

Python Programming

Data Containers

資料容器list, tuple, set, str, dict...

Prof. Chang-Chieh Cheng

Information Technology Service Center

National Chiao Tung University

Lists

- A list is a **ordered** data sequence
- Random access → 根據你輸入的順序輸出也亦然
 - Each element can be accessed by an index
- Duplicate elements are allowed

```
L = [1, 1, 2, 2, 2, 3, 4]
print(L[0])
L[0] = L[1] + L[2]
print(L[0])
print(L)
```

什麼東東都能放

Some useful methods of lists

- `list()`
 - create an empty list

```
L = list()
print(L)          # []
```

- `list.append(x)`
 - add an item `x` to the end of the `list`

```
L = [5, 6, 7]
print(L)          # [5, 6, 7]
L.append(99)
print(L)          # [5, 6, 7, 99]
```

- `list.extend(L2)`
 - add another list `L2` to the end of the `list`

用`append`和`extend`會取代原資料，
這種情況叫做side effect
(但好處就是省記憶體)

```
L1 = ['A', 'B', 'C']
L2 = [1, 2, 3]
L1.extend(L2)
print(L1)          # ['A', 'B', 'C', 1, 2, 3]
print(L2)          # [1, 2, 3]
```

Some useful methods of lists

- `list.insert(i, x)`
 - Insert an item `x` at a given position `i`. Then, `i` will be the index of `x`.

```
L = ['A', 'B', 'C']  
L.insert(1, 'X')  
print(L)          # ??
```

- Let's try it
 - Using `for` loop, `list.append`, `list.extend`, and `list.insert` to merge three lists:
 - `L1 = ['A', 'B', 'C', 'D', 'E']`
 - `L2 = ['X', 'Y', 'Z', 'W']`
 - `L3 = [1, 2, 3, 4, 5]`
 - Then the output list is

```
['A', 'B', 'C', 1, 'D', 2, 'E', 3, 'X', 4, 'Y', 5, 'Z', 'W']
```

Some useful methods of lists

- **list.clear()**
 - Remove all items.

```
L = [5, 6, 7]
print(L)          # [5, 6, 7]
L.clear( )
print(L)          # []
```

- **list.remove(x)**
 - Remove the first item from the list whose value is `x`. It is an error if there is no such item.

```
L = [5, 6, 7, 6]
print(L)          # [5, 6, 7, 6]
L.remove(6)
print(L)          # [5, 7, 6]
L.remove(6)
print(L)          # [5, 7]
L.remove(6)       # An exception of ValueError thrown.
```

- Let's try it 如果找不到欲移除的值會報錯，可以用之前學的try&except來處理
 - Modify the above example of `L.remove(x)`, so that append `x` to `L` if `x` is not in `L`.

Some useful methods of lists

- Avoiding that put `L.remove()` in a loop statement accessing `L`.
- The following code try to clear a list:

```
L = [ 1, 2 ,4 ,3 ,2, 4, 1]
for x in L:
    print(L, ' ==> L[' , i, ' ] = ' , x)
    while True:
        try:
            L.remove(x)
        except:
            break;
    i += 1

print(L) # [3] !?
```

在迴圈對list做操作時，千千萬萬不要在迴圈內對list做新增或刪除的動作，很可能會出錯

Some useful methods of lists

- `list.sort(key=None, reverse=False)`
 - Sorting, where *key* specifies the comparison method. Just let it be *None* in most cases.

```
L = ['cat', 'mouse', 'pig', 'dog', 'bird']
L.sort()
print(L)          # ['bird', 'cat', 'dog', 'mouse', 'pig']
L.sort(key = len)
print(L)          # ['cat', 'dog', 'pig', 'bird', 'mouse']
L.sort(key = len, reverse = True)
print(L)          # ['mouse', 'bird', 'cat', 'dog', 'pig']
```

- Let's try it 相同情況下會維持原來的排序！！
 - Sort a list that contains a set of integers by the descending order of the number of digits
 - For example
 - If `L = [123, 4, 567, 9801, 1234, 0]`
 - The result is `[9801, 1234, 123, 567, 4, 0]`

Some useful methods of lists

- `list.reverse()`
 - Reverse the elements of the list in place.

```
L = [5, 6, 7]
L.reverse()
print(L)          # [7, 6, 5]
```

- Let's try it
 - A list `L` is a palindrome if `L` is the same as `L.reverse()`.
 - Design a program to verify whether a list is a palindrome.
 - For example
 - `L = [5, 6, 7] → L is not a palindrome`
 - `L = [5] → L is a palindrome`
 - `L = [5, 6, 7, 6, 5] → L is a palindrome`
 - `L = [5, 6, 6, 5] → L is a palindrome`
 - `L = [] → L is a palindrome`

Some useful methods of lists

- `list.copy()`
 - Return a copy of the list.

```
L1 = [5, 6, 7]
L2 = L1.copy()
L1.remove(5)
print(L1)          # [6, 7]
print(L2)          # [5, 6, 7]
```

- Let's try it
 - If `L1` is not a palindrome, then `L2 = L1 concatenate L1.reverse()` such that `L2` is a palindrome
 - For example
 - `L1 = [5, 6, 7]`
 - Then `L2 = [5, 6, 7, 7, 6, 5]`

del

- Another way to remove an item from a list

```
L= [-1, 1, 66.25, 333, 333, 1234.5]
del (L[0])
print(L)          # [1, 66.25, 333, 333, 1234.5]
del (L[2:4])
print(L)          # [1, 66.25, 1234.5]
del (L[:])
print(L)          # []
```

- Let's try it
 - A list `L` contains a set of integers
 - Find the range of the longest repeated numbers
 - For example, `L= [1, 1, 6, 3, 3, 3, 4, 4]`
 - Then, the range is `[3:6]`
 - Remove the longest repeated numbers
 - In the above example, `L` will be `[1, 1, 6, 4, 4]`

Multidimensional arrays

- 3x4 array

```
L = [[1, 2, 3 ,4], [5, 6, 7, 8], [9, 10, 11, 12]]
print(L)
print(len(L))          # 3
print(len(L[0]))       # 4
print(L[0][0], L[0][1], L[1][0], L[2][3])
Print(L[2])
```

- 2 x 3 x 2 array

```
L = [[[1, 2], [3 ,4], [5, 6]],
      [[7, 8], [9, 10], [11, 12]]]
print(L)
print(len(L))          # 2
print(len(L[0]))       # 3
print(len(L[0][0]))    # 2
print(L[0][0][0], L[0][1][0], L[1][2][1])
print(L[1])
print(L[0][2])
```

Multidimensional arrays

- 1 x 2 x 3 x 2 array

```
L = [[ [1, 2], [3, 4], [5, 6]],  
      [[7, 8], [9, 10], [11, 12]]]  
print(L)  
print(len(L))           # 1  
print(len(L[0]))         # 2  
print(len(L[0][0]))      # 3  
print(len(L[0][0][0]))   # 2  
print(L[0][0][0][0], L[0][1][2][1])
```

List comprehension

- Using a for-statement to generate a list

```
L1 = [ x for x in range(5) ]  
print(L1)          # [0, 1, 2, 3, 4]  
  
L2 = [ x * 2 for x in range(5) ]  
print(L2)          # [0, 2, 4, 6, 8]  
  
L3 = [ x * x for x in range(5) if x % 2 == 0 ]  
print(L3)          # [0, 4, 16]
```

List comprehension

- Multiple layers of for-statement and 2D list

```
L4 = [y for x in range(3) for y in range(4)]  
print(L4)  
# [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]  
  
L5 = [[y for x in range(3)] for y in range(4)]  
print(L5)  
# [[0, 0, 0], [1, 1, 1], [2, 2, 2], [3, 3, 3]]  
  
L6 = [[y * 3 + x for x in range(3)] for y in range(4)]  
print(L6)  
# [[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]]
```

List comprehension

- Let's try it
 - Using list comprehension to generate a $m * n$ list as follows

- $m = 6, n = 3$:

```
[[0, 1, 2],  
 [5, 4, 3],  
 [6, 7, 8],  
 [11, 10, 9],  
 [12, 13, 14],  
 [17, 16, 15]]
```

- $m = 8, n = 4$:

```
[[0, 1, 2, 3],  
 [7, 6, 5, 4],  
 [8, 9, 10, 11],  
 [15, 14, 13, 12],  
 [16, 17, 18, 19],  
 [23, 22, 21, 20],  
 [24, 25, 26, 27],  
 [31, 30, 29, 28]]
```

Tuples

- A tuple is also a data container to store a set of data objects
- Similar to list, using an index to access an item of a tuple

```
t = 12345, 54321, 'hello!'      # without any parenthesis
print(t[0])                    # 12345
print(t)                       # (12345, 54321, 'hello!')
```



```
u = t, (1, 2, 3, 4, 5)
                                # A tuple also can be an item of another tuple
print(u)                       # ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
                                # Notice the parentheses
```

- Each item is immutable, read only.

```
t = 12345, 54321, 'hello!'
t[1] = 0                       # Error!
```


Tuples

- Let's try it
 - The following code can draw a figure from two lists that are X axis's data and Y axis's data.

```
import matplotlib.pyplot as plt
X = [0, 1, 2, 3]
Y = [10, 5, 8, 9]
plt.plot(X, Y)
plt.scatter(X, Y)
plt.show()
```

- Let's see what kind figure will be drawn if you changed some number in X and Y.

Tuples

- Let's try it
 - Given a set of tuple of two elements

```
T = (0, 10), (1, 5), (2, 8), (3, 9)
```

- Write a program to transform T as two lists, X and Y , such that we can use the program in previous page to draw a figure for T .

Sets

- **Unordered** collection with **no duplicate** elements.

```
A = {7, 9, 1, 1, 9, 2, 1, 2}
print(A)          # {9, 2, 1, 7}
                  # The order is undefined
                  # because set is unordered
print(A[0])       # Error!
```

- Item appending and removing

```
A = {4, 6, 1, 2, 2, 1, 3}
A.add(5)
print(len(A))    # 6
print(A)         # {1, 2, 3, 4, 5, 6}
A.remove(2)
print(len(A))    # 5
print(A)         # {1, 3, 4, 5, 6}
A.clear()
print(len(A))    # 0
print(A)         # {}
```

Sets

- Element accessing

- By for statement

```
A = {7, 9, 1, 1, 9, 2, 1, 2}
for x in A:
    print(x)
```

- By `iter()` and `next()`

- `iter()` returns an iterator to indicate the head of a data container
 - `next(itr)` moves an iterator to the next element of a data container until the iterator reaches to the end of data and throws a `StopIteration` exception.

```
A = {7, 9, 1, 1, 9, 2, 1, 2}
itr = iter(A)
print(next(itr))      # 9
print(next(itr))      # 2
print(next(itr))      # 1
print(next(itr))      # 7
print(next(itr))      # Error! StopIteration
```

Sets

- Element accessing
 - By `iter()` and `next()`
 - `iter()` returns an iterator to indicate the head of a data container
 - `next(itr)` moves an iterator to the next element of a data container until the iterator reaches to the end of data and throws a `StopIteration` exception

```
A = {7, 9, 1, 1, 9, 2, 1, 2}
itr = iter(A)
try:
    print(next(itr))    # 9
    print(next(itr))    # 2
    print(next(itr))    # 1
    print(next(itr))    # 7
    print(next(itr))    # StopIteration
except StopIteration:
    print("Stopped")
```

Sets

- Creating an empty set

```
A = set()
print(len(A))    # 0

A.add(7)
A.add(1)
A.add(1)
A.add(7)
A.add(9)
print(len(A))    # 3
print(A)          # {1, 9, 7}
```

Sets

- Let's try it
 - Use the following to input a series of positive numbers and store them into a list `L`.

```
S = set()
while True:
    x = int(input('Input a positive number: '))
    if x < 0:
        break
    else:
        S.add(x)
print('The size of S is ', len(S))
print(S)
L = list(S);
print(L)
```

Sets

- Let's try it
 - Continued from the previous page.
 - Find the median of \mathbb{L}
 - Using a set S to store all elements in \mathbb{L}
 - Find the median of S
 - For example,
 - if $\mathbb{L} = [4, 6, 1, 2, 2, 1, 3]$, its median is 2
 - Then, S maybe is $\{3, 2, 1, 4, 6\}$, the median of S is 3
 - if $\mathbb{L} = [7, 8, 8, 6, 5, 2, 2, 3, 3]$, its median is 5
 - Then, S maybe is $\{5, 3, 2, 7, 6, 8\}$, the median of S is 6

Sets

- Set is orderless, we cannot sort a set
- But sorted function can return a list of sorted data from a set

```
A = {7, 9, 1, 1, 9, 2, 1, 2}
L = sorted(A)
print(A)          # {9, 2, 1, 7}
print(L)          # [1, 2, 7, 9]
```

Sets

- Union two sets

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = A.union(B)
print(A)          # {1, 2, 3, 4}
print(B)          # {3, 4, 5, 6}
print(C)          # {1, 2, 3, 4, 5, 6}
```

- Intersection of two sets

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = A.intersection(B)
print(A)          # {1, 2, 3, 4}
print(B)          # {3, 4, 5, 6}
print(C)          # {3, 4}
```

Sets

- Check whether two sets are disjoint

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = {5, 6}
print(A.isdisjoint(B))      # False
print(A.isdisjoint(C))      # True
```

- Check set B is a subset of set A

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = {5, 6}
print(C.issubset(A))        # False
print(C.issubset(B))        # True
print(B.issubset(C))        # False
```

Sets

- Check set B is superset of set A

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = {5, 6}
print(C.issuperset(A))      # False
print(C.issuperset(B))      # False
print(B.issuperset(C))      # True
```

- A - B

```
A = {4, 3, 1, 2}
B = {3, 6, 5, 4}
C = A.difference(B)
D = B.difference(A)
print(C)                    # {1, 2}
print(D)                    # {5, 6}
C = A.symmetric_difference(B)
D = B.symmetric_difference(A)
print(C)                    # {1, 2, 5, 6}
print(D)                    # {1, 2, 5, 6}
```

Exercise 1/4

- Given a data set S, and #S is the total number of data elements in S.
- Let A is a subset of S, #A is the number of data elements in A.
- The probability of A in S can be computed by

$$P(A) = \frac{\#A}{\#S}$$

- Given two data sets, S1 and S2, and B is a subset of intersection of S1 and S2. The probability of B is called joint probability that can be computed by

$$P(B) = \frac{\#B}{\#(S1 \cup S2)}$$

- Given two data sets, S1 and S2, and B is a subset of intersection of S1 and S2. The probability of B under S2 is called conditional probability that can be computed by

$$P(B|S2) = \frac{\#B}{\#S2}$$

Exercise 2/4

- Given a set S of positive integers, please compute the probability of even numbers in S .
 - For example, $S = \{4, 3, 1, 2, 6\}$
 - The probability of even numbers in S is $3 / 5 = 0.6$
 - Please modify the following program

```
S = set()
while True:
    x = int(input('Input a positive number: '))
    if x < 0:
        break
    else:
        S.add(x)
```

Exercise 3/4

- Given two sets of positive integers, $S1$ and $S2$, please compute the joint probability of even numbers in $S1$ and $S2$.
 - For example,
 $S1 = \{4, 3, 1, 2, 6\}$
 $S2 = \{4, 2, 1, 5, 8, 10\}$
 - The probability of even numbers in $S1$ and $S2$ is $2 / 8 = 0.25$
- Please compute the conditional probability of even numbers under $S2$.
 - For example,
 $S1 = \{4, 3, 1, 2, 6\}$
 $S2 = \{4, 2, 1, 5, 8, 10\}$
 - $P(\text{even}|S2) = 2 / 6 = 0.3333$
- Please modify the program in the next page.

Exercise 4/4

```
S1 = set()
while True:
    x = int(input('Input a positive number for S1: '))
    if x < 0:
        break
    else:
        S1.add(x)

S2 = set()
while True:
    x = int(input('Input a positive number for S2: '))
    if x < 0:
        break
    else:
        S2.add(x)
```


enumerate

- Creating a sequence of tuples for a data container and each tuple contains (index, and data).

```
L = ['ABC', 'DEF', 'GHI']
E = enumerate(L)
for x in E:
    print(x)

S = {'ABC', 'DEF', 'GHI'}
E = enumerate(S)
for x in E:
    print(x)
```

Dictionaries

- A dictionary is similar to a list, but each element is indexed by a key rather than an integer

```
Scores = {'James':82, 'Mary':98, 'Yamamoto':93}
print(Scores['Mary']) # 98
Scores['Yamamoto'] += 7
print(Scores)          # {'James':82, 'Mary':98, 'Yamamoto':100}
print(len(Scores))     # 3
```

- Like a set, each element is unique in a dictionary.
 - Notice that all elements are unsorted

```
Scores = {'Yamamoto':93, 'James':82, 'Mary':68, 'Mary':98, 'James':80}
print(Scores) # {'Yamamoto': 93, 'James': 80, 'Mary': 98}
```

Dictionaries

- Element updating and appending

```
Scores = {'James':82, 'Mary':98, 'Yamamoto':93}
print(Scores)  # {'James':80, 'Mary':98, 'Yamamoto':93}
Scores.update({'Yamamoto':84})
print(Scores['Yamamoto'])
Scores.update({'Hideo':77})
print(Scores)  # {'James':80, 'Mary':98, 'Yamamoto':93, 'Hideo':77}
```

- Element removing

```
Scores = {'James':82, 'Mary':98, 'Yamamoto':93}
print(Scores)  # {'James':80, 'Mary':98, 'Yamamoto':93}
Scores.pop('James')
print(Scores)  # {'Mary':98, 'Yamamoto':93}
```

Dictionaries

- Create a new dictionary

```
Scores = dict()

Scores.update({'Yamamoto':84})
Scores.update({'Hideo':77})
print(Scores)    # {'Yamamoto':93, 'Hideo':77}
```

Dictionaries

- for loop and dictionaries

```
Scores = {'James':82, 'Mary':98, 'Yamamoto':93}
for key in Scores:
    print(key )                # list all keys

for key in Scores:
    print(key , "=", Scores[key])    # list keys and values
```

Dictionaries

- Sort by key

```
Scores = {'James':82, 'Mary':98, 'Yamamoto':93}

L1 = sorted(Scores)
print(L1)
# ['James', 'Mary', 'Yamamoto']

L2 = sorted(Scores.items())
print(L2)
# [('James', 82), ('Mary', 98), ('Yamamoto', 93)]
```

Dictionaries

- Sort by value

```
from operator import itemgetter

Scores = {'James':82, 'Mary':98, 'Yamamoto':93}
L = sorted(Scores.items(), key = itemgetter(1))
print(L)
# [('James', 82), ('Yamamoto', 93), ('Mary', 98)]
```

Dictionaries

- Let's try it
 - Modify all examples of dictionary such that each student can store **three** scores

Dictionaries

- A dictionary with multiple keys
- Example:

```
student1 = {'Name': 'James', 'ID': '01008', 'Score': 90}
student2 = {'Name': 'Mary', 'ID': '01003', 'Score': 98}
student3 = {'Name': 'Yamamoto', 'ID': '01005', 'Score': 93}
print(student1)
print(student2)
print(student3)
```

```
L = list()
L.append({'Name': 'James', 'ID': '01008', 'Score': 90})
L.append({'Name': 'Mary', 'ID': '01003', 'Score': 98})
L.append({'Name': 'Yamamoto', 'ID': '01005', 'Score': 93})
for student in L:
    print(student)
```

Dictionaries

- Data selection from a list of dictionaries
- Example:

```
L = list()
L.append({'Name': 'James', 'ID': '01008', 'Score': 90})
L.append({'Name': 'Ruby', 'ID': '01024', 'Score': 89})
L.append({'Name': 'Mary', 'ID': '01003', 'Score': 98})
L.append({'Name': 'Yamamoto', 'ID': '01005', 'Score': 93})
L.append({'Name': 'Judy', 'ID': '01021', 'Score': 73})

L2 = [x for x in L if x['Score'] < 90 ]

for student in L2:
    print(student)

L3 = [x for x in L if x['Name'][-1] == 'y' and x['Score'] >= 80 ]

for student in L3:
    print(student)
```

Dictionaries

- Data selection from a list of dictionaries
- Example:

```
from operator import itemgetter
L = list()
L.append({'Name':'James', 'ID':'01008', 'Score':90})
L.append({'Name':'Ruby', 'ID':'01024', 'Score':89})
L.append({'Name':'Mary', 'ID':'01003', 'Score':98})
L.append({'Name':'Yamamoto', 'ID':'01005', 'Score':93})
L.append({'Name':'Judy', 'ID':'01021', 'Score':73})

L4 = [{'ID':x['ID'], 'Score':x['Score']} for x in L ]
L4.sort(key = itemgetter('ID'))
for student in L4:
    print(student)
```

Strings

- All string operations
 - <https://docs.python.org/3/library/stdtypes.html#string-methods>
- Substring finding

```
s = 'Hello! My firends!'
if 'My' in s:
    print('OK')

if 'my' in s:
    print('OK')
```

Strings

- `str.split(sep=None, maxsplit=-1)`
 - Return a list of the words in the string, using `sep` as the delimiter string. If `maxsplit` is given, at most `maxsplit` splits are done.

```
s = 'aaa*bbb*ccc eee fff*ggg '  
L1 = s.split(sep = '*')  
L2 = s.split(sep = ' ')  
L3 = s.split(sep = '*', maxsplit = 2)  
print(L1) # ['aaa', 'bbb', 'ccc eee fff', 'ggg '  
print(L2) # ['aaa*bbb*ccc', 'eee', 'fff*ggg', '']  
print(L3) # ['aaa', 'bbb', 'ccc eee fff*ggg ']
```

- Notice zero-length substrings

```
s = 'aaa*bbb**ccc***eee'  
L = s.split(sep = '*')  
print(L) # ['aaa', 'bbb', '', 'ccc', '', '', 'eee']
```

Strings

- `re.split`
 - Splitting a string with multiple separators

```
import re
s = 'aaa*bbb*ccc eee fff*ggg '
L4 = re.split('ccc|\\*| ', s)
print(L4)
```

Exercise

- Input two strings, `s1` and `s2`.
- Using a `set` to store all words in `s1`.
- Count how many number of words in both `s1` and `s2`.
- For example, `s1` is 'How do you do' and `s2` is 'What do you think'
 - The result is 2, which are 'do' and 'you'.

Strings

- `str.replace(old, new[, count])`
 - Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

```
s = 'AAA ABC ABC ccc abc ABC ABC ddd'
s1 = s.replace('ABC', '*')
s2 = s.replace('ABC', '*', 3)
print(s1)      # AAA * * ccc abc * * ddd
print(s2)      # AAA * * ccc abc * ABC ddd
```


Strings

- `str.count(sub[, start[, end]])`
 - Return the number of non-overlapping occurrences of substring *sub* in the range *[start, end]*.

```
s = 'aaa bbb aaa aaa bbb ccc'
print(s.count('aaa'))      # 3
print(s.count('bbb'))      # 2
print(s.count('ccc'))      # 1
print(s.count('ddd'))      # 0
```

collections:Counter

- `collections` is a standard library of Python that includes a lot of many useful data containers
- `collections/Counter`
 - `Counter` is an unordered collection where elements are stored as dictionary keys and their counts are stored as dictionary values.

```
from collections import Counter
s = 'cccbbaaabbbaaabb'
cnt = Counter(s)
print('-----')
print(cnt)                # Counter({'b': 9, 'a': 6, 'c': 3})
print('-----')
print(cnt.keys())         # dict_keys(['c', 'b', 'a'])
print('-----')
print(cnt.items())        # dict_items([('c', 3), ('b', 9), ('a', 6)])
```

collections::Counter

- Accessing elements of Counter

```
from collections import Counter
s = 'cccbbbbaaabbbaaabb'
cnt = Counter(s)
for item in cnt:
    print(item, '\t', cnt[item])
```

```
"""
```

```
c          3
```

```
b          9
```

```
a          6
```

```
"""
```

without sorting!

Exercise

- Design a program to count the number of occurrences of each word in a text
 - Assuming that each word is separated by a space character.
 - Only count non-zero-length words
 - Just ignore zero-length words
 - Print the results by the number of occurrences in ascending order
 - for example, given a text,
" xyz abc xyz abc abc ABC ",
the results will be
ABC: 1
xyz: 2
abc: 3