TEST 3 programing part

Q1)

Question # 1 A hotel salesperson enters sales in a text file. Each line contains the name of the client and the amount of the sale separated by semicolon. Write a Python program that reads such a file and displays the total and the average sales on the screen.

- You may assume that the name of the input text file is `hotelsales.txt` and the records of the file are formatted correctly.
- if the input file does not exit, display the following message `input file is not found` .
- Sample run is included below for the input file `hotelsales.txt`

Content of the input file

```
Ahmed Alnasir;200
Hattan Alzahrani;520
Abdullah Alghamdi;800
Mohammed Alzahrani;750
Ali Alhasan;420
Mustaf Amer;600
Naser Alsamir;220
```

The output should be as follows

```
The total sales is  3510
The average sales is 501.43
```

```

In [2]:
```
# This function is used to generate the input file
def generateSalesFile():
    try:
        itemNames=['Ahmed Alnasir;','Hattan Alzahrani;','Abdullah Alghamdi;','Mohammed Alzahrani;','Ali Alhasan;',
                   'Mustaf Amer;','Naser Alsamir;']
        itemQuantity=['200\n','520\n','800\n','750\n','420\n','600\n','220']
        outfile = open("hotelsales.txt","w")
        for i in range(len(itemNames)):
            outfile.write(itemNames[i]+itemQuantity[i])
        outfile.close()
    except Exception as e:
        print(e)
    finally:
        outfile.close()
generateSalesFile()
```

In [3]:
```
%%code q01
# YOUR CODE HERE
try:
    infile = open("hotelsales.txt","r")
    SUM = 0
    count = 0
    for line in infile:
        line = line.rstrip()
        mylist = line.split(";")
        SUM = SUM + int(mylist[1])
        count = count + 1
    avg = SUM / count
    print("The total sales is ", SUM)
    print("The average sales is %.2f" % avg)

except FileNotFoundError:
    print("input file is not found")
```

```
The total sales is  3510
The average sales is 501.43
```

In [4]:
```
#Test Cases
flage=False
try:
    from unittest.mock import patch, mock_open
    with patch("__main__.input") as mock_input:
         with patch("__main__.open", mock_open(read_data="Ahmed Alnasir;200\nHattan Alzahrani;520\nAbdullah Alghamdi;800\n\
                                                          Mohammed Alzahrani;750\nAli Alhasan;420\nMustaf Amer;600\n\
                                                          Naser Alsamir;220\n"),

                    create=True) as mock_file:
                mock_input.side_effect=['input','output']
                ip.run_cell_magic('capture','q01_out',_store['q01'])
                out=str(q01_out).lower()
                listout=['total','sales','3510','average' ,'sales','501.43']
                for a in listout:
                    assert a in out , out
except Exception  as e:
    flage=True
    print(e)
if not flage:
    print("Everything passed")
```

```
Everything passed
```
```

Q2)

Question # 2 Write a Python function that merges two lists **a** and **b** into a **new list c** and returns it, where list **c** consists of alternating elements starting from the first element of list **a** and from the last element of list **b**. If one list is shorter than the other one, then alternate as long as there are elements, and then append the remaining elements from the longer list, in the same direction.

You should **only** provide the following function: `def merge(list_a,list_b):`

- The following are sample runs of the program:

Sample Run 1

```
List a Elements:
Enter an element or press enter to finish: 1
Enter an element or press enter to finish: 3
Enter an element or press enter to finish: 5
Enter an element or press enter to finish: 7
Enter an element or press enter to finish: 9
Enter an element or press enter to finish:
List b Elements:
Enter an element or press enter to finish: 10
Enter an element or press enter to finish: 40
Enter an element or press enter to finish: 50
Enter an element or press enter to finish:
List a:  ['1', '3', '5', '7', '9']
List b:  ['10', '40', '50']
List c:  ['1', '50', '3', '40', '5', '10', '7', '9']
```

Sample Run 2

```
List a Elements:
Enter an element or press enter to finish:
List b Elements:
Enter an element or press enter to finish: 20
Enter an element or press enter to finish: 40
Enter an element or press enter to finish: 60
Enter an element or press enter to finish:
List a:  []
List b:  ['20', '40', '60']
List c:  ['60', '40', '20']
```

In [15]:

```
%%code q02

def readElements(myList):
    element = input("Enter an element or press enter to finish: ")
    while element !="":
        myList.append(element)
        element = input("Enter an element or press enter to finish: ")

def merge(list_a,list_b):
    # YOUR CODE HERE
    list_c = []
    fakeA = list(list_a)
    fakeB = list(list_b)
    while len(fakeA)!= 0 or len(fakeB) != 0 :
        if len(fakeA) != 0:
            list_c.append(fakeA.pop(0))
        if len(fakeB) != 0:
            list_c.append(fakeB.pop())
    return list_c
def main():
    list_a = []
    list_b = []
    print("List a Elements:")
    readElements(list_a)
    print("List b Elements:")
    readElements(list_b)
    list_c = merge(list_a,list_b)
    print("List a: ", list_a)
    print("List b: ", list_b)
    print("List c: ", list_c)
#Don't forget to uncomment the function main below.
main()
```

```
List a Elements:
Enter an element or press enter to finish: 20
Enter an element or press enter to finish: 50
Enter an element or press enter to finish:
List b Elements:
Enter an element or press enter to finish:
List a:  ['20', '50']
List b:  []
List c:  ['20', '50']
```

In [16]:

```
# #Test Cases
flage=False
import unittest
test_cases={'1':[['1', '3', '5', '7', '9'],['10', '40', '50'],['1', '50', '3', '40', '5', '10', '7', '9']],
            '2':[['1', '4', '6'],['20', '40', '60', '70', '90'],['1', '90', '4', '70', '6', '60', '40', '20']],
            '3':[['1', '3', '6'],['20', '50', '80'],['1', '80', '3', '50', '6', '20']],
            '4':[[],['20', '40', '60'],['60', '40', '20']]}
for t in test_cases:
    try:
        list_c=merge(test_cases[t][0],test_cases[t][1])

        assert all(a in list_c for a in test_cases[t][2] ),list_c

    except:
        flage=True
        print("FAILED: merge function should return %s not"% test_cases[t][2],list_c)
if not flage:
    print("Everything passed")
```

```
Everything passed
```

Q3)

Question # 3 Develop code for a Python function `compAve` that will return two values: the average of house value prices stored in a list format, and the number of values in that list. You can assume that the list can be of any length greater than or equal to 2. You can also assume that all input is entered correctly.

You should **only** provide the following function: `def compAve(values):`

- The following are sample runs of the program:
- **Sample Run 1**:
  Enter a float value or press enter to finish: 1.5
  Enter a float value or press enter to finish: 0.9
  Enter a float value or press enter to finish: 1.0
  Enter a float value or press enter to finish: 2.0
  Enter a float value or press enter to finish: 0.8
  Enter a float value or press enter to finish: 2.5
  Enter a float value or press enter to finish:
  The average of house value prices is 1.45 and the number of houses equals 6

- **Sample Run 2**:
  Enter a float value or press enter to finish: 1.0
  Enter a float value or press enter to finish: 0.5
  Enter a float value or press enter to finish:
  The average of house value prices is 0.75 and the number of houses equals 2

In [24]:
```
#don't modify the content of this cell just run it
from IPython.core.magic import (register_line_magic,
                                register_cell_magic)
_store = {}
ip = get_ipython()
@register_cell_magic
def code(line, cell):
    _store[line.strip()]=cell
    ip.run_cell(cell)
```

In [25]:
```
%%code q03

'''
This function reads the values in a list and returns it.
'''
def readValues() :
    myList =[]
    value = input("Enter a float value or press enter to finish: ")
    while value !="":
        myList.append(float(value))
        value = input("Enter a float value or press enter to finish: ")
    return (myList)

def compAve(values) :
    # YOUR CODE HERE
    con = len(values)
    total = sum(values)
    AVG = total / con
    return [AVG , con]


def main():
    values = readValues()
    avg = 0.0             # Initialization of the average of house prices to zero.
    count = 0             # Initialization of the number of entered house prices.

    # Enter here your call to the function compAve. Store the average of house values
    # in Variable avg and the number of entered house prices in Variable count and print the results as shown above

    # YOUR CODE HERE
    data = compAve(values)
    avg = data[0]
    count = data[1]
    print("The average of house value prices is %.2f and the number of houses equals %d"% (avg , count))
#Don't forget to uncomment the function main below.
main()
```

```
Enter a float value or press enter to finish: 4
Enter a float value or press enter to finish: 5
Enter a float value or press enter to finish:
The average of house value prices is 4.50 and the number of houses equals 2
```

:
```
##Test Cases
flage=False
import unittest
test_cases={'1':[[1.50, 0.90, 1.0, 2.0, 0.8, 2.5, 3.0, 3.5, 0.25, 0.81],1.626,10],
            '2':[[1.0, 0.5],0.75,2],
            '3':[[1.50, 0.90, 1.0, 2.0, 0.8, 2.50] ,1.45,6]}
for t in test_cases:
    try:
        test_avg, test_count=compAve(test_cases[t][0])
        assert test_avg==test_cases[t][1] and  test_count == test_cases[t][2] ,''
    except:
        flage=True
        print("FAILED: compAve function should return %f, %d not"%(test_cases[t][1],test_cases[t][2]),test_avg,
              ', ',test_count)
if not flage:
    print("Everything passed")

```

```
Everything passed
```