SURIYA S 225229140

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

In [4]:
```
 student=(1,'suri',60,85,70)
def roll(student):
    rollno,name,m1,m2,m3=student
    avg=(m1+m2+m3)/3
    print("avgerage is:",avg)
roll(student)
```

```
avgerage is: 71.66666666666667
```

In [5]:
```
 student=(1,'suri',80,90,95)
def roll(student):
    rollno,name,m1,m2,m3=student
    avg=(m1+m2+m3)/3
    print("avgerage is:",avg)
roll(student)
```

```
avgerage is: 88.33333333333333
```

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 [1]:
```
def weight(list):
    print("1st day weight: ",list[0])
    print("last day weight: ",list[-1])
    print("4th day weight: ",list[3])
    print("heighest weight: ",max(list))
    print("lowest weight: ",min(list))
    tot=0
    for i in range (0,7):
        tot=tot+list[i]
        avg=tot/7
    print(avg)
    if avg<min(list):
        print("Your weight management is excellent")
    else:
        print("Your weight management is not good.Please take care of your health

print("Enter 7 days weight: ")
list=[]
for j in range(0,7):
    ele=float(input())
    list.append(ele)
weight(list)
```

```
Enter 7 days weight:
55.3
56.8
58
60.4
62.8
63.2
64.5
1st day weight:  55.3
last day weight:  64.5
4th day weight:  60.4
heighest weight:  64.5
lowest weight:  55.3
60.142857142857146
Your weight management is not good.Please take care of your health
```

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

In [5]:
```
def lastN(lst, n):

    final_list = []

    for i in range(0, n):

        max1 = 0

        for j in range(len(lst)):

            if lst[j] > max1:

                max1 = lst[j]

        lst.remove(max1)

        final_list.append(max1)

    print(final_list)

x=int(input("How many numbers you want to enter?:"))

print("enter a number")

lst=[]

for i in range(0,x):

    ele=int(input())

    lst.append(ele)

y=int(input("How many largest numbers you want to find?:"))

lastN(lst,y)
```

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

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.

In [3]:

```
a=['bbb', 'ccc', 'axx', 'xzz', 'xaa']
a1=['mix', 'xyz','apple', 'xanadu', 'aardvark','xz']
a2=['ccc', 'bbb', 'aaa', 'xcc', 'xaa']
xlist=[]
def sort(s):
    for elem in s[:]:
        if elem.startswith('x'):
            xlist.append(elem)
            s.remove(elem)
    print(sorted(xlist)+sorted(s))
    del xlist[:]

sort(a)
sort(a1)
sort(a2)
```

```
['xaa', 'xzz', 'axx', 'bbb', 'ccc']
['xanadu', 'xyz', 'xz', 'aardvark', 'apple', 'mix']
['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
```

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.

In [9]:

```
tuple1 = [(1, 7), (1, 3), (3, 4,5), (2,2) ]

List2 =[]
List3 =[]

for t in tuple1:
    List2.append(t[1],)

List2.sort()
print(List2)

for l in List2:
    for q in tuple1:
        if l == int(q[1],):
            List3.append(q)

print(List3)
```

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

In [10]:
```python

tuple1 = [(2,1), (3,2), (1,3) ]

List2 =[]
List3 =[]

for t in tuple1:
    List2.append(t[1],)

List2.sort()
print(List2)

for l in List2:
    for q in tuple1:
        if l == int(q[1],):
            List3.append(q)

print(List3)
```

```
[1, 2, 3]
[(2, 1), (3, 2), (1, 3)]
```

In [11]:
```python
tuple1 = [(2, 3), (1, 2), (3, 1), ]

List2 =[]
List3 =[]

for t in tuple1:
    List2.append(t[1],)

List2.sort()
print(List2)

for l in List2:
    for q in tuple1:
        if l == int(q[1],):
            List3.append(q)

print(List3)
```

```
[1, 2, 3]
[(3, 1), (1, 2), (2, 3)]
```

Define a function first() that receives a tuple and returns its first element

In [13]:
```
data=[(1,'suri'), (2,'rolex'),(3,'john'), (4,'hari'),(5,'josu')]
print(data[0])
```

```
(1, 'suri')
```

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

In [14]:
```
def Sort_Tuple(tup):
    lst = len(tup)
    for i in range(0, lst):
        for j in range(0, lst-i-1):
            if (tup[j][1] > tup[j + 1][1]):
                temp = tup[j]
                tup[j]= tup[j + 1]
                tup[j + 1]= temp
                return tup
tup =[('for', 24), ('is', 10), ('Geeks', 28),
      ('Geeksforgeeks', 5), ]
print(Sort_Tuple(tup))
```

```
[('is', 10), ('for', 24), ('Geeks', 28), ('Geeksforgeeks', 5)]
```

Print lists in sorted order

In [15]:
```
numbers = [1, 3, 4, 2]
numbers.sort()
print(numbers)
```

```
[1, 2, 3, 4]
```

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

In [16]:
```
my_list = [4,3,2,9,10,44,1]
my_list.sort()
print("sorted list is ",my_list)
print("mid value is ",my_list[int(len(my_list)/2)])
```

```
sorted list is  [1, 2, 3, 4, 9, 10, 44]
mid value is  4
```

Define a functino sort_middle() that receives a list of tuples and returns it sorted using the key middle

In [17]:
```
def last(n):
    return n[-1]
def sort(tuples):
    return sorted(tuples, key=last)
a=[(1, 3), (3, 2), (2, 1)]
print("Sorted:")
print(sort(a))
```

```
Sorted:
[(2, 1), (3, 2), (1, 3)]
```

Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order

In [18]:
```
a=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,40)]
a.sort()
print(sort(a))
```

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

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.

In [20]:
```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list
d1 = [1, 2, 2, 3]
d2=[2, 2, 3, 3, 3]
d3 = []
d4 = [2,5,5,6,6,7]
d5 = [6,7,7,8,9,9]
print(Remove(d1))
print(Remove(d2))
print(Remove(d3))
print(Remove(d4))
print(Remove(d5))
```

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

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 [21]:
```
def add_string(str1):
  length = len(str1)
  if length > 2:
    if str1[-3:] == 'ing':
      str1 += 'ly'
    else:
      str1 += 'ing'
  return str1
print(add_string('hail'))
print(add_string('swimming'))
print(add_string('do'))
```

```
hailing
swimmingly
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 [22]:
```
def not_bad(str1):
  snot = str1.find('not')
  sbad = str1.find('bad')
  if sbad > snot and snot>0 and sbad>0:
    str1 = str1.replace(str1[snot:(sbad+4)], 'good')
    return str1
  else:
    return str1
print(not_bad('The dinner is not that bad!'))
```

```
The dinner is good
```

In [ ]: