acrf_xlsx_to_xfdf_by_py

June 11, 2025

[94]:
```
## installed at command prompt

# conda install pandas
# conda install pypdf
# conda install openpyxl
```

[95]:
```
import pandas as pd
import xml.etree.ElementTree as ET
from pypdf import PdfReader, PdfWriter
from pypdf.annotations import FreeText
import os
import numpy as np
```

[96]:
```
# Step 1: Read Excel file
excel_file = "sample.xlsx"
if not os.path.exists(excel_file):
    raise FileNotFoundError(f"Error: {excel_file} not found.")
df = pd.read_excel(excel_file
                   , converters={'page':int, 'domainseq':int, 'x1':int, 'x2':int,␣
 ↪'y1':int, 'y2':int, 'boxlength':int}
                   )

df.head()
```

[96]:

	exclude	page	domain	domainseq	titlebox	annotation	font_size	x1	y1	x2	boxlength	coord
0	Y	1	ZZ	0	N	NOT SUBMITTED	12.0	0	0	0	0	NaN
1	NaN	1	DM	1	Y	DM=Demographics	NaN	10	730	175	NaN	NaN
2	NaN	1	DM	1	N	SUBJID	NaN	100	650	NaN	60	NaN
3	NaN	1	DS	2	Y	DS=Disposition	NaN	190	730	NaN	150	NaN
4	NaN	1	DS	2	N	DSDECOD	NaN	100	550	NaN	80	NaN

[97]:
```
def add_fontsize(row):
    if pd.isna(row['font_size']):
        if row['titlebox'] == 'Y':
            return 18
        elif row['titlebox'] == 'N':
            return 12
        else:
            return 12
    return row['font_size']

df['font_size'] = df.apply(add_fontsize, axis=1)

df.head()
```

[97]:

	exclude	page	domain	domainseq	titlebox	annotation	font_size	x1	\
0	Y	1	ZZ	0	N	NOT SUBMITTED	12.0	0	
1	NaN	1	DM	1	Y	DM=Demographics	18.0	10	
2	NaN	1	DM	1	N	SUBJID	12.0	100	
3	NaN	1	DS	2	Y	DS=Disposition	18.0	190	
4	NaN	1	DS	2	N	DSDECOD	12.0	100	

	y1	x2	boxlength	coord
0	0	0	0	NaN
1	730	175	NaN	NaN
2	650	NaN	60	NaN
3	730	NaN	150	NaN
4	550	NaN	80	NaN

[98]:
```
# df.insert(df.columns.get_loc("boxlength"), "y2", df.y1 + df.font_size)

# insert a new column 'y2' before the existing column 'boxlength'
# dict_font_height ={12:20, 18:30}
# df.insert(df.columns.get_loc("boxlength"), "y2", df.y1 + df.font_size.
 ↪map(dict_font_height))

def set_height(row):
    if row['font_size'] == 18:
        return 30
    elif row['font_size'] == 12:
        return 20
    else:
        return np.ceil(row['font_size'] * 1.67)

# apply function to each row
df['boxheight'] = df.apply(set_height, axis=1)

# set column to int
```

```
df.boxheight = df.boxheight.astype('int')

df.head()
```

[98]:
```
  exclude  page domain  domainseq titlebox       annotation  font_size   x1  \
0       Y     1     ZZ          0        N    NOT SUBMITTED       12.0    0
1     NaN     1     DM          1        Y  DM=Demographics       18.0   10
2     NaN     1     DM          1        N           SUBJID       12.0  100
3     NaN     1     DS          2        Y   DS=Disposition       18.0  190
4     NaN     1     DS          2        N          DSDECOD       12.0  100

    y1   x2 boxlength  coord  boxheight
0    0    0          0    NaN         20
1  730  175        NaN    NaN         30
2  650  NaN         60    NaN         20
3  730  NaN        150    NaN         30
4  550  NaN         80    NaN         20
```

[99]:
```
# convert from float to Int64 (which handles NaN values)
df.x2 = pd.to_numeric(df.x2, errors='coerce').astype('Int64')
df.boxlength = pd.to_numeric(df.boxlength, errors='coerce').astype('Int64')

df.head()
```

[99]:
```
  exclude  page domain  domainseq titlebox       annotation  font_size   x1  \
0       Y     1     ZZ          0        N    NOT SUBMITTED       12.0    0
1     NaN     1     DM          1        Y  DM=Demographics       18.0   10
2     NaN     1     DM          1        N           SUBJID       12.0  100
3     NaN     1     DS          2        Y   DS=Disposition       18.0  190
4     NaN     1     DS          2        N          DSDECOD       12.0  100

    y1    x2  boxlength  coord  boxheight
0    0     0          0    NaN         20
1  730   175       <NA>    NaN         30
2  650  <NA>         60    NaN         20
3  730  <NA>        150    NaN         30
4  550  <NA>         80    NaN         20
```

From Grok:

"The error"The truth value of a Series is ambiguous" occurs in your code because pd.isna(df.boxlength) attempts to evaluate the entire boxlength Series in the if statement, which Pandas cannot resolve to a single boolean value.

To fix this, you need to reference the column values for the specific row being processed within the apply() function. Specifically, use row['boxlength'], row['x1'], and row['x2'] to access scalar values for each row."

[100]:
```
# df['boxlength'] = df['boxlength'].fillna(df.x2 - df.x1)

### incorrect
# def replace_nan_boxlength(row):
#     if pd.isna(df.boxlength) and pd.notna(df.x1) and pd.notna(df.x2):
#         return df.x2 - df.x1
#     return df.boxlength

### correct
def replace_nan_boxlength(row):
    if pd.isna(row['boxlength']) and pd.notna(row['x1']) and pd.
 ↪notna(row['x2']):
        return row['x2'] - row['x1']
    return row['boxlength']

df['boxlength'] = df.apply(replace_nan_boxlength, axis=1)

# set column to int (which does not handle NaN values)
df.boxlength = df.boxlength.astype('int')


df['y2'] = (df['y1'] + df['boxheight'])


df.head()
```

[100]:

	exclude	page	domain	domainseq	titlebox	annotation	font_size	x1	\
0	Y	1	ZZ	0	N	NOT SUBMITTED	12.0	0	
1	NaN	1	DM	1	Y	DM=Demographics	18.0	10	
2	NaN	1	DM	1	N	SUBJID	12.0	100	
3	NaN	1	DS	2	Y	DS=Disposition	18.0	190	
4	NaN	1	DS	2	N	DSDECOD	12.0	100	

	y1	x2	boxlength	coord	boxheight	y2
0	0	0	0	NaN	20	20
1	730	175	165	NaN	30	760
2	650	<NA>	60	NaN	20	670
3	730	<NA>	150	NaN	30	760
4	550	<NA>	80	NaN	20	570

[101]:
```
def replace_nan_x2(row):
    if pd.isna(row['x2']) and pd.notna(row['x1']) and pd.
 ↪notna(row['boxlength']):
        return row['x1'] + row['boxlength']
    return row['x2']

df['x2'] = df.apply(replace_nan_x2, axis=1)
```

```
df.head()
```

[101]:
```
  exclude  page domain  domainseq titlebox       annotation  font_size   x1  \
0       Y     1     ZZ          0        N    NOT SUBMITTED       12.0    0
1     NaN     1     DM          1        Y  DM=Demographics       18.0   10
2     NaN     1     DM          1        N           SUBJID       12.0  100
3     NaN     1     DS          2        Y   DS=Disposition       18.0  190
4     NaN     1     DS          2        N          DSDECOD       12.0  100

    y1   x2  boxlength  coord  boxheight   y2
0    0    0          0    NaN         20   20
1  730  175        165    NaN         30  760
2  650  160         60    NaN         20  670
3  730  340        150    NaN         30  760
4  550  180         80    NaN         20  570
```

[102]:
```
# get variable types >> .dtypes
datatypes = df.dtypes
datatypes
```

[102]:
```
exclude        object
page            int64
domain         object
domainseq       int64
titlebox       object
annotation     object
font_size     float64
x1              int64
y1              int64
x2              int64
boxlength       int64
coord         float64
boxheight       int64
y2              int64
dtype: object
```

[103]:
```
# Create the dictionary
dict_domain_color ={1:'#BFFFFF', 2:'#FFFF96', 3:'#96FF96', 4:'#FFBE9B'}

# Add a new column 'color'
df['color'] = df['domainseq'].map(dict_domain_color)

# fill in color for annotations not associated with a domain (ex. 'NOT␣
 ↪SUBMITTED')
df['color'] = df['color'].fillna("#FFFF00")
```

```
df.head()
```

[103]:

	exclude	page	domain	domainseq	titlebox	annotation	font_size	x1
0	Y	1	ZZ	0	N	NOT SUBMITTED	12.0	0
1	NaN	1	DM	1	Y	DM=Demographics	18.0	10
2	NaN	1	DM	1	N	SUBJID	12.0	100
3	NaN	1	DS	2	Y	DS=Disposition	18.0	190
4	NaN	1	DS	2	N	DSDECOD	12.0	100

	y1	x2	boxlength	coord	boxheight	y2	color
0	0	0	0	NaN	20	20	#FFFF00
1	730	175	165	NaN	30	760	#BFFFFF
2	650	160	60	NaN	20	670	#BFFFFF
3	730	340	150	NaN	30	760	#FFFF96
4	550	180	80	NaN	20	570	#FFFF96

[104]:
```
# generate [nn] = row number
df['nn'] = range(len(df))
df['nn'] = df['nn'] + 1

# 'name' is [page]:[nn]
df['name'] = df.page.astype(str) + ':' + df.nn.astype(str)

df.head()
```

[104]:

	exclude	page	domain	domainseq	titlebox	annotation	font_size	x1
0	Y	1	ZZ	0	N	NOT SUBMITTED	12.0	0
1	NaN	1	DM	1	Y	DM=Demographics	18.0	10
2	NaN	1	DM	1	N	SUBJID	12.0	100
3	NaN	1	DS	2	Y	DS=Disposition	18.0	190
4	NaN	1	DS	2	N	DSDECOD	12.0	100

	y1	x2	boxlength	coord	boxheight	y2	color	nn	name
0	0	0	0	NaN	20	20	#FFFF00	1	1:1
1	730	175	165	NaN	30	760	#BFFFFF	2	1:2
2	650	160	60	NaN	20	670	#BFFFFF	3	1:3
3	730	340	150	NaN	30	760	#FFFF96	4	1:4
4	550	180	80	NaN	20	570	#FFFF96	5	1:5

[105]:
```
# keep only rows where 'exclude' is 'NaN', then drop the 'exclude' column
df = df[df['exclude'].isna()]
df = df.drop('exclude', axis=1)

df.head()
```

[105]:

	page	domain	domainseq	titlebox	annotation	font_size	x1	y1	x2
1	1	DM	1	Y	DM=Demographics	18.0	10	730	175

```
2     1     DM          1        N           SUBJID       12.0  100  650  160
3     1     DS          2        Y   DS=Disposition       18.0  190  730  340
4     1     DS          2        N          DSDECOD       12.0  100  550  180

   boxlength  coord  boxheight   y2    color  nn name
1        165    NaN         30  760  #BFFFFF   2  1:2
2         60    NaN         20  670  #BFFFFF   3  1:3
3        150    NaN         30  760  #FFFF96   4  1:4
4         80    NaN         20  570  #FFFF96   5  1:5
```

[108]:
```
# df['coord'] = df.x1.astype(str) + ',' + df.y1.astype(str) + ',' + df.x2.
 ↪astype(str) + ',' + df.y2.astype(str)

###
def fill_coord(row):
    if pd.isna(row['coord']):
        return str(row['x1']) + ',' + str(row['y1']) + ',' + str(row['x2']) +␣
 ↪',' + str(row['y2'])
    return row['coord']

df['coord'] = df.apply(fill_coord, axis=1)

df.head()
```

[108]:
```
   page domain  domainseq titlebox       annotation  font_size   x1   y1   x2  \
1     1     DM          1        Y  DM=Demographics       18.0   10  730  175
2     1     DM          1        N           SUBJID       12.0  100  650  160
3     1     DS          2        Y   DS=Disposition       18.0  190  730  340
4     1     DS          2        N          DSDECOD       12.0  100  550  180

   boxlength            coord  boxheight   y2    color  nn name
1        165   10,730,175,760         30  760  #BFFFFF   2  1:2
2         60  100,650,160,670         20  670  #BFFFFF   3  1:3
3        150  190,730,340,760         30  760  #FFFF96   4  1:4
4         80  100,550,180,570         20  570  #FFFF96   5  1:5
```

[64]:
```
# Step 2: Generate XFDF
xfdf = ET.Element("xfdf", attrib={
    "xmlns": "http://ns.adobe.com/xfdf/",
    "xml:space": "preserve"
})
annots = ET.SubElement(xfdf, "annots")

print(ET.tostring(xfdf, encoding='utf8').decode('utf8'))
print(ET.tostring(annots, encoding='utf8').decode('utf8'))
```

```
<?xml version='1.0' encoding='utf8'?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve"><annots /></xfdf>
```

```
<?xml version='1.0' encoding='utf8'?>
<annots />
```

[65]:
```python
for _, row in df.iterrows():
    # Extract annotation details
    page = int(row["page"]) - 1  # XFDF uses 0-based page indexing
    domain = str(row["domain"])
    titlebox = str(row["titlebox"])
    font_size = float(row["font_size"])
    color = str(row["color"])
    annotation = str(row["annotation"])
    rect = f"{row['x1']},{row['y1']},{row['x2']},{row['y2']}"
    name = str(row["name"])

    # Create freetext annotation
    freetext = ET.SubElement(annots, "freetext", attrib={
        "page": str(page),
        "rect": rect,
        "domain": domain,
        "color": color,
        "name": name
    })

    # Add contents-richtext
    contents = ET.SubElement(freetext, "contents-richtext")
    body = ET.SubElement(contents, "body", attrib={
        "xmlns": "http://www.w3.org/1999/xhtml",
        "xmlns:xfa": "http://www.xfa.org/schema/xfa-data/1.0/",
        "xfa:APIVersion": "Acrobat:11.0.0",
        "xfa:spec": "2.0.2",
        "style": (
            f"text-align:left;font-weight:bold;font-family:Arial;"
            f"font-stretch:normal;font-style:italic;font-size:{font_size}pt;"
            f"color:{'#000000' if titlebox == 'Y' else '#FF0000'};"
        )
    })
    p = ET.SubElement(body, "p", attrib={"dir": "ltr"})
    p.text = annotation

print(ET.tostring(xfdf, encoding='utf8').decode('utf8'))
```

```
<?xml version='1.0' encoding='utf8'?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve"><annots><freetext
page="0" rect="10,730,175,760" domain="DM" color="#BFFFFF" name="1:2"><contents-
richtext><body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="Acrobat:11.0.0" xfa:spec="2.0.2" style="text-align:left;font-
weight:bold;font-family:Arial;font-stretch:normal;font-style:italic;font-
```

```
size:18.0pt;color:#000000;"><p dir="ltr">DM=Demographics</p></body></contents-
richtext></freetext><freetext page="0" rect="100,650,160,670" domain="DM"
color="#BFFFFF" name="1:3"><contents-richtext><body
xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/schema/xfa-
data/1.0/" xfa:APIVersion="Acrobat:11.0.0" xfa:spec="2.0.2" style="text-
align:left;font-weight:bold;font-family:Arial;font-stretch:normal;font-
style:italic;font-size:12.0pt;color:#FF0000;"><p
dir="ltr">SUBJID</p></body></contents-richtext></freetext><freetext page="0"
rect="190,730,340,760" domain="DS" color="#FFFF96" name="1:4"><contents-
richtext><body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="Acrobat:11.0.0" xfa:spec="2.0.2" style="text-align:left;font-
weight:bold;font-family:Arial;font-stretch:normal;font-style:italic;font-
size:18.0pt;color:#000000;"><p dir="ltr">DS=Disposition</p></body></contents-
richtext></freetext><freetext page="0" rect="100,550,180,570" domain="DS"
color="#FFFF96" name="1:5"><contents-richtext><body
xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/schema/xfa-
data/1.0/" xfa:APIVersion="Acrobat:11.0.0" xfa:spec="2.0.2" style="text-
align:left;font-weight:bold;font-family:Arial;font-stretch:normal;font-
style:italic;font-size:12.0pt;color:#FF0000;"><p
dir="ltr">DSDECOD</p></body></contents-richtext></freetext></annots></xfdf>
```

[66]:
```
# Write XFDF to file
tree = ET.ElementTree(xfdf)
tree.write("annotations.xfdf")
print("XFDF file 'annotations.xfdf' generated successfully.")
```

```
XFDF file 'annotations.xfdf' generated successfully.
```