022 批量爬取B站小视频

* 核心技术

在实现批量爬取B站小视频的爬虫程序时，分别使用了requests模块、time模块、random模块、os模块、re模块以及open()函数。其中技术要点的内容如下：

1．requests模块

requests是python中实现HTTP请求的一种方式，requests是第三方模块，该模块在实现HTTP请求时要比urllib模块简化很多，操作更加人性化。在是使用requests模块时需要通过执行pip install requests代码进行该模块的安装。

requests模块是第三方模块，在使用前需要使用import导入该模块。代码如下：

1. **import** requests # 网络请求模块

（1）在实现网络请求的发送时，是通过requests模块中的get()方法实现的。语法格式如下：

requests.get(url, params=None, \*\*kwargs)

参数说明如下：

* url：表示需要请求的网络地址。
* params：发送网络请求所需要的参数，一般以字典、元组或字节列表。
* \*\*kwargs：表示接受可选参数。

以GET请求方式为例，打印多种请求信息的示例代码如下：

1. **import** requests # 导入模块
2. response = requests.get('http://www.baidu.com')
3. print(response.status\_code) # 打印状态码
4. print(response.url) # 打印请求url
5. print(response.headers) # 打印头部信息
6. print(response.cookies) # 打印cookie信息
7. print(response.text) # 以文本形式打印网页源码
8. print(response.content) # 以字节流形式打印网页源码

（2）在实现视频的下载时，需要使用requests模块内requests.Response类中的iter\_content()方法，该方法可以实现一边下载一边将字节流写（视频文件）入到本地硬盘中。语法格式如下：

requests.Response.iter\_content(chunk\_size=1, decode\_unicode=False)

参数说明如下：

* chunk\_size：表示每次读入内存的字节数。
* decode\_unicode：如果decode\_unicode为True，将根据响应使用最佳可用编码对内容进行解码。

使用iter\_content()方法实现一边下载一边写入本地硬盘的示例代码如下：

1. **with** open('video/'+titlename+'.mp4', 'wb')**as** f: # 将视频写入指定位置  
    **for** data **in** response.iter\_content(chunk\_size=1024): # 循环写入，实现一段一段的写  
    f.write(data) # 写入视频文件  
    f.flush() # 刷新缓存

2. time模块

time模块提供了各种时间相关的函数，该模块中对于时间表示的格式有如下三种：

* timestamp时间戳，时间戳表示的是从1970年1月1日00:00:00开始按秒计算的偏移量。
* struct\_time时间元组，共有九个元素组。分别为：年、月、日、时、分、秒、一周中的第几日、一年中的第几日、夏令时。
* format time格式化时间字符串，已格式化的结构使时间更具可读性。包括自定义格式和固定格式。

time模块是python的内置模块，所以在使用前需要使用import导入该模块。代码如下：

1. **import** time # 时间模块

在实现网络请求的间隔时间时，是通过time模块中的sleep()方法实现的。语法格式如下：

time.sleep(secs)

参数说明如下：

* secs：休眠执行的秒数。“secs”以秒为单位，如果想定时毫秒，可以使用小数，以指示更精确的暂停时间，0.1秒则代表暂停100毫秒。
* 返回值：无。

使用sleep()方法实现延迟5秒，然后通过print()函数输出文本内容。示例代码如下：

1. **import** time # 导入time模块
2. time.sleep(5) # 延迟5秒
3. print('人生苦短，我用Python！')

3.random模块

random模块用于实现各种分布的伪随机数生成器，可以根据不同的实数分布来随机生成值。如随机生成指定范围的整数、浮点数、序列等。该模块中的多数函数都取决于random()函数，该函数通过Mersenne Twister作为核心生成器在[0.0,1.0)区间范围内均匀生成随机浮点数。并且还可以通过random模块中的seed()方法初始化随机数生成器并通过指定种子参数来实现随机生成相同的数值。

random模块是python的内置模块，所以在使用前需要使用import导入该模块。代码如下：

1. **import** random # 随机模块

在实现网络请求的随机间隔时间时，是通过random模块中的randint()方法实现的。语法格式如下：

random.randint(a,b)

参数说明如下：

* a：表示随机整数范围的起始值。
* b：表示随机整数范围的结束值，随机出现的整数包含b参数。
* 返回值：返回指定范围的随机整数。

使用randint()方法获取指定范围的整数。示例代码如下：

1. **import** random # 导入随机数模块
2. print(random.randint(0,2)) # 打印随机整数

4.os模块

os模块是Python内置的与操作系统功能和文件系统相关的模块。该模块依赖于操作系统。通常情况下，如不特别指出，则该模块提供的方法属性在Windows和UNIX上都是可用的。其中，UNXI系统包括Linux和Mac OS X。

os模块是python的内置模块，所以在使用前需要使用import导入该模块。代码如下：

1. **import** os # 操作系统模块

（1）在实现创建指定的目录时，需要使用os模块中的mkdir()方法来实现。语法格式如下：

os.mkdir( path , mode=0o777 , \* , dir\_fd=None )

参数说明如下：

* path：用于指定要创建目录的路径。
* mode：指定目录的模式，默认模式为八进制的777。类似chmod()方法。
* dir\_fd：指定打开文件描述符的路径。默认值为None。
* 返回值：无。

在实现创建指定的一级目录时，示例代码如下：

1. **import** os # 文件与操作系统相关模块
2. os.mkdir(r'e:/mr/wgh') # 创建目录

说明：运行上面的代码，将创建e:/mr/wgh目录。

（2）在不确定某个目录是否存在时，可以使用os.path模块中的exists()方法来判断否存在某个目录。语法格式如下：

os.path.exists(path)

参数说明如下：

* path：为要判断的路径，可以采用绝对路径，也可以采用相对路径。
* 返回值：如果给定的路径存在，则返回True，否则返回False。

注意：，在使用existas()方法时，在某些平台上，如果未授予os.stat()对所请求文件有执行的权限，则该函数可能返回False，即使路径真实存在。

实现判断保存在“E:/mr/test/”目录下的“hello.py”文件是否存在，示例代码如下：

1. **import** os.path # 导入os.path模块
2. path = r'E:/mr/test/hello.py' # 文件
3. **if** os.path.exists(path): # 判断文件是否存在
4. print(path,'文件存在！')
5. **else**:
6. print(path,'文件不存在！')

5.re模块

Python 提供了re模块，用于实现正则表达式的操作。在实现时，可以使用re 模块提供的方法（如search()、match()、findall()等）进行字符串处理，也可以先使用re模块的compile()方法将模式字符串转换为正则表达式对象，然后再使用该正则表达式对象的相关方法来操作字符串。

re模块是python的内置模块，所以在使用前需要使用import导入该模块。代码如下：

1. **import** re # 正则表达式

（1）在实现编译正则表达式对象时，可以使用re模块中的compile()方法来实现。语法格式如下：

re.compile(pattern,flags=0)

参数说明如下：

* pattern：编译时用的表达式字符串。
* flags：编译标志位，用于修改正则表达式的匹配方式，如：是否区分大小写，多行匹配等。
* 返回值：返回一个正则表达式对象。

实现查找包含指定字母的单词，示例代码如下：

1. **import** re # 导入正则表达式re模块
2. # 需要匹配的字符串
3. string = "Tina is a good girl, she is cool, clever, and so on..."
4. match = re.compile(r'\w\*oo\w\*') # 创建正则表达式对象
5. print(match.findall(string )) # 打印所有包含'oo'的单词

（2）在实现字符串根据指定表达式进行替换时，可以使用re模块中的sub()方法来实现。语法格式如下：

re.sub(pattern,repl,string,count=0,flags=0)

参数说明如下：

* pattern：表示模式字符串，由要匹配的正则表达式转换而来。
* repl ：表示替换的字符串。
* string：表示要被查找替换的原始字符串。
* count：可选参数，表示模式匹配后替换的最大次数，默认值为0，表示替换所有的匹配。
* flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。
* 返回值：返回替换后的字符串。

实现排除字符串中的汉字、字母与数字时，可以使用以下示例代码：

1. **import** re
2. string = 'abcdefghijklmnopqrstuvwxyz我爱Python0123456789!#$%&\*+-.^\_`|~'# 定义需要筛选的字符串
3. result = re.sub("[A-Za-z0-9**\u4e00**-**\u9fa5**]", "", string) # 替换字符串
4. print(result) # 打印替换结果

6.open()函数

在Python中，想要操作文件需要先创建或者打开指定的文件并创建文件对象。这可以通过内置的open()函数实现。open()函数的基本语法格式如下：

file = open(filename[,mode[,buffering]])

参数说明如下：

* file：被创建的文件对象；
* filename：要创建或打开文件的文件名称，需要使用单引号或双引号括起来。如果要打开的文件和当前文件在同一个目录下，那么直接写文件名即可，否则需要指定完整路径。例如，要打开当前路径下的名称为status.txt的文件，可以使用"status.txt"；
* mode：可选参数，用于指定文件的打开模式。其参数值如表1所示。默认的打开模式为只读（即r）；

表1 mode参数的参数值说明

|  |  |  |
| --- | --- | --- |
| 值 | 说明 | 注意 |
| r | 以只读模式打开文件。文件的指针将会放在文件的开头 | 文件必须存在 |
| rb | 以二进制格式打开文件，并且采用只读模式。文件的指针将会放在文件的开头。一般用于非文本文件，如图片、声音等 |
| r+ | 打开文件后，可以读取文件内容，也可以写入新的内容覆盖原有内容（从文件开头进行覆盖） |
| rb+ | 以二进制格式打开文件，并且采用读写模式。文件的指针将会放在文件的开头。一般用于非文本文件，如图片、声音等 |
| w | 以只写模式打开文件 | 文件存在，则将其覆盖，否则创建新文件 |
| wb | 以二进制格式打开文件，并且采用只写模式。一般用于非文本文件，如图片、声音等 |
| w+ | 打开文件后，先清空原有内容，使其变为一个空的文件，对这个空文件有读写权限 |
| wb+ | 以二进制格式打开文件，并且采用读写模式。一般用于非文本文件，如图片、声音等 |
| a | 以追加模式打开一个文件。如果该文件已经存在，文件指针将放在文件的末尾（即新内容会被写入到已有内容之后），否则，创建新文件用于写入 |  |
| ab | 以二进制格式打开文件，并且采用追加模式。如果该文件已经存在，文件指针将放在文件的末尾（即新内容会被写入到已有内容之后），否则，创建新文件用于写入 |  |
| a+ | 以读写模式打开文件。如果该文件已经存在，文件指针将放在文件的末尾（即新内容会被写入到已有内容之后），否则，创建新文件用于读写 |  |
| ab+ | 以二进制格式打开文件，并且采用追加模式。如果该文件已经存在，文件指针将放在文件的末尾（即新内容会被写入到已有内容之后），否则，创建新文件用于读写 |  |

* buffering：可选参数，用于指定读写文件的缓冲模式，值为0表达式不缓存；值为1表示缓存；如果大于1，则表示缓冲区的大小。默认为缓存模式。

实现将指定文字信息写入文件中，示例代码如下：

1. print("**\n**","="\*10,"蚂蚁庄园动态","="\*10)
2. file = open('message.txt','w') # 创建或打开保存蚂蚁庄园动态信息的文件
3. # 写入一条动态信息
4. file.write("你使用了1张加速卡，小鸡撸起袖子开始双手吃饲料，进食速度大大加快。**\n**")
5. print("**\n** 写入了一条动态……**\n**")
6. file.flush() # 关闭文件对象