     1. App should store information about the products by category in the store
                     2. Store and automatically update inventory of each products after sale or restocking
                     3. Raise an alert if any product inventory falls below 5 pieces
                     4. Store information about the purchase cost of each product and the sale price per unit
                     5. Allow the store owner to enter sales per item for each customer and generates a total sales
                        receipt after the sale
                     6. for each customer sales checkout:

                             a. Record sales by item and the sales value
                             b. Show total sales by product
                             c. Show total sales by category
                             d. show total sales for each day
                     TASK:

                     Use your acquired knowledge of python to implement the above requirements.

               Please note that this must be a script and not a GUI application. Use python variables,containers,user
               input functions, functions,condtional statements and loops as necessary.
```

```
In [ ]: # Use Case

        What is the use case for the application:
            it is an app that manges inventory and sales

        for inventory management
            1.Enable restocking
            2.Sales reduction from inventory
```

```
Generate sales reciept for customer
1. Compute sales by item and for total items
2. update inventory and sales records per item.
```

In [ ]:
```
Work Flow of Application

Work flow for inventory:
    1.define and store product categories in a list
    2.dictionary to hold products by category
    3.dictionary for item inventory
```

In [57]:
```
#created a variable name called "prodCats
#to store product categories.
#

prodCats = ['beverages', 'phoneAcces','toiletries','pastry','cosmetics']


#created another variable called 'prodDicts'
# to hold the category and the item that belong to each category.


prodDict = {'beverages':['chocolate','drinks','coffee','tea','soyDrinks','pop','soda'],
             'phoneAcces':['c_Case', 'earPieces','s_Guards'],
             'toiletries':['t_paper','b_soap','scrubs','b_creme','shampoo'],
             'pastry':['Pizza','Burgers','Donuts','Muffins','cheeseCakes'],
             'cosmetics':['perfumes','vanishes','n_polish','Deodorants','f_Scrubs']
            }
```

In [123…
```
#demonstrating how to get the names of the categories
prodDict.keys()
```

Out[1230]:
```
dict_keys(['beverages', 'phoneAcces', 'toiletries', 'pastry', 'cosmetics'])
```

In [123…
```
#demonstrating how to get each product that belongs to each category
prodDict['beverages']
```

Out[1231]:
```
['chocolate', 'drinks', 'coffee', 'tea', 'soyDrinks', 'pop', 'soda']
```

In [123…
```
prodDict['phoneAcces']
```

```
Out[1232]: ['c_Case', 'earPieces', 's_Guards']
```

```
In [123…  prodDict['toiletries']
```

```
Out[1233]: ['t_paper', 'b_soap', 'scrubs', 'b_creme', 'shampoo']
```

```
In [723…  prodDict['pastry']
```

```
Out[723]: ['Pizza', 'Burgers', 'Donuts', 'Muffins', 'cheeseCakes']
```

```
In [724…  prodDict['cosmetics']
```

```
Out[724]: ['perfumes', 'vanishes', 'n_polish', 'Deodorants', 'f_Scrubs']
```

```
In [725…  # demonstrating how to access  any product by just using the index number
          list(prodDict['beverages'])[2]
```

```
Out[725]: 'coffee'
```

```
In [726…  # created a dictionary to hold the product and the quantity.

          bevInventDict = {'chocolate':5,'drinks':5,'coffee':10,'tea':15,'soyDrinks':14,'pop':0,'soda':10}
          pAccInventDict = {'c_Case':20, 'earPieces':100,'s_Guards':45}
          toiInventDict = {'t_paper':67,'b_soap':76,'scrubs':36,'b_creme':98,'shampoo':150}
          pastInventDict = {'Pizza':35,'Burgers':76,'Donuts':150,'Muffins':78,'cheeseCakes':32}
          cosInventDict = {'perfumes':43,'vanishes':46,'n_polish':81,'Deodorants':45,'f_Scrubs':54}
          misInventDict = {}
          inventoryList = {'bevInventDict','pAccInventDict ','toiInventDict','pastInventDict','cosInventDict'}
```

```
In [727…  # code to pick a product from the prodDict and gets its inventory from the corresponding inventory dict
          # call inventory for drinks from the beverage dict.

          prodCheck = 'tea'

          if prodCheck in prodDict['beverages']:
              currInvent = bevInventDict[prodCheck]
              print(currInvent)

          else:
              print(f'{prodCheck}, not in inventory')
```

```
15
```

In [728…

```
def checkinventory(prod):
    '''
   this function will accepts a product name and checks if its available in the inventory list
   if found, it will return the current quantity in inventory
   '''

    if prod in prodDict['beverages']:
        return bevInventDict[prod]
    if prod in prodDict['phoneAcces']:
        return pAccInventDict[prod]
    if prod in prodDict['toiletries']:
        return toiInventDict[prod]
    if prod in prodDict['pastry']:
        return pastInventDict[prod]
    if prod in prodDict['cosmetics']:
        return cosInventDict[prod]


prodCheck = input('enter product')


prodCount = checkinventory(prodCheck)
print(prodCount)
```

```
enter productb_soap
76
```

In [12]:

```
for k,v in prodDict.items():
    print(k,v)
```

```
beverages ['chocolate', 'drinks', 'coffee', 'tea', 'soyDrinks', 'pop', 'soda']
phoneAcces ['c_Case', 'earPieces', 's_Guards']
toiletries ['t_paper', 'b_soap', 'scrubs', 'b_creme', 'shampoo']
pastry ['Pizza', 'Burgers', 'Donuts', 'Muffins', 'cheeseCakes']
cosmetics ['perfumes', 'vanishes', 'n_polish', 'Deodorants', 'f_Scrubs']
```

In [14]:

```
# code to update inventory


# flow:
 #check if product is in any dictionary
# if found, reference the coresponding inventory dictionary and increment quantity by the new amount
```

```
#if not found, prompt user to choose a category to add the product
  #  update the corresponding inventory dictionary with product and inventory
```

In [15]:
```
# code to update inventory #restocking

def inventoryUpdater(prod,qty):
    '''
   this funtion checks,and updates the product inventory and the new quantity
   added to the product.
   '''

    if prod in prodDict['beverages']:
        oldQty = bevInventDict[prod]
        bevInventDict[prod] += qty
        print('update successful')
        return [oldQty,bevInventDict[prod]]



    if prod in prodDict['phoneAcces']:
        oldQty = pAccInventDict[prod]
        pAccInventDict[prod] += qty
        print('update successful')
        return [oldQty,pAccInventDict[prod]]


    if prod in prodDict['toiletries']:
        oldQty = toiInventDict[prod]
        toiInventDict[prod] += qty
        print('update successful')
        return [oldQty,toiInventDict[prod]]

    if prod in prodDict['pastry']:
        oldQty = pastInventDict[prod]
        pastInventDict[prod] += qty
        print('update successful')
        return [oldQty,pastInventDict[prod]]


    if prod in prodDict['cosmetics']:
        oldQty = cosInventDict[prod]
        cosInventDict[prod] += qty
        print('update successful')
        return [oldQty,cosInventDict[prod]]
```

```
newInvent = input('specify item')
inventQty = int (input('specify qty'))

oldQty,newQty = inventoryUpdater(newInvent, inventQty)

if newQty !='...':
    print(f'{newInvent} inventory updated from {oldQty} to {newQty}')
else:
    print('not found in inventory, do you want to add as new product?')


'''bevInventDict
'pAccInventDict
toiInventDict
pastInventDict
cosInventDict'''
```

```
specify itemchocolate
specify qty30
update successful
chocolate inventory updated from 5 to 35
```

Out[15]:

```
"bevInventDict\n'pAccInventDict\ntoiInventDict\npastInventDict\ncosInventDict"
```

In [ ]:

```
# code to update sale inventory


# flow:
1. check if product is in any dictionary
2. if found, reference the coresponding inventory dictionary and reduce quantity by the new amount
3. if not found, prompt user to choose a category to reduce the product
     update the corresponding inventory dictionary with product and inventory
```

In [67]:

```
# code to update sale inventory

def inventoryUpdater(prod,qty):
    '''
   this funtion checks,and updates the remaining item left after sales.
   '''

    if prod in prodDict['beverages']:
        oldQty = bevInventDict[prod]
        bevInventDict[prod] -= qty
        print('update successful')
        return [oldQty,bevInventDict[prod]]



    if prod in prodDict['phoneAcces']:
        oldQty = pAccInventDict[prod]
        pAccInventDict[prod] -= qty
        print('update successful')
        return [oldQty,pAccInventDict[prod]]


    if prod in prodDict['toiletries']:
        oldQty = toiInventDict[prod]
        toiInventDict[prod] -= qty
        print('update successful')
        return [oldQty,toiInventDict[prod]]

    if prod in prodDict['pastry']:
        oldQty = pastInventDict[prod]
        pastInventDict[prod] -= qty
        print('update successful')
        return [oldQty,pastInventDict[prod]]


    if prod in prodDict['cosmetics']:
        oldQty = cosInventDict[prod]
        cosInventDict[prod] -= qty
        print('update successful')
        return [oldQty,cosInventDict[prod]]




newInvent = input('specify item')
inventQty = int (input('specify qty'))

oldQty,newQty = inventoryUpdater(newInvent, inventQty)

if newQty !='...':
    print(f'{newInvent} inventory updated from {oldQty} to {newQty}')
else:
    print('not found in inventory, do you want to add as new product?')
```

```
'''bevInventDict
'pAccInventDict
toiInventDict
pastInventDict
cosInventDict'''
```

```
specify itemBurgers
specify qty16
update successful
Burgers inventory updated from 76 to 60
```

Out[67]:
```
"bevInventDict\n'pAccInventDict\ntoiInventDict\npastInventDict\ncosInventDict"
```

In [ ]:
```
# code to enter sales per item for each customer and generate a total sales recipt.
# created a dictionary to store the item and the sale price.
```

In [122…
```
bevSpDict = {'chocolate':5,'drinks':5,'coffee':10,'tea':15,'soyDrinks':14,'pop':0,'soda':10}
pAccSpDict = {'c_Case':20, 'earPieces':100,'s_Guards':45}
toiSpDict = {'t_paper':67,'b_soap':76,'scrubs':36,'b_creme':98,'shampoo':150}
pastSpDict = {'Pizza':35,'Burgers':76,'Donuts':150,'Muffins':78,'cheeseCakes':32}
cosSpDict = {'perfumes':43,'vanishes':46,'n_polish':81,'Deodorants':45,'f_Scrubs':54}
```

In [2]:
```
# this function basically returns sales
# this function takes the product and qty.

def salesCalc(prod, qty):

    sp = ''
    tSales = ''


    if prod in prodDict['beverages']:
        sp = bevSpDict[prod]
        tSales = sp * qty

    if prod in prodDict['phoneAcces']:
        sp= pAccSpDict[prod]
        tSales = sp * qty

    if prod in prodDict['toiletries']:
        sp = toiSpDict[prod]
        tSales = sp * qty
    if prod in prodDict['pastry']:
        sp = pastSpDict[prod]
        tSales = sp * qty

    if prod in prodDict['cosmetics']:
        sp = cosSpDict[prod]
        tSales = sp * qty
    return [sp,tSales]





def salesFunc():

    itemList = input('list all items seperated by comma')
    itemQty = input('list each item quantity seperated by comma')
    salesDict = dict()
    prods = itemList.split(',')
    qtyList = itemQty.split(',')
    qtys = []
    for qty in qtyList:
        qtys.append(float(qty))
    print(prods)
    print(qtys)
    for p  ,q in zip(prods,qtys):
            print(p,q)
            pTsales = salesCalc(p, q)
            print(pTsales)
            unitPrice =pTsales[0]
            totalSale = pTsales[1]
            salesDict[p] = [p, q,unitPrice,totalSale]
            #salesDict[p] = [q,totalSale]
    return  salesDict
```

In [132…

```
salesFunc()
```

```
list all items seperated by commatea,coffee,b_soap,perfumes,vanishes
list each item quantity seperated by comma3,5,2,4,6,7
['tea', 'coffee', 'b_soap', 'perfumes', 'vanishes']
[3.0, 5.0, 2.0, 4.0, 6.0, 7.0]
tea 3.0
[15, 45.0]
coffee 5.0
[10, 50.0]
b_soap 2.0
[76, 152.0]
perfumes 4.0
[43, 172.0]
vanishes 6.0
[46, 276.0]
{'tea': ['tea', 3.0, 15, 45.0], 'coffee': ['coffee', 5.0, 10, 50.0], 'b_soap': ['b_soap', 2.0, 76, 152.0], 'perfumes':
['perfumes', 4.0, 43, 172.0], 'vanishes': ['vanishes', 6.0, 46, 276.0]}
```

In [11]:

```
import pandas as pd

sales = salesFunc()

salesList = list(sales.values())
df = pd.DataFrame(salesList, columns = ['Item','Quantity','Selling Price','Total Sale'])
print(df)
total = sum(df['Total Sale'])
dfArr = df.values.tolist()
dfArr.append(['Total','','',total])

df = pd.DataFrame(dfArr,columns = ['Item','Quantity','Selling Price','Total Sale'])
df
```

```
list all items seperated by commacoffee,tea,b_soap,perfumes,Pizza,vanishes
list each item quantity seperated by comma2,4,6,11,8,10
['coffee', 'tea', 'b_soap', 'perfumes', 'Pizza', 'vanishes']
[2.0, 4.0, 6.0, 11.0, 8.0, 10.0]
coffee 2.0
[10, 20.0]
tea 4.0
[15, 60.0]
b_soap 6.0
[76, 456.0]
perfumes 11.0
[43, 473.0]
Pizza 8.0
[35, 280.0]
vanishes 10.0
[46, 460.0]
       Item  Quantity  Selling Price  Total Sale
0    coffee       2.0             10        20.0
1       tea       4.0             15        60.0
2    b_soap       6.0             76       456.0
3  perfumes      11.0             43       473.0
4     Pizza       8.0             35       280.0
5  vanishes      10.0             46       460.0
```

Out[11]:

	Item	Quantity	Selling Price	Total Sale
0	coffee	2.0	10	20.0
1	tea	4.0	15	60.0
2	b_soap	6.0	76	456.0
3	perfumes	11.0	43	473.0
4	Pizza	8.0	35	280.0
5	vanishes	10.0	46	460.0
6	Total			1749.0

In [55]:

```
# code to raise an alert when product inventory falls below 5.
# created a new dictionary called  BevAlertDict for each product category
bevAlertDict = {'chocolate':0,'drinks':1,'coffee':10,'tea':15,'soyDrinks':14,'pop':0,'soda':10}
pAccAlertDict = {'c_Case':45, 'earPieces':0,'s_Guards':45}
toiAlertDict = {'t_paper':0,'b_soap':76,'scrubs':2,'b_creme':98,'shampoo':150}
pastAlertDict = {'Pizza':35,'Burgers':6,'Donuts':0,'Muffins':78,'cheeseCakes':2}
cosAlertDict = {'perfumes':43,'vanishes':3,'n_polish':1,'Deodorants':45,'f_Scrubs':54}
```

In [65]:
```
def inventoryAlert(prod):
    '''
   this function will accepts a product name and check if its available in the inventory list.
   if found, it will return the current quantity in inventory, if the quantity is less than or below
   5, it will alert.
   '''

    if prod in prodDict['beverages']:
        if  bevAlertDict[prod] < 5:
            print ("ALERT!!!! the stock is running low")
        return bevAlertDict[prod]
    elif prod in prodDict['phoneAcces']:
        if  pAccAlertDict[prod] < 5:
            print ("ALERT!!!! the stock is running low")
        return pAccAlertDict[prod]
    elif prod in prodDict['toiletries']:
        if  toiAlertDict[prod] < 5:
            print ("ALERT!!!! the stock is running low")
        return toiAlertDict[prod]
    elif prod in prodDict['pastry']:
        if  pastAlertDict[prod] < 5:
            print ("ALERT!!!! the stock is running low")
        return pastAlertDict[prod]
    elif prod in prodDict['cosmetics']:
        if  cosAlertDict[prod] < 5:
            print ("ALERT!!!! the stock is running low")
        return cosAlertDict[prod]
   # else:
        #return "Product exist"

prodCheck = input('enter product')


prodCount = inventoryAlert(prodCheck)
print(prodCount ,  " items of this product is in stock")
```

```
enter productvanishes
ALERT!!!! the stock is running low
3  items of this product is in stock
```

In [122…
```
# code to store  information about the purchase cost of each product
# and the sale price.
# i will be using dictionary to store product
# pCost means purchase cost
# qty means quantity
# sP means sale price


#App should store information about the products by category in the store
prodDict = {
            'beverages':
                        [
                            {
                                'item':'chocolate',
                                'qty':5,
                                'pCost':10,
                                'sP':12
                            },
                            {
                                'item':'drinks',
                                'qty':5,
                                'pCost':10,
                                'sP':12
                            },
                            {
                                'item':'coffee',
                                'qty':10,
                                'pCost':8,
                                'sP':10
                            },
                            {
                                'item':'tea',
                                'qty':15,
                                'pCost':5,
                                'sP':7
                            },
                            {
                                'item':'soyDrinks',
                                'qty':14,
                                'pCost':6,
                                'sP':8
                            },
                            {
                                'item':'pop',
                                'qty':0,
                                'pCost':10,
                                'sP':13
                            },
                            {
                                'item':'soda',
                                'qty':10,
                                'pCost':9,
                                'sP':12
                            }
                        ],
            'phoneAcces':
                        [
                            {
                                'item':'c_Case',
                                'qty':20,
                                'pCost':9,
                                'sP':15
                            },
                            {
                                'item':'earPieces',
                                'qty':100,
                                'pCost':11,
                                'sP':17
                            },
                            {
                                'item':'s_Guards',
                                'qty':45,
                                'pCost':9,
                                'sP':15
                            },
                        ],
            'toiletries':
                        [
                            {
                                'item':'t_paper',
                                'qty':67,
                                'pCost':9,
                                'sP':15
                            },
                            {
                                'item':'b_soap',
                                'qty':76,
                                'pCost':10,
                                'sP':13
                            },
                            {
                                'item':'scrubs',
                                'qty':36,
                                'pCost':13,
                                'sP':25
                            },
                            {
                                'item':'b_creme',
                                'qty':98,
                                'pCost':12,
                                'sP':16
                            },
                            {
                                'item':'shampoo',
                                'qty':150,
                                'pCost':11,
                                'sP':15
                            },
                        ],
            'pastry':
                        [
                            {
                                'item':'Pizza',
                                'qty':35,
                                'pCost':11,
                                'sP':17
                            },
                            {
                                'item':'Burgers',
                                'qty':76,
                                'pCost':5,
                                'sP':10
                            },
                            {
                                'item':'Donuts',
                                'qty':150,
                                'pCost':4,
                                'sP':8
                            },
                            {
                                'item':'Muffins',
                                'qty':78,
                                'pCost':11,
                                'sP':15
                            },
                            {
                                'item':'cheeseCakes',
                                'qty':32,
                                'pCost':7,
                                'sP':15
                            },
                        ],
                'cosmetics':
                        [
                            {
                                'item':'perfumes',
                                'qty':43,
                                'pCost':20,
                                'sP':29
                            },
                            {
                                'item':'vanishes',
                                'qty':46,
                                'pCost':19,
                                'sP':23
                            },
                            {
                                'item':'n_polish',
                                'qty':81,
                                'pCost':10,
                                'sP':15
                            },
                            {
                                'item':'Deodorants',
                                'qty':45,
                                'pCost':16,
                                'sP':19
                            },
                            {
                                'item':'f_Scrubs',
                                'qty':54,
                                'pCost':13,
                                'sP':15
                            },
                        ]
            }
```

In [122…

```
#demonstrating how to get each product,purchase cost and sale price per unit that belongs to each category
prodDict['cosmetics']
```

```
Out[1227]: [{'item': 'perfumes', 'qty': 43, 'pCost': 20, 'sP': 29},
            {'item': 'vanishes', 'qty': 46, 'pCost': 19, 'sP': 23},
            {'item': 'n_polish', 'qty': 81, 'pCost': 10, 'sP': 15},
            {'item': 'Deodorants', 'qty': 45, 'pCost': 16, 'sP': 19},
            {'item': 'f_Scrubs', 'qty': 54, 'pCost': 13, 'sP': 15}]
```

```
In [16]: salesRecord = []
```

```
In [26]: # for each customer sales checkout:

         # a. Record sales by item and the sales value
         # b. Show total sales by product
         # c. Show total sales by category
         # d. show total sales for each day


         def salesCheckOut():
             prodType = input('specify product type')
             qty = input('specify quantity sold')
             prodCat = input('specify product category')
             price = input('specify unit price')
             salesDate = input('specify date')


             salesRec = []
             totalSale =  int(qty)*float(price)
             salesRec = [prodType,prodCat,salesDate,qty,totalSale]
             return salesRec
```

```
In [27]: result = salesCheckOut()
         salesRecord.append(result)    # this holds my result no matter how many it is.

         specify product typevanishes
         specify quantity sold12
         specify product categorycosmetics
         specify unit price23
         specify dateNov 8,2022
```

```
In [19]: result

Out[19]: ['tea', 'beverages', 'Dec 5,2022', '8', 56.0]
```

```
In [28]: salesRecord
```

```
Out[28]: [['tea', 'beverages', 'Dec 5,2022', '8', 56.0],
          ['c_Case', 'phoneAcces', 'Nov 2,2022', '10', 150.0],
          ['b_soap', 'toiletries', 'Dec 7,2022', '7', 91.0],
          ['pizza', 'pastry', 'Dec 11,2022', '50', 850.0],
          ['vanishes', 'cosmetics', 'Nov 8,2022', '12', 276.0]]
```

```
In [52]: import pandas as pd
         def salesCheckOut(salesrecord):
             emptytable = pd.DataFrame()
             prodTypeList = []
             qtyList = []
             prodCatList = []
             totalSaleList = []
             saleDateList = []
             for row in salesrecord:


                 prodTypeList.append(row[0])
                 qtyList.append(row[3])
                 prodCatList.append(row[1])
                 totalSaleList.append(row[4])
                 saleDateList.append(row[2])


             emptytable['Type'] = prodTypeList
             emptytable['qty'] = qtyList
             emptytable['category'] = prodCatList
             emptytable['sale value'] = totalSaleList
             emptytable['Date'] = saleDateList

             return emptytable
```

```
In [53]: salesCheckOut = salesCheckOut(salesRecord)
```

```
In [54]: salesCheckOut
```

Out[54]:

	Type	qty	category	sale value	Date
0	tea	8	beverages	56.0	Dec 5,2022
1	c_Case	10	phoneAcces	150.0	Nov 2,2022
2	b_soap	7	toiletries	91.0	Dec 7,2022
3	pizza	50	pastry	850.0	Dec 11,2022
4	vanishes	12	cosmetics	276.0	Nov 8,2022

In [ ]: