1. 关于python字符串涉及到的方法：

replace进行字符串中字符替换

例如：将phone\_number字符串的前9位替换成9个\*

phone\_number.replace(phone\_number[:9],'\*' \* 9)

find找到子串在父串中第一次出现的位置索引

例如：找到第一次出现168的位置索引

search = '168'

num\_a = '1386-168-0006'

num\_a.find(search)

输出为：5

format将原字符串中的空填上值

例如：有如下三种方法进行填充

**print('{} a word she can get what she {} for.'.format('With','came')) print('{preposition} a word she can get what she {verb} for'.format(preposition = 'With',verb = 'came'))**

**print('{0} a word she can get what she {1} for.'.format('With','came'))**

1. python内建函数使用方法见如下网址：<https://docs.python.org/3/library/functions.html>
2. len()函数：返回对象的长度，例如：

len('I Cry Out For Magic!') 结果为该字符串长度

1. 函数的传参方式：

def trapezoid\_area(base\_up, base\_down, height):

return 1/2 \* (base\_up + base\_down) \* height

位置参数：trapezoid\_area(1,2,3)

关键词参数：trapezoid\_area(base\_up=1, base\_down=2, height=3)

默认参数：在函数定义时直接为参数赋值，在调用时若不传入也不会出错（使用默认值），若传入则使用传入值。

def trapezoid\_area(base\_up, base\_down, height=3):

return 1/2 \* (base\_up + base\_down) \* height

调用：trapezoid\_area(1, 2)可以不传入默认的height参数

1. open(url,openway)打开文件，参数一是完整文件路径包括文件名，参数二是打开方式需要在打开的文件中写入信息，此时的打开方式要是’w’
2. \*\*代表幂运算；例如a\*\*b为a的b次幂
3. 条件控制：

if…else

if…elif…else

1. 循环控制：

for…in…

1. range函数：

例如：range(1,11)获得不包含11在内的0~10数字。

1. sum函数。对参数对象进行求和。
2. random.randrange(1,7)产生1~6之间的随机数，需要导入random模块
3. 内置数据结构：存储大量数据的容器。
4. python四种数据结构：列表、字典、元组、集合
5. 列表：

A.列表中的每一个元素都是可变的（可以在列表中添加、删除、修改元素）

B.列表中的元素是有序的，每一个元素都有一个对应的位置

C.列表中可以容纳python中的任意对象

1. 字典：

A.字典中数据必须是以键值对形式出现

B.逻辑上，键是不能重复的，值是可以重复的

C.字典中键是不可变的，值是可变的，可修改的，可以是任何对象

1. 字典是不能切片的。
2. 元组是稳固版的列表，元组是不可修改的，但是元组可以被查看索引。
3. 类：有共同特征和行为事物的抽象概念的总和
4. 类的属性：在类里面复值的变量。
5. 类中的方法有自己固定的参数，self，这个参数就是实例本身。
6. 类中的魔法函数：\_\_init\_\_(self)，若在类中定义了该函数，那么在实例化类对象时，会自动先执行该函数。除了有必须的self参数，还可以有自己的参数。在实例化时直接传入参数（除了self）。
7. 任意循环中使用yield返回结果，可以得到类似于range函数的效果
8. 正则表达式是包含文本和特殊字符的字符串
9. 正则表达式：
10. 择一匹配管道符（|）：从多个模式中选择其一

例如：ar|er 可以用来匹配ar或er

1. 匹配任意单个字符（.）：匹配除了换行符\n以外的任何字符

例如：f.o 匹配f开始，o结束，f与o中间存在一个字符

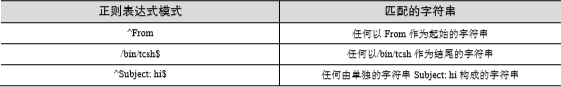
.. 匹配任意两个字符

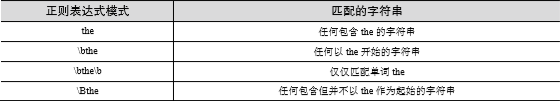
.end 匹配在字符串end之前的任意一个字符

问：怎样才能匹配句点（dot）或者句号（period）字符？

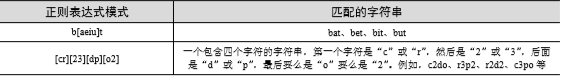
答：要显式匹配一个句点符号本身，必须使用反斜线转义句点符号的功能，例如“\.”。

1. 使用^或\A用来匹配起始字符串（紧挨着的一位字符）
2. 使用$或\Z用来匹配终止字符串（紧挨着的一位字符）
3. 使用\b用来匹配边缘；\B用来匹配中间字符串（紧挨着的一位字符）

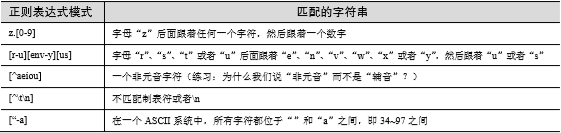




1. []该正则表达式可以匹配一对方括号中包含的任意字符



1. 中括号中使用-（连字符）可以表示一定范围，左方括号后面紧跟^表示不匹配给定字符集中的任意一个



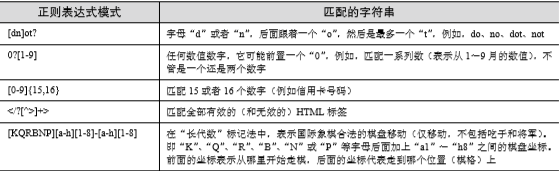
1. 特殊符号\*、+和?：

星号或者星号操作符（\*）将匹配其左边的正则表达式 出现零次或者多次的情况（在计算机编程语言和编译原理中，该操作称为 Kleene 闭包）；

加 号（+）操作符将匹配一次或者多次出现的正则表达式（也叫做正闭包操作符）；

问号（？） 操作符将匹配零次或者一次出现的正则表达式；

大括号操作符（{}），里面 或者是单个值或者是一对由逗号分隔的值。这将最终精 确地匹配前面的正则表达式 N 次（如果是{N}）或者一定范围的次数；例如，{M,N}将匹 配 M～N 次出现。



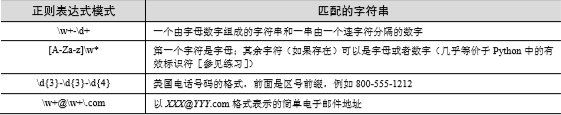
1. 表示字符集的特殊字符：

\d 表示匹配任何十进制数字。（一位）

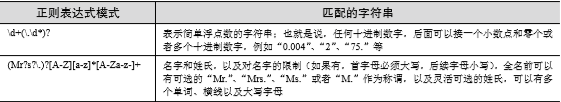
\w能够用于表示全部字母、数字的字符集，相当于[A-Za-z0-9\_]的缩写形式，（一位）

\s 可以用来表示空格字符。（一位）

这些特殊字符的 大写版本表示不匹配；例如，\D 表示任何非十进制数（与[^0-9]相同）



1. 提取任何已经成功匹配的特定字符串或者子字符串，只要用一对圆括号包裹任何正则表达式。



1. 使用compile()函数编译正则表达式，使其成为正则表达式对象
2. 处理正则表达式时，除了正则表达式对象之外，还有另一个对象类型：匹配对象。这 些是成功调用 match()或者 search()返回的对象。匹配对象有两个主要的方法：group()和 groups()。
3. group()要么返回整个匹配对象，要么根据要求返回特定子组
4. groups()则仅返回一个包含 唯一或者全部子组的元组

注意：如果没有子组的要求，那么当group()仍然返回整个匹配时，groups() 返回一个空元组

1. 使用match方法匹配字符串：从字符串的起始部分对模式进行匹配。如果匹配成功，就返回一个匹配对象；如果 匹配失败，就返回 None。match函数的第一个参数为正则表达式字符串。
2. search()的工作方式与 match()完全一致，不 同之处在于 search()会用它的字符串参数，在任意位置对给定正则表达式模式搜索第一次出现 的匹配情况。如果搜索到成功的匹配，就会返回一个匹配对象；否则，返回 None。
3. 在使用（）定义子组时，并列时从左至右依次为子组1…子组n；存在包含时从外至内依次为子组1…子组n
4. 在正则表达式中使用原始字符串是一个好主意：m = re.search(r'\bthe', 'bitethe dog')
5. findall()函数:以列表形式返回全部匹配串，如果没有找到匹配部分，就返回一个空的列表。
6. finditer函数与findall函数类似但是更节省内存，会生成的是一个迭代器

f\_2=re.finditer('(a\w+)','carcarcar carry')

print(f\_2)

#使用for循环进行迭代取值

for i in f\_2:

print (i.group())

print (i.group(1)) #由于不存在（）分组，所以在取group（1）报错

#print (i.group(2)) #由于不存在（）分组，所以在取group（2）报错

1. sub()函数：某字符串中所有匹配正则表达式的部分进行某种形式的替换；subn()函数：某字符串中所有匹配正则表达式的部分进行某种形式的替换，替换后的字符串和表示替换总数的数字一起作为一个拥有两个元素的元组返回。
2. 为了避免ASCII字符与正则表达式的特殊字符冲突，在书写正则表达式时使用原始字符串形式：r’string’
3. 扩展符号，扩展正则表达式：通过使用 (?iLmsux) 系列选项；可以进行组合使用
4. re.I/IGNORECASE忽略大小写(?i)

r\_1=re.findall(r'(?i)yes', 'yes? Yes. YES!!')

print(r\_1)

1. re.M/MULTILINE实现多行混合搜索,在cmd中进入python环境是可以执行成功的；

但是在集成环境中匹配不成功，不会进行跨行检索？是字符串换行符不正确？

r\_2=re.findall(r'(?im)(^th[\w ]+)',"""

This line is the first,

another line,

that line, it's the best

""")

print(r\_2)

1. re.S/DOTALL。该标记表明点号（.）能够用来表示\n 符号（反之其通常 用于表示除了\n 之外的全部字符）

r\_3=re.findall(r'(?is)th.+',"""

The first line,

the second line,

the third line

""")

print(r\_3)

1. re.X/VERBOSE 标记非常有趣；

该标记允许用户通过抑制在正则表达式中使用空白符（除了在字符类中或者在反斜线转义中）来创建更易读的正则表达式

有了(?x)正则表达式可以换行

r\_4= re.search(r'''(?x)

\((\d{3})\)

[ ]

(\d{3})

-

(\d{4})

''', '(800) 555-1212').groups()

print(r\_4)

1. (?:...)使用该符号可以对正则表达式进行分组，但是并不会保存该分组用于后续的检索或者应用

在不需要保存今后永远不会使用的多余匹配时，这个符号可以使用

r\_5=re.findall(r'http://(?:\w+\.)\*(\w+\.com)','http://google.com http://www.google.com http://code.google.com')

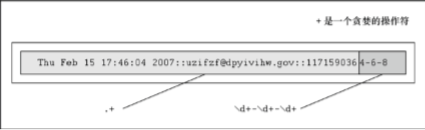
print(r\_5)

r\_6=re.search(r'http://(?:\w+\.)\*(\w+\.com)','http://google.com http://www.google.com <http://code.google.com>')

print(r\_6.group(1))

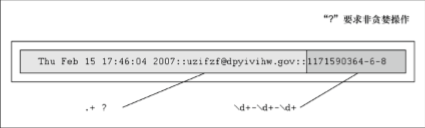
1. 随机整数 的场景都使用 random.randrange()函数
2. time.ctime()函数将该整数转换为日期
3. 正则表达式贪婪匹配（贪婪操作符）：在匹配时对该字符串从左只有按顺序求值，而且试图获取匹配该模式尽可能多的字符。

例如在“Thu Feb 15 17:46:04 2007::uzifzf@dpyivihw.gov::117159036”中使用'.+(\d+-\d+-\d+)'进行匹配，会先尽可能多的匹配’.+’这部分



可以使用“非贪婪”操作符“？”，在“\*”、“ +”或者“?”之后使 用该操作符

例如在“Thu Feb 15 17:46:04 2007::uzifzf@dpyivihw.gov::117159036”中使用'.+?(\d+-\d+-\d+)'进行匹配，就不会先尽可能多的匹配’.+’这部分



1. 网络编程正是遵循客户端/服务器架构的软件模型
2. 客户端/服务器网络编程在服务器响应客户端请求之前会先做一些准备：会先创建一个通信端点，用来监听客户端请求。
3. 客户端所需要做的只是创建它的单一通信端点，然后建立一个到服务器的 连接。然后，客户端就可以发出请求，该请求包括任何必要的数据交换。一旦请求被服务器 处理，且客户端收到结果或某种确认信息，此次通信就会被终止。
4. 套接字是计算机网络数据结构，有两种套接字：基于文件和面向网络。
5. 一个网络地址由主机名+端口号组成。
6. 与网络和套接字编程有关的python模块：

socket模块

async\*模块

socketserver模块

Twisted模块—异步

Concurrence模块—异步

1. 文件传输协议：
2. HTTP主要用于基于Web的文件 下载以及访问 Web 服务，一般客户端无须登录就可以访问服务器上的文件和服务。大部分 HTTP 文件传输请求都用于获取网页（即将网页文件下载到本地）。
3. scp 和rsync 需要用户登录到服务器主机。在传输文件之前必须验证客户端的身份，否则 不能上传或下载文件。
4. FTP 与 scp/rsync 相同，它也可以上传或下载文件，并采用了 UNIX 的多 用户概念，用户需要输入有效的用户名和密码。但FTP 也允许匿名登录（管理员要在FTP服务器上进行设置允许匿名登录）。
5. FTP只使用TCP，FTP 看作客户 端/服务器编程中的特殊情况。因为这里的客户端和服务器都使用两个套接字来通信：一个 是控制和命令端口（21 号端口），另一个是数据端口（有时是 20 号端口）。
6. Python中默认安装的ftplib模块定义了**FTP类**，其中函数有限，可用来实现简单的ftp客户端，用于上传或下载文件，函数列举如下：

|  |  |
| --- | --- |
| ftp登陆连接 | |
| from ftplib import FTP | 加载ftp模块 |
| ftp=FTP() | 设置变量 |
| ftp.set\_debuglevel(2) | 打开调试级别2，显示详细信息 |
| ftp.connect("IP","port") | 连接的ftp sever和端口 |
| ftp.login("user","password") | 连接的用户名，密码 |
| print ftp.getwelcome() | 打印出欢迎信息 |
| ftp.cmd("xxx/xxx") | 进入远程目录 |
| bufsize=1024 | 设置缓冲区大小 |
| filename="filename.txt" | 需要载的文件 |
| file\_handle=open(filename,"wb").write | 以写模式在本地打开文件 |
| ftp.retrbinaly("RETR filename.txt",file\_handle,bufsize) | 接收服务器上文件并写入到本地文件 |
| ftp.set\_debuglevel(0) | 关闭调试模式 |
| ftp.quit() | 退出ftp |
| ftp相关命令操作 | |
| ftp.cwd(pathname) | 设置ftp当前操作的路径 |
| ftp.dir() | 显示目录下所有目录信息 |
| ftp.nlst() | 获取目录下的文件 |
| ftp.mkd(pathname) | 新建远程目录 |
| ftp.pwd() | 返回当前所在位置 |
| ftp.rmd(dirname) | 删除远程目录 |
| ftp.delete(filename) | 删除远程文件 |
| ftp.rename(fromname, toname) | 将fromname修改名称为toname。 |
| ftp.storbinaly("STOR filename.txt",file\_handel,bufsize) | 上传目标文件 |
| ftp.retrbinary("RETR filename.txt",file\_handel,bufsize) | 下载ftp文件 |

1. FTP.quit()与FTP.close()的区别
2. FTP.quit():发送QUIT命令给服务器并关闭掉连接。这是一个比较“缓和”的关闭连接方式，但是如果服务器对QUIT命令返回错误时，会抛出异常。
3. FTP.close()：单方面的关闭掉连接，不应该用在已经关闭的连接之后，例如不应用在FTP.quit()之后。
4. Usenet新闻系统是一个全球存档的电子公告板，但是正在逐渐退出人们的视野，逐渐被在线论坛替代，老的usenet使用UUCP作为其网络传输机制，后期网络流量转向使用TCP/IP。
5. 网络新闻传输协议（NNTP）在新闻组中下载或发表帖子。作为客户端/服务器架构的另一个例子，NNTP 与 FTP 的操作方式相似，但更简单。在 FTP 中，登录、传输数据和控制需要使用不同的端口，而 NNTP 只使用一个标准端口119来通信。用户向服务器发送一个请求，服务器就做出相应的响应。
6. Python中有一个nntplib 库和一个 需要实例化的 nntplib.NNTP 类，详细的可以在：http://docs.python. org/library/nntplib网址中进行查询。
7. Python中有一个smtplib 库和一个 需要实例化的 smtplib.SMTP 类，使用端口号25。用于发送邮件。
8. python还有SMTP\_SSL类，类似于SMTP但是通过加密套接字通信，可以作为SMTP/TLS的替代品，默认端口号465. Python 对 SMTP 的更多支持，可以阅读 http://docs.python.org/library /smtplib。
9. pop协议用来下载邮件，邮局协议（POP）的目的是让用户的工作站可以访问邮箱服务器 里的邮件，并在工作站中。通过简单邮件传输协议（SMTP）将邮件发送到邮件服务。
10. Pop出现之后，出现了IMAP（因特网消息访问协议），具备更完整的解决方案，但是比pop更加复杂。可以适应多邮件客户端，如手机、台式机、笔记本电脑。
11. 使用时导入poplib 模块，之后实例化 poplib.POP3 类。poplib.POP3\_SSL类（Python 2.4 中添加的），该类需要提供一些凭证信息，然后通过加密连接传输邮件。
12. 使用时导入imaplib，实例化其中一个 imaplib.IMAP4\*类。这个模块定义了三个类，分别是 IMAP4、IMAP4\_SSL、IMAP4\_stream，这些类可以用来连接任何兼容 IMAP4 的服务器。IMAP\_SSL 可以通过 SSL 加密的套接字连接 IMAP4 服务器。IMAP 的另一个子类是 IMAP4\_stream，该类可以通过一 个类似文件的对象接口与 IMAP4 服务器交互。
13. 进程（重量级进程）是一个执行中的程序，他们拥有自己的地址空间、内存、数据栈以及其他用于跟踪执行的辅助数据。
14. 线程（轻量级进程）是在同一个进程下执行的，并共享相同的上下文。线程包括开始、执行顺序和结束三部分。它有一个指令指针，用于记录当前运行的上下 文。当其他线程运行时，它可以被抢占（中断）和临时挂起（也称为睡眠）——这种做法叫 做让步（yielding）。
15. 线程一般是以并发方式执行的，正是由于这种并行和数据共享机制，使得多任务间的协作成为可能。、
16. Python多线程编程与Java多线程编程
17. 线程和Python
18. Python 代码的执行是由 Python 虚拟机（又名解释器主循环）进行控制的。Python 在 设计时是这样考虑的，在主循环中同时只能有一个控制线程在执行，所以在任意给定时刻只有一个线程会被解释器执行。
19. Python 虚拟机的访问是由全局解释器锁（GIL）控制的。这个锁就是用来保证同时只 能有一个线程运行的。
20. 在多线程环境中，python虚拟机按照如下方式执行：

1．设置 GIL。

2．切换进一个线程去运行。

3．执行下面操作之一。

a．指定数量的字节码指令。

b．线程主动让出控制权（可以调用 time.sleep(0)来完成）。

4．把线程设置回睡眠状态（切换出线程）。

5．解锁 GIL。

6．重复上述步骤。

1. 当一个线程完成函数的执行时，它就会退出。另外，还可以通过调用诸如 thread.exit()之 类的退出函数，或者 sys.exit()之类的退出 Python 进程的标准方法，亦或者抛出 SystemExit 异常，来使线程退出。不过，你不能直接“终止”一个线程。
2. 在python中使用线程

|  |  |
| --- | --- |
| thread模块python3中改为\_thread | threading模块 |
| （不建议使用）在主线程退出之后，所有 其他线程都会在没有清理的情况下直接退出 | 确保在所有“重要的” 子线程退出前，保持整个进程的存活 |
| 同步原语很少（只有一个） | 1、更先进、更好的线程支持。  2、支持守护线程，。整个 Python 程序（可以解读为：主线程）将在所有非守护线程退出之后才退出，换句话说，就是没有剩下存活的非守护线 程时。 |
|  | 创建线程的三种方法（见E:\studyMyself\aboutGit\testColneProject\aboutPython\testProject文件夹下threading1-threading3）：  • 创建 Thread 的实例，传给它一个函数。  • 创建 Thread 的实例，传给它一个可调用的类实例。  • 派生 Thread 的子类，并创建子类的实例。 |

1. 主线程应该做一个好的管理者，负责了解每个单独的线程需要执行什么，每个派生的 线程需要哪些数据或参数，这些线程执行完成后会提供什么结果。这样，主线程就可以收集 每个线程的结果，然后汇总成一个有意义的最终结果。
2. thread 模块提供了基本的线程和锁定支持；而 threading 模块提供了更高级别、功能更全面的线程管理。使用 Queue 模块，用户可以创建一个队列数据结构，用于在多线程之间进行共享。
3. 守护线程：一般是一个等待客户端请 求服务的服务器。如果没有客户端请求，守护线程就是空闲的。如果把一个线程设置 为守护线程，就表示这个线程是不重要的，进程退出时不需要等待这个线程执行完成。新的子线程会继承父线程的守护标记（thread.daemon = True守护标记为True时即为守护线程）
4. 线程和Java
5. 多线程的同步原语

同步：一般在多线程代码中，总会有一些特 定的函数或代码块不希望（或不应该）被多个线程同时执行，通常包括修改数据库、更新文件或 其他会产生竞态条件的类似情况。如下为python中的两种同步原语：

|  |  |
| --- | --- |
| 锁（简单） | 信号量 |
| 多线程争夺锁时，允许第一个获得锁的线程进入临界区，并执行代码。所有之后到达 的线程将被阻塞，直到第一个线程执行结束，退出临界区，并释放锁。此时，其他等待的线 程可以获得锁并进入临界区。不过请记住，那些被阻塞的线程是没有顺序的：  1、锁有两种状态：  锁定、未锁定  2、锁有两个函数：  获得锁、释放锁 |  |