一、进程与线程

1、区别

对于操作系统来说，一个任务就是一个进程（Process），比如打开一个浏览器就是启动一个浏览器进程，打开一个记事本就启动了一个记事本进程，打开两个记事本就启动了两个记事本进程，打开一个Word就启动了一个Word进程。

有些进程还不止同时干一件事，比如Word，它可以同时进行打字、拼写检查、打印等事情。在一个进程内部，要同时干多件事，就需要同时运行多个“子任务”，我们把进程内的这些“子任务”称为线程（Thread）。

由于每个进程至少要干一件事，所以，一个进程至少有一个线程。当然，像Word这种复杂的进程可以有多个线程，多个线程可以同时执行，多线程的执行方式和多进程是一样的，也是由操作系统在多个线程之间快速切换，让每个线程都短暂地交替运行，看起来就像同时执行一样。当然，真正地同时执行多线程需要多核CPU才可能实现。

**线程是最小的执行单元，而进程由至少一个线程组成**

**多线程和多进程最大的不同在于，多进程中，同一个变量，各自有一份拷贝存在于每个进程中，互不影响，而多线程中，所有变量都由所有线程共享**

2、多进程

（1）在Unix/Linux下，可以使用os.fork()调用实现多进程

（2）要实现跨平台的多进程，可以使用multiprocessing模块，multiprocessing模块提供了一个Process类来代表一个进程对象，要启动大量进程可以使用Pool类

（3）进程间通信是通过multiprocessing模块中Queue、Pipes等实现的

3、多线程

threading模块提供了一个Thread类来代表一个线程，启动一个线程就是把一个函数传入并创建Thread实例，然后调用start()开始执行

（1）多线程编程，变量共享，容易发生冲突，必须用锁（threading.Lock）加以隔离，同时，又要小心死锁的发生。

（2）Python解释器由于设计时有GIL全局锁，导致了多线程无法利用多核。多线程的并发在Python中就是一个美丽的梦。

二、语法

1、list常用方法

pop与remove区别:

(1)pop() 函数用于移除列表中的一个元素（默认最后一个元素），并且返回该元素的值, list.pop([index=-1])

(2)remove() 函数用于移除列表中某个值的第一个匹配项, list.remove(obj)

reverse() 函数用于反向列表中元素, 该方法没有返回值，但是会对列表的元素进行反向排序

sort() 函数用于对原列表进行排序，如果指定参数，则使用比较函数指定的比较函数

2、迭代器与生成器

(1)迭代器

迭代器是一个可以记住遍历的位置的对象。迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。

迭代器有两个基本的方法：iter() 和 next()。

字符串，列表或元组对象都可用于创建迭代器

(2)生成器

在 Python 中，使用了 **yield** 的函数被称为生成器（generator）。跟普通函数不同的是，**生成器是一个返回迭代器的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器**。在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回 yield 的值, 并在下一次执行 next() 方法时从当前位置继续运行。调用一个生成器函数，返回的是一个迭代器对象。

三、JSON数据处理

由键值对组成的数据集合

1、json与python对象转化

json.dumps()：对数据进行编码，将python格式转化成json格式

json.loads()：对数据进行解码，将json格式转化成python格式

2、python读取json文件

Json.dump()：将内容写入文件

Json.load()：将json文件读出来

四、正则表达式

1、中文匹配

方法：转化成unicode，然后匹配

re.comple(r’[\u4e00-\u9fa5]’)

2、贪婪与非贪婪

贪婪：尽可能多匹配，\*表示贪婪

非贪婪：找到符合条件最小内容即可，?表示非贪婪

如： .\*与.\*?区别

默认是贪婪方式

五、XML数据处理

XML语法：处理指令（包含版本，编码）只有一行，只有一个根元素

1、读取

Xml读取技术主要有：SAX 和 DOM

(1)方法：minidom

Import xml.dom.minidom.

(2)方法：etree

Import xml.etree.ElementTree

2、写入

(1)更改XML：

ele.set：修改属性

ele.append：添加子元素

ele.remove：删除元素

(2)创建XML：

SubElement

minidom

etree

六、socket编程

1、UDP编程

实现：

import socket

sock = socket.socket(socket.AF\_INET, socket.DGRAM) Ipv4、UDP方式

Server端流程：

1、建立socket，socket负责通信的一个实例

2、绑定，为socket指定IP和端口

3、接收对方发送内容

4、给对方发送反馈 （非必须步骤）

Client端流程：

1、建立socket

2、发送内容到指定服务器

3、接收服务器反馈内容

2、UDP编程

实现： 面向连接的，每次传输之前需要建立一个连接

import socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) Ipv4、TCP方式

Server端流程：

1、建立socket负责通信，负责接受对方的请求

2、绑定IP和端口

3、监听接入访问的socket

4、接受访问的socket，可以理解为建立一个通讯的通路

5、接收对方发送的内容

6、如果有必要，给对方发送反馈信息

7、关闭连接通路

Client端流程：

1、建立通信socket

2、连接对方，请求跟对方建立通路

3、发送内容到对方服务器

4、接收对方的反馈

5、关闭连接通路