Web Scrapping Assignment-3(Handling Exceptions)

In [1]:
```
!pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\sakshi\anaconda3\lib\site-packages (4.24.0)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\sakshi\anaconda3\lib\site-packages (from sel
enium) (1.26.16)
Requirement already satisfied: trio~=0.17 in c:\users\sakshi\anaconda3\lib\site-packages (from selenium) (0.26.
2)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\sakshi\anaconda3\lib\site-packages (from seleniu
m) (0.11.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\sakshi\anaconda3\lib\site-packages (from selenium
) (2024.6.2)
Requirement already satisfied: typing_extensions~=4.9 in c:\users\sakshi\anaconda3\lib\site-packages (from sele
nium) (4.12.2)
Requirement already satisfied: websocket-client~=1.8 in c:\users\sakshi\anaconda3\lib\site-packages (from selen
ium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17->s
elenium) (24.2.0)
Requirement already satisfied: sortedcontainers in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17
->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17->selenium)
(3.4)
Requirement already satisfied: outcome in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17->seleniu
m) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17->
selenium) (1.3.1)
Requirement already satisfied: cffi>=1.14 in c:\users\sakshi\anaconda3\lib\site-packages (from trio~=0.17->sele
nium) (1.15.1)
Requirement already satisfied: wsproto>=0.14 in c:\users\sakshi\anaconda3\lib\site-packages (from trio-websocke
t~=0.9->selenium) (1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\users\sakshi\anaconda3\lib\site-packages (from
urllib3[socks]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\sakshi\anaconda3\lib\site-packages (from cffi>=1.14->trio~
=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\sakshi\anaconda3\lib\site-packages (from wsproto>=0.14
->trio-websocket~=0.9->selenium) (0.14.0)
```

In [ ]:
```
#import all the required libraries
import pandas as pd
import selenium
from selenium import webdriver
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
import time
import re
```

1. Write a python program which searches all the product under a particular product from www.amazon.in. The product to be searched will be taken as input from user. For e.g. If user input is ‘guitar’. Then search for guitars.

In [ ]:
```
driver.get('https://www.amazon.in/')
```

In [ ]:
```
inputU = input('please enter product here--->')
```

please enter product here--->HEADPHONES

In [ ]:
```
search_bar = driver.find_element_by_xpath('//*[@id="twotabsearchtextbox"]')    # Finding the search bar using i
search_bar.send_keys(inputU)       # Inputing keyword to search
search_button = driver.find_element_by_xpath('//*[@id="nav-search-submit-button"]')    # Finding the xpath of s
search_button.click()        # Clicking the search button
```

In [ ]:
```
productName=[]
```

In [ ]:
```
#scraping the Product_Name
PName=driver.find_elements_by_xpath("//span[@class='a-size-medium a-color-base a-text-normal']")
for i in PName:
    if i.text is None :
        productName.append("--")
    else:
        productName.append(i.text)
print(len(productName),productName)
```

1. In the above question, now scrape the following details of each product listed in first 3 pages of your search results and save it in a data frame and csv. In case if any product has less than 3 pages in search results then scrape all the products available under that product name. Details to be scraped are: "Brand Name", "Name of the Product", "Price", "Return/Exchange", "Expected Delivery", "Availability" and “Product URL”. In case, if any of the details are missing for any of the product then replace it by “-“.

In [ ]:
```
start_page = 0
end_page = 3
```

```
urls = []
for page in range(start_page,end_page+1):
    try:
        page_urls = driver.find_elements_by_xpath('//a[@class="a-link-normal s-no-outline"]')

        # appending all the urls on current page to urls list
        for url in page_urls:
            url = url.get_attribute('href')     # Scraping the url from webelement
            if url[0:4]=='http':                # Checking if the scraped data is a valid url or not
                urls.append(url)                # Appending the url to urls list
        print("Product urls of page {} has been scraped.".format(page+1))

        # Moving to next page
        nxt_button = driver.find_element_by_xpath('//li[@class="a-last"]/a')      # Locating the next_button wh
        if nxt_button.text == 'Next→':                                            # Checking if the button loca
            nxt_button.click()                                                    # Clicking the next button
            time.sleep(5)                                                         # time delay of 5 seconds
        # If the current active button is not next button, the we will check if the next button is inactive or 
        elif driver.find_element_by_xpath('//li[@class="a-disabled a-last"]/a').text == 'Next→':
            print("No new pages exist. Breaking the loop")  # Printing message and breakinf loop if we have rea
            break

    except StaleElementReferenceException as e:             # Handling StaleElement Exception
        print("Stale Exception")
        next_page = nxt_button.get_attribute('href')        # Extracting the url of next page
        driver.get(next_page)
```

In [ ]:

```
prod_dict = {}
prod_dict['Brand']=[]
prod_dict['Name']=[]
prod_dict['Rating']=[]
prod_dict['No. of ratings']=[]
prod_dict['Price']=[]
prod_dict['Return/Exchange']=[]
prod_dict['Expected Delivery']=[]
prod_dict['Availability']=[]
prod_dict['Other Details']=[]
prod_dict['URL']=[]
```

In [ ]:

```
for url in urls[:4]:
    driver.get(url)                                                           # Loading the webpage by url
    print("Scraping URL = ", url)
    #time.sleep(2)

    try:
        brand = driver.find_element_by_xpath('//a[@id="bylineInfo"]')      # Extracting Brand from xpath
        prod_dict['Brand'].append(brand.text)
    except NoSuchElementException:
        prod_dict['Brand'].append('-')

    try:
        name = driver.find_element_by_xpath('//h1[@id="title"]/span')      # Extracting Name from xpath
        prod_dict['Name'].append(name.text)
    except NoSuchElementException:
        prod_dict['Name'].append('-')

    try:
        rating = driver.find_element_by_xpath('//span[@id="acrPopover"]')  # Extracting Ratings from xpath
        prod_dict['Rating'].append(rating.get_attribute("title"))
    except NoSuchElementException:
        prod_dict['Rating'].append('-')

    try:
        n_rating = driver.find_element_by_xpath('//a[@id="acrCustomerReviewLink"]/span')     # Extracting no. o
        prod_dict['No. of ratings'].append(n_rating.text)
    except NoSuchElementException:
        prod_dict['No. of ratings'].append('-')

    try:
        price = driver.find_element_by_xpath('//span[@id="priceblock_ourprice"]')            # Extracting Price
        prod_dict['Price'].append(price.text)
    except NoSuchElementException:
        prod_dict['Price'].append('-')
    try:                                                                                     # Extracting Retur
        ret = driver.find_element_by_xpath('//div[@data-name="RETURNS_POLICY"]/span/div[2]/a')
        prod_dict['Return/Exchange'].append(ret.text)
    except NoSuchElementException:
        prod_dict['Return/Exchange'].append('-')
    try:
        delivry = driver.find_element_by_xpath('//div[@id="ddmDeliveryMessage"]/b')         # Extracting Expect
        prod_dict['Expected Delivery'].append(delivry.text)
    except NoSuchElementException:
        prod_dict['Expected Delivery'].append('-')

    try:
        avl = driver.find_element_by_xpath('//div[@id="availability"]/span')                # Extracting Availa
        prod_dict['Availability'].append(avl.text)
    except NoSuchElementException:
        prod_dict['Availability'].append('-')

    try:                                                                                # Extracting Other 
        dtls = driver.find_element_by_xpath('//ul[@class="a-unordered-list a-vertical a-spacing-mini"]')
        prod_dict['Other Details'].append('  ||  '.join(dtls.text.split('\n')))
    except NoSuchElementException:
        prod_dict['Other Details'].append('-')

    prod_dict['URL'].append(url)                                                        # Saving url
    time.sleep(2)
```

In [ ]:
```
prod_df = pd.DataFrame.from_dict(prod_dict)
prod_df
```

In [ ]:
```
#saving data to csv
prod_df.to_csv('Amazon_{}.csv'.format(inputU))
```

1. Write a python program to access the search bar and search button on images.google.com and scrape 10 images each for keywords ‘fruits’, ‘cars’ and ‘Machine Learning’, ‘Guitar’, ‘Cakes’.

In [ ]:
```
driver.get('https://images.google.com/')
```

In [ ]:
```
search_bar = driver.find_element_by_xpath('//*[@id="sbtc"]/div/div[2]/input')    # Finding the search bar using
search_bar.send_keys("fruits")       # Inputing "banana" keyword to search rock images
search_button = driver.find_element_by_xpath('//*[@id="sbtc"]/button')     # Finding the xpath of search button
search_button.click()         # Clicking the search button
```

In [ ]:
```
print("start scrolling to generate more images on the page...")
# 500 time we scroll down by 10000 in order to generate more images on the website
for _ in range(500):
    driver.execute_script("window.scrollBy(0,10000)")
```

In [ ]:
```
images = driver.find_elements_by_xpath('//img[@class="rg_i Q4LuWd"]')
```

In [ ]:
```
img_urls = []
img_data = []
for image in images:
    source= image.get_attribute('src')
    if source is not None:
        if(source[0:4] == 'http'):
            img_urls.append(source)
len(img_urls)
```

In [ ]:
```
for i in range(len(img_urls)):
    if i >= 100:
        break
    print("Downloading {0} of {1} images" .format(i, 100))
    response= requests.get(img_urls[i])
    file = open("H:/Flip ROBO/banana/img"+str(i)+".jpg", "wb")
    file.write(response.content)
```

1. Write a python program to search for a smartphone(e.g.: Oneplus Nord, pixel 4A, etc.) on www.flipkart.com and scrape following details for all the search results displayed on 1st page. Details to be scraped: “Brand Name”, “Smartphone name”, “Colour”, “RAM”, “Storage(ROM)”, “Primary Camera”, “Secondary Camera”, “Display Size”, “Battery Capacity”, “Price”, “Product URL”. Incase if any of the details is missing then replace it by “- “. Save your results in a dataframe and CSV.

In [ ]:
```
driver = webdriver.Chrome(r"E:\Aniket\chromedriver_win32\chromedriver.exe")
url4="https://www.flipkart.com/search?q=smartphone&otracker=search&otracker1=search&marketplace=FLIPKART&as-sho
driver.get(url4)
```

In [ ]:
```
Brand_Name=[]
Colour=[]
Storage_RAM_ROM=[]
P_F_Camera=[]
Display_size_Resolution=[]
ProcessorAndCores=[]
Battery=[]
Price=[]
Product_URL=[]
```

In [ ]:
```
#scraping the Brand_Name
BName=driver.find_elements_by_xpath("//div[@class='_4rR01T']")
for i in BName:
    if i.text is None :
        Brand_Name.append("--")
    else:
        Brand_Name.append(i.text)
print(len(Brand_Name),Brand_Name)
```

In [ ]:
```
#scraping the Storage_RAM_ROM
```

```python
ram=driver.find_elements_by_xpath("//ul[@class='_1xgFaf']//li[1]")
for i in ram:
    if i.text is None :
        Storage_RAM_ROM.append("--")
    else:
        Storage_RAM_ROM.append(i.text)
print(len(Storage_RAM_ROM),Storage_RAM_ROM)
```

In [ ]:
```python
#scraping the P_F_Camera
PC=driver.find_elements_by_xpath("//ul[@class='_1xgFaf']//li[3]")
for i in PC:
    if i.text is None :
        P_F_Camera.append("--")
    else:
        P_F_Camera.append(i.text)
print(len(P_F_Camera),P_F_Camera)
```

In [ ]:
```python
#scraping the Display_size_Resolution
DS=driver.find_elements_by_xpath("//ul[@class='_1xgFaf']//li[2]")
for i in DS:
    if i.text is None :
        Display_size_Resolution.append("--")
    else:
        Display_size_Resolution.append(i.text)
print(len(Display_size_Resolution),Display_size_Resolution)
```

In [ ]:
```python
#scraping the ProcessorAndCores
P=driver.find_elements_by_xpath("//ul[@class='_1xgFaf']//li[5]")
for i in P:
    if i.text is None :
        ProcessorAndCores.append("--")
    else:
        ProcessorAndCores.append(i.text)
print(len(ProcessorAndCores),ProcessorAndCores)
```

In [ ]:
```python
#scraping the Battery
B=driver.find_elements_by_xpath("//ul[@class='_1xgFaf']//li[4]")
for i in B:
    if i.text is None :
        Battery.append("--")
    else:
        Battery.append(i.text)
print(len(Battery),Battery)
```

In [ ]:
```python
#scraping the Price
price=driver.find_elements_by_xpath("//div[@class='_30jeq3 _1_WHN1']")
for i in price:
    if i.text is None :
        Price.append("--")
    else:
        Price.append(i.text)
print(len(Price),Price)
```

In [ ]:
```python
FlipKart=pd.DataFrame([])
FlipKart['Brand_Name']=Brand_Name
FlipKart['Storage_RAM_ROM']=Storage_RAM_ROM
FlipKart['Amount P_F_Camera']=P_F_Camera
FlipKart['Display_size_Resolution']=Display_size_Resolution
FlipKart['ProcessorAndCores']=ProcessorAndCores
FlipKart['Battery']=Battery
FlipKart['Price']=Price

FlipKart
```

1. Write a program to scrap geospatial coordinates (latitude, longitude) of a city searched on google maps.

In [ ]:
```python
driver = webdriver.Chrome(r"E:\Aniket\chromedriver_win32\chromedriver.exe")
```

In [ ]:
```python
# opening google maps
driver.get("https://www.google.co.in/maps")
time.sleep(3)

city = input('Enter City Name : ')                                          # Enter city to be searched
search = driver.find_element_by_id("searchboxinput")                        # locating search bar
search.clear()                                                              # clearing search bar
time.sleep(2)
search.send_keys(city)                                                      # entering values in search bar
button = driver.find_element_by_id("searchbox-searchbutton")                # locating search button
button.click()                                                              # clicking search button
time.sleep(3)

try:
    url_string = driver.current_url
    print("URL Extracted: ", url_string)
    lat_lng = re.findall(r'@(.*)data',url_string)
    if len(lat_lng):
        lat_lng_list = lat_lng[0].split(",")
        if len(lat_lng_list)>=2:
            lat = lat_lng_list[0]
            lng = lat_lng_list[1]
        print("Latitude = {}, Longitude = {}".format(lat, lng))

except Exception as e:
        print("Error: ", str(e))
```

1. Write a program to scrap all the available details of best gaming laptops from digit.in.

In [ ]:
```
driver = webdriver.Chrome(r"E:\Aniket\chromedriver_win32\chromedriver.exe")
```

In [ ]:
```
url="https://www.digit.in/top-products/best-gaming-laptops-40.html"
```

In [ ]:
```
driver.get(url)
```

In [ ]:
```
Brands=[]
Products_Description=[]
Specification=[]
Price=[]
```

In [ ]:
```
br=driver.find_elements_by_xpath("//div[@class='TopNumbeHeading active sticky-footer']")
len(br)
```

In [ ]:
```
for i in br:

    Brands.append(str(i.text).replace("\n",""))
Brands
```

In [ ]:
```
sp=driver.find_elements_by_xpath("//div[@class='Specs-Wrap']")
len(sp)
```

In [ ]:
```
for i in sp:

    Specification.append(str(i.text).replace("\n",""))
Specification
```

In [ ]:
```
des=driver.find_elements_by_xpath("//div[@class='Section-center']")
len(des)
```

In [ ]:
```
for i in des:

    Products_Description.append(str(i.text).replace("\n",""))
Products_Description
```

In [ ]:
```
pri=driver.find_elements_by_xpath("//td[@class='smprice']")
len(pri)
```

In [ ]:
```
for i in pri:

    Price.append(str(i.text).replace("\n",""))
Price
```

In [ ]:
```
digit_lap=pd.DataFrame([])
digit_lap['Brands']=Brands[0:10]
digit_lap['Price']=Price[0:10]
digit_lap['Specification']=Specification[0:10]
digit_lap['Description']=Products_Description[0:10]
digit_lap
```

1. Write a program to extract at least 500 Comments, Comment upvote and time when comment was posted from any YouTube Video.

In [ ]:
```
driver = webdriver.Chrome('path_to_chromedriver')  # Replace with the path to your WebDriver executable

# Open the YouTube video
video_url = 'https://www.youtube.com/watch?v=your_video_id'  # Replace with the URL of the YouTube video
driver.get(video_url)

# Scroll to load comments
scroll_pause_time = 2  # Adjust the pause time as needed
scrolls = 10  # Adjust the number of scrolls as needed

for _ in range(scrolls):
  driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")
  time.sleep(scroll_pause_time)

# Extract comments, upvotes, and time
```

```python
comments = driver.find_elements_by_xpath('//yt-formatted-string[@id="content-text"]')
upvotes = driver.find_elements_by_xpath('//span[@id="vote-count-middle"]')
times = driver.find_elements_by_xpath('//a[@class="yt-simple-endpoint style-scope yt-formatted-string"]')

# Store the extracted data
extracted_data = []
for comment, upvote, time in zip(comments, upvotes, times):
  extracted_data.append({
  'comment': comment.text,
  'upvote': upvote.text,
  'time': time.text
  })
```