Maheshvaran S

DS205229119

Lab5. List Processing in Python

Question1:Write a function find_average(student) that takes student tuples as input and print student rollno, name, marks and average marks as output.

- Test Cases:
  1.Stud1=(1,"rex",60,85,70)
  find_average(stud1)

Modify the above function find_average(student) so that it processes a tuple of tuples.

2.Stud2=(2,"rex",(80,75,90))
find_average(stud2)

In [3]:
```
def find_average(student):
    roll,name,marks = student
    total = 0
    for mark in marks:
        total += mark
    avg = total / len(marks)
    print("Roll No:",roll,"Name:",name,"Average:",avg)
```

In [4]:
```
stud1 = (1,"rex",(60,85,70))
```

In [5]:
```
find_average(stud1)
```

```
Roll No: 1 Name: rex Average: 71.66666666666667
```

In [6]:
```
stud2 = (2,"Rex",(80,75,90))
```

In [7]:
```
find_average(stud2)
```

```
Roll No: 2 Name: Rex Average: 81.66666666666667
```

Question2:Write a weight management program that prompts that prompts the user to enter in 7 days of their body weight values as float numbers. Store them in list. Then print first day weight, last day weight, 4th day weight, highest weight, lowest weight and average weight.

Finally, print if average weight < lowest weight, then print "Your weight management is excellent". Otherwise print "Your weight management is not good. Please take care of your diet".

In [5]:
```
list=[]
for i in range(7):
    c=float(input("Enter your weight:"))
    list.append(c)
print("\n\nFirst day weight",list[0])
print("\nLast day weight",list[-1])
print("\nHighest weight is",max(list))
```

```
print("\nLowest weight is",min(list))
print("\nAverage of weight is",round(sum(list)/len(list)))
Firstday=list[0]
Lastday=list[-1]
avg=round(sum(list)/len(list))
low_weight=min(list)
if(avg<low_weight):
    print("\nYour weight management is excellent")
else:
    print("\nYour weight management is not good. Please take care of your diet")
```

```
Enter your weight:70
Enter your weight:71
Enter your weight:72
Enter your weight:73
Enter your weight:74
Enter your weight:75
Enter your weight:76


First day weight 70.0

Last day weight 76.0

Highest weight is 76.0

Lowest weight is 70.0

Average of weight is 73

Your weight management is not good. Please take care of your diet
```

QUESTION 3: Write a function lastN(lst,n) that takes a list of integers and n and returns n largest numbers.

How many numbers you want to enter?: 6 Enter a number: 12 Enter a number: 32 Enter a number: Enter a number: Enter a number: Enter a number:

How many largest numbers you want to find?: 3 Largest numbers are: 52, 45, 32

In [13]:

```
def lastN(lst, n):
    final_list =[]

    for i in range(n):
        max1 = 0

        for j in lst:
            if j> max1:
                max1 = j;

        lst.remove(max1);
        final_list.append(max1)
    print("\nlargest numbers:",final_list)

b=int(input("How many number want to enter?: "))
lst=[]
for i in range(b):
    c=int(input("\nEnter a number: "))
    lst.append(c)

n=int(input("\nHow many largest numbers you want to find?: "))
lastN(lst,n)
```

```
How many number want to enter?: 6
```

```
Enter a number: 12

Enter a number: 34

Enter a number: 18

Enter a number: 24

Enter a number: 2

Enter a number: 29

How many largest numbers you want to find?: 3

largest numbers: [34, 29, 24]
```

QUESTION 4: Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with "X" first. Hint: this can be done by making 2 lists and sorting each of them before combining them.

Test Cases:

- 1.Input: ['min','xyz','apple','xanadu','aardvark'] Output:['xanadu','xyz','aardvark','apple','mix']
- 2.Input: ['ccc','bbb','aaa','xcc','xaa'] Output:['xaa','xcc','aaa','bbb','ccc']
- 3.Input: ['bbb','ccc','axx','xzz','xaa'] Output: ['xaa','xzz','axx','bbb','ccc']

In [14]:
```
def front_x(words):
    xlist = []
    alist = []

    for word in words:
        if word.startswith("x"):
            xlist.append(word)
        else:
            alist.append(word)
    return sorted(xlist) + sorted(alist)
```

In [15]:
```
words=['min','xyz','apple','xanadu','aardvark']
front_x(words)
```

Out[15]: ['xanadu', 'xyz', 'aardvark', 'apple', 'min']

In [16]:
```
words=['ccc','bbb','aaa','xcc','xaa']
front_x(words)
```

Out[16]: ['xaa', 'xcc', 'aaa', 'bbb', 'ccc']

In [17]:
```
words=['bbb','ccc','axx','xzz','xaa']
front_x(words)
```

Out[17]: ['xaa', 'xzz', 'axx', 'bbb', 'ccc']

QUESTION 5: Develop a function sort_last(). Given a list of non-empty tuples, return a list sorted in increasing order by the last element in each tuple. Hint: use a custom key= function to extract the last element from each tuple.

Test Cases:

- 1. Input:[(1,7),(1,3),(3,4,5),(2,2)] Output:[(2,2),(1,3),(3,4,5),(1,7)]
- 1. Input:[(1,3),(3,2),(2,1)] Output:[(2,1),(3,2),(1,3)]
- 1. Input:[(2,3),(1,2),(3,1)] Output:[(3,1),(1,2),(2,3)]

In [18]:
```
def last(n): return n[-1]

def sort_list_last(tuples):
  return sorted(tuples, key=last)

print("Test Case:1 \n\tInput:[(1,7),(1,3),(3,4,5),(2,2)] \n\tOutput:",sort_list_last

print("Test Case:2 \n\tInput:[(1,3),(3,2),(2,1)] \n\tOutput:",sort_list_last([(1,3),

print("Test Case:3 \n\tInput:[(2,3),(1,2),(3,1)] \n\tOutput:",sort_list_last([(2,3),
```

```
Test Case:1
        Input:[(1,7),(1,3),(3,4,5),(2,2)]
        Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
Test Case:2
        Input:[(1,3),(3,2),(2,1)]
        Output: [(2, 1), (3, 2), (1, 3)]
Test Case:3
        Input:[(2,3),(1,2),(3,1)]
        Output: [(3, 1), (1, 2), (2, 3)]
```

Question6. Other String Functions

- a) Define a function first() that receives a tuple and returns its first element
- b) Define a function sort_first() that receives a list of tuples and returns the sorted
- c) Print lists in sorted order
- d) Define a function middle() that receives a a tuple and returns its middle element
- e) Define a functino sort_middle() that receives a list of tuples and returns it sorted using the key middle
- f) Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order. Output should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]

In [22]:
```
def first(num):
    for j in num:
        a,b,c=j
        print(a)

def sort_first(num):
    new_list=sorted(num)
    return new_list

def lists(num):
    return sorted(num, key=None, reverse=0)


def middle(num):
    for j in num:
        a,b,c=j
        print(b)

def sort_middle(num):
    return sorted(num,key = lambda mid:mid[1])
num=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,40)]
```

In [23]:
```
first(num)
```

```
1
2
10
20
30
```

In [25]:
```
sort_first(num)
```

Out[25]:
```
[(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 40)]
```

In [21]:
```
sort_middle(num)
```

Out[21]:
```
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]
```

In [24]:
```
middle(num)
```

```
2
1
7
4
6
```

QUESTION7: Develop a function remove_adjacent(). Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

- 1.Input:[1,2,2,3] and Output:[1,2,3]
- 2.Input:[2,2,3,3,3,] and Output:[2,3]
- 3.Input:[].Output:[].
- 4.Input:[2,5,5,6,6,7] and Output:[2,5,6,7]
- 5.Input:[6,7,7,8,9,9] and Output:[6,7,8,9]

In [1]:
```
def remove_adjacent(lst):
    a = []
    for item in lst:
        if len(a):
            if a[-1] != item:
                a.append(item)
        else:
            a.append(item)
    return a
```

In [2]:
```
remove_adjacent([1,2,2,3])
```

Out[2]:
```
[1, 2, 3]
```

In [3]:
```
remove_adjacent([2,2,3,3,3])
```

Out[3]:
```
[2, 3]
```

In [4]:
```
remove_adjacent([2,5,5,6,6,7])
```

Out[4]:
```
[2, 5, 6, 7]
```

In [5]:
```
remove_adjacent([6,7,7,8,9,9])
```

Out[5]:
```
[6, 7, 8, 9]
```

In [6]:
```
remove_adjacent([])
```

Out[6]:
```
[]
```

QUESTION8: Write a function verbing(). Given a string, if its length is at least 3, add'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Return the resulting string. So 'hail' yields: hailing, 'swimming' yields: swimmingly; 'do' yields:do.

In [7]:
```
def verbing(str):
    length = len(str)
    if length > 2:
        if str[-3:] == 'ing':
            str += 'ly'
        else:
            str += 'ing'
    return str
```

In [8]:
```
verbing("hail")
```

Out[8]:
```
'hailing'
```

In [9]:
```
verbing("swimming")
```

Out[9]:
```
'swimmingly'
```

In [10]:
```
verbing("do")
```

Out[10]:
```
'do'
```

QUESTION9: Develop a function not_bad(). Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'..'bad' substring with 'good'. Return the resulting string. So 'This dinner is not that bad! 'Yields: This dinner is good!

In [11]:
```
def not_bad(s):
    snot = s.find('not')
    sbad = s.find('bad')
    if sbad>snot:
        s=s.replace(s[snot:(sbad+3)], 'good')
    return s
```

In [12]:
```
not_bad("This dinner is not really that bad!")
```

Out[12]:
```
'This dinner is good!'
```