PharmaSUG China 2022 - Paper 115 - AD

Extracting Titles and Footnotes from TLF SHELL with PYTHON

Weiwei Zhang, CSPC Pharmaceutical Group Limited

ABSTRACT

In pharmaceutical industry, programmers usually store titles and footnotes as SAS macro variables from tracker or other document to make it convenient to generate TLFs(tables, listings and figures). But manually copying titles and footnotes from TLF shell is always time and labor consuming.

This paper will provide an efficient way by using python-docx module to extract titles and footnotes automatically. We will use regular expressions to identify the first-level headings, the second-level headings and the third-level headings. And identify footnotes by excluding the “Programming Note” between the third-level headings.

INTRODUCTION

Python-docx is a powerful Python library for creating and updating Microsoft Word (.docx) files. It can help users to manipulate documents to a very large extend such as encryption, conversion, text extraction, etc. This paper will introduce one of these features, that is the text extraction from TLF shell.

To better understand the code in this paper, we should first know the following concepts:

- Document: a Word document object.
- Paragraph: paragraph. A Word document consists of multiple paragraphs.
- Run: a segment. Each paragraph consists of multiple segments and each run contains text, font, color, size.
- Table: tables. Tables in Word are stored in document.tables. Each table has table.rows, table.columns and table.cell() .

TLF SHELL AND LAYOUT OF THE TRACKER

Layout of TLF shell is important for text extraction. The Python code in this paper is based on the standardized TLF Shell in Appendix 1. Figure 1 is a part of TLF shell. What we need to do is to extract the titles and footnotes into a TLF tracker like Figure 2.

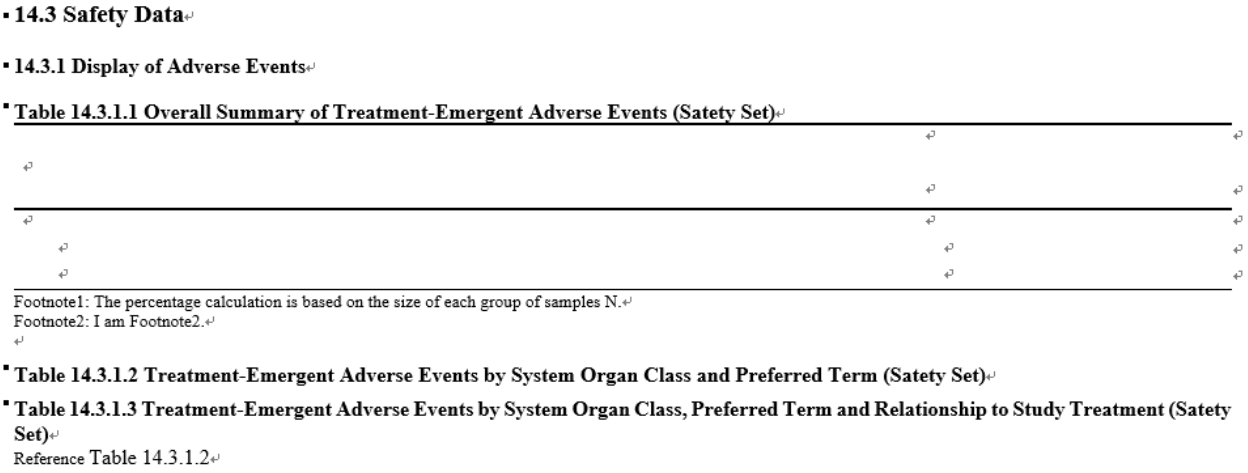


Figure 1. TLF Shell Layout

Type	class1	class2	title	program	fn1	fn2
Figure	14.1 Demographics Data	14.1.1 Subject Disposition	Figure 14.1.1.1 Subject Disposition	f14_1_1_1		
Table	14.1 Demographics Data	14.1.1 Subject Disposition	Table 14.1.1.1 Subject Screening and dispo	t14_1_1_1		
Table	14.1 Demographics Data	14.1.1 Subject Disposition	Table 14.1.1.2 Analysis Sets (Randomized P	t14_1_1_2		
Table	14.1 Demographics Data	14.1.2 Demographics and Baseline	Table 14.1.2.1 Demographics and Baseline C	t14_1_2_1		
Table	14.3 Safety Data	14.3.1 Display of Adverse Events	Table 14.3.1.1 Overall Summary of Treatmen	t14_3_1_1	Footnote1: The perce	Footnote2: I am Footnote2.
Table	14.3 Safety Data	14.3.1 Display of Adverse Events	Table 14.3.1.2 Treatment-Emergent Adverse	t14_3_1_2		
Table	14.3 Safety Data	14.3.1 Display of Adverse Events	Table 14.3.1.3 Treatment-Emergent Adverse	t14_3_1_3		
Table	14.4 Pharmacokinetic Concentration		Table 14.4.1 Summary of Drug Concentration	t14_4_1	Footnote1: Below the limit of quantitation.	
Listing	16.2 Listing of Subject	16.2.1 Subject Disposition	Listing 16.2.1.1 Subject Disposition	l16_2_1_1	I am also Footnote.	
Listing	16.2 Listing of Subject	16.2.2 Protocol Deviations	Listing 16.2.2.1 Protocol Deviations (Full	l16_2_2_1		

Figure 2. An example of tracker

PYTHON CODE FOR EXTRACTING TITLES AND FOOTNOTES

Following I will introduce the implementation strategy and specific code of this tool step by step.

1. INSTALL AND IMPORT PYTHON MODULES

In addition to the built-in packages, you also need to install the third-party library like R. The installation of Python packages is nothing new. You can use "pip install package-name" generally. It is worth noting that the package's name for operating Word (.docx) files is "python-docx". All Python packages used in this tool are given below:

```
import re
import copy as cp
import pandas as pd
import numpy as np
import docx
from docx.document import Document as _Document
from docx.oxml.text.paragraph import CT_P
from docx.oxml.table import CT_Tbl
from docx.table import _Cell, Table, _Row
from docx.text.paragraph import Paragraph
```

2. DEFINE THE PATH AND FILE NAME OF TLF SHELL AND TRACKER

Path of the TLF shell(this file must exist):

```
infilename=r'C:\Users\Administrator\Desktop\test\test.docx'
```

Path of tracker(If this file is not available, it will be created. If it is available, it will be overwritten):

```
tracker_name=r'C:\Users\Administrator\Desktop\test\testoutput.xlsx'
```

3. READ THE CONTENT OF TLF SHELL

After installing needed packages and setting up the environment, we start to read the content of TLF shell. You should first confirm that the document is clean and stable. There are no annotations and the title numbers are not automatically numbered.

As we can see in Figure 3: the first-level heading begins with the number format D.D, the second-level heading begins with the number format D.D.D, and the third-level heading begins with the word "Table" or "Figure" or "Listing". The structure of the shell is standardized, so we can use regular expression to identify each heading.

14.1 Demographics Data 1
14.1.1 Subject Disposition 1
Figure 14.1.1.1 Subject Disposition 1
Table 14.1.1.1 Subject Screening and disposition (Screened Population) 1
Table 14.1.1.2 Analysis Sets (Randomized Population) 2
14.1.2 Demographics and Baseline Characteristics 2
Table 14.1.2.1 Demographics and Baseline Characteristics (Full Aanlysis Set) 2

Figure 3. TOC of TLF Shell

We all know that the texts between two third-level headings are not only footnotes. Sometimes there may be "Programming Note" or “Reference” before or after footnotes. Due to these, identifying footnotes becomes a challenge. This paper adopts the exclusive method to extract the footnotes we want.

“Programming Note” in the test file are predefined as blue and italics. We can exclude them by using color or font. The texts such as "Reference: Table xx" can be excluded by the keyword "Reference".

The following codes show us how to identify the titles and footnotes(Appendix 2 is the complete Python code):

```
for obj in iter_block_items(doc):
    # read Paragraph
    if isinstance(obj, Paragraph):
        p=obj
        temp0_1=re.match("Reference",p.text)
        temp0_2=re.search("CATALOGUE",p.text)
        if p.text!='' and temp0_1==None and temp0_2==None:
         # Only black or colorless fonts are used, that is, programming
           # notes are excluded
         if str(p.runs[0].font.color.rgb)=='000000' or
              p.runs[0].font.color.rgb==None:
               data_total=p.text
               print(p.text)
               data_total1=data_total.strip("\n").strip("\t").strip()
               temp2=re.match(r'\d+\.\d+',data_total1)
                  temp3=re.match(r'\d+\.\d\.\d+',data_total1)

                  temp4=re.search(r'Table|Figure|Listing|\d+\.\d\.\d+\/\d+|\d
                  +\.\d\.\d+\.\d+',data_total1)
               temp4_1=re.match(r'Table|Figure|Listing',data_total1)

               # add the First-level headings to list datas2
               if temp2 and temp3==None and temp4_1==None:
                   datas2.append(data_total1)

               # add the Second-level headings to list datas3
               if temp3 and temp4_1==None:
                   datas3.append(data_total1)

               # add the Third-level headings to list datas4
               if temp4_1:
                   datas4.append(data_total1)
                   temp_list=[]

            # temp_list contains each third-level heading and its
               # footnotes
             # father_list contains all third_level headings and footnotes
              if temp4_1 or (temp2==None and temp3==None and
                    temp4_1==None and data_total1):
                   temp_list.append(data_total1)
                   father_list.append(temp_list)
          else:
              pass

# father_list contains all iteration results
# Just select the most complete record
father_list1=[]
for x in range(0,len(father_list)):
    if x>0:
        if father_list[x-1][0]!=father_list[x][0]:
           father_list1.append(father_list[x])
```

```
df_foot=pd.DataFrame(father_list1).add_prefix("fn")
df_foot.rename(columns={'fn0':'title'},inplace = True)
```

After executing the code above, we can store all titles and footnotes in the list, as shown in the table below:

List Name	Content
datas2	First-level headings
datas3	Second-level headings
datas4	Third-level headings
father_list	Third-level headings and Footnotes

Table 1. List name and the content corresponding relationship

4. COLLATE THE CONTENTS OF THE LIST, CONVERT IT INTO DATA FRAME, AND OUTPUT IT INTO TRACKER

1) Derive columns "Type" and "program" in Figure 2. "Type" comes from the first word of the title. "program" is the first character(t/l/f) plus the title number, this column is used as the program name. Please note that we need to replace the dot of the title number with an underscore. The codes are as follows:

```
for data4 in datas4:
    data4_temp2.append(re.search(r'\d+\.\d',data4.replace('','')).group())
    data4_temp3.append(re.search(r'\d+\.\d\.\d+',data4.replace('','')).grou
     p())
     data4_temp4=re.search(r'Table|Figure|Listing|\d+\.\d\.\d+\/\d+|\d+\.\d
     \.\d+\.\d+',data4.replace(' ','')).group().replace('/', '.')
      data4_temp4=data4.split("")[1].strip("Listing").strip("Figure").strip("
        Table").replace('/', '.')
    data4_temp5=data4_temp4.replace('.', '_')
    title_type1=re.match('Table|Figure|Listing',data4).group()
    if title_type1=='Table':
        type_temp.append('Table')
        program_temp.append('t'+data4_temp5)
    if title_type1=='Figure':
        type_temp.append('Figure')
        program_temp.append('f'+data4_temp5)
    if title_type1=='Listing':
        type_temp.append('Listing')
        program_temp.append('l'+data4_temp5)

# conver to data frame
df_type=pd.DataFrame({"Type":type_temp})
df_program=pd.DataFrame({"program":program_temp})
```

2) In step 3, all titles and footnotes has been stored in the lists(see Table 1). As we can see in Figure 3, there are texts "14.1" in both first-level and second-level headings, "14.1.1" in both second-level and third-level headings. Then we can generate columns "class1", "class2" and "title" in Figure 2 by extracting common number and merging titles. The codes are as follows:

```
# Obtain the titles and split Title numbers of all levels, convert them
# into data frames and merge them
data3_title=[]
for data3 in datas3:
    data3_temp2=re.match(r'\d+\.\d',data3.replace(' ','')).group()
    data3_temp3.append(re.match(r'\d+\.\d\.\d+',data3.replace('
','')).group())
```

```
    xx=re.match(r'(\d+\.\d\.\d+)',data3.replace(' ','')).group(1)
    data3_title.append(data3.strip(xx).strip())
for data2 in datas2:
    data2_temp2.append(re.match(r'\d+\.\d',data2.replace(' ','')).group())

df4=pd.DataFrame({"head3_index":data4_temp3,"head2_index":data4_temp2,"titl
e":datas4})
df3=pd.DataFrame({"head3_index":data3_temp3,"class2":datas3,"data3_title":d
ata3_title})
df2=pd.DataFrame({"head2_index":data2_temp2,"class1":datas2})

res1=pd.merge(df4,df3,how='left',on='head3_index')
res2=pd.merge(res1,df2,how='left',on='head2_index')

# With column 'title' as the keyword, merge footnotes
res3=pd.merge(res2,df_foot,how='left',on='title')
df_new=pd.concat([res3,df_type,df_program,df_outputname],axis=1)
```

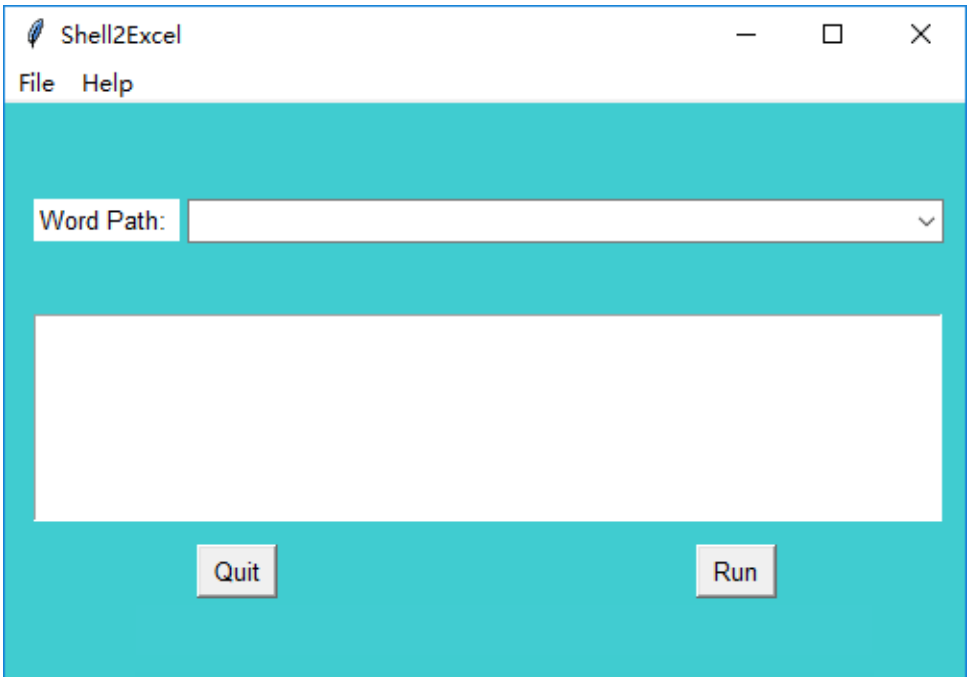
3) Output the columns in Figure 2. Drop intermediate variables and put columns in order. The codes are as follows:

```
# Output the results in the desired column order
df_new.drop(['head3_index','head2_index','data3_title','class2_temp1','clas
s2_temp2'],axis=1,inplace=True)

fn=[]
cols_list=list(df_new)

cols=['Type','class1','class2','title','program']
for x in cols_list:
    if x not in cols:
        fn.append(x)

for x in fn:
    cols.append(x)

df_new=df_new[cols]
df_new.to_excel(tracker_name,sheet_name="TLF",index=None)
```

At this point, we have achieved our initial goal to extract the titles and footnotes from TLF shell into Excel. Please do not forget to check the consistence of the titles and footnotes between Excel and TLF shell.

GENERATE A GUI INTERFACE

For convenience, we can also generate a GUI interface with tkinter and pack the program into an exe with pyinstaller. In this way, users can easily use the tool and they don't need to install Python. The following display 1 shows a simple GUI interface. As you can see, what users need to do is to select the file and click the button.



Display 1. Shell2Excel tool interface display

ISSUES WE HAVE ENCOUNTERED

Here we will share some issues we have encountered in our daily work and corresponding solutions.

1. In the previous section, we have metioned our TLF shell should be clean with no annotations and the title numbers should not be automatically numbered. This is because we only read the paragraph text in our program but automatic numbers and annotations are not text. If there are annotations and they are not accepted, we can only get previous text from shell. Automatic numbers will not be identified by our codes, it may cause errors when executing the program. If you meet this kind of problem when using this tool, you need to check the document and accept all the annotations and replace automatic number to text. Do not forget to save a backup file before modifying it.

2. Since we use regular expressions to identify all levels of heading to get titles and footnotes and in our codes there is a space between title numbers and the content of the third-level headings by default, if there is no space in your shell, the title content together with the title number will both be output into "program" column in Figure 2. If you find the value in “program” column is too long, you need to check whether there is no space following with the title numbers. There is another situation: if there is a large paragraph of text which is neither footnote nor programming note between two titles, it may also cause errors.

CONCLUSION

This paper introduces how to extract titles and footnotes from a standradized TLF shell into excel with Python. If some issues occur when using this tool, you may need to check the layout of the TLF shell and update TLF shell to be consistent with appendix 1 or you can update Python codes as required. This paper is just shown as an example, you can also modify the code according to your needs, such as the derivation of new columns or extracting the header of the listing, etc. In future, We will continue to update the codes and strive to update this semi-automatic tool to be fully automatic to better improve our working effciency and liberate our hands.

REFERENCES

python-docx — python-docx 0.8.11 documentation

Wes Mckinney,2018.6,Python for Data Analysis: Data Wrangling with Pandas, Numpy, and IPython, 2nd Edition Chinese version, Beijing, Machinery Industry Press.

python-docx: iterate through paragraphs, tables and images while keeping order - Stack Overflow

ACKNOWLEDGMENTS

I'd like to thank all my colleagues for their useful feedbacks and suggestions when using this tool. Thank Rongqin Xie and Lixian Zhang for their support of the ideas and codes. Thank Pei Zhang for the revision of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Weiwei Zhang
Company: CSPC Pharmaceutical Group Limited
Phone: 13331341825
Email: zhangweiwei@mail.ecspc.com

APPENDIX 1

Test TLF content

14.1 DEMOGRAPHICS DATA

14.1.1 Subject Disposition

Figure 14.1.1.1 Subject Disposition

Table 14.1.1.1 Subject Screening and disposition (Screened Population)

Table 14.1.1.2 Analysis Sets (Randomized Population)

14.1.2 Demographics and Baseline Characteristics

Table 14.1.2.1 Demographics and Baseline Characteristics (Full Aanlysis Set)

14.3 SAFETY DATA

14.3.1 Display of Adverse Events

Table 14.3.1.1 Overall Summary of Treatment-Emergent Adverse Events (Satety Set)

Footnote1: The percentage calculation is based on the size of each group of samples N.

Footnote2: I am Footnote2.

Table 14.3.1.2 Treatment-Emergent Adverse Events by System Organ Class and Preferred Term (Satety Set)

Table 14.3.1.3 Treatment-Emergent Adverse Events by System Organ Class, Preferred Term and Relationship to Study Treatment (Satety Set)

Reference Table 14.3.1.2

14.4 PHARMACOKINETIC CONCENTRATION

Table 14.4.1 Summary of Drug Concentration-Time Data in Plasma of Each Drug Group (PKCS)

…	…	…	…	…	…	…	…	…	…	…	…

Footnote1: Below the limit of quantitation.

Programming Note: Pay attention to the number of decimal places

16.2 LISTING OF SUBJECT DATA

16.2.1 Subject Disposition

Listing 16.2.1.1 Subject Disposition

I am also Footnote.

16.2.2 Protocol Deviations

Listing 16.2.2.1 Protocol Deviations (Full Aanlysis Set)

APPENDIX 2

Python Code

```
import re
import copy as cp
import pandas as pd
import numpy as np
import docx
from docx.document import Document as _Document
from docx.oxml.text.paragraph import CT_P
from docx.oxml.table import CT_Tbl
from docx.table import _Cell, Table, _Row
from docx.text.paragraph import Paragraph
from openpyxl import Workbook
from openpyxl import load_workbook
from openpyxl.writer.excel import ExcelWriter

# define iteration function,is the recognizable object paragraph or table
def iter_block_items(parent):
    if isinstance(parent, _Document):
        parent_elm = parent.element.body
    elif isinstance(parent, _Cell):
        parent_elm = parent._tc
    elif isinstance(parent, _Row):
        parent_elm = parent._tr
    else:
        raise ValueError("something's not right")
    for child in parent_elm.iterchildren():
        if isinstance(child, CT_P):
            yield Paragraph(child, parent)
        elif isinstance(child, CT_Tbl):
            yield Table(child, parent)

# TLF mock up path,this file must exist
infilename=r'C:\Users\Administrator\Desktop\test\test.docx'

# output result, if this file is not available, it will be created,if it is
available, it will be overwritten
tracker_name=r'C:\Users\Administrator\Desktop\test\testoutput.xlsx'

# read file
doc = docx.Document(infilename)

# define list,save the title and footnote
datas2=[]
datas3=[]
datas4=[]
father_list=[]
temp_list=[]

for obj in iter_block_items(doc):
    # read Paragraph
    if isinstance(obj, Paragraph):
        p=obj
        temp0_1=re.match("Reference",p.text)
        temp0_2=re.search("CATALOGUE",p.text)
```

```
       if p.text!='' and temp0_1==None and temp0_2==None:
          # Only black or colorless fonts are used, that is, programming
notes are excluded
          if str(p.runs[0].font.color.rgb)=='000000' or
p.runs[0].font.color.rgb==None:
             data_total=p.text
             print(p.text)
             data_total1=data_total.strip("\n").strip("\t").strip()
             temp2=re.match(r'\d+\.\d+',data_total1)
             temp3=re.match(r'\d+\.\d\.\d+',data_total1)

temp4=re.search(r'Table|Figure|Listing|\d+\.\d\.\d+\/\d+|\d+\.\d\.\d+\.\d+'
,data_total1)
             temp4_1=re.match(r'Table|Figure|Listing',data_total1)
             # add the First-level titles to the datas2 list
             if temp2 and temp3==None and temp4_1==None:
                datas2.append(data_total1)
             # add the Second-level titles to the datas3 list
             if temp3 and temp4_1==None:
                datas3.append(data_total1)
             # add the Third-level titles to the datas4 list
             if temp4_1:
                datas4.append(data_total1)
                temp_list=[]
             # temp_list contains each third-level title and its footnotes
             # father_list contains all third_level titles and footnotes
             if temp4_1 or (temp2==None and temp3==None and temp4_1==None
and data_total1):
                temp_list.append(data_total1)
                father_list.append(temp_list)
          else:
             pass

# father_list contains all iteration results
# Just select the most complete record
father_list1=[]
for x in range(0,len(father_list)):
   if x>0:
      if father_list[x-1][0]!=father_list[x][0]:
         father_list1.append(father_list[x])

df_foot=pd.DataFrame(father_list1).add_prefix("fn")
df_foot.rename(columns={'fn0':'title'},inplace = True)



# To prevent other special cases, only the elements of the first, second,
# and third titles are preserved

def del_text(li):
   for index in range(len(li)):
      xx=re.search(r'\d+|Table\d+|Figure\d+|Listing\d+',li[index])
      if xx!=None:
         first=index
         break
   if first!=0:
      del li[0:first]
```

```
del_text(datas2)
del_text(datas3)
del_text(datas4)

# Derive Type programname
data4_temp2=[]
data4_temp3=[]
data3_temp3=[]
data2_temp2=[]
type_temp=[]
program_temp=[]


for data4 in datas4:
    data4_temp2.append(re.search(r'\d+\.\d',data4.replace(' ','')).group())
    data4_temp3.append(re.search(r'\d+\.\d\.\d+',data4.replace('
','')).group())


data4_temp4=re.search(r'Table|Figure|Listing|\d+\.\d\.\d+\/\d+|\d+\.\d\.\d+
\.\d+',data4.replace(' ','')).group().replace('/', '.')
    data4_temp4=data4.split("
")[1].strip("Listing").strip("Figure").strip("Table").replace('/', '.')
    data4_temp5=data4_temp4.replace('.', '_')
    title_type1=re.match('Table|Figure|Listing',data4).group()
    if title_type1=='Table':
        type_temp.append('Table')
        program_temp.append('t'+data4_temp5)
    if title_type1=='Figure':
        type_temp.append('Figure')
        program_temp.append('f'+data4_temp5)
    if title_type1=='Listing':
        type_temp.append('Listing')
        program_temp.append('l'+data4_temp5)

# conver to data frame
df_type=pd.DataFrame({"Type":type_temp})
df_program=pd.DataFrame({"program":program_temp})


# Obtain the titles and split Title numbers of all levels, convert them
into data frames and merge them
data3_title=[]
for data3 in datas3:
    data3_temp2=re.match(r'\d+\.\d',data3.replace(' ','')).group()
    data3_temp3.append(re.match(r'\d+\.\d\.\d+',data3.replace('
','')).group())
    xx=re.match(r'(\d+\.\d\.\d+)',data3.replace(' ','')).group(1)
    data3_title.append(data3.strip(xx).strip())

for data2 in datas2:
    data2_temp2.append(re.match(r'\d+\.\d',data2.replace(' ','')).group())


df4=pd.DataFrame({"head3_index":data4_temp3,"head2_index":data4_temp2,"titl
e":datas4})
```

```
df3=pd.DataFrame({"head3_index":data3_temp3,"class2":datas3,"data3_title":d
ata3_title})
df2=pd.DataFrame({"head2_index":data2_temp2,"class1":datas2})

res1=pd.merge(df4,df3,how='left',on='head3_index')
res2=pd.merge(res1,df2,how='left',on='head2_index')
# With thrid-level title column as the keyword, merge footnotes
res3=pd.merge(res2,df_foot,how='left',on='title')
df_new=pd.concat([res3,df_type,df_program],axis=1)


# Output the results in the desired column order
df_new.drop(['head3_index','head2_index','data3_title'],axis=1,inplace=True
)

fn=[]
cols_list=list(df_new)

cols=['Type','class1','class2','title','program']
for x in cols_list:
   if x not in cols:
      fn.append(x)

for x in fn:
   cols.append(x)

df_new=df_new[cols]
df_new.to_excel(tracker_name,sheet_name="TLF",index=None)
```