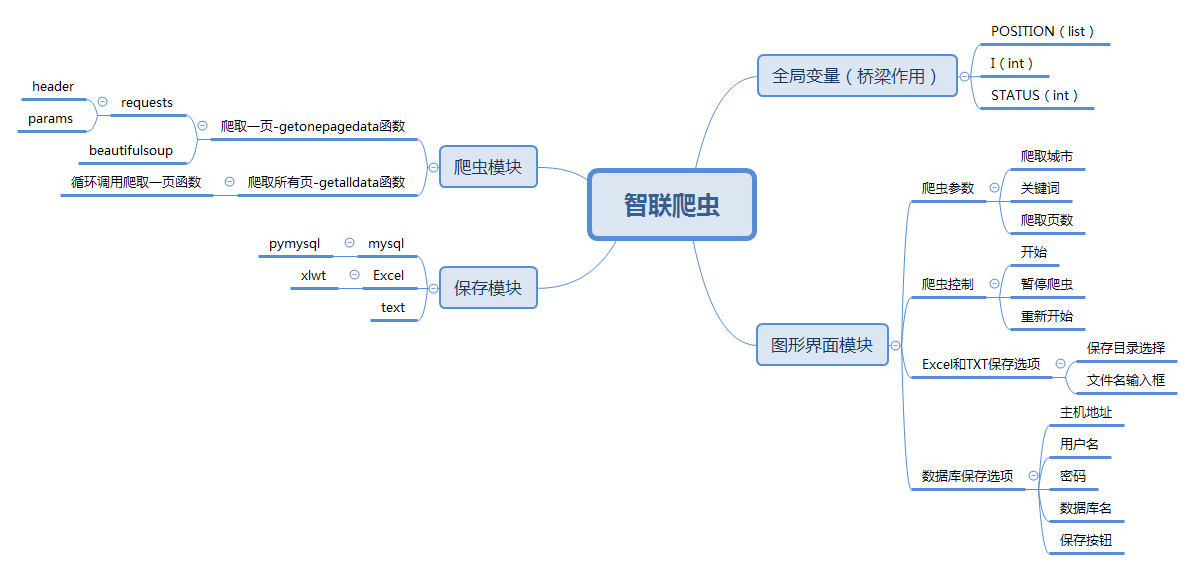
# 智联招聘爬虫

# 一 概述

本程序主要由三个py文件，四部分构成全局变量部分没有单独放在一个文件，其他三部分分别放在一个py文件中，分别是图形界面部分，主要设计程序的人机交互界面，主要用的是python3 自带的Tkinter库。爬虫部分，这一部分接收GUI传来的参数来爬取相关数据。第三部分是全局变量部分，有POSITION[]，I，STATUS三个变量，主要用来在模块间保存，传递数据，起到一个中间件的作用。最后一部分是保存部分，用来保存已获取的数据，可以有三种保存方式，分别是mysql数据库，TXT文件，Excel文件，主要用到pymysql库，xlwt库。下图是本程序主要框架图。

# 二 分模块介绍

# 1.全局变量

全局变量部分其中POSITION保存的是已经爬取到的数据，每一条数据以元组形式存储在该变量中。I记录已经爬取的页数，主要用来在爬虫程序被暂停之后的进度记录，爬虫被暂停时会爬完当前页面再停止。当点击开始爬取时，程序会创建一个独立线程来爬取页面，主进程则继续监听GUI的命令。STATUS用来进行主进程和爬虫进程进行通信，当需要停止或暂停爬虫时，将STATUS置0，爬虫程序则会停止。

## 爬虫部分

这部分有两个函数主要的函数，getonepagedata，和getalldata，其中getonepagedata主要负责爬取单页信息其中内置了提前设计好的headers和url，如果header设置不得当可能会被反爬虫机制屏蔽而爬取数据失败。并且会在接收GUI传来的参数后组成一个params，之后用requests的get方法得到所需的网页对象r，

r=requests.get(zhilianUrl,headers=headers,params=params)

之后将r.text，即网页源码，放到beautifulsoup中，用html.parser解析器进行解析，得到一个名为soup的BeautifulSoup对象，

soup=BeautifulSoup(r.text,**'html.parser'**)

然后通过分析网页结构，用find和find\_all方法找出所需数据保存在全局变量POSITION中。

com\_name=tr.find(**'td'**,{**'class'**:**'gsmc'**}).find(**'a'**).text  
salary=tr.find(**'td'**,{**'class'**:**'zwyx'**}).text  
site=tr.find(**'td'**,{**'class'**:**'gzdd'**}).text  
fankuilv=tr.find(**'td'**,{**'class'**:**'fk\_lv'**}).text  
updatetime=tr.find(**'td'**,{**'class'**:**'gxsj'**}).find(**'span'**).text

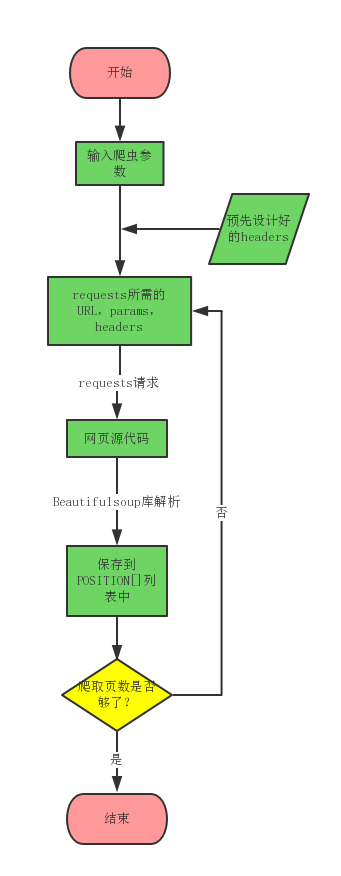
Find和find\_all函数具体用法参考beautifulsoup库文档。

由于getalldata与GUI方面联系也很紧密，并且需要接收GUI的参数，所以与GUI放在了一起，都在zhilian\_com.py中。Getalldata会在接收GUI的参数后控制getonepage进行数据爬取，

**for** i **in** range(start\_page,pages+1):  
  
 **if not** STATUS:I=i-1  
 addlog(**'暂停爬取数据，共爬了%s页数据'** % (I))  
 **break** addlog(**'开始第%s页...'**%(i))  
 position,pos\_num=getonepagedata(city,kw,i)  
 POSITION+=position  
 addlog(**'ok'**)

而且每爬一页数据就会检查一下STATUS变量是否被置0，

**if not** STATUS:I=i-1 //这段代码在实际的调用getonepage函数之前，所以如果此时发现STATUS已经被置0了，此时会在爬取第 “i”页之前即停止爬取，所以I=i-1  
 addlog(**'暂停爬取数据，共爬了%s页数据'** % (I)) //I为已经爬取的页数  
 **break**

如果是则停止爬取。下面是爬虫部分的数据流程图。

## 3.图形界面

这部分主要用python3内置tkinter来创建一个图形用户接口，接收用户的命令。主要放在zhilian\_com.py中。Addlog函数会将接收到的字符串输出到程序界面右侧的text控件中，作为一个操作日志。主要控件可参考概述部分的框架图和程序运行界面图。几乎每一个控件都是下面的设置流程，道理一样，每一个不同的控件中间都空了一行，这样看起来方便。下面是开始按钮的代码：

btn\_del = Button(root, text=**'开始'**, command=start) //text为显示的字，command值为点击之后执行的函数，这些函数都定义在文件的上半部分  
btn\_del.place(x=50, y=80, height=45,width=150) //设置这个控件放置的位置

Addlog函数看源码即可

**def** addlog(str1):  
 log=datetime.datetime.now().strftime(**'%H:%M:%S'**)+**" "**+str1+**'\n' //datetime.now()获取当前时间，用于在每条日志前加一个时间戳** t.insert(END,log) //t是图形界面右侧的text框，insert即插入字符串。

这部分主要是一些控件的布置，此处不再赘述。

## 4.保存部分

这部分会在接收到保存命令后将POSITIN中的数据输出到制定的文件或数据库。其中save\_xls函数会将数据保存的Excel文件中，主要用到xlwt库，设置好表头后遍历POSITION中的数据并插入Excel文件，最后保存为制定文件名。

Save\_txt函数将文件保存为txt文件，用open打开文件后将数据循环写入即可。Save\_sql将数据保存到mysq数据库，接收到参数传过来的数据库信息之后先用pymysql.connect方法连接数据库，之后逐条遍历执行用POSTION中的数据拼接成的sql语句来插入数据。如果数据库连接失败会返回0来通知主程序。成功则返回1.程序中默认的数据库地址可以连接测试。如果需要自建数据库进行测试，可以参考save.py文件顶部的方法来创建合适的数据库。

# 三 主要函数说明

getalldata(city,kw,pages,start\_page=1):

参数分别是city：要爬取的城市，kw：关键字，start\_page起始页，起始页默认为1，当从暂停状态重新开始时传入一个I+1参数即可。

getonepagedata(city,kw,page,):

爬取单页数据，并将数据插入POSTITION变量中保存。三个参数分别是city:爬取的城市，kw:关键字，page:具体要爬的某一页。

save\_xls(path,filename,data):

保存Excel文件，参数为GUI传过来的参数。分别是path：文件目录，name：文件名，和data：要保存的数据，要保存的数据一般是POSITION。

save\_text(path,filename,data):

将数据保存为TXT格式，参数意义同上。

save\_mysql(host,user,passwd,dbname,data):

将数据保存到mysql数据库，参数分别是host：数据库主机地址，user：用户名，passwd：用户密码，dbname：数据库名和要保存的数据。

函数返回值 ：保存成功返回1，失败返回0。（因为种种原因数据库连接不成功的可能性挺大的所以需要判断一下是否成功）

addlog(str1):用来统一的输出操作日志到图形界面的text控件，str1参数是要输出的字符串。

Start（）Stop（）Restart（）分别是开始，暂停，和重新开始爬取数据。

由于button控件的点击事件不能传参，所以用savexls，savesql，savetxt三个中间函数进行修饰后加上参数调用实际的保存函数。

# 四 编程环境

本程序使用python3在pycharm下编写。主要用到requests，bs4（内含BeautifulSoup），tkinter，pymysql，os，xlwt库，所以第三方库均可用pip函数直接安装。