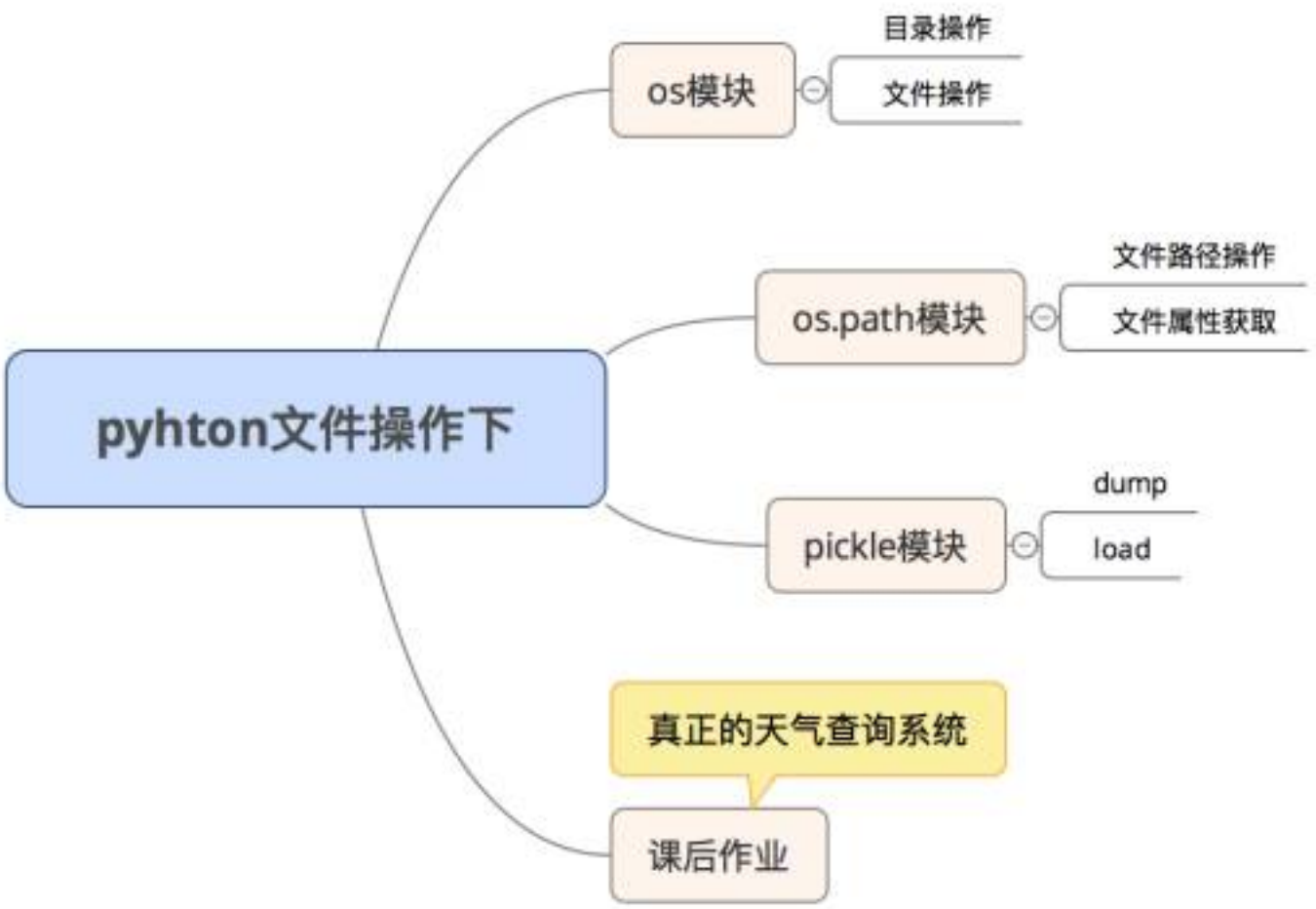
Python文件（下）与文件操作相关的一些模块

Original 2016-12-04 哈哈小 菜猿观世界

通过本节的学习，你将了解以下内容：



上节课简单介绍了pyhton对文件的操作，open，write和read，以及close，这些方法，想必大家此时暂停10秒回想一下应该还是能想起来的。

文件读写既已完结，文件系统的讲解是不是就此打住呢？

nope，文件系统，既然是系统，就此了结，那让“系统”情何以堪啊？

想象这么一个过程，c盘下某个目录下的某个目录下的某个目录下有某个文件，用一个路径去描述是这样的：

c:/a/b/c/biedakai.avi

电脑启动后，某位同志急迫地需要找到这个文件并打开。

数小时后，再关闭。

整个过程，我们上节课的内容好像只涉及到最后几个步骤－打开，读取再关闭。

前面路径那一块可是没做丝毫的介绍。所以呢，今天的重心就落在这一part了。

一，os模块

模块（module），对于只看过哈哈小python原创的伙伴，这个词应该是陌生的。

这样讲啊，模块，谐音某块。对，没错，就是某块。

某块啥？写好的方法或接口或实现，分装打包放在某块。当你的程序要使用里面的功能时，直接把它导进来（import）就行了。

这个首先要介绍的是os模块，全名叫operating system－操作系统。

我们都知道，现在市面上普遍使用的操作系统，windows，mac-os，linux三大操作系统。由于操作系统不同，导致从计算机底层搭建起来的文件实现策略不同，实现方式的不同，可能就会带来某一种能在mac系统上运行的软件搬运到windows上时，就不能执行了的不方便。

所以我们在mac上写的一个pyhton程序在windows上就不能执行了？

怎么可能，pyhton作为一门跨平台的运行，怎么说也不允许这种事情发生。os这个模块听它名字也知道，在对跨平台问题的解决上，一定是python的得力帮手之一。

我们来对os模块一探究竟：

记得刚才提到，要使用模块，因为它放在某地，要让此模块为你的程序所用，第一步要做的就是导入模块：

导入模块，关键字import

```
>>> import os
```

导入某块要看看这个模块大概包括写什么方法和属性，可以调用哈哈小朵多次提到的一个方法dir

```
>>> dir(os)
```

注：由于此模块内容过多，所以输出内容就不在此显示了，大家可自行查阅。

下面从该模块挑选几个频繁用到的几个方法进行讲解：

> 返回当前工作路径

```
>>> os.getcwd()
'/Users/hahaxiao/Documents'
```

> 改变工作路径

```
>>> os.chdir('/Users/hahaxiao/Desktop/python')
>>> os.getcwd()
'/Users/hahaxiao/Documents'
>>>
```

> 列举指定目录的文件名－listdir(path=".")

```
>>> os.listdir()
['.DS_Store', 'a', 'a2', 'a3', 'city.py', 'weather.py']
>>>
```

关键字参数默认为.表示当前目录。可以看到，在当前的python目录下有三个子目录a,a2,a3和三个文件。

> 比如我们再进一步看看a子文件夹下有些什么，可以这样给路径

```
>>> os.listdir("./a")
['.DS_Store', 'b']
```

```
>>>
```

> 创建单层目录，如该目录已经存在，抛出异常

```
>>> os.mkdir("./d")
>>> os.listdir()
['.DS_Store', 'a', 'a2', 'a3', 'city.py', 'd', 'weather.py']
>>>
```

> 也可以如下创建多层目录

```
>>> os.makedirs('./d/a')
>>> os.listdir("./d")
['a']
>>>
```

> 删除文件

```
>>> os.remove("./city.py")
>>> os.listdir()
['.DS_Store', 'a', 'a2', 'a3', 'd', 'weather.py'] # 删除成功
>>>
```

> 删除单层目录，非空抛异常

```
>>> os.rmdir("./d")
Traceback (most recent call last): # d文件夹下已存在a文件夹，非空所以抛异常了
  File "<pyshell#25>", line 1, in <module>
    os.rmdir("./d")
OSError: [Errno 66] Directory not empty: './d'
>>> os.rmdir("./a3") # a3是空文件夹
>>> os.listdir()
['.DS_Store', 'a', 'a2', 'd', 'weather.py']
```

```
>>>
```

> 将文件old重命名为new－rename(old,new)

```
>>> os.rename("weather.py","tianqi.py")
>>> os.listdir()
['.DS_Store', 'a', 'a2', 'd', 'tianqi.py']
>>>
```

执行到这，有一种在黑框框敲命令的感觉。

别急，让你真正有敲命令的感觉的是下面这个方法

> 运行系统的shell命令－system(command)

可以试着玩玩下面这些代码

对于windows系统

```
os.system("calc") # 打开计算器
os.system("cmd") # 打开命令提示符窗口
os.system("notepad") # 打开记事本
os.system("shutdown -s -t 0")#立即关机
```

除此，还有一些支持路径操作常用的定义

```
>>> os.curdir # 获取当前目录
'.'
>>> os.pardir # 获取上一级目录
'..'
>>> os.sep # 对应操作系统的路径分割符（win下为"\\", Linux 下为'/'）
'/'
>>> os.name # 指代当前使用的操作系统（包括：“posix”，“mac”，“os2”，“java”等）
'posix'
```

注，其他方法大家可以结合help(模块.方法名) 函数自行探索。

二，os.path模块

os.path这名字看着好像是os板块的某个儿子，其实并没有什么关系。

来看看这个模块有哪些好用的方法

> basename(path)去掉目录路径，返回文件名

```
>>> import os.path
>>> os.path.basename("/python/weather.py")
'weather.py'
```

> dirname(path)去掉文件名，返回目录路径

```
>>> os.path.dirname("/python/weather.py")
'/python'
```

> split(path)分割文件名和路径，以元组的形式返回

```
>>> os.path.split("/python/weather.py")
('/python', 'weather.py')
```

> splitext(path)分离文件名和拓展名，以元组的形式返回

```
>>> os.path.splitext("/python/weather.py")
('/python/weather', '.py')
```

> getsize(file)返回指定文件的尺寸，单位为字节

```
>>> os.path.getsize("tianqi.py")
794
```

> getatime(file)返回文件最近的访问时间，返回浮点型的秒数

```
>>> os.path.getatime("tianqi.py")
1480774581.0
```

[黑人问号]，这么大个数，表示需要解释一下。

这里调用getatime返回的时间叫做unix时间戳（timestamp），是自1970年1月1日（00:000:00 GMT）以来的秒数。

GMT(greenwich mean time)也就是格林尼治协调世界时间UTC(coordinated universial time）。

格林尼治时间是世界时间的参考，只要知道了UTC 时间，世界上其他地方的时间也就知道了。比如，北京位于东八区，只需在UTC时间多加八个小时，就是北京时间了。

unix时间戳对我们来讲，不具备实用意义，所以当我们得到unix时间戳时，通常要做的事情就是把它转换成格林尼治世界标准时间或当地时间。

在python，另一个叫做time的模块提供给了我们这样两个方法：

gmtime（unix时间戳）转换为GMT时间

localtime（unix时间戳）转换为当地时间

我们不妨来试试：

```
>>> import time
>>> time.gmtime(os.path.getatime("tianqi.py"))
time.struct_time(tm_year=2016, tm_mon=12, tm_mday=3, tm_hour=14, tm_min=16, tm_sec=21, tm_wday=5, tm_yday=338, tm_isdst=0)

>>> time.localtime(os.path.getatime("tianqi.py"))
```

time.struct_time(tm_year=2016, tm_mon=12, tm_mday=3, tm_hour=22, tm_min=16, tm_sec=21, tm_wday=5, tm_yday=338, tm_isdst=0)
>>>

有发现，当地时间确实是比 GMT多了八个小时。

与getatime（file）类似的两个方法是：

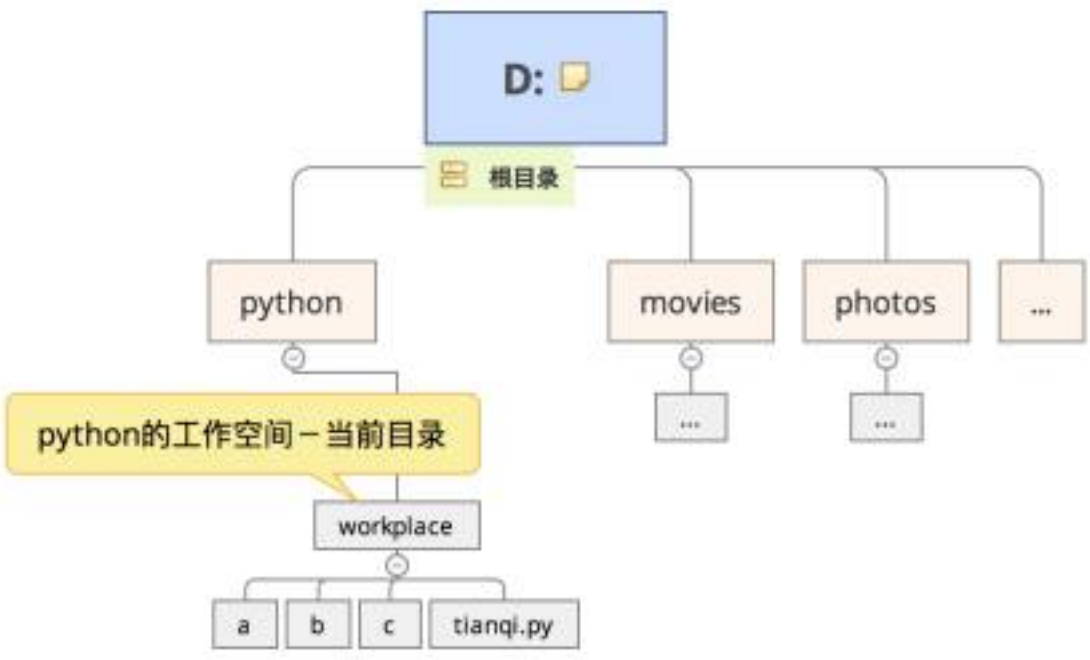
getctime（file）：返回指定文件的创建时间（create）

getmtime（file）：返回指定文件的修改时间（modify）

> 判断指定路径是否为绝对路径

```
>>> os.path.isabs("tianqi.py")
False
```

这里提到绝对路径，哈哈小又得在这啰嗦几句了。



假如某windows D盘有如上文件目录结构，当前python目录假设为workplace目录，像上例给的“tian-qi.py”，相对workplace来讲，位于其根目录下，这样的路径称为相对路径。

与相对路径相对应的是绝对路径，依照上图所示目录结构，tianqi.py的绝对路径是：D://python//work-place//tianqi.py。

结合上图及上文所学，下面哈哈小来个小练习。给定以下文件路径，请迅速说出下面代码执行的结果是什么?

os.listdir("..//..//")

os.listdir("D://")

os.path.isabs("D://moives")

三，pickle模块

pickle这个词翻译成中文，是泡菜，腌制的意思，至于python作者为什么起这么个名字，哈哈小想或许跟作者的口味有某种联系。但是哈哈小觉得最大的联系应该是它的用途了。

腌制泡菜，大概就两个过程，第一步准备好菜，倒入（dump）缸内，腌制许久，第二步就是再从缸内取出食用（load）。

python的pickle模块的使用要与上述过程对应的话，这倒入之前的“菜”就好比前面哈哈小讲过的非常重要的pyhton中的数据结构，像列表，元组，字典等等。

倒入（dump）这个过程就是要靠pickle提供给我们的dump方法来实现了，其作用就是把这些数据结构中的数据做一个转换，然后放到“缸”里，这“缸”就是我们这里说的文件啦。

放入之后，腌制许久（保存在文件中），第二步需要做的就是，取出为程序所用，这一过程的实现又是要借助pickle模块的另一个方法了－load方法。

文字描述，到此打住，下面我们通过具体的代码来看看如何操作。

dump操作

```
>>> f=open("pickle.pkl","wb") # 以二进制写的模式打开一个文件
>>> import pickle # 倒入模块
>>> pickle.dump([1,"ha",2,"ha",3,"xiao"],f) # 倒入数据到文件
>>> f.close() # 关闭文件
>>>
```

经过上步，我们就成功将一个列表（你也可以去尝试倒入一个元祖或字典等其他数据结构）倒入到一个叫pickle.pkl（至于为什么文件后缀名是pkl，答案是：你也可以给任何你想给符合要求的文件后缀名）的文件中。

注：如果你要跑到工作空间，直接去打开这个文件，我保证你看到的会是乱码。因为这文件只有pickle懂。

所以，我们接下来要进行第二步，做一个还原，取出里面的数据，为程序所用。

load方法

```
>>> f=open("pickle.pkl","rb") # 以二进制读的形式打开一个文件
>>> pickle.load(f) # 从该文件中取出数据
[1,"ha",2,"ha",3,"xiao"] # 发现，显示的数据中正是我们倒入前的模样
>>>
```

四，课后作业

哈哈小从互联网上捕获下面这么个小应用。

```
你想查询那个城市的天气？北京
北京
晴 :-2°C~16°C

你想查询那个城市的天气？天津
天津
晴 :2°C~16°C
```

具体实现过程描述：

weather 文件（部分）

```python
#coding=utf-8
import urllib.request
import json

from city import city

cityname = input("你想查询那个城市的天气? ")
citycode=""
print(cityname)
try:
    citycode =city[cityname]
except:
    print ("not Found")
if citycode:
    try:
        url= "http://www.weather.com.cn/data/cityinfo/"+citycode+".html"#构造网址
        content = urllib.request.urlopen(url).read()#读取网页源代码
```

city文件（部分）

```python
#coding=utf-8

city = {
    '北京': '101010100',
    '海淀': '101010200',
    '朝阳': '101010300',
    '顺义': '101010400',
    '怀柔': '101010500',
    '通州': '101010600',
    '昌平': '101010700',
    '延庆': '101010800',
    '丰台': '101010900',
    '石景山': '101011000',
    '大兴': '101011100',
```

实现原理简单描述：

有发现，city是一个只放了一个叫做city的字典的py文件，字典内容记录的是城市编号与城市名称的对应。

当用户输入一个文件名称时，程序会去city这个py文件（在这里city.py就相当于一个被weather.py调用的模

块了）找到该城市对应的编号。

得到编号后，通过天气提供的天气数据接口，向互联网发送数据获取请求，获取后，将数据做一个处理，最终显示给用户。这就是天气查询程序的大致原理了。

借助这个程序，我们需要对我们今天所学加以运用了。

由于city.py文件存放的是几乎所有中国县市对应编码的信息，高达两千多行。

我们要做的是对city文件做些修改。要求是，通过使用pickle模块，首先将city字典保存（dump）在一个叫做city.pickle的文件中，当需要使用时，再去从city.pickle中获取。

经测试没啥问题后，city.py文件中city字典将不再列出，最终city.py文件代码行数将不会超过10行。

经过上面一步一步的改造，再回到weather程序，运行，天气小程序依然正常执行。

注：限于目前所学内容，程序中出现的urllib.request，except，json等字眼不必深究。日益增进，自会清晰明朗。而且这些内容对今天课后题的解答也不会有什么影响。

获取作业源代码，请回复1035

注：我们不想给你带来多少知识，只想尽绵薄之力给你带来可能的思想的启迪。