Python笔记（七）：字典、类、属性、对象实例、继承

（一）  简单说明

   字典是Python的内置数据结构，将数据与键关联（例如：姓名：张三，姓名是键，张三就是数据）。例如：下面这个就是一个字典

{'姓名': '张三', '出生日期': '2899-08-12', '成绩': ['3.21', '3.10', '3.01']}

创建字典、添加数据、访问字典数据的方式如下：

d = {}  #直接用{}创建字典  
f = dict()  
#通过工厂函数dict()创建字典  
#通过下面的方式添加数据  
d['姓名'] = '张三'  
d['出生日期'] = '2899-08-12'  
d['成绩'] = ['3.21', '3.10', '3.01', '2.45', '2.34', '2.22', '2.01']  
print(d)  
#通过键访问字典的数据  
print(d['姓名'])  
print(d['出生日期'])  
print(d['成绩'])  
print(d['成绩'][2])

输出如下所示：

（二）  将列表转换为字典

（1）    
创建一个文件

james2.txt 第一项是姓名，第二项是出生日期，后面的是成绩

James Lee,2002-3-14,2-34,3:21,2.34,2.45,3.01,2:01,2:01,3:10,2-22,2-01,2.01,2:16

（2）    
要求

在屏幕上输出下面格式的字典

{'姓名': 'James Lee', '出生日期': '2002-3-14', '成绩': ['3.21', '3.10', '3.01']}

（3）    
主程序代码

（4）   the\_dict模块代码

（三）  类、属性、对象实例

 简单的说，类和属性都是一个抽象的概念，对象实例是一个具体的“存在”。例如：

类：人

属性：姓名、身高、体重

对象实例：张三、李四

人 指一类东西，身高、姓名、体重 是这类东西都有的属性，张三、李四指的是具体的某一个人。

（四）  创建类、创建对象实例

说明：上面这部分是必须有的，其中 People是类名，自己自定义，\_\_init\_\_(self)方法控制如何初始化对象，self也是必须有的（这是一个目标标识符，标识当前对象具体是什么）

举个例子：

（1）   创建一个类

class People:  
    def \_\_init\_\_(self,name,date=None,achievement=[]):

# date，achievement有缺省（默认）值  
        self.name = name  
        self.date = date  
        self.achievement = achievement

（2）    
创建对象实例

#创建 name为张三的对象实例

说明：使用zs = People('张三')时，会自动调用people类的 \_\_init\_\_（）方法，其中self = zs,name=’张三’，然后创建一个 name = ‘张三’date=None，achievement=[] 的对象实例zs

（3）   类中可以定义很多方法，不过每个方法的第一个参数都必须是self（没有设置这个参数，那么第一参数就会被当做self参数处理）。

例如：

（五）  继承

  可以从零开始创建一个新类，也可以继承已经创建好的类，在这个基础上新增属性、方法。

继承的概念：继承父类的所有方法及属性，子类可以新增方法、属性，也可以重写父类的方法。简单的说，比如你继承了你父亲的所有能力及天赋（包括身高、体重等），然后你可以通过学习去掌握更多的能力，继承自你父亲的能力你也可以根据自己的需要去改变（比如：杰出的沟通能力，你父亲可能用在商业谈判上，你可能想用在泡妞上面）。比喻可能不太形象，请多多包涵。

（1）   通过继承的方式创建一个类

class PeopleList(list):  
    def \_\_init\_\_(self):  
        list.\_\_init\_\_([])

PeopleList(list)新类将派生list类，list.\_\_init\_\_([]) 初始化所派生的类

（2）    
举个例子，下面这个类就能继承list的所有方法

from FirstPython import the\_dict as td  
  
the\_james2 = td.chdict('F:\Python\Python文件\james2.txt')  
print(the\_james2)

def sanitize(time\_str):  
    #传入字符串，将'-'和':'修改为'.'并返回，否则直接返回  
    if '-' in time\_str:  
        (x,y) = time\_str.split('-',1)  
        return(x+"."+y)  
    elif ':' in time\_str:  
        (x,y) = time\_str.split(':',1)  
        return (x + "." + y)  
    else:  
        return(time\_str)  
  
def chdict(the\_file):  
    #传入文件，返回一个字典  
    d = dict()  
    with open(the\_file) as james\_file:  
        the\_list = james\_file.readline().strip().split(',')  
        #分割数据，返回一个列表  
    d['姓名'] = the\_list.pop(0)  
    #pop()删除指定位置的数据项并返回  
    d['出生日期'] = the\_list.pop(0)  
    d['成绩'] = sorted(set([sanitize(t) for t in the\_list ]),reverse=True)[0:3]  
    #set()删除重复数据并返回一个无序的集合，sorted()排序  
    return d

class People:  
    def \_\_init\_\_(self):

zs = People('张三')

def chdict(self,the\_file):  
    with open(the\_file) as new\_file:  
        the\_list = new\_file.readline().strip().split(',')  
    return People(the\_list.pop(0),the\_list.pop(0),the\_list)  
def top3(self):  
    return(sorted(set(self.sanitize(t) for t in self.achievement),reverse=True)[0:3])

class PeopleList(list):  
    def \_\_init\_\_(self,name,date=None,achievement=[]):  
        list.\_\_init\_\_([])  
        self.name = name  
        self.date = date  
        self.extend(achievement)

可以在编辑器中进行测试：

james = PeopleList('james')  
james.append(5)  
print(james)  
james.extend([1,2,3])  
print(james)

