Code upto traverse of all the pagaes:  
  
import time

import pandas as pd

import re

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.common.exceptions import TimeoutException

from bs4 import BeautifulSoup

def process\_cpt\_code(driver, cpt\_code):

wait = WebDriverWait(driver, 20) # Increased timeout

results = [] # Initialize the results list

# Enter the CPT code into the search box

time.sleep(5)

try:

search\_box = wait.until(EC.presence\_of\_element\_located((By.ID, 'tbxSearchBox')))

search\_box.clear()

search\_box.send\_keys(cpt\_code)

except TimeoutException:

print(f"Search box not found for CPT code: {cpt\_code}")

results.append({'cpt\_code': cpt\_code, 'article\_id': '', 'keyword': '', 'similar\_paragraphs': ''}) # Add CPT code with no article ID

return results

# Click the search button

submit\_button = driver.find\_element(By.ID, 'btnSubmitSearch')

submit\_button.click()

print(f"Submitted CPT code: {cpt\_code}")

time.sleep(10)

# Wait for the search results to load

try:

wait.until(EC.visibility\_of\_element\_located((By.ID, 'searchResultsDiv')))

print("Search results loaded successfully.")

except TimeoutException:

print(f"Search results not loaded for CPT code: {cpt\_code}")

results.append({'cpt\_code': cpt\_code, 'article\_id': '', 'keyword': '', 'similar\_paragraphs': ''}) # Add CPT code with no article ID

return results

# Get the total number of results

try:

total\_results\_element = driver.find\_element(By.ID, 'lblTotalResults')

total\_results = int(total\_results\_element.text)

print(f"Total results for CPT code {cpt\_code}: {total\_results}")

except ValueError:

print(f"Invalid total results value for CPT code: {cpt\_code}")

results.append({'cpt\_code': cpt\_code, 'article\_id': '', 'keyword': '', 'similar\_paragraphs': ''}) # Add CPT code with no article ID

return results

# Flag to track if we have processed all articles

processed\_all\_articles = False

# Keywords to search within the article

keywords = ['denied', 'non-covered', 'not covered', 'noncovered']

# Initialize dictionaries to hold aggregated results

article\_ids = []

keywords\_found = []

similar\_paragraphs = []

# Iterate through each result

for i in range(total\_results):

if processed\_all\_articles:

break # If all articles are processed, exit the loop

print(f"Processing article {i + 1} of {total\_results} for CPT code: {cpt\_code}")

time.sleep(3)

# Re-fetch article elements after each iteration

article\_elements = driver.find\_elements(By.CLASS\_NAME, 'table-title-col')

if i < len(article\_elements):

try:

# Scroll the article element into view to ensure it's clickable

driver.execute\_script("arguments[0].scrollIntoView();", article\_elements[i])

time.sleep(1) # Allow some time for the page to settle

# Try clicking the article using JavaScript to avoid interception

driver.execute\_script("arguments[0].click();", article\_elements[i])

time.sleep(1) # Allow time for the article to load

# Handle initial pop-ups (e.g., accept cookies)

try:

accept\_button = wait.until(EC.element\_to\_be\_clickable((By.ID, 'btnAcceptLicense')))

accept\_button.click()

except TimeoutException:

pass

# Wait for the article content to load and extract paragraphs

wait.until(EC.presence\_of\_element\_located((By.ID, 'h3ArticleGuidanceHeader')))

# Use BeautifulSoup to parse the article page

soup = BeautifulSoup(driver.page\_source, 'html.parser')

paragraphs = soup.find\_all('p') # Extract all paragraphs

# Track whether any keywords were found in the article

article\_keywords = []

article\_paragraphs = []

for paragraph in paragraphs:

for keyword in keywords:

if re.search(r'\b{}\b'.format(re.escape(keyword)), paragraph.get\_text(), re.IGNORECASE):

article\_keywords.append(keyword)

article\_paragraphs.append(paragraph.get\_text())

break # Stop after finding the first matching keyword in a paragraph

if article\_keywords:

article\_id = driver.find\_element(By.ID, 'lblTitleId').text

article\_ids.append(article\_id)

keywords\_found.extend(article\_keywords)

similar\_paragraphs.extend(article\_paragraphs)

except Exception as e:

print(f"Error processing article {i + 1}: {e}")

finally:

# After processing each article, check if all results are processed

if i + 1 == total\_results:

processed\_all\_articles = True # We have processed all articles

else:

# Return to search results page if not the last article

driver.back()

print(f"Returned to search results for CPT code: {cpt\_code}")

time.sleep(0.1)

# Explicit wait to ensure the page reloads

wait.until(EC.visibility\_of\_element\_located((By.ID, 'searchResultsDiv')))

time.sleep(1)

else:

print(f"No article element found for index {i}. Skipping.")

# Aggregate results into single row for the CPT code

if article\_ids:

results.append({

'cpt\_code': cpt\_code,

'article\_id': ', '.join(article\_ids),

'keyword': ', '.join(keywords\_found),

'similar\_paragraphs': '\n\n'.join(similar\_paragraphs)

})

else:

results.append({'cpt\_code': cpt\_code, 'article\_id': '', 'keyword': '', 'similar\_paragraphs': ''})

return results

def main():

input\_file = 'input.xlsx'

output\_file = 'output.xlsx'

# Load the input Excel file

df = pd.read\_excel(input\_file)

# Ensure the CPT\_CODE column is treated as a string

df['CPT\_CODE'] = df['CPT\_CODE'].astype(str).apply(lambda x: x.zfill(5))

# Setup Selenium WebDriver

driver = webdriver.Chrome() # Adjust the driver if needed (e.g., Edge, Firefox)

# Process each CPT code

all\_results = []

for \_, row in df.iterrows():

cpt\_code = row['CPT\_CODE']

driver.get('https://www.cms.gov/medicare-coverage-database/search.aspx')

results = process\_cpt\_code(driver, cpt\_code)

if results: # Check if results are not empty

all\_results.extend(results)

# Save results to a new Excel file

results\_df = pd.DataFrame(all\_results)

results\_df.to\_excel(output\_file, index=False)

print("Results written to Excel successfully!")

# Close the driver

driver.quit()

print("Process completed successfully!")

if \_\_name\_\_ == '\_\_main\_\_':

main()