day07 : 元组，字典

笔记本:	Python基础		
创建时间:	2018/6/11 15:59	**更新时间:**	2018/7/2 19:38
作者:	liuchang_0412@163.com		

day07 笔记:

1.列表和字符串的比较:
1.都是序列有先后顺序，都是可迭代对象
2.列表是可变的，字符串是不可变的
3.列表可以存储任意类别的数据，字符串只能存储字符

2.字符串的文本解析方法:
S.split(sep=None) #将字符串S分割为字符串列表
例: S = "welcome to tarena"
words = S.split(" ") #words --> ["welcome", "to", "tarena"]

S.join(iterable) #将可迭代对象进行拼接，中间用字符串进行分隔
例: L = ["C:", "windows" , "system32"]
"\\".join(L) #L --> "C:\windows\system32"

#练习：有字符串"hello"，生成新字符串"h e l l o" 和 "h-e-l-l-o"

3.元组(tuple)：元组是不可改变的序列，同list一样，元组可以存放任意值。
①表示方法：用小括号( )括起来。单个元素括起来后加逗号区分单个对象还是元组
②创建空元组:
t = ( ) #空元组
t = tuple( ) #空元组
③创建非空元组:

t = (20,) #t --> (20,) 元组类型
t = 20, #t --> (20,) 元组类型
t = (10, 20, 30) #t --> (10, 20, 30) 元组类型
t = 10, 20, 30 #t --> (10, 20, 30) 元组类型
④创建元组错误示例:
t = (20) #t --> 20 数值类型
x , y = 100, 200 # x 绑定100，y 绑定200 数值类型
⑤元组的构造（生成）函数:
tuple( ) #生成一个空元组，等同于( )
tuple(iterable) #用于可迭代对象生成一个元组

4.元组的运算：

①算术运算： + , += , * , *=

+运算符：拼接

+= 运算符：拼接后对变量赋值

* 运算符：生成重复的元组

*= 生成重复的元组并赋值给变量

例： x = (1, 2, 3) + (4, 5, 6)　　# x --> (1, 2, 3, 4, 5, 6)

x += (7,)　　# x --> (1, 2, 3, 4, 5, 6, 7)

②比较运算：> , >= , < , <= , == , !=

同列表(list)的比较方法相同

③元组的 in/ not in 运算符：

同列表(list)的使用方法相同，用于判断一个值是否存在与元组中

④索引(index)，切片(slice)，等同于字符串(seq)的使用规则。元组不能索引/切片赋值!

5.可用于序列的函数总结

len(x)	返回序列的长度
max(x)	返回序列的最大值的元素
min(x)	返回序列最小值的元素
sum(x)	返回序列所有元素的和
any(x)	真值测试，如果列表中其中一个值为真值则返回 True
all(x)	真值测试，如果列表中所有值都是真值则返回 True
reversed(seq)	返回反向序列顺序的迭代器
sorted(iterable , key = None , reverse = False)	返回已排序的对象的**列表**

#练习：

1.输入任意一个字符串或数字，判断是否为回文 （12321 和 "ABCDCBA"为回文）

2.编写程序，获取一个数值，计算并打印其中每个数字的出现个数

例如：输入2234524，打印如下：

数字2出现3次，数字3出现1次，数字4出现2次，数字5出现1次

6.字典(dict)：

①字典的定义：

1.字典是一种可变容器，可以存储任意类型的数据

2.字典中每个数据都是用“键”(key) 进行索引，而不像序列可以用下标进行索引

3.字典中的数据没有先后关系，字典的存储是无序的

4.字典的数据是以键(key)-值(value)对的形式进行存储的

5.字典的表示方式是以{ }括起来，以冒号 ':' 进行分隔的键值对，各键值对间用 ',' 分隔

6.字典的键不能重复

②创建空字典：

```
d = { }          #空字典
d = dict( )    #空字典
```

③创建非空字典：

```
d = { "name" : "liuchang" , "age" : 27 }
```

④字典的值(value)可以为任意类型（布尔，数值，字符串，None，列表，元组，字典等)

⑤字典的键(key)必须为不可变类型的值（None，布尔，数值，字符串，元组）

7.字典的基本操作：

①字典的访问：用[ ] 运算符访问字典内的成员

```
字典 [键]
例：  d = {"name": "tarena", "age": 15}
        print("姓名", d["name"], "年龄", d["age"])
```

②添加/修改字典的元素：键不存在，创建键并绑定键对应的值；键存在，修改键绑定的值

```
字典[键] = 值
例：  d = { }
        d['name'] = 'tarena'          #创建新的键值对
        d['age'] = 15                    #创建新的键值对
        d['age'] = 16                    #修改'age'键所对应的值
```

③删除字典元素 del

```
del 字典[键]
例：  del d['age']              #删除'age'和其对应的值
```

④获取字典中元素的个数len函数：

```
len(字典)        #返回字典中的元素个数（键值对）
```

⑤字典的成员资格判断 in/not in 运算符：判断一个键是否存在与字典中

```
键 in 字典
例：  d = { "name" : "liuchang" , "age" : 27 }
        "liuchang" in d      #返回True
```

⑥字典的生成函数dict( )：

```
dict( )                    #生成一个空字典，等同于{ }
dict( iterable )          #用可迭代对象初始化一个字典
dict(**kwargs)          #关键字参数形式生成一个字典
例：  d = dict([( 'name' , 'tarena' ) , ( 'age' , 15 )])
        d = dict(name = "tarena", age = 16)
```

8.字典的方法：（D代表字典对象）

```
D.clear( )              #清空字典
D.pop(key)              #移除键
D.copy( )               #返回字典D的副本，只复制一层（浅拷贝）
D.update(D2)            #将字典D2合并到D中，如果键相同，则此键的值取D2的值
D.get(key, default)     #返回key所对应的值，如果没有此键则返回default
D.keys( )               #返回可迭代的dict_keys集合对象
D.values( )             #返回可迭代的dict_values值对象
D.items( )              #返回可迭代的dict_item对象
```

9.字典推导式：

①语法： {键 : 值表达式 for 变量 in 可迭代对象 (if 条件表达式)}

例：number = [1001, 1002 ,1003, 1004]

names = ["Tom", "Jerry", "Spike", "Tyke"]

animals = { numbers[i] : name[i] for i in range(4) }

#练习：

1.已知两个等长的列表 list1 和 list2，以 list1 中的元素为键，以 list2 中的元素为值，生成相应的字典，如list1 = ["a","b","c"] list2=[1,2,3]。生成字典为{"a":1,"b":2,"c":3}。

2.输入5个学生的姓名和年龄，每个学生的信息形成字典后存入列表中，内部存储格式如下：[{"name":"aaa","age":20},{"name":"bbb","age":23},...]。输入完成后，打印所有学生信息如下：

```
+---------------+----------------+
|       姓名      |       年龄       |
+---------------+----------------+
|       aaa       |         20       |
|       bbb       |         23       |
|       .....     |        .....     |
+---------------+----------------+
```

输入学生年龄，把低于此年龄的学生信息打印出来

知识点：列表和字典组合使用。