List Processing in Python

In [ ]:
```
#NAME : ANNAPOORNIMA
#ROLL NO: 225229101
'''Question1. Write a function find_average(student) that takes student tuple as
print student rollno, name, marks and average marks as output.
Test Cases:
1. stud1 = (1, “rex”, 60, 85, 70)
find_average(stud1)'''
```

In [6]:
```
print("case : 1")
print()
def find_averge(s):
    print("Roll No. : ",s[0])
    print("Name : ",s[1])
    print("Mark1 : ",s[2])
    print("Mark2 : ",s[3])
    print("Mark3 : ",s[4])
    for i in s:
        m=(s[2]+s[3]+s[4])/3
    print("Average Mark :",m)
stud=(1,"Rex",60,85,70)
stud1=tuple(stud)
find_averge(stud1)
```

```
case : 1

Roll No. :  1
Name :  Rex
Mark1 :  60
Mark2 :  85
Mark3 :  70
Average Mark : 71.66666666666667
```

In [ ]:
```
'''Modify the above function find_average(student) so that it processes a tuple c
2. stud2 = (2, “rex”, (80, 75, 90))
find_average(stud2)'''
```

In [8]:
```
print("case : 2")
print()

def find_averge(s):
    print("Roll No. : ",s[0])
    print("Name : ",s[1])
    print("Mark1 : ",s[2][0])
    print("Mark2 : ",s[2][1])
    print("Mark3 : ",s[2][2])
    for i in s:
        m=(s[2][0]+s[2][1]+s[2][2])/3
    print("Average Mark :",m)
#main:
stud=(2,"Rex",(80,75,90))
stud2=tuple(stud)
find_averge(stud2)
```

```
case : 2

Roll No. :  2
Name :  Rex
Mark1 :  80
Mark2 :  75
Mark3 :  90
Average Mark : 81.66666666666667
```

In [ ]:
```
'''Question2.
Write a weight management program that prompts the user to enter in 7 days of the
weight values as float numbers. Store them in list. Then print first day weight,
weight, 4th day weight, highest weight, lowest weight and average weight. Finally
average weight < lowest weight, then print “Your weight management is excellent”.
Otherwise print “Your weight management is not good. Please take care of your die
```

In [10]:
```
a=[]
d1=float(input("Day 1 : "))
d2=float(input("Day 2 : "))
d3=float(input("Day 3 : "))
d4=float(input("Day 4 : "))
d5=float(input("Day 5 : "))
d6=float(input("Day 6 : "))
d7=float(input("Day 7 : "))
a.append(d1)
a.append(d2)
a.append(d3)
a.append(d4)
a.append(d5)
a.append(d6)
a.append(d7)
print("First Day Weight : ",a[0])
print("4th Day Weight : ",a[6])
print("late Day Weigth : ",a[6])
print("Highest Weight : ",max(a))
print("Lowest Weight : ",min(a))
print("Averge Weight : ",sum(a)/len(a))
if sum(a)<min(a):
    print("Your Weight Management is Excellent")
else:
    print("Your Weight Management is NOT So Good. Please take Care of your DIET")
```

```
Day 1 : 35
Day 2 : 38
Day 3 : 39
Day 4 : 42
Day 5 : 43
Day 6 : 44
Day 7 : 46
First Day Weight :  35.0
4th Day Weight :  46.0
late Day Weigth :  46.0
Highest Weight :  46.0
Lowest Weight :  35.0
Averge Weight :  41.0
Your Weight Management is NOT So Good. Please take Care of your DIET
```

In [ ]:
```
'''Question3. Write a function lastN(lst, n) that takes a list of integers and n
largest numbers.'''
```

In [14]:
```
def lastN(lst,n):
    lst.sort()
    return lst[-n: ]
li=[]
n=int(input("how many number you want to enter?:"))
for i in range(0,n):
    e=int(input("enter a number: "))
    li.append(e)
n=int(input("how many largest number you want to find?:"))
print(n,"largest number are:")
print(lastN(li,n))
```

```
how many number 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 number you want to find?:3
3 largest number are:
[32, 45, 52]
```

In [ ]:
```
'''Question4. Given a list of strings, return a list with the strings in sorted c
all the strings that begin with 'x' first. Hint: this can be done by making 2 lis
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 [30]:
```
def sort(list):
    l=list
    x=[]
    nox=[]
    for i in l:
        if i[0].lower() == "x":
            x.append(i)
        else:
            nox.append(i)
    x.sort(),nox.sort()
    return x + nox
list1=['mix','xyz','apple','xanadu','aardvark']
list2=['ccc','bbb','aaa','xcc','xaa']
list3=['bbb','ccc','axx','xzz','xaa']
print("Input :",list1)
print("Output :",sort(list1))
print("Input :",list3)
print("Output :",sort(list3))
```

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

In [ ]:
```
'''Question5. Develop a function sort_last(). Given a list of non-empty tuples, r
sorted in increasing order by the last element in each tuple. Hint: use a custom
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 [31]:
```
def sort_last(t):
    l=len(t)
    for i in range(0,1):
        for j in range(0,l-i-1):
            if(t[j][1] > t[j + 1][1]):
                temp = t[j]
                t[j] = t[j + 1]
    return t
tp=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,20)]
print("Input :",tp)
print("Output :")
print(sort_last(tp))
```

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

In [ ]:
```
'''Question6. Other String Functions
a) Define a function first() that receives a tuple and returns its first element
```

In [43]:
```
def first(s):
    print(s[0])
t=(100,2,3,4,5,6,7,8,9,0)
print("The first element of the Tuple :")
first(t)
```

```
The first element of the Tuple :
100
```

In [ ]:
```
'''b) Define a function sort_first() that receives a list of tuples and returns t
```

In [34]:
```
def sort_first(s):
    return sorted(s)
t=[(4,1,5),(9,4,3),(1,2,3),(10,23,5)]
print("sorted list:")
print(sort_first(t))
```

```
sorted list:
[(1, 2, 3), (4, 1, 5), (9, 4, 3), (10, 23, 5)]
```

In [ ]:
```
'''c) Print lists in sorted order'''
```

In [35]:
```
def sort_first(s):
    print("sorted list:",sorted(s))
t=[(4,1,5),(9,3),(1,2),(10,23,5)]
sort_first(t)
```

```
sorted list: [(1, 2), (4, 1, 5), (9, 3), (10, 23, 5)]
```

In [ ]:
```
'''d) Define a function middle() that receives a a tuple and returns its middle e
```

In [36]:
```
def middle(s):
    a=len(s)/2
    print(s[int(a)])
t=(100,21,34,4,500)
print("Middle element of Tuble :")
middle(t)
```

```
Middle element of Tuble :
34
```

In [ ]:
```
'''e) Define a functino sort_middle() that receives a list of tuples and returns
the key middle'''
```

In [ ]:
```
```

In [ ]:
```
'''f) Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sor
should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]'''
```

In [37]:
```
def sort_last(t):
    l=len(t)
    for i in range(0,1):
        for j in range(0,l-i-1):
            if (t[j][1] > t[j + 1][1]):
                temp = t[j]
                t[j]= t[j + 1]
                t[j + 1]=temp
    return t
tp=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,20)]
print("Input :",tp)
print("Output :")
print(sort_last(tp))
```

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

In [ ]:
```
'''Question7. Develop a function remove_adjacent(). Given a list of numbers, retu
where all adjacent same elements have been reduced to a single element. You may c
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 [40]:
```
def remove_adjacent(l):
    li=list(dict.fromkeys(l))
    print("Output :")
    print(li)
l1=[1,2,2,3]
l2=[2,2,3,3,3]
l3=[]
l4=[2,5,5,6,6,7]
l5=[6,7,7,8,9,9]
print("Input :",l1)
remove_adjacent(l1)
print("Input :",l2)
remove_adjacent(l2)
print("Input :",l3)
remove_adjacent(l3)
print("Input :",l4)
remove_adjacent(l4)
print("Input :",l5)
remove_adjacent(l5)
```

```
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]
```

In [ ]:
```
'''Question8. Write a function verbing(). Given a string, if its length is at lea
end. Unless it already ends in 'ing', in which case add 'ly' instead. If the stri
than 3, leave it unchanged. Return the resulting string. So „hail‟ yields: hailir
yields: swimmingly; „do‟ yields: do.'''
```

In [41]:
```
def verbing(w):
    l=len(w)
    if l<=2:
        print(w)
    if l>=3:
        if w[-3:]=='ing':
            w+='ly'
    else:
        w+='ing'
    print(w)
verbing('hall')
verbing('swimming')
verbing('do')
```

```
hall
swimmingly
do
doing
```

In [ ]:
```
'''Question9. Develop a function not_bad(). Given a string, find the first appear
substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not
with 'good'.
Return the resulting string. So 'This dinner is not that bad!' yields: This dinne
```

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

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