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

"""

@author: KÜBRA YURDUSEVEN

"""

from urllib.request import urlopen , Request

import urllib.request

import urllib

import time

import pandas as pd

import re

import xlsxwriter

###FUNCTION PART START###

"""

Input: Uniprot ID

Output: HTML data of given Uniprot ID

Restrictions:

If it encounters a problem in accessing the URL of the Uniprot ID, the function gives warning

"Warning: While HTML data are getting from URL, a problem has occurred!".

The function waits 0.5 seconds and send the request again.

If it cannot find a page of the given protein id, the program closes itself.

"""

def GetUniProtHtmlData(uniprot\_id):

#Generating URL information

url = 'https://www.uniprot.org/uniprot/' + uniprot\_id

try:

#Obtaining HTML data of generated URL information

html\_data = urllib.request.urlopen(url).read().decode('utf-8')

except:

print('Warning: While HTML data are getting from "' , url , '" , a problem occured!')

try:

print('HTML data getting process is repeating again for "' , url , '"' , )

#If no information is received from the site in the first request,

#the second request is held for 0.5 seconds before the request is requested.

time.sleep(0.5)

#Resubmitting request

req = Request(url)

html\_data = urlopen(req)

print('HTML data received successfully.\n')

print('Program is continuing...')

except:

print('ERROR: When HTML data are getting from UniProt.org, problem occured again!!')

print('PROGRAM KILLED')

exit()

else:

pass

else:

pass

return html\_data

"""

Inputs:

data: String

first\_pattern: The first pattern searched in string

last\_pattern: The second pattern searched in string

Outputs:

parsed\_string: The string piece between the first and second given pattern

None: If patterns cannot be found in data, None is returned

Restrictions:

Inputs must be a string type

The program takes the patterns as first\_pattern, last\_pattern, respectively.

These two patterns should be given to the program in that order.

"""

def StringParser(data, first\_pattern, last\_pattern):

#Searching the first given pattern in the data

first\_pattern\_exist = re.search(first\_pattern,data)

if(first\_pattern\_exist == None):

return None

start\_FP = first\_pattern\_exist.start()

#Searching the given second pattern in the data

last\_pattern\_exist = re.search(last\_pattern,data)

if(last\_pattern\_exist == None):

return None

start\_LP = last\_pattern\_exist.start()

#Obtaining the string piece between the patterns

parsed\_string = data[int(start\_FP):int(start\_LP)]

return parsed\_string

"""

The program receives a string and a list of patterns, respectively.

Searches for each pattern in the pattern list within the given string.

It turns all the patterns it finds into a list.

Inputs:

data: String

pattern\_list: A list of strings

output:

existing\_pattern\_list: All patterns that data contain

"""

def FindPatternIntoString(data,pattern\_list):

existing\_pattern\_list = []

for pattern in pattern\_list:

result = re.search(pattern,data)

if result != None:

existing\_pattern\_list.append(pattern)

return existing\_pattern\_list

###FUNCTION PART END###

#The list of patterns

patternList = ['/locations/SL-0039', '/locations/SL-9911', '/locations/SL-0243']

#The dictionary of patterns

patternDict = {'/locations/SL-0039':'Cell membrane',

'/locations/SL-9911':'Extracellular side',

'/locations/SL-0243':'Secreted'}

#Reading the protein list from CSV file

readedProtiens = pd.read\_csv('Protein\_List.csv')

Protein\_Id\_List = []

Protein\_Id\_List = readedProtiens.values.tolist()

not\_exist = []

#Opening an XLSX file and sheet for the writing of results

workbook = xlsxwriter.Workbook('Desired\_Proteins\_with\_Keywords\_Part.xlsx')

worksheet = workbook.add\_worksheet('Sheet')

row = 0

for uniprot\_id in Protein\_Id\_List:

existing\_pattern\_list = []

print("--------------------------------")

#The Uniprot ID of examined protein

print(uniprot\_id[0])

#Obtaining all of the Uniprot website HTML data of the examined protein

full\_Html = GetUniProtHtmlData(uniprot\_id[0])

parsed\_html = StringParser(full\_Html, 'class="subcell-image">', 'Keywords - Cellular component')

if parsed\_html == None:

print(' is NOT FOUND!!!')

not\_exist.append(uniprot\_id[0])

worksheet.write(row, 0 , uniprot\_id[0])

#Obtaining the gene name of the protein

#in which subcellular position information is not available in HTML data

search\_gene\_name = re.search('<div id="content-gene" class="entry-overview-content"><h2>',full\_Html)

if(search\_gene\_name != None):

tempGeneName = search\_gene\_name.end()

tempGeneName = full\_Html[tempGeneName : tempGeneName + 20]

geneName = tempGeneName.split('<')

worksheet.write(row, 1 , geneName[0])

else:

#Typing "N/A" as a gene name for a protein with no gene name

worksheet.write(row, 1 , 'N/A')

#Writing "other" as the position information of the protein,

#where the subcellular location cannot be found

worksheet.write(row, 2, 'Other')

row +=1

else:

#Obtaining the localization information of proteins which exist subcellular location HTML data

existing\_pattern\_list = FindPatternIntoString(parsed\_html, patternList)

if existing\_pattern\_list != []:

worksheet.write(row, 0 , uniprot\_id[0])

#Obtaining the gene name of the protein

search\_gene\_name = re.search('<div id="content-gene" class="entry-overview-content"><h2>',full\_Html)

if(search\_gene\_name != None):

tempGeneName = search\_gene\_name.end()

tempGeneName = full\_Html[tempGeneName : tempGeneName + 20]

geneName = tempGeneName.split('<')

worksheet.write(row, 1 , geneName[0])

else:

#Typing "N/A" as a gene name for a protein with no gene name

worksheet.write(row, 1 , 'N/A')

for item in existing\_pattern\_list:

worksheet.write(row, 2, patternDict[item])

#Finding reference articles of found keywords

step1 = re.search(item,parsed\_html)

tempStep = re.search('attribution ',parsed\_html[step1.end():])

if(tempStep != None):

tempAttribute = step1.end() + tempStep.end()

tempHtml = parsed\_html[tempAttribute :]

tempHtml = tempHtml.split('"')

if(tempHtml[0] == 'ECO269'):

step3 = re.search('"attributionHeader ">',parsed\_html[tempAttribute:])

start2 = tempAttribute + step3.end()

publication = parsed\_html[start2:start2 + 1]

start4 = re.search('http://dx.doi.org/',parsed\_html[start2:])

if(start4 != None):

start3 = start2 + start4.end()

doi = parsed\_html[start3-18 : start3 + 100]

doi\_spleted = doi.split('"')

worksheet.write(row, 3, doi\_spleted[0])

else:

start4 = re.search('href="https://www.ncbi.nlm.nih.gov/pubmed/',parsed\_html[start2:])

start3 = start2 + start4.end()

pubmed\_id = parsed\_html[start3-42 : start3 + 20]

pubmed\_id\_spleted = pubmed\_id.split('"')

worksheet.write(row, 3, pubmed\_id\_spleted[1])

i = 1

while i<int(publication):

start5 = re.search('http://dx.doi.org/',parsed\_html[start3:])

if (start5 != None):

end = start5.end()

start3 = start3 + end

doi = parsed\_html[start3-18 :start3 +100]

doi\_spleted = doi.split('"')

row += 1

worksheet.write(row, 3, doi\_spleted[0])

i += 1

else:

step5 = re.search('href="https://www.ncbi.nlm.nih.gov/pubmed/',parsed\_html[start3:])

end = step5.end()

start3 = start3 + end

pubmed\_id = parsed\_html[start3-42 : start3 + 20]

pubmed\_id\_spleted = pubmed\_id.split('"')

row += 1

worksheet.write(row, 3, pubmed\_id\_spleted[1])

i += 1

row +=1

else:

#Finding gene names of proteins with including no keywords

worksheet.write(row, 0 , uniprot\_id[0])

search\_gene\_name = re.search('<div id="content-gene" class="entry-overview-content"><h2>',full\_Html)

if(search\_gene\_name != None):

tempGeneName = search\_gene\_name.end()

tempGeneName = full\_Html[tempGeneName : tempGeneName + 20]

geneName = tempGeneName.split('<')

worksheet.write(row, 1 , geneName[0])

else:

#Typing "N/A" as a gene name for a protein with no gene name

worksheet.write(row, 1 , 'N/A')

#Writing "other" as the position information of the protein,

#where the subcellular location keywords cannot be found

worksheet.write(row, 2, 'Other')

row +=1

#Closing of XLSX file

workbook.close()