Digit Frequency - Assessment problem

Given a string,s, consisting of alphabets and digits , Find the frequency of numbers in the given string
from 0 t0 9

In [1]:
```python
# S = a11472o5t6   Output: 0 2 1 0 1 1 1 1 0 0

# s= 213abc456def111
# 0 4 1 1 1 1 1 1 0 0 0 -> Frequency of sorted numbers
#count(1)-> 4

# S=c
# 0 0 0 0 0 0 0 0 0 0

# s=1234567890
# 1 1 1 1 1 1 1 1 1 1

def uniqueData(allnumbers):
    unique=[]
    for n in allnumbers:
        if n not in unique:
            unique.append(n)
    return unique

def digitfrequency1(s):
    allnumbers=[]
    for i in s:
        if i.isdigit():
            allnumbers.append(i)
    unique=uniqueData(allnumbers)
    for i in range(0,10):
        if str(i) not in unique:
            print(0,end=" ")
        else:
            count=allnumbers.count(str(i))
            print(count,end=" ")

digitfrequency1("213abc456def111")
```

```
0 4 1 1 1 1 1 0 0 0
```

Marks Analysis Application

- Generate marks file - marks file for n students
- Input: Marks text file- each line contain marks of one students
- Generates a report with the following information

- Class Average
- % of students passed
- % of students failed
- % of students with distinction
- Highest Mark Frequency
- Lowest Mark Frequency

In [54]:

```python
### Marks Analysis

from random import randint

def generateMarks(n,lb,ub):
    filename='DataFiles/marks.txt'
    with open(filename,'w') as f:
        for i in range(0,n):
            r=randint(lb,ub)
            f.write(str(r)+'\n')
    print(n,"Marks added successfully")

generateMarks(20,0,100)

```

```
20 Marks added successfully
```

In [55]:

```python
def classaverage(filepath):
    sum=0
    count=0
    with open(filepath,'r') as f:
        for i in f:
            sum=sum+int(i)
            count=count+1
    print(sum/count)
classaverage('DataFiles/marks.txt')
```

```
43.75
```

In [56]:

```python
# Function to find passpercentage of students in a file

def passpercentage(filepath):
    count=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if int(i)>=35:
                count=count+1
    print((count/mc)*100)
passpercentage('DataFiles/marks.txt')

```

```
60.0
```

In [57]:
```
# Function to find fail percentage of students in a file

def failpercentage(filepath):
    count=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if (int(i)<35):
                count=count+1
    print((count/mc)*100)
failpercentage('DataFiles/marks.txt')

def failedpercentage(filepath):
    failpercentage=100-(passpercentage(filepath))
    print(failpercentage)


```

```
40.0
```

In [48]:
```
# Function to find disti

def distinction(filepath):
    count=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if int(i)>=75:
                count=count+1
    print((count/mc)*100)
distinction('DataFiles/marks.txt')
```

```
25.0
```

In [58]:
```
# Function find frequency of highest marks

def frequencyHighest(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        print(max(sp))
        print(sp.count(max(sp)))
frequencyHighest('DataFiles/marks.txt')
```

```
91
1
```

In [59]:

```python
# Function to find frequency of lowest marks

def frequencyLowest(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        print(min(sp))
        print(sp.count(min(sp)))
frequencyLowest('DataFiles/marks.txt')
```

```
1
1
```

In [61]:

```python
def marksanalysis(filepath):
    while True:
        n=int(input("Choose ur option:\n1).Generation of marks\n2).Class Ave
        if(n==1):
            st=int(input())
            generateMarks(st,0,100)
        elif(n==2):
            classaverage(filepath)
        elif(n==3):
            passpercentage(filepath)
        elif(n==4):
            failpercentage(filepath)
        elif(n==5):
            distinction(filepath)
        elif(n==6):
            frequencyHighest(filepath)
        elif(n==7):
            frequencyLowest(filepath)
        else:
            break
marksanalysis('DataFiles/marks.txt')
```

```
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
1
30
30 Marks added successfully
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
2
51.93333333333333
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
3
63.33333333333333
Choose ur option:
1).Generation of marks
2).Class Average
```

```
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
4
36.666666666666664
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
5
33.333333333333333
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
6
98
1
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
7
5
1
Choose ur option:
1).Generation of marks
2).Class Average
3).Pass percentage
4).Failed percentage
5).Distinction
6).Frequency of Highest
7).Frequency of Lowest
8
```

Contacts Application

In [3]:

```python
import re

def phonenumbervalidator(number):
    pattern='^[6-9][0-9]{9}$|^[0][6-9][0-9]{9}$|^[+][9][1][6-9][0-9]{9}$'
    if re.match(pattern,str(number)):
        return True
    return False
def emailvalidator(email):
    pattern='^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z][@][0-9a-z]{3,18}[.][a-z]{2,4}
    if re.match(pattern,email):
        return True
    return False
emailvalidator("reddyh580@gmail.com")
```

Out[3]:

```
True
```

In [5]:

```python
def contact_exists(name):
    filename='DataFiles/contacts.txt'
    with open(filename,'r') as f:
        p=name+','
        fd=f.read()
    return re.search(p,fd)
contact_exists('anu')
```

Out[5]:

```
<re.Match object; span=(0, 4), match='anu,'>
```

In [9]:

```python
def addcontact(name,phone,email):
    filename='DataFiles/contacts.txt'
    if not contact_exists(name):
        if emailvalidator(email) and phonenumbervalidator(phone):
            with open(filename,'a') as f:
                line=name+','+str(phone)+','+email+'\n'
                f.write(line)
            print(name,"is added to contact list")
        else:
            print('invalid phone or email')
    else:
        print(name,'already exists')
    return
addcontact('baby',9705079252,'baby_123@gmail.com')
```

```
baby is added to contact list
```

In [20]:
```
def searchcontact(name):
    filepath='DataFiles/contacts.txt'
    if contact_exists(name):
        with open(filepath,'r') as f:
            for i in f:
                i=i.split(',')
                if i[0]==name:
                    print(i[0],i[1],i[2])
    else:
        print("contact does not exists")
searchcontact('baby')

```

```
baby 9705079252 baby_123@gmail.com
```

In [23]:
```
def listallcontacts():
    filename='DataFiles/contacts.txt'
    with open(filename,'r') as f:
        x=f.read().split()
        if len(x)!=0:
            print(x)
        else:
            print('Empty list of contacts')
    return
listallcontacts()
```

```
['anu,9866296799,anu.13284@gmail.com', 'baby,9705079252,baby_123@gmail.com']
```

In [71]:
```python
# Function to check if two strings are anagrams
# abc cba -> True
# {a:1,b:1,c:1} {c:1,b:1,a:1}
# abc abc
# aabbcc ccbbaaa -> False
# aabbcc aaabbcc
# {a:2,b:2,c:2} {a:3,b:2,c:2}

def checkAnagrams(s1,s2):
    if len(s1)!=len(s2):
        return False
    if sorted(s1)==sorted(s2):
        return True
    return False
checkAnagrams('abc','bcc')

def charDeletionsAnagrams(s1,s2):
    uncommon=[]
    for i in s1:
        if i not in s2:
            uncommon.append(i)
    for i in s2:
        if i not in s1:
            uncommon.append(i)
    count=len(uncommon)
    freqs1={}
    freqs2={}
    uniqs1=[]
    uniqs2=[]

    for i in s1:
        if i not in uncommon and i not in uniqs1:
            freqs1[i]=s1.count(i)
            uniqs1.append(i)
    for i in s2:
        if i not in uncommon and i not in uniqs2:
            freqs2[i]=s2.count(i)
            uniqs2.append(i)
    for key in freqs1.keys():
        count=count+abs(freqs1[key]-freqs2[key])
    return count
charDeletionsAnagrams('aaabcc','abbcddd')
```

Out[71]:
```
0
```

In [73]:
```python
def averageRange(lb,ub):
    sum=0
    for i in range(lb,ub+1):
        sum=sum+i
        count=ub-lb+1
    return sum//count
averageRange(1000,123456)
```

Out[73]:
```
62228
```

In [ ]:
```python
###### {a:4,g:9,i:6,p:213,c=6}
# [4,6,6,9,213]
# [a,c,g,i,p]
# k=3
# li=[]
# for item in d.items():
# if item[1]
#       li.append(item[0])
# li=[i,c]

def kLargestFrequency(s,k):
    #Construct the frequency dic
    unique=[]
    freq={}
    for i in s:
        if i not in unique:
            freq[i]=s.count(i)
    values=sorted(freq.values(),reversed=True)
    uniquevalues=list(set(values))
    uniquevalues=sorted(uniqueValues,reversed=True)
    if k<=len(uniquevalues):
        kvalue=uniqueValues[k-1]
    else:
        return -1
    for item in freq.items():
        if item[1]==kvalue:
            li.append
    return min(li)
with open('../input.txt') as f:
    t=int(f.readline())
    for i in range(t):
        s=f.readline()
        k=int(f.readline())
        print(KLargestFrequency(s,k))
```

In [ ]:
```python

```