In [1]:

```python
#Importing the library
import pandas as pd
import numpy as np
import requests
import selenium
from selenium import webdriver
from bs4 import BeautifulSoup
#importing the Web driver
driver=webdriver.Chrome(r"C:\Users\lakhi\Downloads\chromedriver_win32 (1)\chromedriver.exe")
import time

# Importing required Exceptions which needs to handled
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
import re
```

Q1. Write a python program which searches all the product under a particular product vertical from www.amazon.in. The product verticals to be searched will be taken as input from user. For e.g. If user input is ‘guitar’. Then search for guitars.

Q2. 2. 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 dataframe and csv. In case if any product vertical has less than 3 pages in search results then scrape all the products available under that product vertical. Details to be scraped are: "Brand Name", "Name of the Product", "Rating", "No. of Ratings", "Price", "Return/Exchange", "Expected Delivery", "Availability", "Other Details" and “Product URL”. In case, if any of the details are missing for any of the product then replace it by “-“.

In [2]:

```python
atr=[]
name=[]
brand=[]
rating=[]=[]
n_rating=[]
price=[]
exchange=[]
expected_del=[]
avlblty=[]
other=[]
def amz():
    q=input(str())
    url='https://www.amazon.in/s?k='+q+'&ref=nb_sb_noss_2'
    driver.get(url)
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    for i in range(0,3):
        for r in list(driver.find_elements_by_xpath("//a[@class='a-link-normal s-no-outline']")):
            atr.append(r.get_attribute('href'))
    driver.find_element_by_xpath('//li[@class="a-last"]/a').click()
    for p in atr:
        driver.get(p)
        try:
            name_tags=driver.find_element_by_xpath("//div[@class='a-section a-spacing-none']/span[2]")
            name.append(name_tags.text)
        except NoSuchElementException:
            name.append("-")
        try:
            barnd_tag=driver.find_element_by_xpath("//div[@class='a-section a-spacing-none']/a")
            brand.append(barnd_tag.text)
        except NoSuchElementException:
            brand.append("-")
        try:
            rat = driver.find_element_by_xpath('//span[@id="acrPopover"]')  # Extracting Ratings from xpath
            rating.append(rat.get_attribute("title"))
        except NoSuchElementException:
            rating.append("-")
        try:
            price_tag=driver.find_element_by_xpath("//span[@class='a-size-medium a-color-price priceBlockSalePriceString']")
            price.append(price_tag.text)
        except NoSuchElementException:
            price.append("-")
        try:
            n_rat = driver.find_element_by_xpath('//a[@id="acrCustomerReviewLink"]/span')
            n_rating.append(n_rat.text)
        except NoSuchElementException:
            n_rating.append("-")
        try:
            exc=driver.find_element_by_xpath("//span[@class='a-declarative']/div[2]")
            exchange.append(exc.text)

        except NoSuchElementException:
            exchange.append("-")
        try:
            exp_del=driver.find_element_by_xpath("//div[@class='a-section a-spacing-mini']/b")
            expected_del.append(exp_del.text)
        except NoSuchElementException:
            expected_del.append("-")
        try:
            avl=driver.find_element_by_xpath("//span[@class='a-size-medium a-color-success']")
            avlblty.append(avl.text)
        except NoSuchElementException:
            avlblty.append("-")

        try:
            dls=driver.find_element_by_xpath("//ul[@class='a-unordered-list a-vertical a-spacing-mini']")
            other.append(dls.text)
        except NoSuchElementException:
            other.append("-")
    Amzn=pd.DataFrame({"Brand":brand,"Name":name,"Price":price,"Rating":rating,
                      "NO_OF_Rating":n_rating,"Exchange & Refund":exchange,
                      "Avibility":avlblty,"Other_details":other,"Product Url":atr})
    Amzn.to_csv("Amazone.csv")
    time.sleep(3)
    return(Amzn.head())
```

In [3]:

```python
amz()
```

```
bat
```

Out[3]:

	Brand	Name	Price	Rating	NO_OF_Rating	Exchange & Refund	Avibility	Other_details	Product Url
0	Brand: LYCAN	Lycan Beast Bigger Edge 2.5 Inch Full Size Har...	-	4.2 out of 5 stars	46 ratings	10 day Refund/Replacement	In stock.	suitable for soft ball cricket u can use (ten...	https://www.amazon.in/Lycan-Beast-Bigger-Plast...
1	Brand: Generic		-	2.3 out of 5 stars	7 ratings	10 day Refund/Replacement	In stock.	-	https://www.amazon.in/Balaji-Traders-Popular-W...
2	Brand: Generic		-	2.5 out of 5 stars	2 ratings	10 day Refund/Replacement	In stock.	-	https://www.amazon.in/Balaji-Traders-Nicolls-P...
3	Brand: GM	GM Six6 F2 909 English Willow Cricket Bat Shor...	-	5.0 out of 5 stars	3 ratings	10 day Refund/Replacement	In stock.	Size: short handle\nWillow: English Willow\nCo...	https://www.amazon.in/GM-English-Willow-Cricke...
4	Brand: SG	-	-	4.0 out of 5 stars	51 ratings	10 day Refund/Replacement	-	In Box contents: 1 Cricket Bat with Cover | Ma...	https://www.amazon.in/SG-Sierra-Kashmir-Willow...

Q3. Write a python program to access the search bar and search button on images.google.com and scrape 100 images each for keywords ‘fruits’, ‘cars’ and ‘Machine Learning’.

In [4]:

```python
driver.get('https://images.google.com/')
search_bar = driver.find_element_by_xpath('//*[@id="sbtc"]/div/div[2]/input')
search_bar.send_keys(input("Searching for→ ",))
search_button = driver.find_element_by_xpath('//*[@id="sbtc"]/button')
search_button.click()
time.sleep(10)
for _ in range(500):
    driver.execute_script("window.scrollBy(0,100000)")
images = driver.find_elements_by_xpath('//img[@class="rg_i Q4LuWd"]')
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)
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(r"E:\abc\Google"+str(i)+".jpg", "wb")
    file.write(response.content)
```

```
Searching for→ Fruit
Downloading 0 of 100 images
Downloading 1 of 100 images
Downloading 2 of 100 images
Downloading 3 of 100 images
Downloading 4 of 100 images
Downloading 5 of 100 images
Downloading 6 of 100 images
Downloading 7 of 100 images
Downloading 8 of 100 images
Downloading 9 of 100 images
Downloading 10 of 100 images
Downloading 11 of 100 images
Downloading 12 of 100 images
Downloading 13 of 100 images
Downloading 14 of 100 images
Downloading 15 of 100 images
Downloading 16 of 100 images
Downloading 17 of 100 images
Downloading 18 of 100 images
Downloading 19 of 100 images
Downloading 20 of 100 images
Downloading 21 of 100 images
Downloading 22 of 100 images
Downloading 23 of 100 images
Downloading 24 of 100 images
Downloading 25 of 100 images
Downloading 26 of 100 images
Downloading 27 of 100 images
Downloading 28 of 100 images
Downloading 29 of 100 images
Downloading 30 of 100 images
Downloading 31 of 100 images
Downloading 32 of 100 images
Downloading 33 of 100 images
Downloading 34 of 100 images
Downloading 35 of 100 images
Downloading 36 of 100 images
Downloading 37 of 100 images
Downloading 38 of 100 images
Downloading 39 of 100 images
Downloading 40 of 100 images
Downloading 41 of 100 images
Downloading 42 of 100 images
Downloading 43 of 100 images
Downloading 44 of 100 images
Downloading 45 of 100 images
Downloading 46 of 100 images
Downloading 47 of 100 images
Downloading 48 of 100 images
Downloading 49 of 100 images
Downloading 50 of 100 images
Downloading 51 of 100 images
Downloading 52 of 100 images
Downloading 53 of 100 images
Downloading 54 of 100 images
Downloading 55 of 100 images
Downloading 56 of 100 images
Downloading 57 of 100 images
Downloading 58 of 100 images
Downloading 59 of 100 images
Downloading 60 of 100 images
Downloading 61 of 100 images
Downloading 62 of 100 images
Downloading 63 of 100 images
Downloading 64 of 100 images
Downloading 65 of 100 images
Downloading 66 of 100 images
Downloading 67 of 100 images
Downloading 68 of 100 images
Downloading 69 of 100 images
Downloading 70 of 100 images
Downloading 71 of 100 images
Downloading 72 of 100 images
Downloading 73 of 100 images
Downloading 74 of 100 images
Downloading 75 of 100 images
Downloading 76 of 100 images
Downloading 77 of 100 images
Downloading 78 of 100 images
Downloading 79 of 100 images
Downloading 80 of 100 images
Downloading 81 of 100 images
Downloading 82 of 100 images
Downloading 83 of 100 images
Downloading 84 of 100 images
Downloading 85 of 100 images
Downloading 86 of 100 images
Downloading 87 of 100 images
Downloading 88 of 100 images
Downloading 89 of 100 images
Downloading 90 of 100 images
Downloading 91 of 100 images
Downloading 92 of 100 images
Downloading 93 of 100 images
Downloading 94 of 100 images
Downloading 95 of 100 images
Downloading 96 of 100 images
Downloading 97 of 100 images
Downloading 98 of 100 images
Downloading 99 of 100 images
```

Q4. 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”, “Display Resolution”, “Processor”, “Processor Cores”, “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 [5]:

```python
link=[]
brand=[]
Smart_phone=[]
color=[]
ram=[]
storage=[]
primary_camera=[]
secondary_camare=[]
display_size=[]
Display_Resolution=[]
Processor=[]
Processor_Cores=[]
Battery_Capacity=[]
Battery_Type=[]
Price=[]
def flip():
    q=input("Please enter Brand name   ")#brand
    q1=input("please enter  model name  ")#model
    url="https://www.flipkart.com/search?q="+q+"%20"+q1+"&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off"
    driver.get(url)
    for i in list(driver.find_elements_by_xpath("//a[@class='_1fQZEK']")):
        link.append(i.get_attribute('href'))
    for l in link:
        driver.get(l)
        try:
            read_more = driver.find_element_by_xpath('//button[@class="_2KpZ6l _1FH0tX"]')
            read_more.click()
        except NoSuchElementException:
            print("Exception Occured. Moving to next page")
        try:
            brnd = driver.find_element_by_xpath('//span[@class="B_NuCI"]')
            brand.append(brnd.text.split()[0])
        except NoSuchElementException:
            brand.append('-')
        try:
            prc = driver.find_element_by_xpath('//div[@class="_30jeq3 _16Jk6d"]')
            Price.append(prc.text)
        except NoSuchElementException:
            Price.append('-')

        try:
            name = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][1]/table/tbody/tr[3]/td[2]/ul/li')
            Smart_phone.append(name.text)
        except NoSuchElementException:
            Smart_phone.append('-')

        try:
            clr = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][1]/table/tbody/tr[4]/td[2]/ul/li')
            color.append(clr.text)
        except NoSuchElementException:
            color.append('-')

        try:
            disp_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][2]/div')
            if disp_chk.text != "Display Features" : raise NoSuchElementException
            disp_size = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][2]/table[1]/tbody/tr[1]/td[2]/ul/li')
            display_size.append(disp_size.text)
        except NoSuchElementException:
            display_size.append('-')

        try:
            disp_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][2]/div')
            if disp_chk.text != "Display Features" : raise NoSuchElementException
            disp_res = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][2]/table[1]/tbody/tr[2]/td[2]/ul/li')
            Display_Resolution.append(disp_res.text)
        except NoSuchElementException:
            Display_Resolution.append('-')

        try:
            pro_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[2]/td[1]')
            if pro_chk.text != "Processor Type" : raise NoSuchElementException
            prsr = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[2]/td[2]/ul/li')
            Processor.append(prsr.text)
        except NoSuchElementException:
            Processor.append('-')

        try:
            core_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[3]/td[1]')
            if core_chk.text != "Processor Core" :
                core_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[2]/td[1]')
                if core_chk.text != "Processor Core" :
                    raise NoSuchElementException
                else :
                    cores = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[2]/td[2]/ul/li')
            else :
                cores = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][3]/table[1]/tbody/tr[3]/td[2]/ul/li')
                Processor_Cores.append(cores.text)
        except NoSuchElementException:
            Processor_Cores.append('-')

        try:
            rom = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][4]/table[1]/tbody/tr[1]/td[2]/ul/li')
            storage.append(rom.text)
        except NoSuchElementException:
            storage.append('-')

        try:
            rm = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][4]/table[1]/tbody/tr[2]/td[2]/ul/li')
            ram.append(rm.text)
        except NoSuchElementException:
            ram.append('-')

        try:
            pri_cam = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][5]/table[1]/tbody/tr[2]/td[2]/ul/li')
            primary_camera.append(pri_cam.text)
        except NoSuchElementException:
            primary_camera.append('-')

        try:
            cam_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][5]/table[1]/tbody/tr[6]/td[1]')
            if cam_chk != "Secondary Camera" :
                if driver.find_element_by_xpath('//div[@class="_3k-BhJ"][5]/table[1]/tbody/tr[5]/td[1]').text == "Secondary Camera":
                    sec_cam = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][5]/table[1]/tbody/tr[5]/td[2]/ul/li')
                else :
                    raise NoSuchElementException
            else :
                sec_cam = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][5]/table[1]/tbody/tr[6]/td[2]/ul/li')
                secondary_camare.append(sec_cam.text)
        except NoSuchElementException:
            secondary_camare.append('-')

        try:
            if driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/div').text != "Battery & Power Features" :
                if driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/div').text == "Battery & Power Features" :
                    bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/table/tbody/tr/td[1]')
                    if bat_chk.text != "Battery Capacity" : raise NoSuchElementException
                    bat_cap = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/table/tbody/tr/td[2]/ul/li')
                elif driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/div').text == "Battery & Power Features" :
                    bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/table/tbody/tr/td[1]')
                    if bat_chk.text != "Battery Capacity" : raise NoSuchElementException
                    bat_cap = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/table/tbody/tr/td[2]/ul/li')
                else:
                    raise NoSuchElementException
            else :
                bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/table/tbody/tr/td[1]')
                if bat_chk.text != "Battery Capacity" : raise NoSuchElementException
                bat_cap = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/table/tbody/tr/td[2]/ul/li')
            Battery_Capacity.append(bat_cap.text)
        except NoSuchElementException:
            Battery_Capacity.append('-')

        try:
            if driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/div').text != "Battery & Power Features" :
                if driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/div').text == "Battery & Power Features" :
                    bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/table/tbody/tr[2]/td[1]')
                    if bat_chk.text != "Battery Type" : raise NoSuchElementException
                    bat_typ = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][9]/table/tbody/tr[2]/td[2]/ul/li')
                elif driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/div').text == "Battery & Power Features" :
                    bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/table/tbody/tr[2]/td[1]')
                    if bat_chk.text != "Battery Type" : raise NoSuchElementException
                    bat_typ = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][8]/table/tbody/tr[2]/td[2]/ul/li')
                else:
                    raise NoSuchElementException
            else :
                bat_chk = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/table/tbody/tr[2]/td[1]')
                if bat_chk.text != "Battery Type" : raise NoSuchElementException
                bat_typ = driver.find_element_by_xpath('//div[@class="_3k-BhJ"][10]/table/tbody/tr[2]/td[2]/ul/li')
        except NoSuchElementException:
                Battery_Type.append('-')
    flipkart=pd.DataFrame({"Brand_Name":brand,"Smart_phone_name":Smart_phone,"Colour":color,"RAM":ram,
                        "Storage(ROM)":storage,"Primary_camera":primary_camera,
                        "Display_size":display_size,
                       "Display_Resolution":Display_Resolution,
                        "Processor":Processor,
                        "Battery_Capacity":Battery_Capacity,"URL":link,
                         "Price":Price})
    flipkart.to_csv("Flipkart.csv")

    time.sleep(5)
    return(flipkart.head(10))
```

In [6]:

```python
flip()
```

```
Please enter Brand name   mi
please enter  model name  9
```

Out[6]:

	Brand_Name	Smart_phone_name	Colour	RAM	Storage(ROM)	Primary_camera	Display_size	Display_Resolution	Processor	Battery_Capacity	URL	Price
0	Redmi	Redmi 9 Prime	Mint Green	4 GB	64 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-mint-gr...	₹9,499
1	Redmi	Redmi 9 Prime	Sunrise Flare	4 GB	64 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-sunrise...	₹9,499
2	Redmi	Redmi 9 Prime	Matte Black	4 GB	64 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-matte-b...	₹9,499
3	Redmi	Redmi 9 Prime	Mint Green	4 GB	128 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-mint-gr...	₹10,999
4	Redmi	Redmi 9 Prime	Sunrise Flare	4 GB	128 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-sunrise...	₹10,999
5	Redmi	Redmi 9 Prime	Space Blue	4 GB	128 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-space-b...	₹10,999
6	Redmi	Redmi 9 Prime	Matte Black	4 GB	128 GB	13MP Rear Camera	16.59 cm (6.53 inch)	2340 x 1080 Pixels	MediaTek Helio G80	5020 mAh	https://www.flipkart.com/redmi-9-prime-matte-b...	₹11,799
7	Redmi	Redmi 8A	Midnight Black	3 GB	32 GB	12MP Rear Camera	15.8 cm (6.22 inch)	1520 x 720 Pixels	Qualcomm Snapdragon 439	5000 mAh	https://www.flipkart.com/redmi-8a-midnight-bla...	₹8,398
8	Redmi	Redmi 8A Dual	Sky White	2 GB	32 GB	13MP + 2MP	15.8 cm (6.22 inch)	1520 x 720 Pixels	Qualcomm Snapdragon 439	5000 mAh	https://www.flipkart.com/redmi-8a-dual-sky-whi...	₹7,959
9	Redmi	Redmi 9	Sky Blue	4 GB	128 GB	13MP + 8MP	16.59 cm (6.53 inch)	720 x 1600$$pixel	MediaTek Helio G35	5000 mAh	https://www.flipkart.com/redmi-9-sky-blue-128-...	₹10,199

Q5 Write a program to scrap geospatial coordinates (latitude, longitude) of a city searched on google maps.

In [7]:

```python
driver.get("https://www.google.co.in/maps")
time.sleep(5)

city = input('Please Enter City Name : ')
search = driver.find_element_by_id("searchboxinput")
search.clear()
time.sleep(3)
search.send_keys(city)
button = driver.find_element_by_id("searchbox-searchbutton")
button.click()
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))
```

```
Please Enter City Name : noida
URL Extracted:  https://www.google.co.in/maps/place/Noida,+Uttar+Pradesh/@13.0054777,77.6854536,15z/data=!4m5!3m4!1s0x390ce5a43173357b:0x37ffce30c87cc03f!8m2!3d28.5355161!4d77.3910265
Latitude = 13.0054777, Longitude = 77.6854536
```

In [ ]:

In [ ]: