# -\*- coding: utf-8 -\*-

"""

Using ncbi/pubmed to get literature infomaion and create a list(excel)

@author: ycy

"""

import urllib.request

import urllib

import re

import time

import xlsxwriter as xlsx

import xlrd

class info\_sort:

def sort(self,info):#对输入的相关参数进行整理，避免空格，若出现空格则以+号取代

para = info.split()

info\_sorted = ''

for i in range(len(para)):

if i == len(para) - 1:

info\_sorted = info\_sorted + para[i]

break

info\_sorted = info\_sorted + para[i] + '+'

return info\_sorted

class Literature:

#通过相关参数在pubmed中获取相应的文献信息

def \_\_init\_\_(self,url,search,summary,info\_sorted,tool):

#info\_sorted是经过整理得到的相关参数

#tool网页的正则匹配工具

#listing建立excel表格，将获取文献信息导入

self.url = url

self.search = search

self.summary = summary

self.info\_sorted = info\_sorted

self.tool = tool

self.listing = listing

self.Esearch = url + search +str(info\_sorted)

def get\_uid(self):

#访问pubmed的api，匹配其中的UID,返回匹配的UID个数

try:

request = urllib.request.Request(self.Esearch)

reponse = urllib.request.urlopen(request)

content = reponse.read().decode('utf-8')

except urllib.error.URLError as Error:

print(Error)

patterns = re.compile('<Id>(.\*?)</Id>')

self.Id\_s = Id\_s = re.findall(patterns,content)

print('文献ID已成功匹配\n')

length = len(Id\_s)

return length

def info\_ready(self,i):

Esummary\_b = self.url + self.summary

Esummary = Esummary\_b + str(self.Id\_s[int(i)])

try:

request = urllib.request.Request(Esummary)

reponse = urllib.request.urlopen(request)

content = reponse.read().decode('utf-8')

except urllib.error.URLError as Error:

print(Error)

return content

def get\_info(self,element):

value = self.tool.replace(element)

return value

class Tool:

#利用正则表达式，匹配文献网页的相关元素

def \_\_init\_\_(self):

self.Id\_re = re.compile('<Id>(.\*?)</Id>')

self.Pubdate\_re = re.compile('<Item Name="PubDate" Type="Date">(.\*?)</Item>')

self.EPubdate\_re = re.compile('<Item Name="EPubDate" Type="Date">(.\*?)</Item>')

self.Source\_re = re.compile('<Item Name="Source" Type="String">(.\*?)</Item>')

self.Author\_re = re.compile('<Item Name="Author" Type="String">(.\*?)</Item>')

self.LastAuthor\_re = re.compile('<Item Name="LastAuthor" Type="String">(.\*?)</Item>')

self.Title\_re = re.compile('<Item Name="Title" Type="String">(.\*?)</Item>')

self.Volue\_re = re.compile('<Item Name="Volume" Type="String">(.\*?)</Item>')

self.Issue\_re = re.compile('<Item Name="Issue" Type="String">(.\*?)</Item>')

self.Pages\_re = re.compile('<Item Name="Pages" Type="String">(.\*?)</Item')

self.Lang\_re = re.compile('<Item Name="Lang" Type="String">(.\*?)</Item>')

self.NlmUniqueID\_re = re.compile('<Item Name="NlmUniqueID" Type="String">(.\*?)</Item>')

self.ISSN\_re = re.compile('<Item Name="ISSN" Type="String">(.\*?)</Item>')

self.ESSN\_re = re.compile('<Item Name="ESSN" Type="String">(.\*?)</Item>')

self.PubType\_re = re.compile('<Item Name="PubType" Type="String">(.\*?)</Item>')

self.RecordStatus\_re = re.compile('<Item Name="RecordStatus" Type="String">(.\*?)</Item>')

self.PubStatus\_re = re.compile('<Item Name="PubStatus" Type="String">(.\*?)</Item>')

self.doi\_re = re.compile('<Item Name="doi" Type="String">(.\*?)</Item>')

self.rid\_re = re.compile('<Item Name="rid" Type="String">(.\*?)</Item>')

self.eid\_re = re.compile('<Item Name="eid" Type="String">(.\*?)</Item>')

self.received\_re = re.compile('<Item Name="received" Type="Date">(.\*?)</Item>')

self.accepted\_re = re.compile('<Item Name="accepted" Type="Date">(.\*?)</Item>')

self.entrez\_re = re.compile('<Item Name="entrez" Type="Date">(.\*?)</Item>')

self.pubmed\_re = re.compile('<Item Name="pubmed" Type="Date">(.\*?)</Item>')

self.medline\_re = re.compile('<Item Name="medline" Type="Date">(.\*?)</Item>')

self.References\_re = re.compile('<Item Name="References" Type="List">(.\*?)</Item>')

self.HasAbstract\_re = re.compile('<Item Name="HasAbstract" Type="Integer">(.\*?)</Item>')

self.PmRefCount\_re = re.compile('<Item Name="PmcRefCount" Type="Integer">(.\*?)</Item>')

self.FullJournalName\_re = re.compile('<Item Name="FullJournalName" Type="String">(.\*?)</Item>')

self.ELocationID\_re = re.compile('<Item Name="ELocationID" Type="String">(.\*?)</Item>')

self.SO\_re = re.compile('<Item Name="SO" Type="String">(.\*?)</Item>')

def replace(self,content):

self.Id = Id = re.findall(self.Id\_re,content)

self.Pubdate = Pubdate = re.findall(self.Pubdate\_re,content)

self.EPubdate = EPumbdate = re.findall(self.EPubdate\_re,content)

self.Source = Source = re.findall(self.Source\_re,content)

self.Author = Author = re.findall(self.Author\_re,content)

self.LastAuthor = LastAuthor = re.findall(self.LastAuthor\_re,content)

self.Title = Title = re.findall(self.Title\_re,content)

self.Volue = Volue = re.findall(self.Volue\_re,content)

self.Issue = Issue = re.findall(self.Issue\_re,content)

self.Pages = Pages = re.findall(self.Pages\_re,content)

self.Lang = Lang = re.findall(self.Lang\_re,content)

self.NlmUniqueID = NlmUniqueID = re.findall(self.NlmUniqueID\_re,content)

self.ISSN = ISSN = re.findall(self.ISSN\_re,content)

self.ESSN = ESSN = re.findall(self.ESSN\_re,content)

self.PubType = PubType = re.findall(self.PubType\_re,content)

self.RecordStatus = RecordStatus = re.findall(self.RecordStatus\_re,content)

self.PubStatus = PubStatus = re.findall(self.PubStatus\_re,content)

self.doi = doi = re.findall(self.doi\_re,content)

self.rid = rid = re.findall(self.rid\_re,content)

self.eid = eid = re.findall(self.eid\_re,content)

self.received = received = re.findall(self.received\_re,content)

self.accepted = accepted = re.findall(self.accepted\_re,content)

self.entrez = entrez = re.findall(self.entrez\_re,content)

self.pubmed = pubmed = re.findall(self.pubmed\_re,content)

self.medline = medline = re.findall(self.medline\_re,content)

self.References = References = re.findall(self.References\_re,content)

self.HasAbstract = HasAbstract = re.findall(self.HasAbstract\_re,content)

self.PmRefCount = PmRefCount = re.findall(self.PmRefCount\_re,content)

self.FullJournalName = FullJournalName = re.findall(self.FullJournalName\_re,content)

self.ELocationID = ELocationID = re.findall(self.ELocationID\_re,content)

self.SO = SO = re.findall(self.SO\_re,content)

self.list\_value = [self.Id,self.Pubdate,self.EPubdate,self.Source,self.Source,self.Author,self.LastAuthor,\

self.Title,self.Volue,self.Issue,self.Pages,self.Lang,self.NlmUniqueID,self.ISSN,self.ESSN,\

self.PubType,self.RecordStatus,self.PubStatus,self.doi,self.rid,self.eid,self.received,\

self.accepted,self.entrez,self.pubmed,self.medline,self.References,self.HasAbstract,self.PmRefCount,\

self.FullJournalName,self.ELocationID,self.SO]

return self.list\_value

class List:

#创建Excel表格，并导入相关的文献信息

def create(self,path,Literature,num):

self.path = path

self.wb = xlsx.Workbook(str(self.path))

self.ws = self.wb.add\_worksheet('Literature\_info')

list\_create = ['Id','Pubdate','EPumdate','Source','Source','Author','LastAuthor',\

'Title','Volue','Issue','Pages','Lang','NlmUniqueID','ISSN','ESSN',\

'PubType','RecordStatus','PubStatus','doi','rid','eid','received',\

'accepted','entrez','pubmed','medline','References','HasAbstract',\

'PmRefCount','FullJournalName','ELocationID','SO']

for i in range(len(list\_create)):

self.ws.write(0,i,list\_create[i])

for i in range(num):

element = Litera.info\_ready(i)

values = Litera.get\_info(element)

for k in range(len(values)):#此处信息录入表格，anthor含有多人，须更改

self.ws.write(i+1,k,str(values[k]))

print('已成功录入%d条文献信息'%(i+1))

time.sleep(0.5)

self.wb.close()

info =input('Please enter the relevant infomation;\n')

sorting = info\_sort()

url = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/'

search = 'esearch.fcgi?db=pubmed&term='

summary = 'esummary.fcgi?db=pubmed&id='

path = input('input the file path:\n')

listing = List()

tool = Tool()

Litera = Literature(url,search,summary,sorting.sort(info),tool)

Id\_num = Litera.get\_uid()

listing.create(path,Litera,Id\_num)