Problem solving and Programming in Python June 2019

Date 14 June 2019

Day Objectives

- Python Data Structures
  - Lists
  - Tuples
- Dictionaries
- Basic Problem set on Data Structures
- Advanced Problem set

Python Data Structures

Lists

In [48]:
```
li = [123,978,654]
li #Access the entire list
li[1]# Accessing an elemnt with index in a list
li[1:]# Access all element from second element to end element
li = li[::-1]
li
li = li[::-1]
li[::2]#Accessing even index elements
li[1::2]
# If you want to add the elements to the lis we are having a two ways
        #Direct Referencing---->[index]
        #Indirectbreferencing-->through functions
li.append(345)#  Adding an element to end of the list
# Adding an element at a particular index
li.insert(1,234)
li.sort() # Sort elements in ascending order
li.pop() # Remove the last element in a list
li.pop(1)#remove an element at particular index in a list


li2 = [234,456,789]
li.extend(li2)# Merge list2 into list1
sum(li)# Sum of all elements in alist
max(li)# Maximum element in the list
len(li)# Number of elements in a list

# Average of list elements
sum(li)/len(li)
# Average of all alternative elements
sum(li[1::2])/len(li[1::2])
min(li)
li.index(234)
li
```

Out[48]:
```
[123, 345, 654, 234, 456, 789]
```

In [36]:
```
# Function to identify the second largest element in a unique list(no duplicates)
# Sort the data and select the second last element
# sort the data in reverse order , and select the second element
def secondLargest(li):
    li.sort()
    return li[-2]


# Function that returns the nth largest
def genericLargest(li,n):
    li.sort()
    return li[-n]
secondLargest(li)
genericLargest(li,6)
```

Out[36]:
```
123
```

In [50]:
```
# Function to search for data in a list
# Search for key in the list return the index of the key.return -1 if key not fou
def linearsearch(li,key):
    if key in li:
        for i in range(0,len(li)):
            if key == li[i]:
                return i
        return 1
    else:
        return -1
def linaersearch2(li,key):
    for element in li:
        if element==key:
            return li.index(element)
    return -1

def linearSearch3(li,key):
    if key in li:
        return li.index(key)
    return -1
#linaersearch2(li,654)
#linearsearch(li,654)
linearSearch3(li,1000)
```

Out[50]: -1

In [60]:
```
# Function to count the occurances of a carachter  in a string
#"Python Programming"
def findingRepeatedValues(s,key):
    count=0
    for i in range(0,len(s)):
        if s[i] == key:
            count+=1
    return count
def countCharOccurances2(s,c):
    return s.count(c)

countCharOccurances2("Python Programming",'Py')
#findingRepeatedValues("Python Programming",'Py')
```

Out[60]: 1

In [62]:
```
# Function to find the number of occurances of a substring
#"abscdabcd"====>"ab"---->2
def occurancesOfSubstring(s,subs):
    count=0
    for subs in s:
        count+=1
    return count
occurancesOfSubstring("mastan Vali","Vali")
```

Out[62]: 11

In [71]:
```
s="1 2 3 4 5 6 7 8 9"
li=s.split()
numberslistr=[]
for i in li:
    numberslistr.append(int(i))
numberslistr
```

Out[71]: [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [79]:
```
n=int(input())
def sumofSquare(n):
    sum=0
    for i in range(1,n+1):
        sum+=i**2
    return sum
sumofSquare(n)
```

```
3
```

Out[79]: 14

Problem ColsestZero

Explanation

- li = [3,2,-1,-2,-3](Orginal List)
- Sort the Data
- li = [-3,-2,-1,2,3](Sorted List)
- pl = [1,2,2,3,3](Positive sorted List)
- pl[0] -> check if this number is -ve or +ve in the orginal list
- if pl[0] in li:
  - return pl[0]
- else:
  - return -(pl[0])

In [84]:
```
n = int(input())
s = input()
# "1 2 3 4 5"
li = []
for i in s.split():
    li.append(int(i))
def cosestZero(li):
    # Seperate all numbers less than zero and take max
    # All numbers greater than=zero ,get the minimum
    #-1 2 5 -10 -20
    # -1 -10 -20 ->-1
    #2 5 ->2
    #-1 2->
min(li)
```

```
5
0 1 2 3 4
```

Out[84]:
```
0
```

In [7]:
```
# Closest Zero Problem simple solution
li = [-1,-2,2,3,1]
li.sort()
pl = []
for i in li:
    pl.append(abs(i))
pl.sort()
if pl[0] in li:
    print(pl[0])
else:
    print(-pl[0])
```

```
1
```

In [11]:
```
# FarthestFromZero
li = [-1,-2,2,3,1,-100]
li.sort()
pl = []
for i in li:
    pl.append(abs(i))
pl.sort()
if pl[-1] in li:
    print(pl[-1])
else:
    print(-pl[-1])
```

```
-100
```

You are given three numbers a,b, and c.write program to find the largest number which is less than or equal to c and leaves remainder b when devided by a. 3 2 9====>8 9%3 == 0 8%3 == 2

In [24]:
```
def findingLargest(a,b,c):
    for i in range(c,a-1,-1):
        if i%a == b:
            return i
    return -1
findingLargest(3,2,9)
```

Out[24]: 8

In [26]:
```
s='123456'
for i in range(6,1-1,-1):
    print(i)
```

```
6
5
4
3
2
1
```

In [28]:
```
s=input()
for ch in range(0,len(s)):
    if ch.upper():
        ch.lower()
    else:
        ch.upper()
s
```

```
MastanVali

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-28-22171f665263> in <module>
      1 s=input()
      2 for ch in range(0,len(s)):
----> 3     if ch.upper():
      4         ch.lower()
      5     else:

AttributeError: 'int' object has no attribute 'upper'
```

In [ ]:
```

```