Exercise 1. Design an interactive product management system using Python function. (60 points) In the stock.txt file, there are product names and their corresponding stock numbers.

1. Import the stock.txt file and save the content into a dictionary named stock. (10pts) Example: {'coke': 10, 'milk': 13, 'juice': 5}
2. Write a function to generate overall stats for all products (10pts) Function name: gen_stats Parameters: stock dictionary Return: None Description: Print all basic stats of all products, such as the total amount for all products, the average, the maximum and the minimum.
3. Write a function to calculate the amount of a given product (20pts) Function name: check_stock Parameters: Stock dictionary, product name Return: The number in stock for that product. Description: Retrieve the corresponding number in stock for a given product. The given product is obtained from the user (use input function). Your program should be able to handle the invalid case where the input is not in stock. Sample Run 1 Enter a product name: coke The number is: 10 Sample Run 2 Enter a product name: orange We don't have this product
4. Write a function to update the amount for a given product (20pts) Function name: update_stock Parameters: The stock dictionary, a given product, the number Return: The updated stock dictionary Description: Update the number in stock for a given product. The given product and the number are obtained from the user (use input function). Your program should be able to handle the invalid case where the input is not in stock. Sample Run 1 Enter a product name and a number: coke, 11 The updated dictionary is: {'coke': 11, 'milk': 13, 'juice': 5, ...} Sample Run 2 Enter a product name: orange, 11 We don't have this product

In [5]:

```python
def import_stock(file):
    stock = {}
    with open(file, 'r') as f:
        for line in f:
            name, quantity = line.strip().split(',')
            stock[name] = int(quantity)
    return stock

def gen_stats(stock):
    total = sum(stock.values())
    average = total / len(stock)
    maximum = max(stock.values())
    minimum = min(stock.values())
    print("Total amount: ", total)
    print("Average: ", average)
    print("Maximum: ", maximum)
    print("Minimum: ", minimum)

def check_stock(stock, product):
    if product in stock:
        print("The number is: ", stock[product])
    else:
        print("We don't have this product")

def update_stock(stock, product, quantity):
    if product in stock:
        stock[product] = quantity
        print("The updated dictionary is: ", stock)
    else:
        print("We don't have this product")

stock = import_stock("stocks.txt")

while True:
    print("1. Generate overall stats for all products")
    print("2. Calculate the amount of a given product")
    print("3. Update the amount for a given product")
    print("4. Quit")
    choice = input("Enter your choice (1-4): ")

    if choice == '1':
        gen_stats(stock)
    elif choice == '2':
        product = input("Enter a product name: ")
        check_stock(stock, product)
    elif choice == '3':
        product, quantity = input("Enter a product name and a number: ").split('
        update_stock(stock, product, int(quantity))
    elif choice == '4':
        break
    else:
        print("Invalid choice. Try again.")
```

```
1. Generate overall stats for all products
2. Calculate the amount of a given product
3. Update the amount for a given product
4. Quit
Total amount:  28
Average:  9.333333333333334
Maximum:  13
Minimum:  5
1. Generate overall stats for all products
2. Calculate the amount of a given product
3. Update the amount for a given product
4. Quit
The number is:  10
1. Generate overall stats for all products
2. Calculate the amount of a given product
3. Update the amount for a given product
4. Quit
The updated dictionary is:  {'coke': 20, 'juice': 5, 'milk': 13}
1. Generate overall stats for all products
2. Calculate the amount of a given product
3. Update the amount for a given product
4. Quit
```

Exercise 2. Top-selling products (40 points) Read the transaction data: retail.txt. Each row in the file is one purchase transaction from a customer. The first column is transaction ID, column 2-3 are the product IDs purchased in this transaction. Columns are separated by spaces. Co-purchased products are defined as a pair of products purchased in the same transaction. For example, a row is: "3 26 28", Then 26 and 28 is a pair of co-purchased product IDs. It is worth mentioning that if the value in third column is zero, it means this customer only purchased one product (the one in second column). Task: Use Python programming to find top 10 most popular products and top 10 most commonly copurchased product pairs, then save the result into a new file “output.txt”. Hint: To find co-purchased product in each transaction, you might use a nested loop.

In [16]:

```
valArr = []
counters = {}

with open('retail.txt', 'r') as f:
    data = f.readlines()

for line in data:
    line = line.strip().split()
    products = line[0:]
    valArr.append(" ".join(products))


def updateCnt(key):
    if key in counters:
        counters[key] += 1
    else:
        counters[key] = 1

def process(prodId1, prodId2):
    if int(prodId1) != 0:
        updateCnt(prodId1)

    if int(prodId2) != 0:
        updateCnt(prodId2)

    if int(prodId1) != 0 and int(prodId2) != 0:
        updateCnt("_".join([prodId1, prodId2]))


for i in range(len(valArr)):
    currEl = valArr[i]
    if(len(currEl) > 0):
        parsedArr = currEl.split(' ')
        id, prod1, prod2 = parsedArr
        process(prod1, prod2)

sortedByValsDESC = sorted(counters.items(), key = lambda x:x[1], reverse = True)

first10ns=[[], []]
for i in range(len(sortedByValsDESC)):
    key, counter = sortedByValsDESC[i]
    if len(key) <= 2:
        if(len(first10ns[0]) <= 9):
            first10ns[0].append((key, counter))
    else:
        if(len(first10ns[1]) <= 9):
            first10ns[1].append((key, counter))

with open('output.txt', 'w') as f:
    f.write('Top 10 most popular products:\n')
    for product in first10ns[0]:
        f.write('{}\t{}\n'.format(product[0], product[1]))
    f.write('\nTop 10 most commonly copurchased product pairs:\n')
    for product_pair in first10ns[1]:
        f.write('{}\t{}\n'.format(" and ".join(product_pair[0].split('_')), prod
```