

Frequency of numbers in a string

```
In [3]: 1 # Frequency of every digit in a string
        2 # s = 123andj48739292
        3 n=input()
        4 for i in range(0,10):
        5     print(n.count(str(i)),end=' ')
```

```
sjkls72828273hjsje
0 0 3 1 0 0 0 2 2 0
```

```
In [1]: 1 def digitFrequency2(s):
        2     for i in range(0,10):
        3         count = s.count(str(i))
        4         print(count, end=' ')
        5     return
        6 digitFrequency2('213abc456def111')
```

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

Problem Statement

- Given a string, s , consisting of alphabets and digits, find the frequency of each digit in the given string.
- Input Format
 - The first line contains a string, num, which is the given number.
- Constraints
 - $1 \leq \text{len}(\text{num}) \leq 1000$
 - All the elements of num are made of english alphabets and digits.
- Output Format
 - Print ten space-separated integers in a single line denoting the frequency of each digit from 0 to 9

In [9]:

```

1  # s = 123abc456def
2  # 0 1 1 1 1 1 1 0 0 0 --> frequency of sorted numbers
3
4  # s = c
5  # 0 0 0 0 0 0 0 0 0 0
6
7  # s 1234567890
8  # 1 1 1 1 1 1 1 1 1 1
9
10 def uniqueData(allnumbers):
11     unique = []
12     for n in allnumbers:
13         if n not in unique:
14             unique.append(n)
15     return unique
16
17 def digitFrequency1(s):
18     allnumbers = []
19     for i in s:
20         if i.isdigit():
21             allnumbers.append(i)
22     unique=uniqueData(allnumbers)
23     for i in range(0,10):
24         if str(i) not in unique:
25             print(0,end=' ')
26         else:
27             count = allnumbers.count(str(i))
28             print(count, end=' ')
29
30
31 digitFrequency1('2938729198281919919288829')
32

```

0 4 5 1 0 0 0 1 6 8

In [9]:

```

1  # Generation of marks
2  from random import randint
3  def generatemarks(n,lb,ub):
4      filepath='DataFiles/marks.txt'
5      with open(filepath,'w') as f:
6          for i in range(0,n):
7              r=randint(lb,ub)
8              f.write(str(r)+'\n')
9      print(n,'marks stored in file ')
10 generatemarks(30,1,100)

```

30 marks stored in file

```
In [10]: 1 # def classAverage(filepath):
2 #     sum=0
3 #     count=0
4 #     with open(filepath,'r') as f:
5 #         for i in f:
6 #             sum=sum+int(i)
7 #             count=count+1
8 #             print(sum/count)
9
10 # classAverage('DataFiles/marks.txt')
```

60.1

```
In [11]: 1 # # % passed percentage
2 # def passpercentage(filepath):
3 #     count=0
4 #     sum=0
5 #     with open(filepath,'r') as f:
6 #         for i in f:
7 #             if(int(i)>=35):
8 #                 sum=sum+int(i)
9 #                 count=count+1
10 #             print(sum/count)
11 # passpercentage('DataFiles/marks.txt')
```

71.43478260869566

```
In [16]: 1 # # % Pass
2 # def Passpercentage(filepath):
3 #     count=0
4 #     mc=0
5 #     with open(filepath,'r') as f:
6 #         for i in f:
7 #             mc=mc+1
8 #             if(int(i)>=35):
9 #                 count=count+1
10 #             return ((count/mc)*100)
11 # Passpercentage('DataFiles/marks.txt')
```

Out[16]: 76.66666666666667

```
In [17]: 1 # # % Fail Percen
2 # def failper(filepath):
3 #     fcount=100 - Passpercentage(filepath)
4 #     return (fcount)
5
6 # failper('DataFiles/marks.txt')
```

23.33333333333333

In [13]:

```

1 # def failpercentage(filepath):
2 #     count=0
3 #     mc=0
4 #     with open(filepath,'r') as f:
5 #         for i in f:
6 #             mc=mc+1
7 #             if(int(i)<35):
8 #                 count=count+1
9 #             return ((count/mc)*100)
10 # failpercentage('DataFiles/marks.txt')

```

23.333333333333332

In [22]:

```

1 # # Frequency of highest marks -->
2 # def frequencyHighest(filepath):
3 #     with open(filepath,'r') as f:
4 #         sp=f.read().split()
5 #         sp=list(map(int,sp))
6 #         # sp = list(map(str,sp)) # --> can convert int of list element in
7 #         #print(sp) # --> printing the list of int values of list
8 #         return (sp.count(max(sp)))
9 #     #return
10 # frequencyHighest('DataFiles/marks.txt')

```

1

In [4]:

```

1 # # % Pass
2 # def Disctionpercentage(filepath):
3 #     count=0
4 #     mc=0
5 #     with open(filepath,'r') as f:
6 #         for i in f:
7 #             mc=mc+1
8 #             if(int(i)>75):
9 #                 count=count+1
10 #             return ((count/mc)*100)
11 # Disctionpercentage('DataFiles/marks.txt')

```

In [26]:

```

1 # # Frequency of Lowest marks -->
2 # def frequencyLowest(filepath):
3 #     with open(filepath,'r') as f:
4 #         sp=f.read().split()
5 #         sp=list(map(int,sp))
6 #         # sp = list(map(str,sp)) # --> can convert int of list element in
7 #         #print(sp) # --> printing the list of int values of list
8 #         return (sp.count(min(sp)))
9 #     return
10 # frequencyLowest('DataFiles/marks.txt')

```

2

In [1]:

```

1  # Generation of marks
2  from random import randint
3  def generatemarks(n,lb,ub):
4      filepath='DataFiles/marks.txt'
5      with open(filepath,'w') as f:
6          for i in range(0,n):
7              r=randint(lb,ub)
8              f.write(str(r)+'\n')
9      #print(n,'marks stored in file ')
10 #generatemarks(30,1,100)
11
12 def classAverage(filepath):
13     sum=0
14     count=0
15     with open(filepath,'r') as f:
16         for i in f:
17             sum=sum+int(i)
18             count=count+1
19     return (sum/count)
20
21 #classAverage('DataFiles/marks.txt')
22
23
24 # % Pass
25 def Passpercentage(filepath):
26     count=0
27     mc=0
28     with open(filepath,'r') as f:
29         for i in f:
30             mc=mc+1
31             if(int(i)>=35):
32                 count=count+1
33     return ((count/mc)*100)
34 #Passpercentage('DataFiles/marks.txt')
35
36 def failpercentage(filepath):
37     count=0
38     mc=0
39     with open(filepath,'r') as f:
40         for i in f:
41             mc=mc+1
42             if(int(i)<35):
43                 count=count+1
44     return ((count/mc)*100)
45 #failpercentage('DataFiles/marks.txt')
46
47 # Frequency of highest marks -->
48 def frequencyHighest(filepath):
49     with open(filepath,'r') as f:
50         sp=f.read().split()
51         sp=list(map(int,sp))
52         # sp = list(map(str,sp)) # --> can convert int of list element into
53         #print(sp) # --> printing the list of int values of list
54         return (sp.count(max(sp)))
55     #return
56 #frequencyHighest('DataFiles/marks.txt')

```

```
57
58 # Frequency of Lowest marks -->
59 def frequencyLowest(filepath):
60     with open(filepath, 'r') as f:
61         sp=f.read().split()
62         sp=list(map(int,sp))
63         # sp = list(map(str,sp)) # --> can convert int of list element into
64         # print(sp) # --> printing the list of int values of list
65         return (sp.count(min(sp)))
66     #return
67 #frequencyLowest('DataFiles/marks.txt')
68
69 # %% disction
70 def Disctionpercentage(filepath):
71     count=0
72     mc=0
73     with open(filepath, 'r') as f:
74         for i in f:
75             mc=mc+1
76             if(int(i)>75):
77                 count=count+1
78     return ((count/mc)*100)
79 #Disctionpercentage('DataFiles/marks.txt')
```

```

In [3]: 1 def marksAnalysis(filepath):
        2     while True:
        3         n=int(input("Choose option :\n1).Generation of Marks\n2).Class Average\n3).% of pass\n4).% of fail\n5).% Disction\n6).Highest Frequency\n7). Lowest Frequency\n8)."))
        4         if(n==1):
        5             st=int(input())
        6             generatemarks(st,1,100)
        7         elif(n==2):
        8             print(classAverage(filepath))
        9         elif(n==3):
        10            print(Passpercentage(filepath))
        11         elif(n==4):
        12            print(failpercentage(filepath))
        13         elif(n==5):
        14            print(Discionpercentage(filepath))
        15         elif(n==6):
        16            print(frequencyHighest(filepath))
        17         elif(n==7):
        18            print(frequencyLowest(filepath))
        19         else:
        20             break
        21     return
        22 marksAnalysis('DataFiles/marks.txt')

```

Choose option :
 1).Generation of Marks
 2).Class Average
 3). % of pass
 4). % of fail
 5).% Disction
 6).Highest Frequency
 7). Lowest Frequency

1
50

Choose option :
 1).Generation of Marks
 2).Class Average
 3). % of pass
 4). % of fail
 5).% Disction
 6).Highest Frequency
 7). Lowest Frequency

5
24.0

Choose option :
 1).Generation of Marks
 2).Class Average
 3). % of pass
 4). % of fail
 5).% Disction
 6).Highest Frequency
 7). Lowest Frequency

8

```
In [7]: 1 # Function to check if two strings are anagrams
2 # abc cbc --> True
3 # {'ea':1,'b':1,'c':1} {'c':1, 'a':1, 'b':1} --> both dictionaries are sam
4 # aabbcc ccbbaaa --> False
5 # By Sorting the strings we will get as --> aabbcc aaabbcc
6 def checkAnagrams(s1,s2):
7     if len(s1) != len(s2):
8         return False
9     if sorted(s1) == sorted(s2):
10         return True
11     return False
12 checkAnagrams('abc','bcc')
13
```

Out[7]: False

```
In [9]: 1 t=int(input())
2 for i in range(0,t):
3     s1=sorted(input())
4     s2=sorted(input())
5
6     count=0
7     for i in s1:
8         if i not in s2:
9             count+=1
10    for i in s2:
11        if i not in s1:
12            count+=1
13    print(count)
```

```
2
cde
abc
4
dfdskc
acjdd
5
```



```

In [15]: 1 def CharDeletionsAnagrams(s1,s2):
2         # characters occurring only string
3         uncommon = [] # to collect all uncommon characters i.e., characters occur
4         for i in s1:
5             if i not in s2:
6                 uncommon.append(i)
7         for i in s2:
8             if i not in s1:
9                 uncommon.append(i)
10        count = len(uncommon)
11        # freqs1 -> Frequency of common charactes which are only in s1
12        # freqs2 -> Frequency of common charactes which are only in s2
13        freqs1 = {}
14        freqs2 = {}
15        # uniqs1 ,uniqs2 --> Unique characters in s1 and s2
16        uniqs1 = []
17        uniqs2 = []
18        # Frequency of common unique characters in s1
19        for i in s1:
20            if i not in uncommon and i not in uniqs1:
21                freqs1[i] = s1.count(i)
22                uniqs1.append(i)
23        # Frequency of common unique characters in s2
24        for i in s2:
25            if i not in uncommon and i not in uniqs2:
26                freqs2[i] = s2.count(i)
27                uniqs2.append(i)
28        # Diffence in frequencies for common characters
29        for key in freqs1.keys():
30            count +=abs(freqs1[key] - freqs2[key])
31        return count
32    t=int(input())
33    for i in range(0,t):
34        s1=input()
35        s2=input()
36        print(CharDeletionsAnagrams(s1,s2))

```

```

2
cde
abd
4
jskjdsaa
aajdjksjx
4

```

In [2]:

```

1 n= int(input())
2 for i in range(n):
3     s1=input()
4     s2=input()
5     x1=[]
6     x2=[]
7     x=[]
8     temp= ord('a')
9     for i in range(0,26):
10         x1.append(s1.count(chr(temp)))
11         x2.append(s2.count(chr(temp)))
12         temp = temp+1
13     for j in range(0,26):
14         x.append(abs(x2[j]-x1[j]))
15     print(sum(x))
16

```

```

2
cde
abc
4
abc
cds
4

```

In [9]:

```

1 # Function to kLarest Frequency
2 # {a:4, g:9, i:6, p:213, c:6}
3 # [4,6,6,9,213]
4 # [213,9,6,6,4]
5 # [a,c,g,i,p]
6 # k = 3
7 # li = []
8 # for item in d.items():
9 #     if item[1] == 6
10 #         li.append(item[0])
11 # li = [i, c]
12
13 def KLargestFrequency(s, k):
14     # Consturct the frequency dictionary
15     unique = []
16     freq = {}
17     for i in s:
18         if i not in freq.keys():
19             freq[i] = s.count(i)
20             values = sorted(freq.values(), reverse=True)
21             uniqueValues = list(set(values))
22             uniqueValues = sorted(uniqueValues, reverse = True)
23     kvalue = uniqueValues[k-1]
24     return kvalue
25
26 KLargestFrequency([2,3,4,3,2,4,2,4,9,6,5,4],2)

```

Out[9]: 3

In []:

1

