大学计算机入门课

Class #8

［教学目的］

- Get more familiar and confident about using dictionary
- Know how to get information from a file
- And write file via python

［课程大纲］

- Part 1: more on dictionary
- Part 2: read file
- Part 3: write file.

Part 1: more function about using dictionary

- Recall: dictionary

Dict = {Key: Value}
we can get the value of particular key by the following command:

```
>>> Dict[Key]
Value
```

We can add a new key with the value in this dict by:

```
>>> Dict[newKey] = ValueForNewKey
# this modify the original dictionary
>>> Dict
{Key: Value, newKey: ValueForNewKey}
```

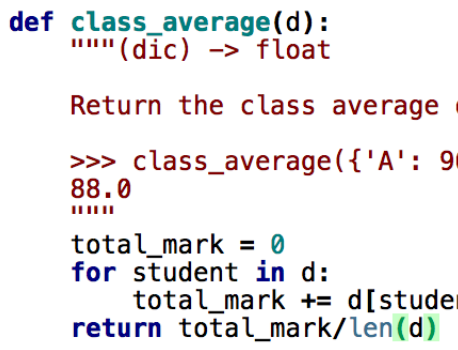
And we can modify the value for particular by the same way of adding a new key in the dictionary.

And recall the dictionary operation

Method	Description
D.clear()	Removes all key/value pairs from dictionary D.
D.get(k)	Returns the value associated with key k, or None if the key isn't present. (Usually you'll want to use D[k] instead.)
D.get(k, v)	Returns the value associated with key k, or a default value v if the key isn't present.
D.keys()	Returns dictionary D's keys as a set-like object—entries are guaranteed to be unique.
D.items()	Returns dictionary D's (key, value) pairs as set-like objects.
D.pop(k)	Removes key k from dictionary D and returns the value that was associated with k—if k isn't in D, an error is raised.
D.pop(k, v)	Removes key k from dictionary D and returns the value that was associated with k; if k isn't in D , returns v.
D.setdefault(k)	Returns the value associated with key k in D.
D.setdefault(k, v)	Returns the value associated with key k in D; if k isn't a key in D, adds the key k with the value v to D and returns v.
D.values()	Returns dictionary D's values as a list-like object—entries may or may not be unique.
D.update(other)	Updates dictionary D with the contents of dictionary other; for each key in other, if it is also a key in D, replaces that key in D's value with the value from other; for each key in other, if that key isn't in D, adds that key/value pair to D.

- 掌握了这些 dictionary 的基本运用，上节课我们也接触了几个与 dictionary 有关的 function

```
def pair_name_age(l1, l2):
    """(list[str], list[int]) -> dic

    Precondition: len(l1) == len(l2)

    Return the dictionary with name int l1 as the key and age in l2 as the value.

    >>> result = pair_name_age(["Annie", "Cherry", "Alex"], [18, 21, 22])
    >>> result == {'Annie': 18, 'Cherry': 21, 'Alex': 22}
    True
    """
    dic_result  = {}
    i = 0
    for i in range(len(l1)):
        dic_result[l1[i]] = l2[i]

    return dic_result
```

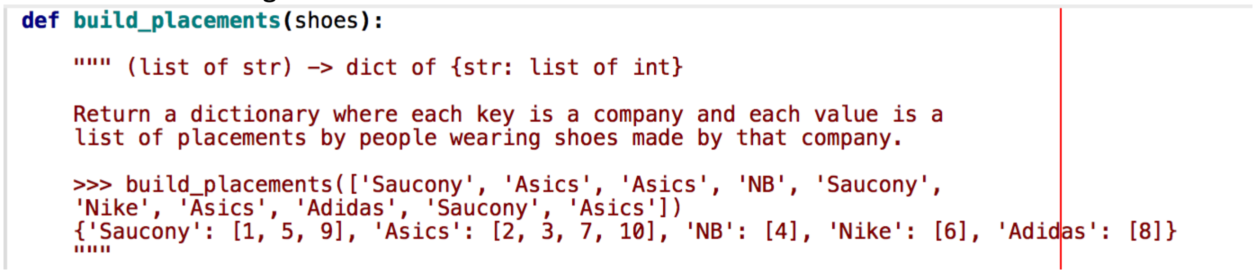


```python
def class_average(d):
    """(dic) -> float

    Return the class average of the value in each key in d.

    >>> class_average({'A': 96, 'B': 89, 'C': 67, 'D': 100})
    88.0
    """
    total_mark = 0
    for student in d:
        total_mark += d[student]
    return total_mark/len(d)
```

更多的 function：

– let’s consider a function that return a dictionary where each key is a company and each value is a list of placements by people wearing shoes made by the company.

Look at the docstring:

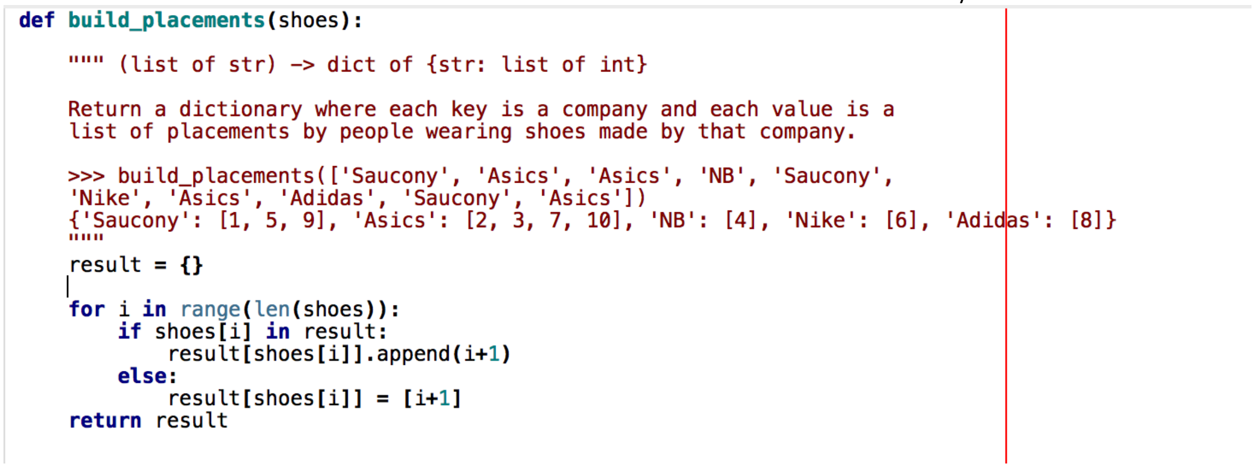


```python
def build_placements(shoes):

    """ (list of str) -> dict of {str: list of int}

    Return a dictionary where each key is a company and each value is a
    list of placements by people wearing shoes made by that company.

    >>> build_placements(['Saucony', 'Asics', 'Asics', 'NB', 'Saucony',
    'Nike', 'Asics', 'Adidas', 'Saucony', 'Asics'])
    {'Saucony': [1, 5, 9], 'Asics': [2, 3, 7, 10], 'NB': [4], 'Nike': [6], 'Adidas': [8]}
    """
```

# want each shoe brand as a key in the dictionary
# each key appears once
# if we start with an empty dictionary, need to be able to
# add keys and their value (list of ints)
# if we find a key that's already in the dictionary, we
# need to update its value WITHOUT overwriting it

and how should we iterate over shoes list?
We will use range (other ways will work through)

So now, let’s complete this function body:



```
def build_placements(shoes):

    """ (list of str) -> dict of {str: list of int}

    Return a dictionary where each key is a company and each value is a
    list of placements by people wearing shoes made by that company.

    >>> build_placements(['Saucony', 'Asics', 'Asics', 'NB', 'Saucony',
    'Nike', 'Asics', 'Adidas', 'Saucony', 'Asics'])
    {'Saucony': [1, 5, 9], 'Asics': [2, 3, 7, 10], 'NB': [4], 'Nike': [6], 'Adidas': [8]}
    """
    result = {}

    for i in range(len(shoes)):
        if shoes[i] in result:
            result[shoes[i]].append(i+1)
        else:
            result[shoes[i]] = [i+1]
    return result
```

- now, let’s consider a more complicate function
  given a list L = [1,2,3,4,5]
  this function will generate a dictionary such that each value in the given list is a key, and the value will be the term next to it. The last term will have the value for the first item

  首先，我们先考虑一下这个 function， 它的每个 value 用什么形式来表示比较好呢？
  如果 given
  >>> L = [1,2,3,4,5]
  >>> next_item(L)
  {1:2, 2:3, 3:4, 4:5, 5:1}
  所以，我们会想它的每个 value 可以是 the data type in L。

  但是我们知道 list 会有 duplicate 的值出现，但是作为 dictionary 的 key 的话，我们只能够记录一次，所以选择记忆会让我们丧失一些信息。

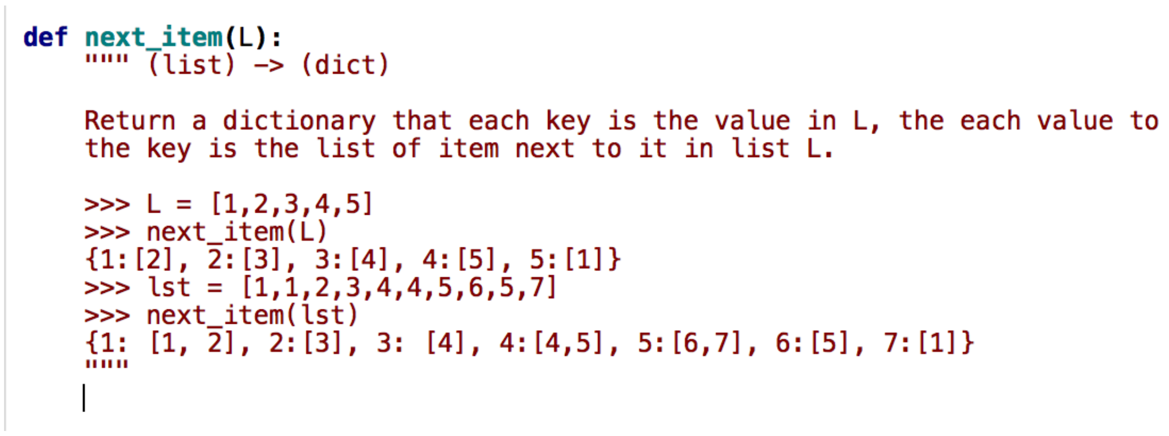
  所以每个 value 用 list 来表示更合适

  >>> L = [1,2,3,4,5]
  >>> next_item(L)
  {1:[2], 2:[3], 3:[4], 4:[5], 5:[1]}

  >>> lst = [1,1,2,3,4,4,5,6,5,7]
  >>> next_item(lst)
  {1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]}

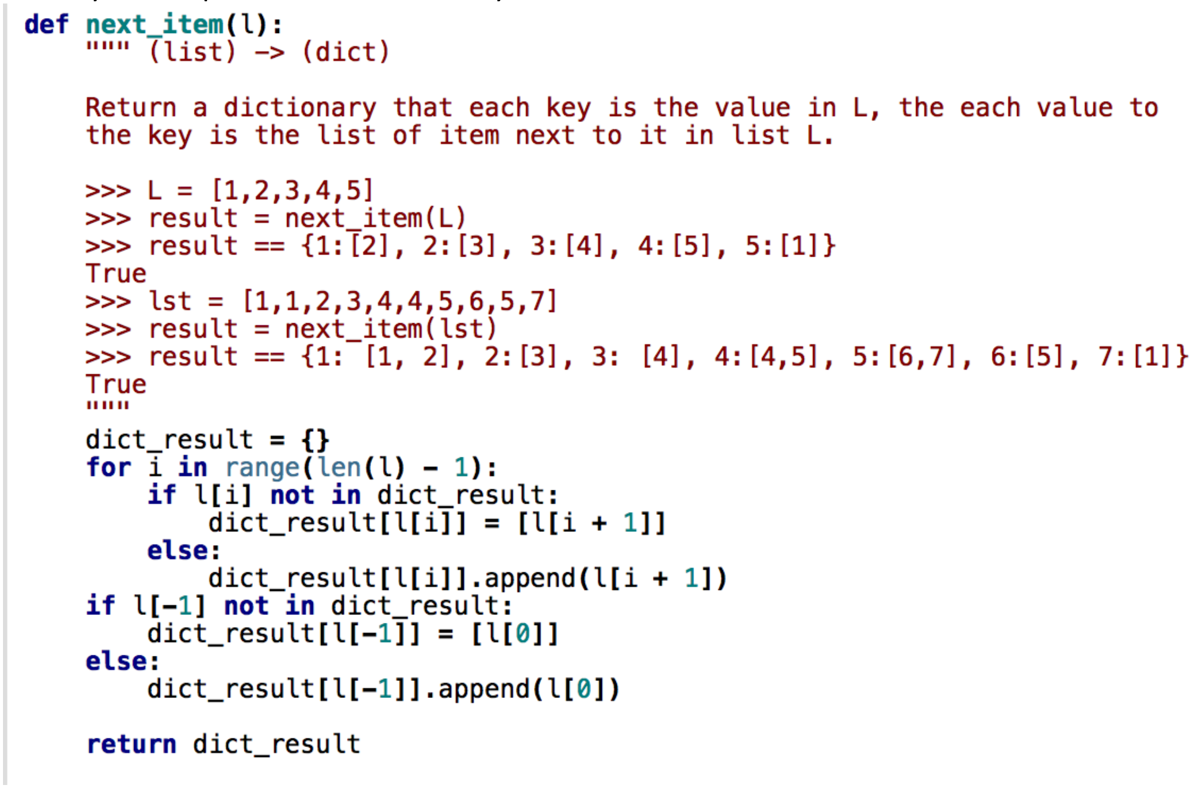
  用这种方式的话，我们所有的信息都可以储存进去。

现在，让我们 complete our function docstring。



```python
def next_item(L):
    """ (list) -> (dict)

    Return a dictionary that each key is the value in L, the each value to
    the key is the list of item next to it in list L.

    >>> L = [1,2,3,4,5]
    >>> next_item(L)
    {1:[2], 2:[3], 3:[4], 4:[5], 5:[1]}
    >>> lst = [1,1,2,3,4,4,5,6,5,7]
    >>> next_item(lst)
    {1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]}
    """
```

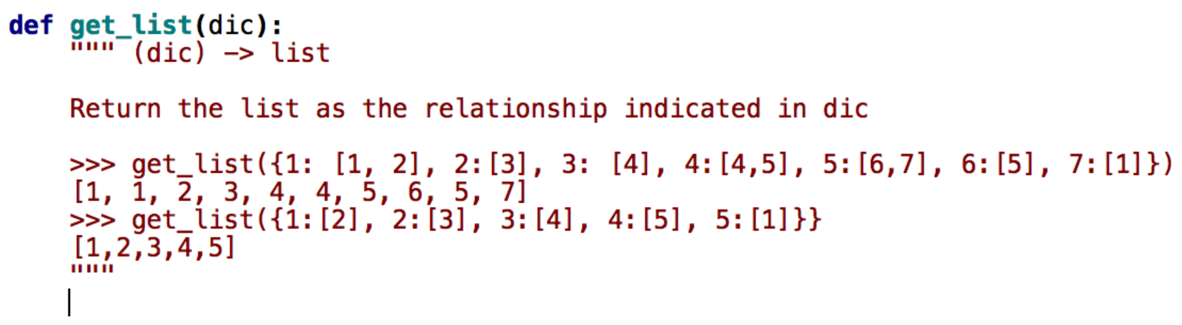
let's try to complete the function body.



```python
def next_item(l):
    """ (list) -> (dict)

    Return a dictionary that each key is the value in L, the each value to
    the key is the list of item next to it in list L.

    >>> L = [1,2,3,4,5]
    >>> result = next_item(L)
    >>> result == {1:[2], 2:[3], 3:[4], 4:[5], 5:[1]}
    True
    >>> lst = [1,1,2,3,4,4,5,6,5,7]
    >>> result = next_item(lst)
    >>> result == {1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]}
    True
    """
    dict_result = {}
    for i in range(len(l) - 1):
        if l[i] not in dict_result:
            dict_result[l[i]] = [l[i + 1]]
        else:
            dict_result[l[i]].append(l[i + 1])
    if l[-1] not in dict_result:
        dict_result[l[-1]] = [l[0]]
    else:
        dict_result[l[-1]].append(l[0])

    return dict_result
```

pay attention to our examples in the docstring.

- Reverse thinking!
  Now, consider a function given the dictionary like previous question we generating, and given a list such represent the relationship.
  And the key was written in the presence of order in the list.



```
def get_list(dic):
    """ (dic) -> list

    Return the list as the relationship indicated in dic

    >>> get_list({1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]})
    [1, 1, 2, 3, 4, 4, 5, 6, 5, 7]
    >>> get_list({1:[2], 2:[3], 3:[4], 4:[5], 5:[1]}}
    [1,2,3,4,5]
    """
```

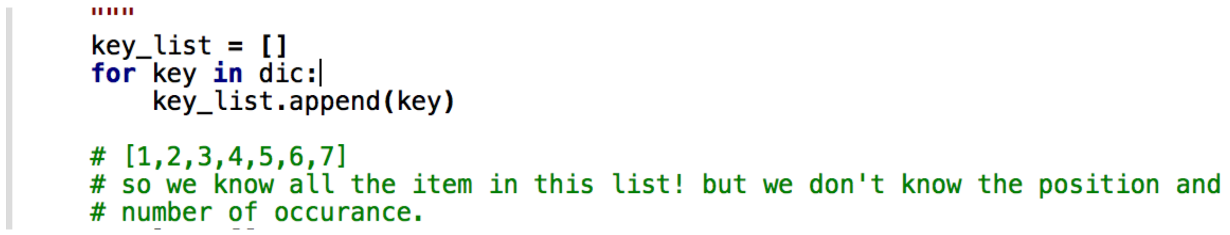
how to achieve this?
Let's thinking deeply!
consider the work process

{1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]}

1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6 -> 5 -> 7 -> | stop

and the key moves!

Firstly, about order!
Since dictionary does not have order! So we have to put the key into a list, in order to maintain the order.



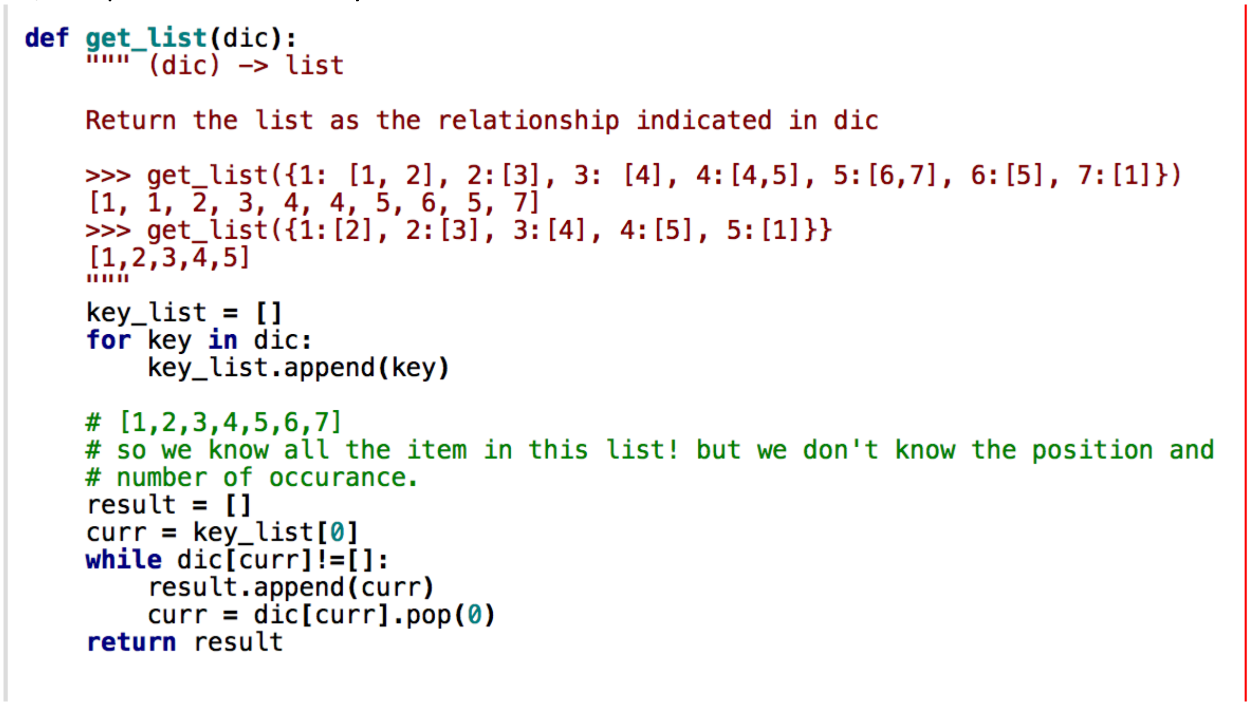
```
    """
    key_list = []
    for key in dic:
        key_list.append(key)

    # [1,2,3,4,5,6,7]
    # so we know all the item in this list! but we don't know the position and
    # number of occurance.
```

we start with key_list[0] as the key
and start to move the key!

but there is still a problem! If one key has a list of more than one item, how can we know, which one will we pass the key to?
And when to stop???

We need to stop the process, so while loop is the good choice! And once we move the current key to a value in Value list, we can remove this value!
After every key has an empty list we can stop!

So, complete the code body now!



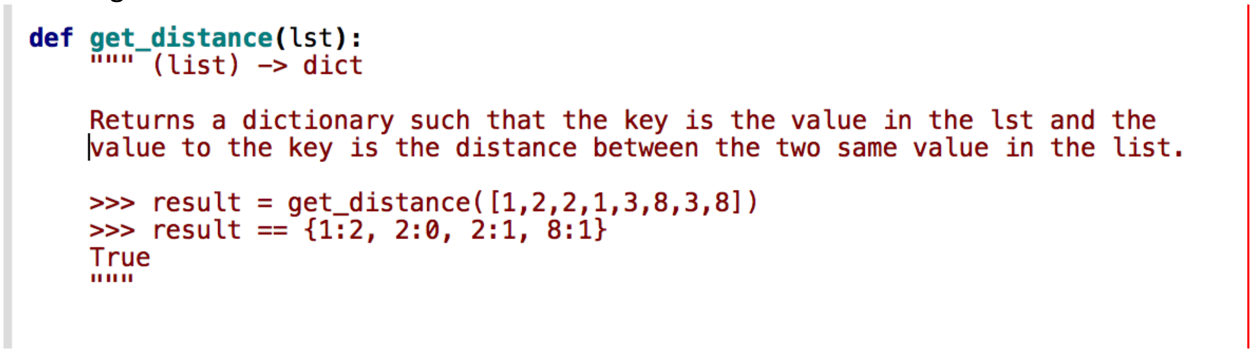
```python
def get_list(dic):
    """ (dic) -> list

    Return the list as the relationship indicated in dic

    >>> get_list({1: [1, 2], 2:[3], 3: [4], 4:[4,5], 5:[6,7], 6:[5], 7:[1]})
    [1, 1, 2, 3, 4, 4, 5, 6, 5, 7]
    >>> get_list({1:[2], 2:[3], 3:[4], 4:[5], 5:[1]})
    [1,2,3,4,5]
    """
    key_list = []
    for key in dic:
        key_list.append(key)

    # [1,2,3,4,5,6,7]
    # so we know all the item in this list! but we don't know the position and
    # number of occurance.
    result = []
    curr = key_list[0]
    while dic[curr]!=[]:
        result.append(curr)
        curr = dic[curr].pop(0)
    return result
```

- Let’s see another example

Given a list with every value in this list appear exact twice.
Consider a function that returns a dictionary such that the key is the value in the list and the value to the key is the distance between the two same value in the list.

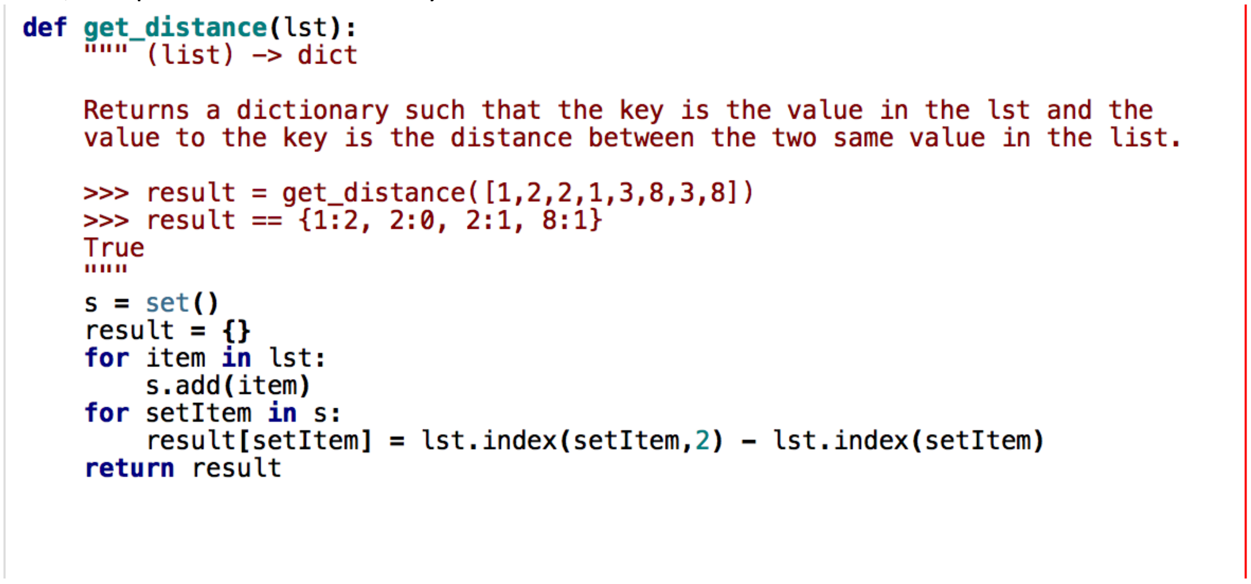
[1,2,2,1,3,8,3,8]
{1:2, 2:0, 2:1, 8:1}

docstring:



```python
def get_distance(lst):
    """ (list) -> dict

    Returns a dictionary such that the key is the value in the lst and the
    value to the key is the distance between the two same value in the list.

    >>> result = get_distance([1,2,2,1,3,8,3,8])
    >>> result == {1:2, 2:0, 2:1, 8:1}
    True
    """
```

Now, complete the function body now.

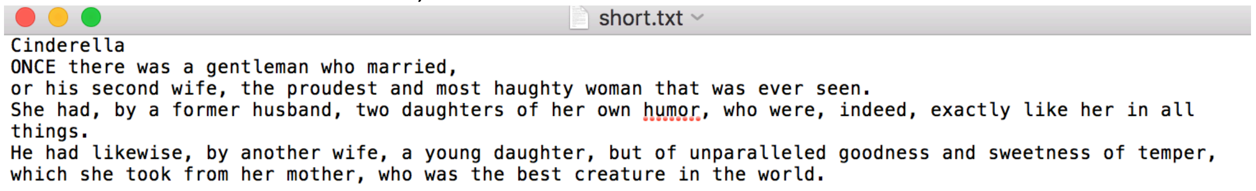


```python
def get_distance(lst):
    """ (list) -> dict

    Returns a dictionary such that the key is the value in the lst and the
    value to the key is the distance between the two same value in the list.

    >>> result = get_distance([1,2,2,1,3,8,3,8])
    >>> result == {1:2, 2:0, 2:1, 8:1}
    True
    """
    s = set()
    result = {}
    for item in lst:
        s.add(item)
    for setItem in s:
        result[setItem] = lst.index(setItem,2) - lst.index(setItem)
    return result
```

Part 2– read file.

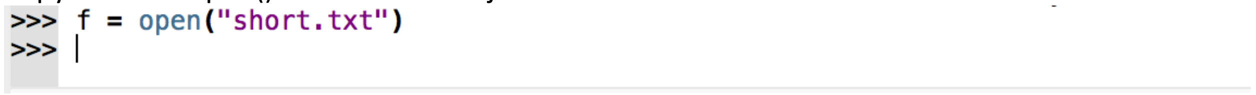
在现实生活中，各个软件中都会存在 database，要有文件来储存信息。比如公司每个人的信息或者账务单。所以阅读 file 读取信息是必不可少的。但是当信息量特别大的时候，人工阅读找到想要的信息会花费大量的时间与人力，所以我们今天学习用 python 读取文件，并且提取相应的信息。同时，我们还会学习怎么用 python 编写。这些都是在学习计算机时非常重要的技能。

Look at the file we want to read,


short.txt

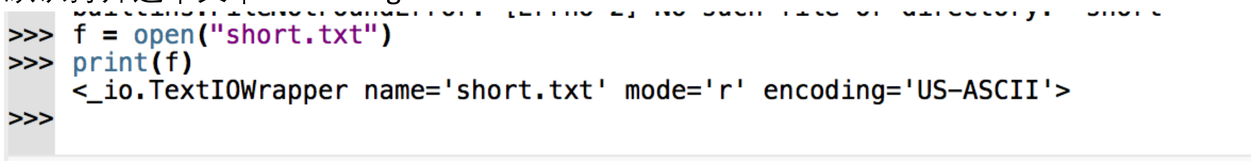
```
Cinderella
ONCE there was a gentleman who married,
or his second wife, the proudest and most haughty woman that was ever seen.
She had, by a former husband, two daughters of her own humor, who were, indeed, exactly like her in all
things.
He had likewise, by another wife, a young daughter, but of unparalleled goodness and sweetness of temper,
which she took from her mother, who was the best creature in the world.
```

In python use open() returns a file object.



```
>>> f = open("short.txt")
>>> |
```

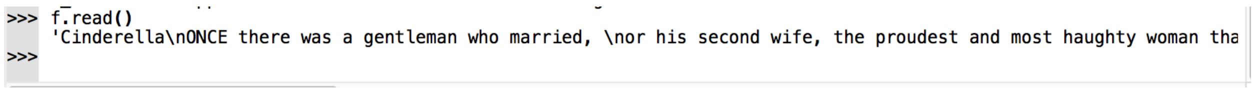
>>> open(filename)
默认打开这个文章 to reading.



```
>>> f = open("short.txt")
>>> print(f)
<_io.TextIOWrapper name='short.txt' mode='r' encoding='US-ASCII'>
>>>
```

(the mode here is “r” representing reading.)

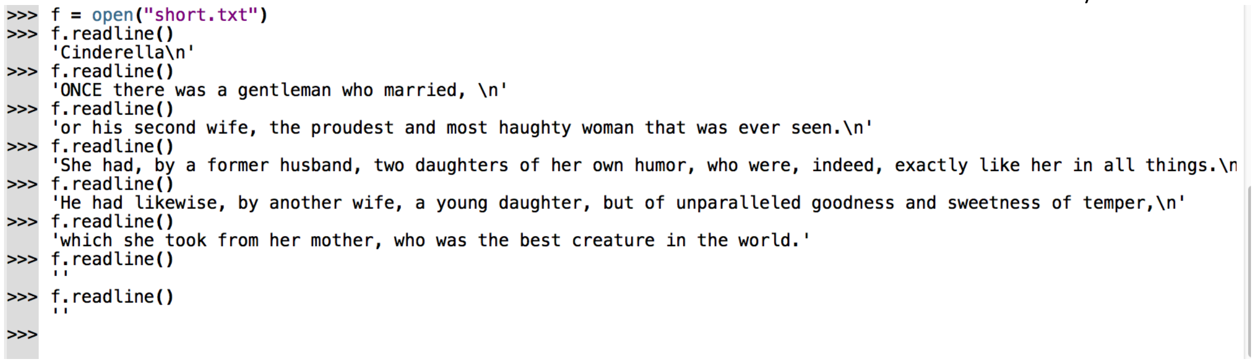
用 f.read()
来读取整个文件内容。



```
>>> f.read()
'Cinderella\nONCE there was a gentleman who married, \nor his second wife, the proudest and most haughty woman tha
>>>
```

注: “\n” 代表换行，
在文件里换行的时候这么用。

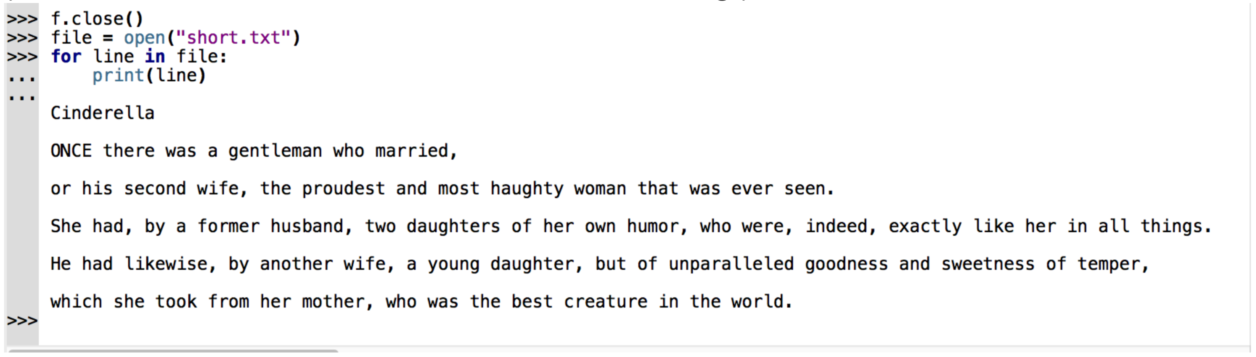
如果想一句一句的读文件，我们可以用
>>> f.readline()
注意，当我们没有关闭这个文件的话，readline 是累加记忆。每一次实施 readline command 都会 move 到上一次阅读的下一行。



```
>>> f = open("short.txt")
>>> f.readline()
'Cinderella\n'
>>> f.readline()
'ONCE there was a gentleman who married, \n'
>>> f.readline()
'or his second wife, the proudest and most haughty woman that was ever seen.\n'
>>> f.readline()
'She had, by a former husband, two daughters of her own humor, who were, indeed, exactly like her in all things.\n
>>> f.readline()
'He had likewise, by another wife, a young daughter, but of unparalleled goodness and sweetness of temper,\n'
>>> f.readline()
'which she took from her mother, who was the best creature in the world.'
>>> f.readline()
''
>>> f.readline()
''
>>>
```

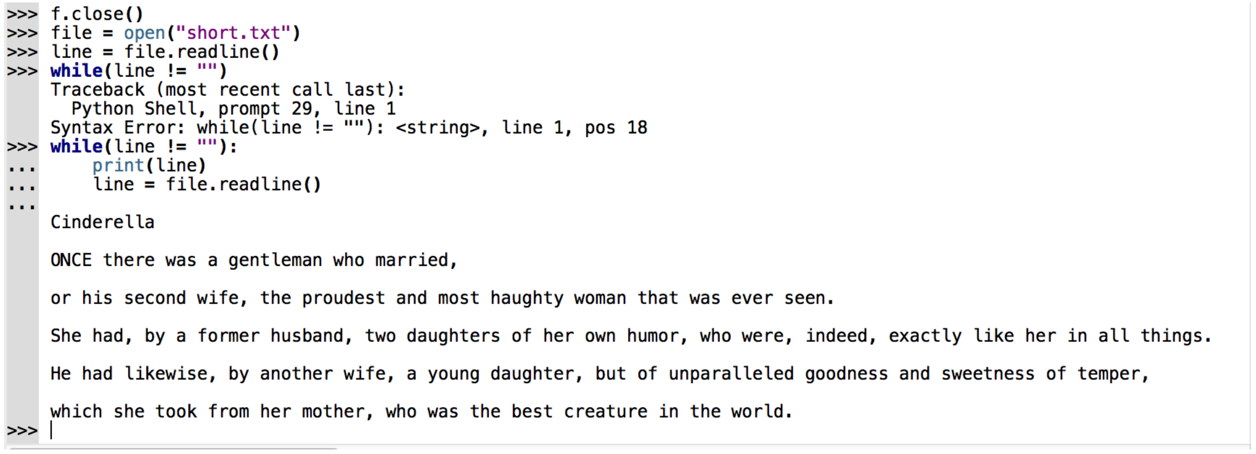
当下一行没有内容的时候，会 return 一个 empty 的 string。

Use for loops to read de whole thing line by line:
(note: remember to close the file after last time reading!)



```
>>> f.close()
>>> file = open("short.txt")
>>> for line in file:
...     print(line)
...
Cinderella

ONCE there was a gentleman who married,

or his second wife, the proudest and most haughty woman that was ever seen.

She had, by a former husband, two daughters of her own humor, who were, indeed, exactly like her in all things.

He had likewise, by another wife, a young daughter, but of unparalleled goodness and sweetness of temper,

which she took from her mother, who was the best creature in the world.
>>>
```

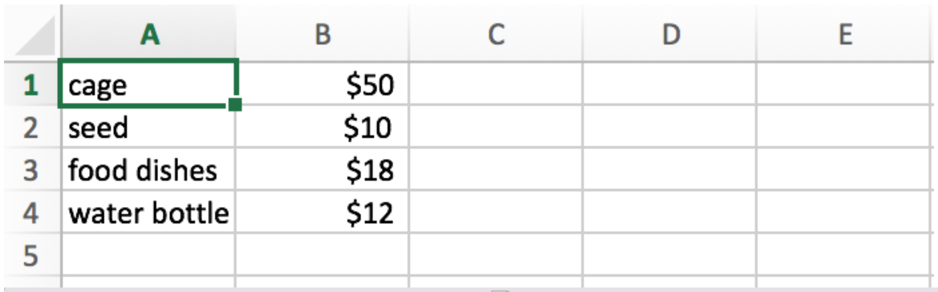
同样的，我们也可以用 while loop 来实现。



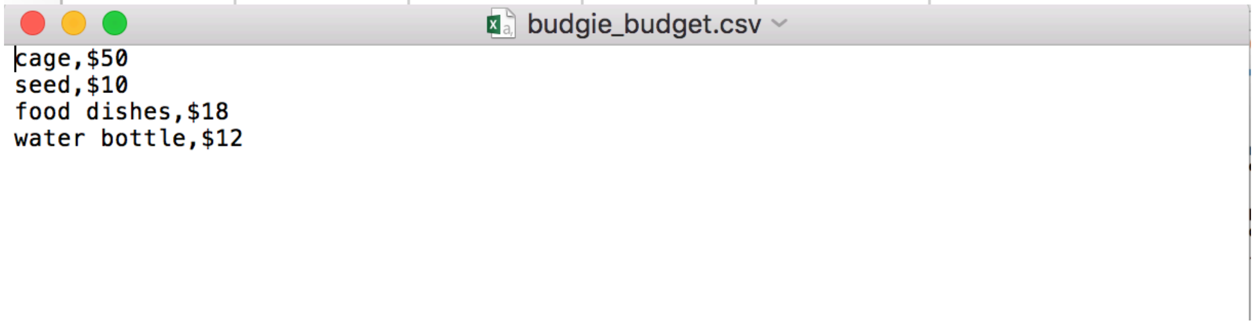
```
>>> f.close()
>>> file = open("short.txt")
>>> line = file.readline()
>>> while(line != "")
Traceback (most recent call last):
  Python Shell, prompt 29, line 1
Syntax Error: while(line != ""): <string>, line 1, pos 18
>>> while(line != ""):
...     print(line)
...     line = file.readline()
...
Cinderella

ONCE there was a gentleman who married,

or his second wife, the proudest and most haughty woman that was ever seen.

She had, by a former husband, two daughters of her own humor, who were, indeed, exactly like her in all things.

He had likewise, by another wife, a young daughter, but of unparalleled goodness and sweetness of temper,

which she took from her mother, who was the best creature in the world.
>>> |
```

注意：每条 line 都是一个 string。

Write a function that return a dictionary with the information in budgie_budget.csv



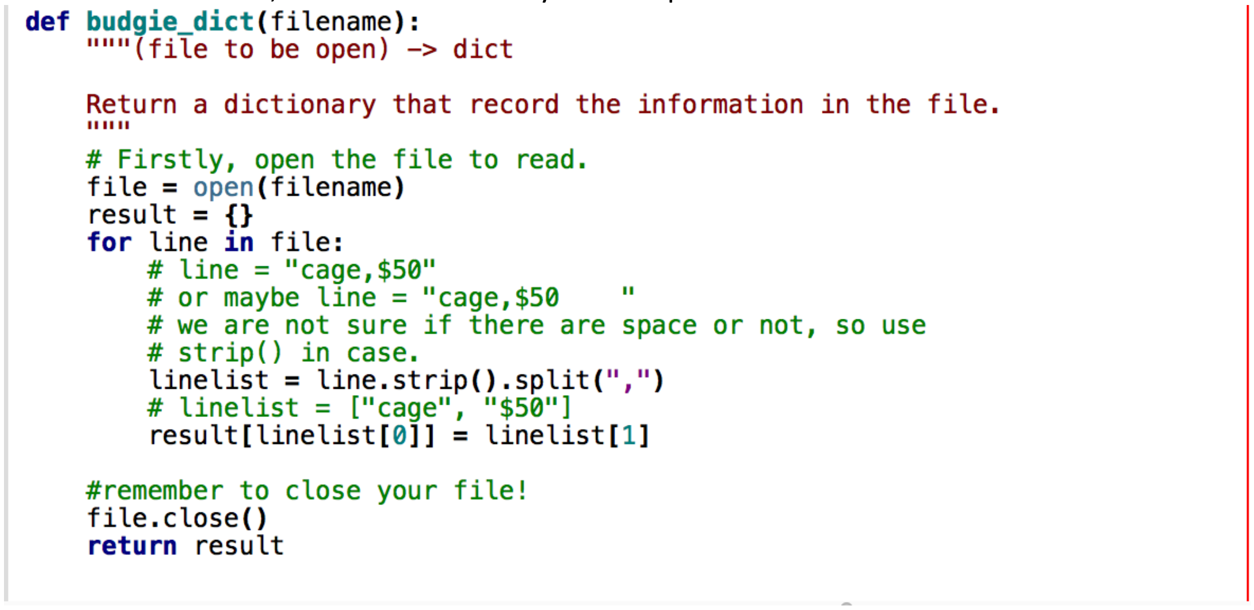
	A	B	C	D	E
1	cage	$50			
2	seed	$10			
3	food dishes	$18			
4	water bottle	$12			
5					



budgie_budget.csv

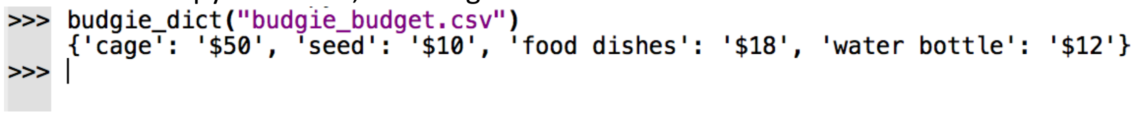
```
cage,$50
seed,$10
food dishes,$18
water bottle,$12
```

we want in each line, the name be the key and the price be the value.



```python
def budgie_dict(filename):
    """(file to be open) -> dict

    Return a dictionary that record the information in the file.
    """
    # Firstly, open the file to read.
    file = open(filename)
    result = {}
    for line in file:
        # line = "cage,$50"
        # or maybe line = "cage,$50    "
        # we are not sure if there are space or not, so use
        # strip() in case.
        linelist = line.strip().split(",")
        # linelist = ["cage", "$50"]
        result[linelist[0]] = linelist[1]

    #remember to close your file!
    file.close()
    return result
```

run this in the python shell, we will get what we want!



```
>>> budgie_dict("budgie_budget.csv")
{'cage': '$50', 'seed': '$10', 'food dishes': '$18', 'water bottle': '$12'}
>>> 
```

part 3 – write file

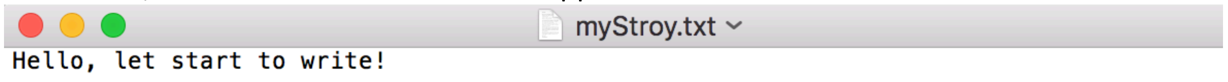
- Using “w” to indicate the mode is write.

```
>>> file = open("myStroy.txt", "w")
>>> print(file)
    <_io.TextIOWrapper name='myStroy.txt' mode='w' encoding='US-ASCII'>
>>>
```

- The file named myStroy.txt will be created in your folder!
- Start to write

```
>>> file.write("Hello, let start to write!\n")
    27
```

- Close your file,
- After close, the new file with the content appears!


myStroy.txt

```
Hello, let start to write!
```

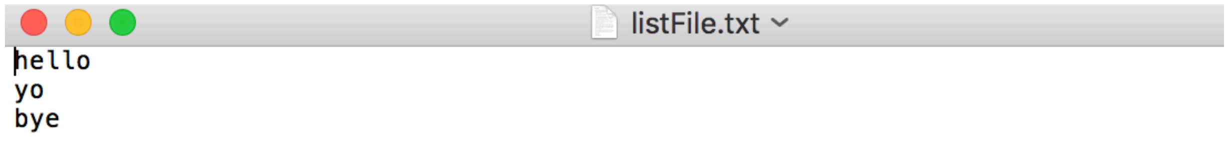
- Consider a function that record the thing in the list to a file named listFile.txt

```
def my_list_file(lst):
    """(list) -> None

    Record the value in lst into a file named listFile.txt line by line.
    """
    file = open("listFile.txt", "w")
    for message in lst:
        file.write(str(message) + "\n")

    file.close()
```

after run

```
>>> my_list_file(["hello", "yo", "bye"])
>>>
```

the listFile.txt has been created and with the content inside!



listFile.txt

```
hello
yo
bye
```