Lecture 5 List, Dictionary, and Tuple (2021.10.9 & 14)

§1 List

1. len( )

1° Return the number of elements in any sequence

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x=[1,2,'joe',99]
>>> print(len(x))
4
```

2. Built-in functions about list

```
>>> numbers = [3,41,12,9,74,15]
>>> print(len(numbers))
6
>>> print(max(numbers))
74
>>> print(min(numbers))
3
>>> print(sum(numbers))
154
>>> print(sum(numbers)/len(numbers))
25.666666666666668
```

3. range( )

- The range() function returns a list of numbers (with Range type)
- We can construct an index loop using for and an integer iterator

```
>>> x=range(4)
>>> x
range(0, 4)
>>> x[0]
0
>>> x[1]
1
>>> x[2]
2
>>> x[3]
3
```

```
>>> x=range(2,10,2)
>>> x[0]
2
>>> x[3]
8
>>> x[4]
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    x[4]
IndexError: range object index out of range
```

4. Concatenating lists using +

Add two existing lists together to create a new list

```
>>> a=[1,2,3]
>>> b=[4,5,6]
>>> c=a+b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

5. in / not in

1° Logical operators that return True or False

2° Do not modify the list

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
```

6. Two loops

Example

```
friends = ['Tom','Jerry','Bat']

for friend in friends:
    print('Happy new year,', friend)

for i in range(len(friends)):
    friend = friends[i]
    print('Happy new year,', friend)
```

Output

```
Happy new year, Tom
Happy new year, Jerry
Happy new year, Bat
Happy new year, Tom
Happy new year, Jerry
Happy new year, Bat
>>>
```

7. dir( )

```
>>> x=list()
>>> type(x)
<class 'list'>
>>> dir(x)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

8. copy( ) and =

1° copy( )

To copy the content of an array.

2° =

To copy a reference of an array.

```
friends = ['Tom','Jerry']

newFriends = friends.copy()

newFriends[0] = 'David'

print(friends)          # ['Tom', 'Jerry']
print(newFriends)       # ['David', 'Jerry']

oldFriends = friends

oldFriends[0] = 'David'

print(friends)          # ['David', 'Jerry']
print(oldFriends)       # ['David', 'Jerry']
```

9. append( )

We can create an empty list using list( ), and then add elements using append( ) method
The list stays in order, new elements are added at the end of the list.

```
numlist = list()
while True:
    inp = input('Enter a number:')
    if inp =='done': break
    value = float(inp)
    numlist.append(value)

average = sum(numlist)/len(numlist)
print('The average is:',average)
```

10. sort( )

The sort( ) method means "sort yourself"

```
>>> friend = ['Tom','Jerry','Bat']
>>> friends.sort()
>>> print(friends)
['Bat', 'Jerry', 'Tom']
>>> print(friends[1])
Jerry
>>>
>>> numbers = [1,2,5,100,32,7,97,1001]
>>> numbers.sort()
>>> print(numbers)
[1, 2, 5, 7, 32, 97, 100, 1001]
```

11. split( )

Use the split( ) method to break up a string into a list of strings

```
>>> myStr = 'Catch me if you can'
>>> words = myStr.split()
>>> print(words)
['Catch', 'me', 'if', 'you', 'can']
>>> print(len(words))
5
>>> print(words[0])
Catch
>>> for w in words: print(w)

Catch
me
if
you
can
```

When you do not specify a delimiter, multiple spaces are treated like "one" delimiter.

You can specify what delimiter character to use in splitting

```
>>> line = 'A lot                of spaces'
>>> etc = line.split()
>>> print(etc)
['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print(thing)
['first;second;third']
>>> len(thing)
1
>>>
>>> thing = line.split(';')
>>> print(thing)
['first', 'second', 'third']
>>> print(len(thing))
3
```

§2 Dictionary

1. Two collections

List: a linear collection of values that stay in order.

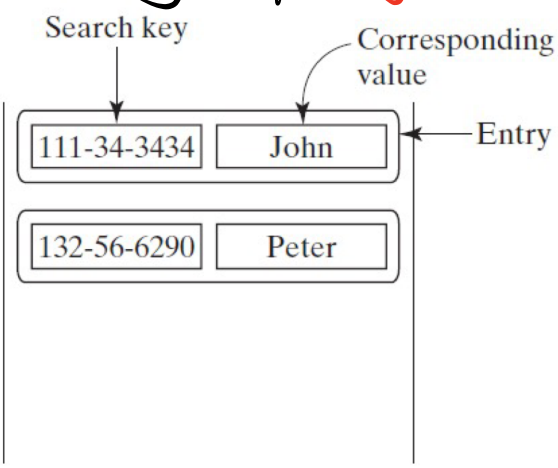
List index their entries based on the position in the list.

Dictionary: a "bag" of values, each with its own label.

Dictionaries index the elements with a key

key: the lookup tag

The dictionary maps keys to their element values



```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> dct = dict()
>>> dct['age'] = 21
>>> dct['height'] = 183
>>> print(dct)
{'age': 21, 'height': 183}
>>> dct['age'] = 23
>>> print(dct)
{'age': 23, 'height': 183}
```

2. Dictionary

```
>>> purse = dict()
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 3}
>>> print(purse['candy'])
3
>>> purse['candy']=purse['candy']+2
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 5}
>>> purse[3] = 77
>>> print(purse)
{3: 77, 'money': 12, 'tissues': 75, 'candy': 5}
```

3. Dictionary literals (constant)

Dictionary literals use curly braces and have list of "key: value" pairs

You can make an empty dictionary using empty curly braces

```
>>> jjj = {'chuck':1,'fred':42, 'jan':100}
>>> print(jjj)
{'fred': 42, 'chuck': 1, 'jan': 100}
>>> ooo={}
>>> print(ooo)
{}
```

4. Tracebacks

It is an error to reference a key which is not in the dictionary

We can use the in operator to see if a key is in the dictionary

```
>>> ccc=dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    print(ccc['csev'])
KeyError: 'csev'
>>> 'csev' in ccc
False
```

5. Counting with a dictionary

Write a program to instruct the user to continuously input some words, and use dictionary to count how many times a word has been inputted before.

```
wordDict = dict()

while True:
    word = input('Enter a word:')

    if word in wordDict:
        wordDict[word]=wordDict[word]+1
    elif word=='done':
        break
    else:
        wordDict[word]=1

print('The result of word count:')
print(wordDict)
```

6. get()

Check if a key is already in a dictionary, and assuming a default value if the key is not there

```
>>> counts = {'aaa':1,'bbb':2,'ccc':5}
>>> print(counts.get('eee',0))
0
```

Write a program to instruct the user to input a line of texts, and use dictionary to count how many times each word has been seen in this line. You should use the get() method in this program.

```
wordDict = dict()

textLine = input('Enter a line:')
words = textLine.split()

print('Counting...')
for word in words:
    wordDict[word] = wordDict.get(word,0) + 1

print('The result of word count:')
print(wordDict)
```

7. Definite loops

A for loop goes through all the keys in that dictionary and looks up the values.

```
counts = {'chuck':1,'fred':42,'jan':100}
for key in counts:
    print(key, counts[key])
```

reserved word

```
jan 100
fred 42
chuck 1
```

CASE STUDY: COUNTING THE OCCURRENCES OF WORDS IN A TEXT

- Write a program counts the occurrences of words in a text and displays the words and their occurrences in ascending order of the words
- The program uses a dictionary to store a pair consisting of a word and its count
- For each word, check whether it is already a key in the map:
  - If not, add the key and value 1 to the map
  - Otherwise, increase the value for the word (key) by 1 in the map

ANSWER

```
filename = input("input the file name:")
file = open(filename, "r")
wordDict = dict()
for line in file:
        words = line.split()
        for word in words:
                if word in wordDict:
                        wordDict[word] += 1
                else:
                        wordDict[word] = 1

for word in wordDict:
        print(word,":",wordDict[word])
```

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

ANSWER 2

```
filename = input("input the file name:")
file = open(filename, "r")
wordDict = dict()
for line in file:
        words = line.split()
        for word in words:
                wordDict[word] = wordDict.get(word, 0) + 1

for word in wordDict:
        print(word,":",wordDict[word])
```

8. Retrieving lists of keys and values

You can get a list of keys, values or items (both) from a dictionary.

```
>>> jjj = {'chuck':1,'fred':42,'jan':100}
>>> print(list(jjj))
['jan', 'fred', 'chuck']

>>> print(list(jjj.keys()))
['jan', 'fred', 'chuck']

>>> print(list(jjj.values()))
[100, 42, 1]
>>> print(list(jjj.items()))
[('jan', 100), ('fred', 42), ('chuck', 1)]
```

9. Two iteration variables

We loop through the key-value pairs in a dictionary using two iteration variables

```
counts = {'chuck':1,'fred':42,'jan':100}
for key,value in counts.items():
    print(key,value)
```

each key — corresponding value

```
chuck 1
fred 42
jan 100
```

§3 Tuples

1. Basic information

1° Tuples are another type of sequence that function more like a list. They have elements which are indexed starting from 0.

```
>>> x=('Glenn','Sally','Joseph')
>>> print(x)
('Glenn', 'Sally', 'Joseph')
>>> y=(1,9,2)
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for i in y:
        print(i)

1
9
2
```

2° But unlike a list, you cannot change tuple contents after you create it —— like a string

List

```
>>> x=[9,8,7]
>>> x[2]=6
>>> print(x)
[9, 8, 6]
```

String

```
>>> y='abc'
>>> y[2]='e'
Traceback (most recent call last)
:
  File "<pyshell#23>", line 1, in
<module>
    y[2]='e'
TypeError: 'str' object does not
support item assignment
```

Tuple

```
>>> z=(5,4,3)
>>> z[2]
3
>>> z[2]=0
Traceback (most recent call last)
:
  File "<pyshell#28>", line 1, in
<module>
    z[2]=0
TypeError: 'tuple' object does no
t support item assignment
```

```
>>> x=(1,2,3)
>>> x.sort()
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    x.sort()
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    x.append(5)
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    x.reverse()
AttributeError: 'tuple' object has no attribute 'reverse'
```

3° Tuples are more efficient than list.

2. Tuples and dictionaries

The item() method in dictionaries return a list of (key, value) tuples

```
>>> d=dict()
>>> d['csev']=2
>>> d['cwen']=4
>>> for (k,v) in d.items():
        print(k,v)


csev 2
cwen 4
>>> tups = d.items()
>>> print(tups)
dict_items([('csev', 2), ('cwen', 4)])
>>> print(list(tups))
[('csev', 2), ('cwen', 4)]

>>> tups = list(tups)
>>> tups[1]
('cwen', 4)
```

3. Tuples are comparable

The comparison operators work with tuples and other sequences. If the first item is equal, Python goes on to the next element, until it finds the elements which are different.

```
>>> (0, 1, 2)<(5, 1, 2)
True
>>> (0, 1, 200000)<(0, 3, 4)
True
>>> ('Jones','Sally')<('Jones','Fred')
False
>>> ('Jones','Sally')>('Adams','Sam')
True
```

4. sort & sorted()

sort a list of tuples to get a sorted version of a dictionary

```
>>> d={'a':10,'b':1,'c':22}
>>> t=d.items()
>>> t=list(t)
>>> t
[('c', 22), ('b', 1), ('a', 10)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
```

We can take a sequence as a parameter and returns a sorted sequence

```
>>> d={'a':10,'b':1,'c':22}
>>> t=d.items()
>>> t=list(t)
>>> t
[('c', 22), ('b', 1), ('a', 10)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
```

```
>>> d={'a':10,'b':1,'c':22}
>>> d.items()
dict_items([('c', 22), ('b', 1), ('a', 10)])
>>> t=sorted(list(d.items()))
>>> t
[('a', 10), ('b', 1), ('c', 22)]

>>> for k,v in t:
        print(k,v)


a 10
b 1
c 22
```

5. Sort by values instead of key

If we could construct a list of tuples of the form (value, key), we could sort by value

Write a program, which sorts the elements of a dictionary by the value of each element

```
myDict = {'a':100,'b':10,'c':11,'d':21}
myList = myDict.items()

newList = list()
for e in myList:
    newTuple = (e[1],e[0])
    newList.append(newTuple)

print('The list before sorting:')
print(newList)
print('The list after sorting:')
print(sorted(newList))
```

Write a program to open a text file, and find out the 10 most common words in this file.

```
fhand = open('myhost.txt','r')
counts = dict()
for line in fhand:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0)+1

lst = list()
for key,val in counts.items():
    lst.append((val,key))

lst.sort(reverse = True)

for val,key in lst[:10]:
    print(key,val)
```