1. Range(X,Y) 生成的序列是 包括前面X,不包括后面Y
2. List[X:Y] 列表切割是 包括前面X,不包括后面Y. 且List[X:] 包括X直接切割到列表尾部，包括列表最后一个元素。
3. List[X:Y:Z] 列表从X到Y,但不包含Y，且Z为步长。默认情况下不指定为Z=1

Range(x,y,z) 从x到y,但不含y,且z为步长，默认情况下不指定为z=1

1. 为多个变量同时申明与赋值(例如a,b,c = [1,2,3],”GG”,{1:2,3:4})
2. Python申明多行字符串，以’’’申明结束

Message = ‘’’

Info …..

Location …

‘’’’

1. Python申明常量全部大写字母，但只是表示显示常量不推荐更改，但是python允许所有变量都可以更改不会报错
2. PYC文件的生成。首先我们编程用到的语言都是高级语言(C, JAVA,RUBY, PYTHON),而机器在执行代码中只认识机器语言。所以存在把高级语言翻译成机器语言的过程。这个过程可以大概分为编译型与解释型两种方法。

编译型（C例如）是在程序执行前，就把代码编译成机器码，然后机器去执行该对应机器码。再执行中无需进行任何编译。

解释型（Pyhon例如），是在程序执行中，把代码一条条翻译成机器码执行。在执行过程中再翻译成机器码。并且由于采用虚拟机技术，并不是一步直接翻译成机器码。细节是上，首先把代码编译成PyCodeObject，然后再交由Python虚拟机再翻译成机器码。并且Python为了优化时间，PycodeObject再执行完以后会以PYC文件保存，从而下一次执行文件便不需要再编译。

----Python只为引入的包所对应的.py文件进行编译，执行的主文件不会编译。

----当.py文件修改后，Python会有自己的检测重新编译。

----手动删除pyc文件对程序并不影响，再次执行Python只会为其再次编译。

1. Python3中没有Long 与Int整数类型之分，不管多大都是Int. Python2中才有Long与Int,并且Python根据数值他将自己转化。
2. Python3中列表如果有数字和字符串，不能直接排序。 Python2中可以。
3. id(obj) 查看obj变量的内存地址
4. isdigit(var)函数可以用于判断将变量是否可以转为数字，而无需int()时候去catch这个错误
5. 当我们需要一个标志位推出死循环的逻辑时可以：

While flag:

……

Flag = False

这样的话如果标志位变化则退出循环。而不需要while 1, 然后在循环内部进行break语句。

1. Enumerate(obj)返回一个元组，第一个是index,第二个元素是里面的元素。用途在于产生index以及对应的元素
2. Python 的API说明中，中括号表示可以选参数
3. Json中使用双引号，不使用单引号。数字可以不用引号。
4. Linux下ctrl + C结束python程序并且进程消失。而ctrl + Z 是挂起python程序。进程依然存在。
5. 带小数点的字符串可以直接通过float()转型。

带小数点的字符串不可以直接通过int()转型。

不带小数点的字符串可以直接通过int()转型。

1. Python中的int, string, dict, list,set 实际上是对应与Python中的int, str, dict, list, set的类的实例。所以每一个实例中的方法都定义保存在class中，而不需要每个实例都将方法保存到内存中。
2. Python中的dict(), list(), set()方法可以用于创建字典，列表以及集合。他们将调用对应类中的\_\_init\_\_方法，并且返回对应的数据类型。而如果我们以example = {}, example = [x,x,x..], example = {x,x,x,x,} 这种方式创建的时候，实际上Python还是调用dict(), list(), set()方法去创建，即是\_\_init\_\_方法创建。
3. Python的Set集合是无序不重复的列表。可以用于执行交集，差集，并集等集合操作。
4. Python必须先定义方法，然后才可以调用方法。这是由于Python的执行顺序从上到下决定的。而JAVA, C#是整体预加载然后执行的，所以不需要先定义再运行。
5. 定义函数的参数叫做形式参数（[formal parameter](http://www.baidu.com/link?url=QbY23OT87imXDzEt4t7GoTp4ffbn1xSxpGLWKoDMtp-X8f4TOmGh3A6y0Xbs-qhJase6_5z4Y-nwxBcL_hKtobA9QJr5ucXV_pF6lT5GKNF6R81Am-eWHN86PjXJ_LRN)），调用函数时传递的参数叫做实参（actual parameter）
6. Python解释器自动带有垃圾回收机制，定时的把没有引用的内存空间回收。
7. 虽然数字，字符串，字典，列表在Python中都是对应类的实例，在传递参数的时候，Python会对传递参数的不同而分别采取传递值或者传递引用的方式。

def Test(a):

a.append(“G”)

test\_list = [1,2,3]

Test(test\_list) # 执行完后test\_list值为[1,2,3,”G”]，Python将list的引用传入的函数当中。Dict也是这样的情况。

def Test(a):

a = 3

b = 4

Test(b) # 执行完后b值为4，Python将数字，字符串传入函数中，完全拷贝一份传入。

1. Python定义函数时，开头使用”””与”””进行块注释，说明程序的用途，传入的参数意义，以及返回的值
2. Python的三元表达式与其他语言的三元表达式格式不太一样。三元表达式的意义在于对简单条件语句的简写，即是简单的if …. else …。

格式为: 为真时的结果 if 判定条件 else 为假时的结果

Example : name = “True” if 5==3 else “False”

Note: 为真为假时都只能执行一个语句。

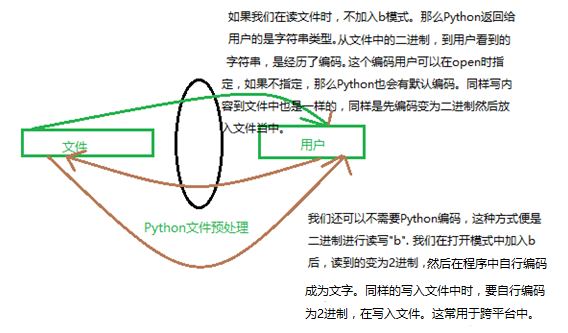
1. Python的lamda表达式，lamda表达式的意义在与定义一个含有简单逻辑的函数，并且返回。

返回函数名称 = lambda 需要传入参数（可以多个）: 操作 (默认函数将返回该值，并且只能有一句话)

Example:

F1 = lambda a: a+100 # 调用时F1(100)返回200

1. 在utf-8中，一个汉字占有3个字节，每个字节转为8位的二进制。
2. Python读取文件时对文件内容的编码预处理



Note: Python的二进制表达是bytes, bytes是用16进制表达的2进制串，为了表达简洁。

从bytes转为字符串, str(bytesObj, encoding = “Encoding”)

从字符串转为bytes, bytes(str Obj, encoding = “Encoding”)

1. 对文件句柄进行循环(即是for line in File\_Handle)则会将文件内容在每次循环中逐次读出。
2. Pycharm中Ctrl + 鼠标单击进行函数定义体

Pycharm中 Atl + Enter忽略这个PEP8的格式警告

1. Python函数如果不手动指定任何东西，那么Python函数默认返回None.
2. 将数组中的元素连接起来并且输出，便捷的方法””.join(list)
3. len()内置函数，在Python3中是默认以字符作为长度。在Python2中默认以字节作为长度。
4. Python字符串的strip()函数，会去除开头结尾处的回车还行以及空格。但是中间的不去除。
5. 字符串转为列表,直接通过list转换不行，需要先eval再list转换。即list(eval(str))

Example:

test = “[1,2,3,4]”

test = list(eval(test)) # test = [1,2,3,4]

1. Python序列化：将数据类型（包括基本数据类型，复杂数据类型）变为字符串。

Python反序列化：将字符串读入程序中，形成数据类型。

Python主要的序列化与反序列化为JSON以及 PICKLE.

JSON可以转换普通数据类型，包括数字，字符串，列表字典。无法转化对象等复杂数据对象。转出来的字符串可以跨平台使用。

PICKLE模块可以转换所有数据类型，但是无法跨平台使用，只能Python自己读取。

1. Python在路径中找寻引入的模块时候，当找到的时候便不再寻找，而不是继续找进行覆盖行为。
2. Python的模块实际上对应的是一个Python文件。

\_\_init\_\_.py文件主要有2个作用，第一个是将文件夹变为Python的一个包，使得可以批量的引入包内的模块，而不需要每一个都执行import语句逐一引入。第二个重要的原因是在当引入这个包时，实际上执行的是\_\_init\_\_.py文件。\_\_init\_\_.py中可以定义当\*时，具体引入此包下的什么模块（如果为空，则什么都不引入）， 可以将一些深层次的包在\_\_init\_\_.py中引入，使得引入这个包的程序可以以同一名称使用。

1. Python的Logging模块，是Python内置的日志模块。可是向日志中输出，时间，多少行引起，什么函数引起。从而不需要自己手写。但是需要配置。Logging模块配置的过程分为,logger, handler, formatter.

流程为：建立logging的一个实例，然后我们将handler注册到这个实例当中，handler表示输出到屏幕或者文件的渠道。然后建立formatter将输出的格式再加入到handler当中。当有日志信息需要输出的时候，logging实例将通过各个handler，以及handler上面的格式输出日志。

在日志中信息由4种等级, DEBUG - > INFO -> ERROR -> CRITICAL.

P.S 在handler中可以各自设置输出日志的最低等级，全局logger实例也可以设置。全局充当的作用是全局过滤，然后交给handler中的等级再过滤。（所以如果全局日志等级中设为CRITICAL,各个handler中设为INFO,也不会看到DEBUG, INFO, ERROR的信息）

40. OS模块主要函数

|  |  |
| --- | --- |
| os.path.abspath(filename) | 返回文件的绝对路径 |
| os.path.dirname(filename) | 返回路径的上一层目录路径 |
| os.path.join(path,filename) | 返回拼接完的路径 |
| os.getcwd() | 返回当前的工作目录 |
| os.listdir([path]) | 返回当前目录的子目录以及文件 |
| os.chdir(path), os.mkdir([path]), os.rmdir([path]), | 改变工作目录, 创建文件夹，删除文件夹 |
| isfile([path]),isdir([path]), exist([path]) | 判断是否为文件，是否为文件夹，路径所指是否存在 |

1. Python字符串有两种格式化方法，一种是百分号%，一种是使用format函数。
2. Python使用hashlib模块进行加密，里面含有SHA, MD5算法。过程为，先建立hashlib的md5对象，可选设置密钥，然后对传入的字符串进行加密。MD5为单向加密，从明文可以转为密文，但是从密文无法转为明文。通过MD5算法，相同的密钥，相同的明文，所得出的密文相同。所以密码比较时，直接比较密文。

import hashlib

hash\_obj = hashlib.md5(bytes(“密钥”,encoding=”utf-8”))

hash\_obj.update(bytes(“需要加密字符串”,encoding=”utf-8”))

hash\_obj.hexdigest() # 生成以16进制表示的密文

1. 有时候在 Windows下面使用Notepad++等文本编辑器编辑Python文件以后，上传到Linux服务器中执行会报错: python^M: bad interpreter: No such file or directory

原因在于不同系统编码方式引起的。

解决方法是在Linux服务器上执行 sed -i 's/^M//g'  filename ，意思是去掉filename中的每一行中的^M。（注意^M 在linux 下写法 按^M 是回车换行符,输入方法是按住CTRL+v,松开v,按m，而不是直接打字形成^再M)

1. Python的virtualenv是python自带的虚拟环境，这个虚拟环境在于当使用pip的时候，安装的包都局限在这个虚拟环境当中不会影响外部的Python安装包。另外virtualenvwrapper是virtualenv辅助包，更加方便。可以统一虚拟环境存储地址，自动结束虚拟环境等。
2. Python判断变量是否为None, 是 变量 is None判断，而不是变量 == None
3. Python的异常捕获语句，try except else finally

当有异常的时候执行except语句

当没有异常的时候执行else语句

无论是否存在异常执行finally语句

47.