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

INTRODUCTION TO COMPUTER SCIENCE: PROGRAMMING METHODOLOGY

TUTORIAL 7

LIST

1.List is a Sequence

- Similar to strings, lists are sequences; lists are sequences of some elements, which can be any objects.

Table 10.1 Common Operations for Sequence s

Operation	*Description*
`x in s`	True if element x is in sequence s.
`x not in s`	True if element x is not in sequence s.
`s1 + s2`	Concatenates two sequences s1 and s2.
`s * n, n * s`	n copies of sequence s concatenated.
`s[i]`	ith element in sequence s.
`s[i : j]`	Slice of sequence s from index i to j − 1.
`len(s)`	Length of sequence s, i.e., the number of elements in s.
`min(s)`	Smallest element in sequence s.
`max(s)`	Largest element in sequence s.
`sum(s)`	Sum of all numbers in sequence s.
`for loop`	Traverses elements from left to right in a `for` loop.
`<, <=, >, >=, =, !=`	Compares two sequences.

```
>>> lst1=[1,2,3,'a','b',print,input]
>>> lst1[5]
<built-in function print>
>>> lst1[1:6]
[2, 3, 'a', 'b', <built-in function print>]
>>> lst2=[1,2]
>>> lst3=[3,4,5]
>>> lst2+lst3
[1, 2, 3, 4, 5]
>>> lst2*3
[1, 2, 1, 2, 1, 2]
>>> len(lst1)
7
>>> max(lst2)
2
>>> sum(lst3)
12
>>> 'a' in lst1
True
>>> lst2>lst3
False
>>>
```

2.Methods of List

list	
append(x: object): None	Adds an element x to the end of the list.
count(x: object): int	Returns the number of times element x appears in the list.
extend(l: list): None	Appends all the elements in l to the list.
index(x: object): int	Returns the index of the first occurrence of element x in the list.
insert(index: int, x: object): None	Inserts an element x at a given index. Note that the first element in the list has index 0.
pop(i): object	Removes the element at the given position and returns it. The parameter i is optional. If it is not specified, list.pop() removes and returns the last element in the list.
remove(x: object): None	Removes the first occurrence of element x from the list.
reverse(): None	Reverses the elements in the list.
sort(): None	Sorts the elements in the list in ascending order.

```
>>> lst=[]
>>> lst.append('a')
>>> lst
['a']
>>> lst2=[1,2,3]
>>> lst.extend(lst2)
>>> lst
['a', 1, 2, 3]
>>> lst.insert(1,'b')
>>> lst
['a', 'b', 1, 2, 3]
>>> lst.append(1)
>>> lst
['a', 'b', 1, 2, 3, 1]
>>> lst.count(1)
2
>>> lst.pop()
1
>>> lst
['a', 'b', 1, 2, 3]
>>> lst.remove(1)
>>> lst
['a', 'b', 2, 3]
>>> lst.reverse()
>>> lst
[3, 2, 'b', 'a']
>>>
```

3.Dictionary

Dictionary

- 1.Dictionary labels each element with its key.
- 2.Elements in a dictionary have no order.

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

The get() method

- This pattern of checking to see if a key is already in a dictionary, and assuming a default value if the key is not there is so common, that there is a method called get() that does this for us.

```
>>> counts={'aaa':1,'bbb':2,'ccc':5}
>>> counts.get('bbb',0)
2
>>> counts.get('ddd',0)
0
>>> counts.get('eee',3)
3
>>> counts
{'aaa': 1, 'ccc': 5, 'bbb': 2}
>>> counts['eee']=counts.get('eee',3)
>>> counts
{'aaa': 1, 'ccc': 5, 'bbb': 2, 'eee': 3}
>>>
```

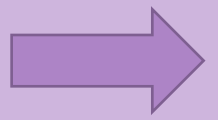
4.Retrieving keys and values

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

```
>>> LeBron={'James':23,'Durant':35,'Harden':13,'Irving':11,'Leonard':2}
>>> print(list(LeBron))
['Irving', 'Leonard', 'James', 'Harden', 'Durant']
>>> print(list(LeBron.keys()))
['Irving', 'Leonard', 'James', 'Harden', 'Durant']
>>> print(list(LeBron.values()))
[11, 2, 23, 13, 35]
>>> print(list(LeBron.items()))
[('Irving', 11), ('Leonard', 2), ('James', 23), ('Harden', 13), ('Durant', 35)]
>>> type(LeBron.keys()),type(LeBron.values()),type(LeBron.items())
(<class 'dict_keys'>, <class 'dict_values'>, <class 'dict_items'>)
>>>
```

- Bonus: two iteration variables.

```
>>> for key,value in LeBron.items():
        print(key,value)
```



```
Irving 11
Leonard 2
James 23
Harden 13
Durant 35
```

Q1: Count occurences

- Write a program that reads some integers between 1 and 100 and counts the occurrences of each.

```
Enter integers between 1 and 100: 2 5 6 5 4 3 23 43 2 ↵ Enter
2 occurs 2 times
3 occurs 1 time
4 occurs 1 time
5 occurs 2 times
6 occurs 1 time
23 occurs 1 time
43 occurs 1 time
```

Note that if a number occurs more than one time, the plural word “times” is used in the output.

Q2: Display distinct numbers

- Write a program that reads in numbers separated by a space in one line and displays distinct numbers (i.e., if a number appears multiple times, it is displayed only once).

Hint: Read all the numbers and store them in list1. Create a new list list2. Add a number in list1 and list2. If the number is already in the list, ignore it.

Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2 ↵Enter
The distinct numbers are: 1 2 3 6 4 5
```

Q3: Compute mean and deviation

- Compute the standard deviation of n numbers:

$$mean = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \ldots + x_n}{n} \qquad deviation = \sqrt{\frac{\sum_{i=1}^{n} (x_i - mean)^2}{n - 1}}$$

Your program should contain the following functions: def deviation(x): and def mean(x):. Write a test program that prompts the user to enter a list of numbers and displays the mean and standard deviation, as shown in the following sample run:

```
Enter numbers: 1.9 2.5 3.7 2 1 6 3 4 5 2 ↵Enter
The mean is 3.11
The standard deviation is 1.55738
```

Q4:Test sorted list

- Write the following function that return True if the list is already sorted in increasing order: def isSorted(lst): .Write a test program that prompts the user to enter a list and displays whether the list is sorted or not. Here is a sample run:

```
Enter list: 1 1 3 4 4 5 7 9 10 30 11 Enter
The list is not sorted
```

```
Enter list: 1 1 3 4 4 5 7 9 10 30 Enter
The list is already sorted
```

Q5:Word recitation

- The file Dictionary.txt stores some words along with their meanings in English, using colon ":" to separate. Write a program to randomly select an English word from the file and allow users to guess it by looking at the description of it on the screen. The program will give you feedback whether your guess is correct or not. If wrong, it will tell you the correct answer. Words in the file won't appear twice and after finishing guessing all words in the file, the program will tell you how many words you guess correctly.