Python笔记

## 格式化输出（print()）

使用格式化说明符构成字符串

%s、%d、%f

格式：

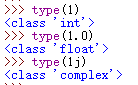
1. 'my name is %s' % ('weiguo')
2. ‘I am like %s and %s’ % (‘炸鸡’,’啤酒’)
3. 'my name is %10s' % ('weiguo') ps: %10s，一个长度10的字符串，而给它的是‘weiguo’，长度为6，所以前面会有4个空格

Python 中反斜杠\为转义字符，如果用到（如路径）需要转义\ ， [\\即可](file:///\\即可)

或者 r”C:\WINDWE\DD”

## 基本数值类型

可以通过 python 内建函数 type()，查看类型



Python中数值提供了如下三种类型：

### 整型

### 浮点型

### 虚数

## 运算符

### 基本算术

加 减 乘 除 取余（%）\*\* 乘方

## 变量

名称存储，数据的引用

## 元组、列表、集合、字典

除了字符串、数值、还有4种重要的基本类型

### 共同点

1. 都允许在一个名称下组织多条数据
2. 每个类型都允许在它们中查找元素 name[index] index:0 ~ len(name)-1
3. len() 函数，返回长度
4. 列表、元组，访问最后一个元素，a[-1] -1表示从右往左
5. 分片:

a[:] 复制a全部元素

a[0:2] 以0开始，不包含下标2

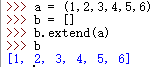
1. 序列的append方法，在一个序列的末尾添加序列



append方法是把整个当作整体加到末尾



1. 如果根据元组的内容，创建一个新的列表，可以使用列表、元组 的extend方法



1. pop方法：

参数是下标，返回值是对应的值，执行完了，列表对应位置被删除了，后续的提上来



1. sorted（[3，2，1]） 排序

>>> [1,2,3]

### 元组tuple（不可改变的数据序列）

1. 圆括号（）
2. 创建后不可改变里面的值
3. 每个值都可以单独访问（查看）
4. 创建一个元素的元组

必须加 ，

Name = (‘weiguo’ , )

不加一个逗号，会被认为是字符串

### 列表list(可以改变的数据序列)

1. 方括号[]
2. 创建后可改变元素的值
3. 每个值都可以单独访问（查看）

### 集合（set()）

1. 和字典类似，集合仅包含键，没有与键相关联的值
2. 不包含重复数据的数据集
3. 在处理重复数据时，集合非常有用
4. 分类：
   1. 可变集合：

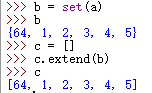
可以CURD它的元素

* 1. 不可变集合

创建后，就不能修改



1. 也可以通过.extend(set(a)) 转成list



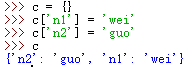
### 字典（以名称索引的分组数据）

1. 这些名称可以是字母、数值、字符串、符号
2. 不可以有两个相同的键
3. 花括号 { }
4. 创建字典

第一种方式：



第二种方式：



1. 从字典中获取键

c.keys() 返回一个含有c字典中的所有键的视图，通过list()，创建一个list对象，list(c.keys())



1. 从字典中获取值

list(c.values())返回一个含有c字典中的所有值的视图，通过list()，创建一个list对象

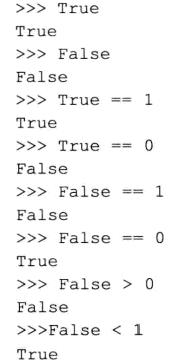


## 像列表一样处理字符串

Python能够把字符串当作单个字符串的列表处理

这样字符串，就可以遍历，单个查询，修改，也可以用list的方法

## 特殊类型（None、True : 1、False : 0）



## 条件判断

Ps:

1. 在Python中序列也可以进行比较，每个序列中同一个位置的每一个元素都相同那么两个序列就相等
2. 字典也可以比较，一个字典中的每个键与值（一组）必须与另一个字典中的键与值一一对应，
3. 给结果取反 not True 、not False (not 是运算符)



1. and or !

## 循环（while for）

### While



为True就执行，直到为False

### For



基本格式：

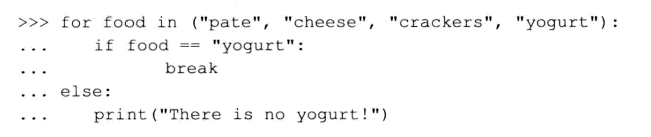
For … in []:

### 跳出循环(break)

### 继续循环（continue）

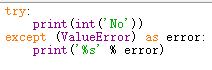
### 循环的末尾使用else

只有在不是因为break结束的循环，才会执行到else中的代码



## 错误处理

try……..except

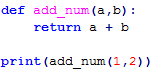


ValueError as error 中 as 是将ValueError指定给一个名称error

1. 可以在except 子句中通过pass忽略处理

## 函数

基本语法：



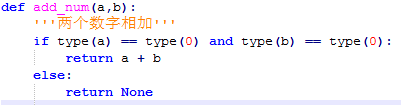
定义函数的时候，在函数中通过 ’’’ xxxxxx ‘’’ 写函数描述，通过\_\_doc\_\_查看

函数名.\_\_doc\_\_

### 检查参数

两种方式：

1. type

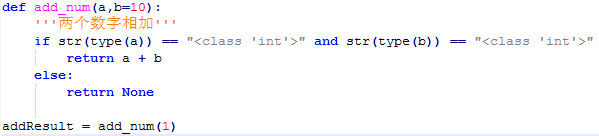


1. str()



### 为参数设置默认值

Python允许创建带默认值的参数，必须在参数列表的末尾



## 类与对象

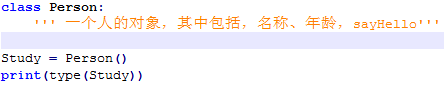
类：封装了数据和函数

### 对象的含义

Python每条数据都是对象

### 定义类

使用class关键字



Study = Person() 🡪由类创建对象

## 组织程序（模块）

模块：

1. 它是一个工具，它将程序划分为不同的命名片段，而不必要使用类来实现这一点
2. 包含与一个公共主题相关的一组函数、方法、数据（网络、文件处理的、图形GUI）

Python有很多自带的模块，第三方模块

### 如何编写、共享模块

### Python中捆绑的模块

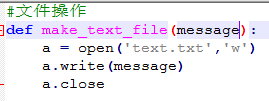
### 导入模块的概念

两个操作：

1. 首先要在操作系统是安装模块（三方）
2. Import 模块名
3. From 模块名 import 需要导入的

## 文件和目录

### 文件对象



如果这个文件不存在，创建它，存在会删除所有数据后写入

### 检查一个文件时候存在



### 先文件中追加

‘a’ 追加参数

1. write(‘’’

这里不需要 \n 来换行

‘’’)

### 读文本文件

‘r’读参数

1. a.readline() 返回一行
2. a. readlines() 返回一个每一行作为元素的列表
3. a.read() 返回全部

### 文件异常

当python进行文件操作遇到错误的时候，会抛IOError错误：

1. 试图打开并读取一个不存在的文件
2. 试图在一个不存在的目录下创建文件
3. 试图打开一个没有读访问权限的文件
4. 试图在一个没有写访问权限的目录下创建文件
5. 计算机遇到磁盘错误

### 路径和目录

Python 将路径和目录操作的细节封装在了os目录中

Os 模块在执行时，可能抛出OSError异常，需要处理它

路径：

模块os中包含另一个模块path，它提供了操作路径的函数，使用os.path操作路径，便于移植

1. **os.path.join(‘a’,’b’)** 将目录名称组合成路径，根据操作系统自动匹配\ \or /

>>> “a\\b”

1. **os.path.split(“c:\\A\\B\\C”)** 将路径最后一组件提取出来，返回元组

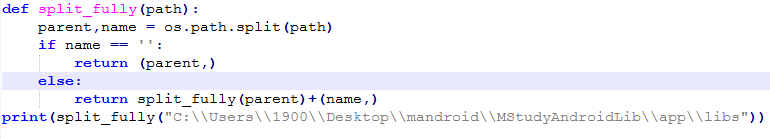
>>> (‘c:\\A\\B’,’C’)

A , B = os.path.split(“c:\\A\\B\\C”)

A = ‘c:\\A\\B’

B = ‘C’

***递归拆分路径***



1. **os.path.splitext**(‘abc.jpg’) 将文件名和文件扩展名分开，返回元组
2. 目录内容查询 **os.listdir(‘c:\\aaa’)** 返回列表

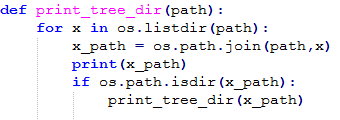
通过os.path.join构造路径

1. **获取文件信息**
   1. **os.path.isfile(‘路径’) 路径指向的是一个文件，返回True**
   2. **os.path.isdir(‘路径’) 路径指向的是一个目录，返回True**

路径不存在，两个函数都返回false

结合这个：

递归打印出一个路径下的所有文件



1. 获取文件的大小

os.path.getsize(‘filename’) 以字节为单位返回文件大小

1. 获取文件修改的时间

os.path.getmtime(‘filename’) 返回距离1970年毫秒数

格式化时间：**time模块**

1. 重命名、移动、删除文件（模块 shutil）
   1. shutil.move(‘’,’’) 第一个参数，文件名，第二个参数，可以是路径，可以是文件名，路径，移动，名称，重命名
   2. shutil.copy(‘’,’’) 复制文件
   3. os.remove(‘’) 删除文件
2. 文件权限os.chmod