In [ ]:

```python
#import all required packages
import csv
from datetime import datetime
import requests
from bs4 import BeautifulSoup
import time
from selenium import webdriver
import re
import pandas as pd
import functools
from functools import reduce
```

In [ ]:

```python
#function to get the url for desired job role
def get_url(position,page,post):
    template='https://www.naukri.com/{}-jobs-{}?k={}'
    url=template.format(position,page,post)
    return url

#user input for url
post=input("Search job role: ")
position=post
page=1               #To initialize the url from first page.
url=get_url(position,page,post)
print(url)
```

In [ ]:

```python
#To check if the website allows web scraping
response = requests.get(url)
print(response)
```

In [ ]:

```python
response.reason
```

In [ ]:

```python
#To trim the html tags from required data using Regular expressions
def cleanhtml(raw_html):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', raw_html)
    return cleantext
```

In [ ]:

```python
#create a new dataframe for scraped data
df = pd.DataFrame(columns=['Company','Description','Experience','Location','Salary','
df
```

In [ ]:

```python
for i in range(1,21):#to navigate through 20 pages
    print("page : ",i)
    url=get_url(position,i,post)#pass i for page
    driver = webdriver.Chrome("D:\chromedriver.exe")
    driver.get(url)

    time.sleep(10)

    soup = BeautifulSoup(driver.page_source,'html5lib')
    #print(soup.prettify())

    driver.close()

    #to get a job card
    temp = soup.find(class_='list')
    cards = temp.find_all('article',class_='jobTuple bgWhite br4 mb-8')
    #print(len(cards))

    for cards in cards:#navigate through all job cards in a page
        comp_a = cards.find('a',class_='subTitle ellipsis fleft')

        if comp_a is not None:#if company name is given
                Company = comp_a.text
                print(Company)
        else:#if company name is not given
            pass

        # Location
        Loc = cards.find('li',class_='fleft grey-text br2 placeHolderLi location')
        if Loc is not None:#check if location is given
            Loc_span = Loc.find('span',class_='ellipsis fleft fs12 lh16')
            if Loc_span is not None:#if location is given
                Location = Loc_span.text.strip()
                print(Location)

            else:
                pass

        else:#if location not is given
            pass

        # Years of experience Required
        Exp = cards.find('li',class_='fleft grey-text br2 placeHolderLi experience')
        if Exp is not None:#if experience is given
            Exp_span = Exp.find('span',class_='ellipsis fleft fs12 lh16')
            if Exp_span is not None:#if experience is given
                Experience = Exp_span.text
                print(Experience)

            else:#if experience is not given
                pass

        else:
            pass


        # Salary offered
        Sal = cards.find('li',class_='fleft grey-text br2 placeHolderLi salary')
        #Sal_span = Sal.find('span',class_='ellipsis fleft fs12 lh16')
        if Sal is not None:#if Salary is given
            Sal_span = Sal.find('span',class_='ellipsis fleft fs12 lh16')
            if Sal_span is not None:
                Salary = Sal_span.text
                print(Salary)

            else:
                pass#if salary is not given

        else:
            pass

       #To get all skills in a jobcard finding all li tags in ul and printing the tex
        Skill=[]
        sk1 = cards.find(class_='tags has-description')
        sk2 = sk1.find_all('li',class_='fleft fs12 grey-text lh16 dot')
        for li in sk2 :
            li=str(li)
            li=cleanhtml(li)
            Skill.append(li)
        print(Skill)


        #Description
        desc = cards.find('div',class_='job-description fs12 grey-text')
        if desc is not None:
                Description = desc.text
                print(Description)

        else:
            pass#if description is not given




        #Append rows in Dataframe
        df=df.append({'Company':Company,'Description':Description,'Experience':Experie
```

In [ ]:

```python
print(df)
```

In [ ]:

```python
print(df.shape)
print(df.head(10))
print(df.tail(10))
```

In [ ]:

```python
#store dataframe in a file
df.to_csv("D:/Scrape_Naukri.csv",index=False)
```

In [ ]:

```python
print(df['Skills'])
```

In [ ]:

```python
print('DataFrame information: ',df.info())
```

In [ ]:

```python
sk_set=df['Skills'].to_list()
print(sk_set)
```

In [ ]:

```python
#flatten list(2d to 1d)
Skills_1d = reduce(lambda z, y :z + y, sk_set)
print(Skills_1d)
```

In [ ]:

```python
#To print unique skill element count using dictionary(skill_count)
Skill_count = dict(zip(list(Skills_1d),[list(Skills_1d).count(i) for i in list(Skills_
print("Dictionary : ",Skill_count)
print("Count : ",len(Skill_count))
print(Skill_count)
```

In [ ]:

```python
#storing this dictionary to a dataframe S_count
S_count = pd.DataFrame.from_dict(Skill_count , orient ='index')
S_count
```

In [ ]:

```python
#store dataframe in a file
S_count.to_csv("D:/Skill_count.csv")
```

In [ ]:

In [ ]: