In [2]:
```
from selenium import webdriver
import time
import pandas as pd
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
```

1. Write a python program which searches all the product under a particular product from www.amazon.in (http://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 [94]:
```
driver=webdriver.Chrome()
```

In [95]:
```
driver.get("https://www.amazon.in/")
```

In [96]:
```
search_bar=driver.find_element(By.ID,"twotabsearchtextbox")
```

In [97]:
```
print("what would you like to search today?")
search_for=input()
```

```
what would you like to search today?
Laptop
```

In [98]:
```
search_bar.clear()
search_bar.send_keys(search_for)
```

In [99]:
```
search_button=driver.find_element(By.ID,"nav-search-submit-button")
search_button.click()
```

In [ ]:
```

```

3. 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 [101]:
```
driver=webdriver.Chrome()
```

In [102]:
```
driver.get("https://images.google.com")
```

In [107]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[1]/div[3]/form/div|
search_bar.send_keys("Fruits")
```

In [108]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[1]/div[3]/form/
search_button.click()
```

In [120]:
```
images=[]
```

In [121]:
```
images=driver.find_elements(By.TAG_NAME,"img")
```

In [122]:
```
images_tags=driver.find_elements(By.XPATH,"/html/body/div[4]/div/div[15]/di
for i in images_tags:
    title=i.text
    images.append(title)
```

In [125]:
```
len(images)
```

Out[125]:
```
251
```

In [126]:
```
images
```

Out[126]:
```
[<selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.980")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.1389")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.1391")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.1392")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.1461")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.164B3C08A733BEA0A40906F2F5F6AC0C.e.1463")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
```

In [131]:
```
search_bar.clear()
```

In [132]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/div
search_bar.send_keys("Cars")
```

In [134]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/
search_button.click()
```

In [135]:
```
cars=[]
```

In [137]:
```
cars=driver.find_elements(By.TAG_NAME,"img")
```

In [138]:
```
cars_tags=driver.find_elements(By.XPATH,"/html/body/div[4]/div/div[15]/div/
for i in cars_tags:
    title=i.text
    cars.append(title)
```

In [139]:
```
len(cars)
```

Out[139]:
```
249
```

In [140]:
```
cars
```

Out[140]:
```
[<selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.5566")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.6173")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.6175")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.6176")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.6245")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.1D74147685BFE97C073B9F5E62F39261.e.6247")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
```

In [145]:
```
search_bar.clear()
```

In [146]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/div[
search_bar.send_keys("Machine Learning")
```

In [147]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/d
search_button.click()
```

In [148]:
```
machine_learning=[]
```

In [149]:
```
machine_learning=driver.find_elements(By.TAG_NAME,"img")
```

In [150]:
```
machine_learning_tags=driver.find_elements(By.XPATH,"/html/body/div[4]/div/
for i in machine_learning_tags:
    title=i.text
    machine_learning.append(title)
```

In [151]:
```
len(machine_learning)
```

Out[151]:
```
254
```

In [152]:
```
machine_learning
```

Out[152]:
```
[<selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.9890")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.10035")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.10037")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.10038")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.10107")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.7E5D4196BA666975A858F86D16B7CC97.e.10109")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
```

In [156]:
```
search_bar.clear()
```

In [157]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/div[
search_bar.send_keys("guitar")
```

In [158]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/d
search_button.click()
```

In [159]:
```
guitar=[]
```

In [160]:
```
guitar=driver.find_elements(By.TAG_NAME,"img")
```

In [161]:
```
guitar_tags=driver.find_elements(By.XPATH,"/html/body/div[4]/div/div[15]/di
for i in guitar_tags:
    title=i.text
    guitar.append(title)
```

In [162]:
```
len(guitar)
```

Out[162]:
```
252
```

In [163]:
```
guitar
```

Out[163]:
```
[<selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14280")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14883")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14885")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14886")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14955")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.845F8225D34486ADDEF44B57428F6CF7.e.14957")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
```

In [168]:
```
search_bar.clear()
```

In [169]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/div[
search_bar.send_keys("Cakes")
```

In [170]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[2]/div[2]/form/d
search_button.click()
```

In [171]:
```
cakes=[]
```

In [172]:
```
cakes=driver.find_elements(By.TAG_NAME,"img")
```

In [173]:
```
cakes_tags=driver.find_elements(By.XPATH,"/html/body/div[4]/div/div[15]/div
for i in cakes_tags:
    title=i.text
    cakes.append(title)
```

In [174]:
```
len(cakes)
```

Out[174]:
```
254
```

In [175]:
```
cakes
```

Out[175]:
```
[<selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.18906")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.19547")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.19549")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.19550")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.19619")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
2a8c79b4463214cef611bd", element="f.06DFAA9B9FEF07F095E728D4A35FE2E5.
d.EB42F4CC5B626FF7D323473657693D4B.e.19621")>,
 <selenium.webdriver.remote.webelement.WebElement (session="123fc61beb
```

In [ ]:
```

```

4. Write a python program to search for a smartphone(e.g.: Oneplus Nord, pixel 4A, etc.) on www.flipkart.com (http://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 [134]:
```
driver=webdriver.Chrome()
```

In [135]:
```
driver.get("https://www.flipkart.com/")
```

In [142]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div/div/div[1]/div[1]/d
search_bar.send_keys("smartphone")
```

In [143]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div/div/div[1]/div[1
search_button.click()
```

In [144]:
```
brand_name=[]
```

In [162]:
```
brand_tags=driver.find_elements(By.CLASS_NAME,"KzDlHZ")
for i in brand_tags:
    title=i.text
    brand_name.append(title)
```

In [163]:
```
brand_name
```

Out[163]:
```
['Motorola G34 5G (Ocean Green, 128 GB)',
 'Motorola G34 5G (Ocean Green, 128 GB)\n4.21,00,698 Ratings & 7,042 Revi
ews\n8 GB RAM | 128 GB ROM\n16.51 cm (6.5 inch) HD+ Display\n50MP + 2MP |
16MP Front Camera\n5000 mAh Battery\nSnapdragon 695 5G Processor\nVegan L
eather Design\n1 Year on Handset and 6 Months on Accessories',
 'Motorola G34 5G (Ocean Green, 128 GB)',
 'Motorola G34 5G (Ocean Green, 128 GB)',
 'Motorola g64 5G (Mint Green, 128 GB)',
 'POCO M6 Pro 5G (Power Black, 128 GB)',
 'Motorola g64 5G (Pearl Blue, 128 GB)',
 'Motorola G34 5G (Charcoal Black, 128 GB)',
 'Motorola G34 5G (Ice Blue, 128 GB)',
 'Motorola g04s (Satin Blue, 64 GB)',
 'Motorola g64 5G (Ice Lilac, 128 GB)',
 'POCO C65 (Pastel Green, 128 GB)',
 'Motorola g64 5G (Ice Lilac, 256 GB)',
 'REDMI 13C (Starshine Green, 128 GB)',
 'Motorola Edge 50 Fusion (Hot Pink, 256 GB)',
 'POCO C65 (Matte Black, 128 GB)',
 'POCO C65 (Pastel Blue, 128 GB)',
 'POCO C65 (Pastel Blue, 128 GB)',
 'Motorola Edge 50 Fusion (Marshmallow Blue, 256 GB)',
 'OnePlus Nord CE 3 Lite 5G (Chromatic Gray, 256 GB)',
 'OnePlus Nord CE 3 Lite 5G (Chromatic Gray, 128 GB)',
 'REDMI 13C (Stardust Black, 128 GB)',
 'REDMI 13C (Starfrost White, 128 GB)',
 'OnePlus Nord CE 3 Lite 5G (Pastel Lime, 256 GB)',
 'Motorola Edge 50 Fusion (Marshmallow Blue, 128 GB)',
 'POCO M6 Pro 5G (Forest Green, 128 GB)',
 'POCO C61 (Diamond Dust Black, 64 GB)']
```

In [ ]:
```
```

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

In [27]:
```
driver=webdriver.Chrome()
```

In [28]:
```
driver.get("https://www.google.com/maps/")
```

In [30]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[1]/div[3]/div[8]/di
```

In [31]:
```
print("Which city would like to search?")
city=input()
```

```
Which city would like to search?
Indore
```

In [32]:
```
search_bar.send_keys(city)
```

In [33]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[1]/div[3]/div[8]
search_button.click()
```

In [34]:
```
url=driver.find_element(By.XPATH,"/html/body/div[1]/div[3]/div[8]/div[10]/d
```

In [35]:
```
a=url.get_attribute("href")
a
```

Out[35]:
```
'https://accounts.google.com/ServiceLogin?hl=en&passive=true&continue=htt
ps%3A%2F%2Fwww.google.com%2Fmaps%2Fplace%2FIndore%2C%2BMadhya%2BPradesh%2
F%4022.7239728%2C75.8638499%2C11z%2Fdata%3D!3m1!4b1!4m6!3m5!1s0x3962fcad1
b410ddb%3A0x96ec4da356240f4!8m2!3d22.7195687!4d75.8577258!16zL20vMDFfeXZ
5%3Fentry%3Dttu&service=local&ec=GAZAcQ'
```

In [36]:
```
latitude=a.replace("//","").split("2F%40")[1].split("%2C")[0]
```

In [37]:
```
latitude
```

Out[37]:
```
'22.7239728'
```

In [38]:
```
longitude=a.replace("//","").split("%2C")[2]
```

In [39]:
```
longitude
```

Out[39]:
```
'75.8638499'
```

In [48]:
```
df=pd.DataFrame({})
df["City"]=[city]
df["Latitude"]=[latitude]
df["Longitude"]=[longitude]
```

In [49]:
```
df
```

Out[49]:

	City	Latitude	Longitude
0	Indore	22.7239728	75.8638499

In [ ]:
```

```

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

In [93]:
```
driver=webdriver.Chrome()
```

In [94]:
```
driver.get("https://www.digit.in")
```

In [96]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/div[5]/div/form/input[1
search_bar.send_keys("Best Gaming Laptops")
```

In [101]:
```
search_button=driver.find_element(By.CLASS_NAME,"icon-search-onclick")
search_button.click()
```

In [ ]:
```

```

7. Write a python program to scrape the details for all billionaires from www.forbes.com (http://www.forbes.com). Details to be scrapped: “Rank”, “Name”, “Net worth”, “Age”, “Citizenship”, “Source”, “Industry”.

In [35]:
```
driver=webdriver.Chrome()
```

In [36]:
```
driver.get("https://www.forbes.com")
```

In [38]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/main/form/div/div/input
search_bar.send_keys("all billionaries")
```

In [39]:
```
search_button=driver.find_element(By.XPATH,"/html/body/main/form/div/div/di
search_button.click()
```

In [34]:
```
import pandas as pd
```

In [41]:
```
names=[]
```

In [42]:
```
name_tags=driver.find_elements(By.CLASS_NAME,"Table_personName__Bus2E")
for i in name_tags:
    title=i.text
    names.append(title)
```

In [43]:
```
len(names)
```

Out[43]: 201

In [44]:
```
names
```

Out[44]:
```
['NAME',
 'Bernard Arnault & family',
 'Elon Musk',
 'Jeff Bezos',
 'Mark Zuckerberg',
 'Larry Ellison',
 'Warren Buffett',
 'Bill Gates',
 'Steve Ballmer',
 'Mukesh Ambani',
 'Larry Page',
 'Sergey Brin',
 'Michael Bloomberg',
 'Amancio Ortega',
 'Carlos Slim Helu & family',
 'Francoise Bettencourt Meyers & family',
 'Michael Dell',
 'Gautam Adani',
 'Jim Walton & family',
```

In [45]:
```
rank=[]
```

In [48]:
```
rank_tags=driver.find_elements(By.CLASS_NAME,"Table_rank__X4MKf")
for i in rank_tags:
    title=i.text
    rank.append(title)
```

In [49]:
```
len(rank)
```

Out[49]:
```
401
```

In [50]:
```
net_worth=[]
```

In [51]:
```
net_worth_tags=driver.find_elements(By.CLASS_NAME,"Table_finalWorth__UZA6k"
for i in net_worth_tags:
    title=i.text
    net_worth.append(title)
```

In [53]:
```
len(net_worth)
```

Out[53]:
```
201
```

In [54]:
```
net_worth
```

Out[54]:
```
['NET WORTH',
 '$233 B',
 '$195 B',
 '$194 B',
 '$177 B',
 '$141 B',
 '$133 B',
 '$128 B',
 '$121 B',
 '$116 B',
 '$114 B',
 '$110 B',
 '$106 B',
 '$103 B',
 '$102 B',
 '$99.5 B',
 '$91 B',
 '$84 B',
 '$78.4 B',
```

In [72]:
```
detail=[]
```

In [73]:
```
detail_tags=driver.find_elements(By.CLASS_NAME,"Table_tableRow__lF_cY")
for i in detail_tags:
    title=i.text
    detail.append(title)
```

In [74]:
```
len(detail)
```

Out[74]:
```
200
```

In [75]:
```
detail[:100]
```

Out[75]:
```
['1.\nBernard Arnault & family\n$233 B\nFashion & Retail',
 '2.\nElon Musk\n$195 B\nAutomotive',
 '3.\nJeff Bezos\n$194 B\nTechnology',
 '4.\nMark Zuckerberg\n$177 B\nTechnology',
 '5.\nLarry Ellison\n$141 B\nTechnology',
 '6.\nWarren Buffett\n$133 B\nFinance & Investments',
 '7.\nBill Gates\n$128 B\nTechnology',
 '8.\nSteve Ballmer\n$121 B\nTechnology',
 '9.\nMukesh Ambani\n$116 B\nDiversified',
 '10.\nLarry Page\n$114 B\nTechnology',
 '11.\nSergey Brin\n$110 B\nTechnology',
 '12.\nMichael Bloomberg\n$106 B\nFinance & Investments',
 '13.\nAmancio Ortega\n$103 B\nFashion & Retail',
 '14.\nCarlos Slim Helu & family\n$102 B\nTelecom',
 '15.\nFrancoise Bettencourt Meyers & family\n$99.5 B\nFashion & Retail',
 '16.\nMichael Dell\n$91 B\nTechnology',
 '17.\nGautam Adani\n$84 B\nDiversified',
 '18.\nJim Walton & family\n$78.4 B\nFashion & Retail',
 '19.\nRob Walton & family\n$77.4 B\nFashion & Retail',
 '20.\nJensen Huang\n$77 B\nTechnology',
 '21.\nAlice Walton\n$72.3 B\nFashion & Retail',
 '22.\nDavid Thomson & family\n$67.8 B\nMedia & Entertainment',
 '23.\nJulia Koch & family\n$64.3 B\nDiversified',
 '24.\nZhong Shanshan\n$62.3 B\nFood & Beverage',
 '25.\nCharles Koch & family\n$58.5 B\nDiversified',
 '26.\nGiovanni Ferrero\n$43.8 B\nFood & Beverage',
 '27.\nPrajogo Pangestu\n$43.4 B\nDiversified',
 '27.\nZhang Yiming\n$43.4 B\nTechnology',
 '29.\nTadashi Yanai & family\n$42.8 B\nFashion & Retail',
 '30.\nPhil Knight & family\n$40.9 B\nFashion & Retail',
 '31.\nMark Mateschitz\n$39.6 B\nFood & Beverage',
 '32.\nKlaus-Michael Kuehne\n$39.2 B\nLogistics',
 '33.\nColin Huang\n$38.9 B\nTechnology',
 '34.\nStephen Schwarzman\n$38.8 B\nFinance & Investments',
 '35.\nJacqueline Mars\n$38.5 B\nFood & Beverage',
 '35.\nJohn Mars\n$38.5 B\nFood & Beverage',
 '37.\nDieter Schwarz\n$38 B\nFashion & Retail',
 '38.\nLi Ka-shing\n$37.3 B\nDiversified',
 '39.\nShiv Nadar\n$36.9 B\nTechnology',
 '40.\nAlain Wertheimer\n$36.8 B\nFashion & Retail',
 '40.\nGerard Wertheimer\n$36.8 B\nFashion & Retail',
 '42.\nKen Griffin\n$36.4 B\nFinance & Investments',
 '43.\nMacKenzie Scott\n$35.6 B\nTechnology',
 '44.\nThomas Peterffy\n$34 B\nFinance & Investments',
 '45.\nReinhold Wuerth & family\n$33.6 B\nManufacturing',
 '46.\nWilliam Ding\n$33.5 B\nTechnology',
 '46.\nSavitri Jindal & family\n$33.5 B\nMetals & Mining',
 '48.\nGianluigi Aponte\n$33.1 B\nLogistics',
 '48.\nRafaela Aponte-Diamant\n$33.1 B\nLogistics',
 '50.\nChangpeng Zhao\n$33 B\nFinance & Investments',
 '51.\nMasayoshi Son\n$32.7 B\nFinance & Investments',
 '52.\nLen Blavatnik\n$32.1 B\nDiversified',
 '53.\nMiriam Adelson & family\n$32 B\nGambling & Casinos',
 '54.\nFrançois Pinault & family\n$31.6 B\nFashion & Retail',
 '55.\nJim Simons\n$31.4 B\nFinance & Investments',
 '56.\nGina Rinehart\n$30.8 B\nMetals & Mining',
 '57.\nMa Huateng\n$30.2 B\nTechnology',
 '58.\nAbigail Johnson\n$29 B\nFinance & Investments',
 '59.\nVagit Alekperov\n$28.6 B\nEnergy',
 '60.\nEduardo Saverin\n$28 B\nTechnology',
 '60.\nLukas Walton\n$28 B\nFashion & Retail',
```

```
'62.\nGermán Larrea Mota Velasco & family\n$27.9 B\nMetals & Mining',
'63.\nLee Shau Kee\n$27.7 B\nReal Estate',
'64.\nJeff Yass\n$27.6 B\nFinance & Investments',
'65.\nAndrea Pignataro\n$27.5 B\nFinance & Investments',
'66.\nLow Tuck Kwong\n$27.4 B\nEnergy',
'66.\nLeonid Mikhelson & family\n$27.4 B\nEnergy',
'68.\nStefan Quandt\n$27.3 B\nAutomotive',
'69.\nDilip Shanghvi\n$26.7 B\nHealthcare',
'70.\nVladimir Lisin\n$26.6 B\nMetals & Mining',
'71.\nR. Budi Hartono\n$26.5 B\nFinance & Investments',
'71.\nSusanne Klatten\n$26.5 B\nAutomotive',
'73.\nThomas Frist, Jr. & family\n$26.2 B\nHealthcare',
'73.\nDaniel Gilbert\n$26.2 B\nFinance & Investments',
'75.\nIris Fontbona & family\n$25.7 B\nMetals & Mining',
'76.\nEmmanuel Besnier\n$25.5 B\nFood & Beverage',
'76.\nMichael Hartono\n$25.5 B\nManufacturing',
'76.\nAlexey Mordashov & family\n$25.5 B\nMetals & Mining',
'79.\nJohn Menard, Jr.\n$25.2 B\nFashion & Retail',
'80.\nHe Xiangjian & family\n$25.1 B\nManufacturing',
'81.\nJack Ma\n$24.5 B\nTechnology',
'81.\nElaine Marshall & family\n$24.5 B\nDiversified',
'83.\nAutry Stephens\n$24.3 B\nEnergy',
'84.\nEyal Ofer\n$24 B\nDiversified',
'85.\nVladimir Potanin\n$23.7 B\nMetals & Mining',
'86.\nGennady Timchenko\n$23.4 B\nEnergy',
'87.\nTakemitsu Takizaki\n$23.1 B\nManufacturing',
'88.\nVinod Adani\n$23 B\nDiversified',
'89.\nRobin Zeng\n$22.9 B\nAutomotive',
'90.\nCyrus Poonawalla\n$21.3 B\nHealthcare',
'91.\nAndrey Melnichenko & family\n$21.1 B\nMetals & Mining',
'92.\nDiane Hendricks\n$20.9 B\nConstruction & Engineering',
'92.\nKushal Pal Singh\n$20.9 B\nReal Estate',
'94.\nVicky Safra & family\n$20.6 B\nFinance & Investments',
'94.\nEric Schmidt\n$20.6 B\nTechnology',
'94.\nDavid Tepper\n$20.6 B\nFinance & Investments',
'97.\nSteve Cohen\n$19.8 B\nFinance & Investments',
'98.\nKumar Birla\n$19.7 B\nDiversified',
'99.\nRick Cohen & family\n$19.6 B\nTechnology',
'100.\nRupert Murdoch & family\n$19.5 B\nMedia & Entertainment']
```

In [78]:
```
df=pd.DataFrame({'Rank':rank[:100],'Name':names[:100],'Net Worth':net_worth
```

In [79]:
```
df
```

Out[79]:

	Rank	Name	Net Worth
0	RANK	NAME	NET WORTH
1	1.	Bernard Arnault & family	$233 B
2	1.	Elon Musk	$195 B
3	2.	Jeff Bezos	$194 B
4	2.	Mark Zuckerberg	$177 B
...	...	...	...
95	48.	Eric Schmidt	$20.6 B
96	48.	David Tepper	$20.6 B
97	48.	Steve Cohen	$19.8 B
98	48.	Kumar Birla	$19.7 B
99	50.	Rick Cohen & family	$19.6 B

100 rows × 3 columns

In [ ]:
```
```

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

In [62]:
```
driver=webdriver.Chrome()
```

In [63]:
```
driver.get("https://www.youtube.com")
```

In [64]:
```
search_bar=driver.find_element(By.XPATH,"/html/body/ytd-app/div[1]/div/ytd-
search_bar.send_keys("Data Science")
```

In [67]:
```
search_button=driver.find_element(By.XPATH,"/html/body/ytd-app/div[1]/div/y
search_button.click()
```

In [61]:
```
video=driver.find_elements(By.XPATH,"/html/body/ytd-app/div[1]/ytd-page-mar
```

In [ ]:
```
```

9. Write a python program to scrape a data for all available Hostels from https://www.hostelworld.com/ (https://www.hostelworld.com/) in “London” location. You have to scrape hostel name, distance from city centre, ratings, total reviews, overall reviews, privates from price, dorms from price, facilities and property description.

In [3]:
```
driver=webdriver.Chrome()
```

In [4]:
```
driver.get("https://www.hostelworld.com/")
```

In [9]:
```
location=driver.find_element(By.XPATH,"/html/body/div[3]/div/div[2]/main/he
location.send_keys("London")
```

In [13]:
```
search_button=driver.find_element(By.XPATH,"/html/body/div[3]/div/div[2]/ma
search_button.click()
```

In [20]:
```
hostel_name=[]
```

In [23]:
```
name_tags=driver.find_elements(By.XPATH,"/html/body/div[3]/div/div/div[2]/d
for i in name_tags:
    title=i.text
    hostel_name.append(title)
```

In [25]:
```
hostel_name
```

Out[25]:
["Wombat's City Hostel London",
 "Featured Properties\nPalmers Lodge - Swiss Cottage\n8.7\nFabulous\nFrom \n€33\nSt Christopher's Village\n7.7\nVery Good\nFrom\n€25\nAll properties\nSorted by our organic, commission-free algorithm.\nWombat's City Hostel London\n9.3\nSuperb\n(16688)\nHostel\n- 3.6km from city centre\n-20%\nPrivates From\n€228\n€183\n-20%\nDorms From\n€55\n€44\nSelling out fast!\nPalmers Lodge - Swiss Cottage\n8.7\nFabulous\n(16307)\nHostel\n- 6.5km from city centre\n-26%\nPrivates From\n€202\n€149\n-26%\nDorms From\n€44\n€33\nSelling out fast!\nOnefam Notting Hill\n9.7\nSuperb\n(2880)\nHostel\n- 5.5km from city centre\nPrivates From\n€185\nDorms From\n€77\nSelling out fast!\nUrbany Hostel London\n9.5\nSuperb\n(1351)\nHostel\n- 5.4km from city centre\nPrivates From\n€160\nDorms From\n€51\nOnefam Waterloo\n9.5\nSuperb\n(405)\nHostel\n- 0.7km from city centre\nNo Privates Available\nDorms From\n€48\nSelling out fast!\nNX London Hostel\n8.8\nFabulous\n(2491)\nHostel\n- 6.1km from city centre\nNo Privates Available\n-15%\nDorms From\n€38\n€32\nSt Christopher's Village\n7.7\nVery Good\n(13155)\nHostel\n- 1.8km from city centre\n-10%\nPrivates From\n€94\n€85\n-10%\nDorms From\n€28\n€25\nClink261\n8.2\nFabulous\n(719)\nHostel\n- 3.2km from city centre\nNo Privates Available\nDorms From\n€36\nGenerator London\n7.2\nVery Good\n(8335)\nHostel\n- 3km from city centre\nPrivates From\n€147\nDorms From\n€35\nNo.8 Seven Sisters\n7.7\nVery Good\n(4315)\nHostel\n- 9km from city centre\nNo Privates Available\nDorms From\n€15\nAstor Museum Inn\n8.9\nFabulous\n(9562)\nHostel\n- 2.2km from city centre\nNo Privates Available\n-10%\nDorms From\n€33\n€30\nSelling out fast!\nSafestay London Elephant & Castle\n7.1\nVery Good\n(5387)\nHostel\n- 1.7km from city centre\nPrivates From\n€196\nDorms From\n€22\nNo.8 Willesden Hostel London\n6.9\nGood\n(5303)\nHostel\n- 10km from city centre\nNo Privates Available\nDorms From\n€17\nSelling out fast!\nDestinations Hostels @ The Gallery\n9.1\nSuperb\n(408)\nHostel\n- 1.7km from city centre\nNo Privates Available\nDorms From\n€48\nSmart Russell Square Hostel\n7.5\nVery Good\n(10319)\nHostel\n- 2.6km from city centre\nNo Privates Available\nDorms From\n€28\nQueen Elizabeth Chelsea\n8.0\nFabulous\n(3815)\nHostel\n- 5.7km from city centre\nNo Privates Available\nDorms From\n€21\nAstor Hyde Park\n7.9\nVery Good\n(12419)\nHostel\n- 4.3km from city centre\nNo Privates Available\n-10%\nDorms From\n€28\n€26\nSelling out fast!\nLondon Backpackers\n8.5\nFabulous\n(4645)\nHostel\n- 11.9km from city centre\nPrivates From\n€94\nDorms From\n€26\nHostelle - women only hostel London\n8.5\nFabulous\n(221)\nHostel\n- 5.1km from city centre\nNo Privates Available\nDorms From\n€42\nPhoenix Hostel\n7.0\nVery Good\n(4815)\nHostel\n- 4.2km from city centre\nNo Privates Available\n-5%\nDorms From\n€24\n€23\nSmart Camden Inn Hostel\n8.8\nFabulous\n(3213)\nHostel\n- 4.4km from city centre\nNo Privates Available\nDorms From\n€30\nSaint James Backpackers\n8.0\nFabulous\n(1990)\nHostel\n- 5.5km from city centre\nNo Privates Available\n-10%\nDorms From\n€43\n€39\nYHA London Oxford Street\n9.2\nSuperb\n(4728)\nHostel\n- 2.1km from city centre\nPrivates From\n€115\nNo Dorms Available\nSelling out fast!\nSmart Hyde Park Inn Hostel\n7.7\nVery Good\n(6878)\nHostel\n- 5km from city centre\nPrivates From\n€191\nDorms From\n€24\nSelling out fast!\nBarmy Badger Backpackers\n9.2\nSuperb\n(2095)\nHostel\n- 5.5km from city centre\nPrivates From\n€138\nDorms From\n€45\nSelling out fast!\nSt Christopher's Hammersmith\n7.8\nVery Good\n(4475)\nHostel\n- 7.5km from city centre\n-10%\nPrivates From\n€103\n€93\n-10%\nDorms From\n€25\n€22\nSelling out fast!\nAstor Victoria\n7.0\nVery Good\n(15179)\nHostel\n- 1.8km from city centre\n-10%\nPrivates From\n€110\n€99\n-10%\nDorms From\n€26\n€24\nKabannas London St Pancras\n8.4\nFabulous\n(3060)\nHostel\n- 3.3km from city centre\nPrivates From\n€201\nDorms From\n€50\nSelling out fast!\nSt Christopher's Inn - London Bridge\n7.4\nVery Good\n(3677)\nHostel\n- 1.8km from city centre\n-10%\nPrivates From\n€103\n€93\n-10%\nDorms From\n€35\n€31\nSt Christopher's Camden\n8.2\nFabulous\n(4195)\nHostel\n- 4.3km from city centre\nNo Privates Available\n-10%\nDorms From\n€30\n€27\n1\n2\n3"]