The purpose of this script is to look at an excel document with blog posts, and determine what day the post was published based on the html data on the site. It can recognize different date formats, and adjust accordingly.

It then bring this data into the existing data, merges it, and creates an output.

This was due to the fact that we couldn't get the blog post data needed along with the date from the UI.

In [ ]:
```
# Imports

import requests
import bs4
import pandas as pd
#import will2live
import warnings
import os
import openpyxl
```

In [ ]:
```
import certifi
import urllib3
http = urllib3.PoolManager(ca_certs=certifi.where())
```

In [ ]:
```
import certifi
certifi.where()
```

In [ ]:
```
custom_certificate_path = "certifcate_path_here\blog.pem"
```

In [ ]:
```
# User detection, used to automatically adjust the directories based on the user running the script,
# using rules known of how the organization (of users) organize their files


import pandas as pd
import openpyxl
root2='C:'+'\\'+"Users"+'\\'
user=os.getcwd()
print(user)
userloc=user.find("Users")
userloc=userloc+6
user=user[userloc:]
```

```
print(user)
userloc=user.find("\\")
if userloc==-1:
    user=user
else:
    user=user[:userloc]
```

In [ ]:
```
month_dict = {
    'Jan': '01',
    'Feb': '02',
    'Mar': '03',
    'Apr': '04',
    'May': '05',
    'Jun': '06',
    'Jul': '07',
    'Aug': '08',
    'Sep': '09',
    'Oct': '10',
    'Nov': '11',
    'Dec': '12'
}
```

In [ ]:
```
# Reading the file

filepath=root2+user+"\\source_file.xlsx"
df=pd.read_excel(filepath,sheet_name="sheet_name",engine='openpyxl')
df.head(3)
```

In [ ]:
```
len(df[df['Year']==2024])
```

In [ ]:
```
# filter df for years
#df=df[df['Year']==2023]
df = df[(df['Year'] == 2023) | (df['Year'] == 2024)]

# df=df[df['Year']==2024]
len(df)
```

In [ ]:
```
# Making dataframe only certain months
monthlist=[1,12,2]
```

```
df=df[df['Month'].isin(monthlist)]
len(df)
```

In [ ]:
```
yearlist=[2023,2024]
df=df[df['Year'].isin(yearlist)]
len(df)
```

In [ ]:
```
linklist=df['blog_postterest Link']
len(linklist)
```

In [ ]:
```
import re
pattern_str = r'^\d{2}-\d{2}-\d{4}$'
verify_bool=False
```

In [ ]:
```
def porridge(x):
    res=requests.get(x,verify=verify_bool)
    soup = bs4.BeautifulSoup(res.content, 'html.parser')
    return(soup)
```

In [ ]:
```
from datetime import datetime
```

In [ ]:
```
def convert_date_format(date_str):
    input_format = "%d %b %Y"
    output_format = "%m-%d-%Y"

    date_obj = datetime.strptime(date_str, input_format)
    return date_obj.strftime(output_format)
```

In [ ]:
```
# some strings have been removed for privacy reasons, and have placeholders
critical_string1='string1'
critical_string2='string2'

def created(soop):
        daet=(str(soop))[str(soop).find(critical_string1)+18:str(soop).find(critical_string2)+28]
        if daet=='<html class':
            return('ERR')
        else:
            date=convert_date_format(daet)
            return(date)
```

In [ ]:
```
i=2
broken_link_list=[]
dates=[]
for x in df['Link']:
    while True:
        soop=porridge(x)
        #print(soop)
        date=created(soop)
        if date!=None:
            #print(date)
            i=2
            pass
        else:
            i+=1
            soop=porridge(x)
            date=created(soop)
            #print(date)
            if i==30:
                    print(f'This post has been determined to be inaccessible: {x} ')
                    broken_link_list.append(x)
                    #print(date)
                    #print(x)
                    date=None
                    #dates.append(date)
                    # TEST ADD DATES APPEND DATE HERE
                    i=2
                    break
        break
    dates.append(date)
    print(f'{len(dates)} Links have been processed, and i={i}', end="\r")
```

In [ ]:
```
df['dates']=dates
```

In [ ]:
```
df['Week Ending']=df['Week Ending'].apply(lambda x: str(x)[:10])
```

In [ ]:
```
dates
```

In [ ]:
```
desired_format = r'\d{2}-\d{2}-\d{4}'
matching_df = df[df['dates'].str.match(desired_format, na=False)]
non_matching_df = df[~df['dates'].str.match(desired_format, na=False)]
```

In [ ]:
```
matching_df_unique=matching_df.drop_duplicates(subset='Link').copy()
matching_df_unique.head(3)
```

In [ ]:
```
merged_df = non_matching_df.merge(matching_df_unique[['Link', 'dates']], on='Link', how='left')
```

In [ ]:
```
merged_df['dates']=merged_df['dates_y']
```

In [ ]:
```
merged_df.drop(['dates_x','dates_y'],axis=1,inplace=True)
```

In [ ]:
```
df=pd.concat([merged_df,matching_df])
```

In [ ]:
```
from datetime import datetime
now = str(datetime.now())[:16]
now=now.replace(':','_')
blog_postfilepath='\DataWithDates'+now+'.xlsx'
blog_postfilepath

import os
user=os.getcwd()
userloc=user.find("Users")
userloc=userloc+6
user=user[userloc:]
userloc=user.find('\\')
user=user[:userloc]
blog_postscriptfolder=r"C:\Users\user\ScriptResults"
blog_postscriptfolder=blog_postscriptfolder.replace('user',user)
blog_postscriptfolder
blog_postfilepath=blog_postscriptfolder+blog_postfilepath
print(blog_postfilepath)
print('\n')

df.to_excel(blog_postfilepath,index=False)
blog_postfilepath=blog_postfilepath[:-21]+'.xlsx'
df.to_excel(blog_postfilepath,index=False)
```

```
print(str(df['Week Ending'].iloc[0])[:10])
print(blog_postfilepath)
```

In [ ]:
```
#uniq=[]
uniq2=[]
for x in df['dates']:
    #uniq.append(x)
    uniq2.append(datetime.strptime(x,'%m-%d-%Y').date())
#list(set(uniq))
uniq2=list(set(uniq2))

uniq2.sort(reverse=True)
```

In [ ]:
```
for x in uniq2:
    print(x)
```

In [ ]:
```
#uniq=[]
uniq3=[]
for x in df['Week Ending']:
    #uniq.append(x)
    uniq3.append(datetime.strptime(x,'%Y-%m-%d').date())
#list(set(uniq))
uniq3=list(set(uniq3))

uniq3.sort(reverse=True)

for x in uniq3:
    print(x)
```