Problem Solving Using Python and R Lab

Lab5. List Processing in Python

```
# Name:P.Asha Belcilda
Rollno:225229104
```

Question1. Write a function find_average(student) that takes student tuple 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 [4]:
```
def find_average(student):
    rollno,name,marks=student
    total=0
    for i in marks:
        total += i
    average=total/3
    print("Rollno:",rollno,"\nName:",name,"\nAverage:",average)
stud1 = (1,"rex",(60, 85, 70))
find_average(stud1)
```

```
Rollno: 1
Name: rex
Average: 71.66666666666667
```

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

```
Rollno: 2
Name: rex
Average: 81.66666666666667
```

Question2.Write a weight management program 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 [8]:
```
x=[]
a1=float(input("Day 1:"))
a2=float(input("Day 2:"))
a3=float(input("Day 3:"))
a4=float(input("Day 4:"))
a5=float(input("Day 5:"))
a6=float(input("Day 6:"))
a7=float(input("Day 7:"))
x.append(a1)
x.append(a2)
x.append(a3)
x.append(a4)
x.append(a5)
x.append(a6)
x.append(a7)
print("First day weight:",x[0])
print("Last day weight",x[6])
print("Highest weight:",max(x))
print("Lowest weight:",min(x))
print("Average weight:",sum(x)/len(x))
if sum(x)/len(x)<min(x):
    print("Your Weight Management is Excellent")
else:
    print("Your Weight Management is not good.Please take care of your diet")
```

```
Day 1:100
Day 2:98
Day 3:97.3
Day 4:91
Day 5:90
Day 6:89.0
Day 7:87
First day weight: 100.0
Last day weight 87.0
Highest weight: 100.0
Lowest weight: 87.0
Average weight: 93.18571428571428
Your Weight Management is not good.Please take care of your diet
```

Question3. Write a function lastN(lst, n) that takes a list of integers and n and returns nlargest numbers.

How many numbers you want to enter?: 6 Enter a number: 12 Enter a number: 32 Enter a number: 10 Enter a number: 9 Enter a number: 52 Enter a number: 45 How many largest numbers you want to find?: 3 Largest numbers are: 52, 45, 32

In [11]:
```
n=int(input("How many numbers you want to enter?:"))
x=[int(input("Enter a number:"))for i in range(n)]
y=[]
num=int(input("How many largest numbers you want to find?:"))
for a in range(0,num):
    N=0
    for b in range(len(x)):
        if x[b]>N:
            N=x[b]
    x.remove(N)
    y.append(N)
print(num,"Largest number:")
for i in y:
    print(i)
```

```
How many numbers you want to enter?:6
Enter a number:12
Enter a number:32
Enter a number:10
Enter a number:9
Enter a number:52
Enter a number:45
How many largest numbers you want to find?:3
3 Largest number:
52
45
32
```

Question4. 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: ['mix', '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 [25]:
```
def Sort(li):
    m=li
    n=[]
    o=[]
    for i in m:
        if i[0].lower() == "n" :
            n.append(i)
        else:
            o.append(i)
    o.sort(),n.sort()
    return o + n
li1=['mix', 'xyz','apple','xanadu','aardvark']
li2=['ccc','bbb','aaa','xcc','xaa']
li3=['bbb','ccc','axx','xzz','xaa']
print("Input:",li1)
print("Output:",Sort(li1))
print("Input:",li3)
print("Output:",Sort(li3))
```

```
Input: ['mix', 'xyz', 'apple', 'xanadu', 'aardvark']
Output: ['aardvark', 'apple', 'mix', 'xanadu', 'xyz']
Input: ['bbb', 'ccc', 'axx', 'xzz', 'xaa']
Output: ['axx', 'bbb', 'ccc', 'xaa', 'xzz']
```

Question5. 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 form 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)]
2. Input: [(1,3),(3,2),(2,1)] Output: [(2,1),(3,2),(1,3)]
3. Input: [(2,3),(1,2),(3,1)] Output: [(3,1),(1,2),(2,3)]

In [27]:
```
def Sort_last(a):
    b=len(a)
    for i in range(0,b):
        for j in range(0,b-i-1):
            if (a[j][1] > a[j + 1][1]):
                temp = a[j]
                a[j]=a[j+1]
                a[j + 1]=temp
    return a
a1=[(1, 7), (1, 3), (3, 4, 5), (2, 2)]
a2=[(1,3),(3,2),(2,1)]
a3=[(2,3),(1,2),(3,1)]
print("Input:",a1)
print("Output:",Sort_last(a1))
print("Input:",a2)
print("Output:",Sort_last(a2))
print("Input:",a3)
print("Output:",Sort_last(a3))
```

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

In [29]:
```
def first(a):
    print(a[0])
b=(34,100,2,7,78,90,0,98,34,67)
print("First element:")
first(b)
```

```
First element:
34
```

```
b) Define a function sort_first() that receives a list of tuples and returns
the sorted
```

In [30]:
```
def sort_first(a):
    return sorted(a)
b=[(4,1,5),(9,4,3),(1,2,3),(10,23,5)]
print("Sorted List:")
print(sort_first(b))
```

```
Sorted List:
[(1, 2, 3), (4, 1, 5), (9, 4, 3), (10, 23, 5)]
```

```
c) Print lists in sorted order
```

In [31]:
```
def sort_first(a):
    print("Sorted List:",sorted(a))
b=[(4,1,5),(9,4,3),(1,2,3),(10,23,5)]
sort_first(b)
```

```
Sorted List: [(1, 2, 3), (4, 1, 5), (9, 4, 3), (10, 23, 5)]
```

```
d) Define a function middle() that receives a a tuple and returns its middle
element
```

In [32]:
```
def middle(a):
    b=len(a)/2
    print(a[int(b)])
c=(34,100,2,7,78,90,0,98,34,67)
print("Middle element:")
middle(c)
```

```
Middle element:
90
```

```
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 [34]:
```
def Sort_last(a):
    b=len(a)
    for i in range(0,b):
        for j in range(0,b-i-1):
            if (a[j][1] > a[j + 1][1]):
                temp=a[j]
                a[j]=a[j + 1]
                a[j + 1]=temp
    return a
c=[(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)]
print("Input:",c)
print("Output:")
print(Sort_last(c))
```

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

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] Output: [2,5,6,7]
5. Input: [6,7,7,8,9,9] Output: [6,7,8,9]

In [38]:
```
def remove_adjacent(a):
    lst=list(dict.fromkeys(a))
    print("Output:")
    print(lst)
b1=[1, 2, 2, 3]
b2=[2, 2, 3, 3, 3]
b3=[]
b4=[2,5,5,6,6,7]
b5=[6,7,7,8,9,9]
print("Input:",b1)
remove_adjacent(b1)
print("Input:",b2)
remove_adjacent(b2)
print("Input:",b3)
remove_adjacent(b3)
print("Input:",b4)
remove_adjacent(b4)
print("Input:",b5)
remove_adjacent(b5)
```

```
Input: [1, 2, 2, 3]
Output:
[1, 2, 3]
Input: [2, 2, 3, 3, 3]
Output:
[2, 3]
Input: []
Output:
[]
Input: [2, 5, 5, 6, 6, 7]
Output:
[2, 5, 6, 7]
Input: [6, 7, 7, 8, 9, 9]
Output:
[6, 7, 8, 9]
```

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 [40]:
```
def verbing(a):
    b=len(a)
    if b<=2:
        print(a)
    if b>=3:
        if a[-3:] == 'ing':
            a+='ly'
    else:
        a+='ing'
    print(a)
verbing('hail')
verbing('swimming')
verbing('do')
```

```
hail
swimmingly
do
doing
```

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 [42]:
```
def not_bad(str1):
    ab=str1.find('not')
    cd=str1.find('bad')
    if cd>ab and ab>0 and cd>0:
        str1 = str1.replace(str1[ab:(cd+4)],'good')
        return str1
    else:
        return str1
str2='This dinner is not bad!'
print("Input:",str2)
print("Output:",not_bad(str2))
```

```
Input: This dinner is not bad!
Output: This dinner is good
```