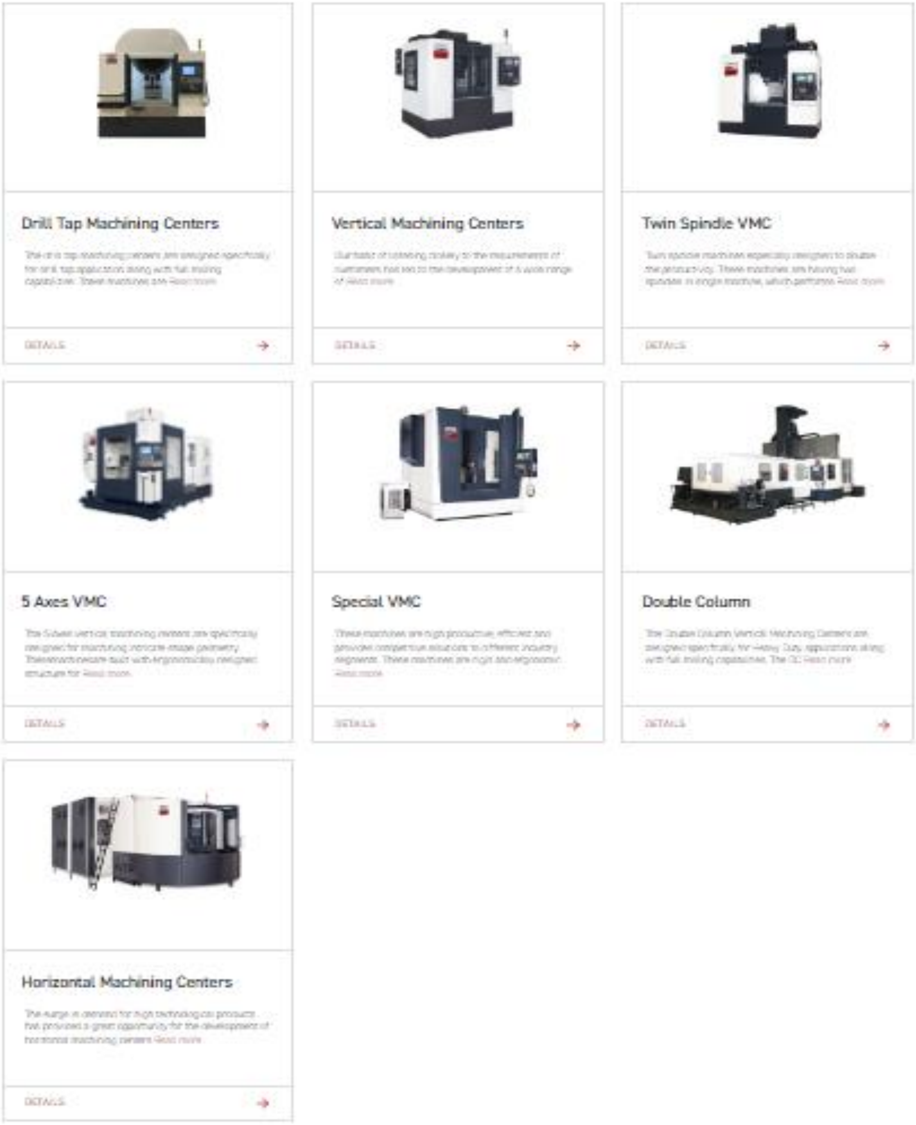
Web Scraping Project

Objective: scrape product specifications from the webpage
https://www.acemicromatic.net/product_cat/milling/



Drill Tap Machining Centers

The drill tap machining centers are designed specifically for drill tap application along with full milling capabilities. These machines are Read more

DETAILS →

Vertical Machining Centers

Our habit of listening closely to the requirements of customers has led to the development of a wide range of Read more

DETAILS →

Twin Spindle VMC

Twin spindle machines especially designed to double the productivity. These machines are having two spindles in single machine, which performs Read more

DETAILS →

5 Axes VMC

The 5 axes vertical machining centers are specifically designed for machining intricate shape geometry. These machines are built with ergonomically designed structure for Read more

DETAILS →

Special VMC

These machines are high productive, efficient and provides competitive solutions to different industry segments. These machines are rigid and ergonomic. Read more

DETAILS →

Double Column

The Double Column Vertical Machining Centers are designed specifically for Heavy Duty applications along with full milling capabilities. The DC Read more

DETAILS →

Horizontal Machining Centers

The surge in demand for high technological products has provided a great opportunity for the development of horizontal machining centers Read more

DETAILS →

1 Libraries used :

Following libraries are used for web scrapping

1.1 requests:

The requests library in Python is primarily used for making HTTP requests to websites and APIs. It simplifies tasks like fetching web pages, accessing web services, handling authentication, customizing headers, managing sessions, and downloading files. It's an essential tool for web scraping, data retrieval, and interacting with web-based resources in Python applications.

1.2 bs4 :

BeautifulSoup, imported via from bs4 import BeautifulSoup, is a Python library for parsing and manipulating HTML and XML documents. It's a go-to choice for web scraping tasks, enabling developers to extract, navigate, and modify data within web pages or XML files efficiently. When paired with the requests library, it becomes a valuable tool for collecting and processing data from the web for various applications, from data analysis to content aggregation and more.

1.3 pandas :

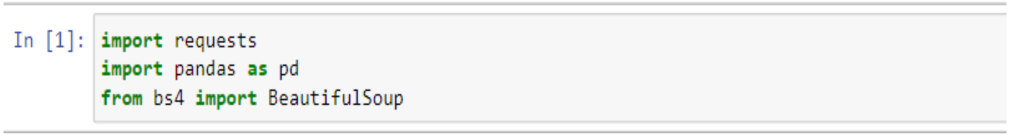
pandas is a Python library essential for web scraping projects. It simplifies data management, cleaning, and analysis, making it a valuable tool for organizing and working with scraped data efficiently.

2 Steps used for web scraping :

The following steps are used for web scaping

2.1 import required libraries :

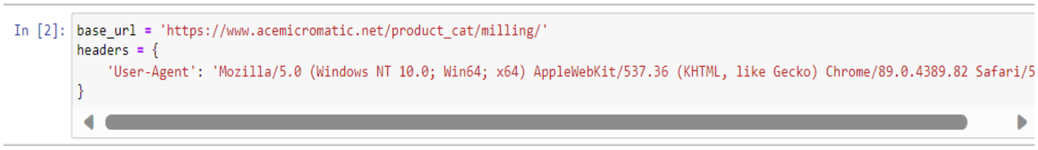
we import above mentioned python libraries



```
In [1]: import requests
        import pandas as pd
        from bs4 import BeautifulSoup
```

2.2 define base url and headers :

create a variable for assign URL of the base Page



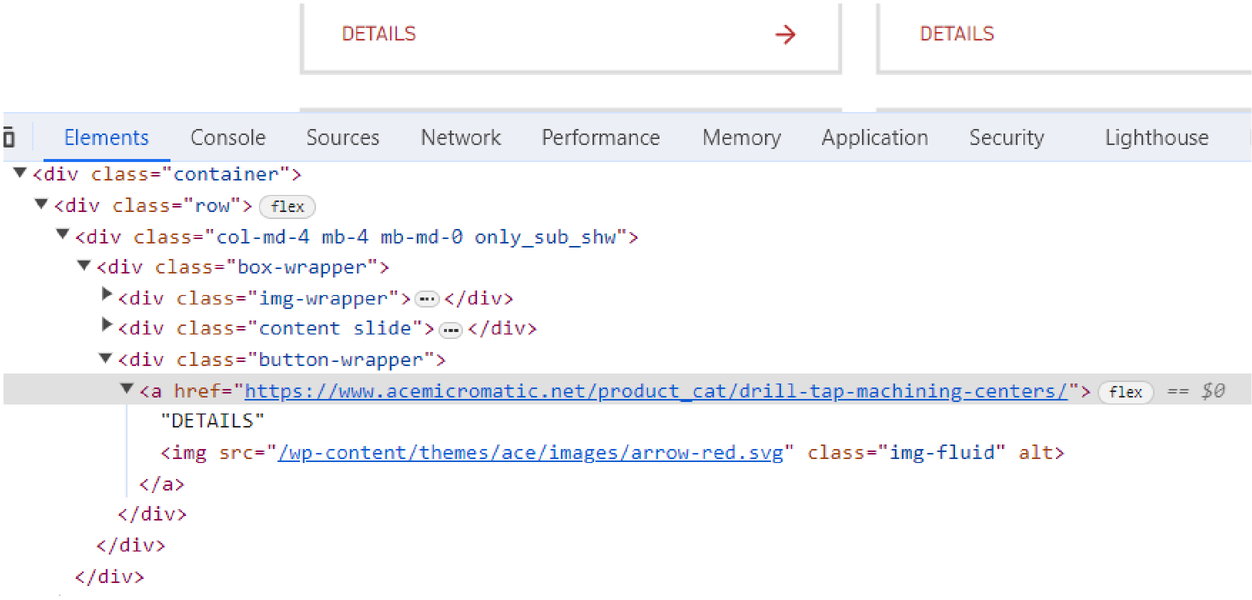
```
In [2]: base_url = 'https://www.acemicromatic.net/product_cat/milling/'
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/5
        }
```

2.3 create function for scraping data from website:

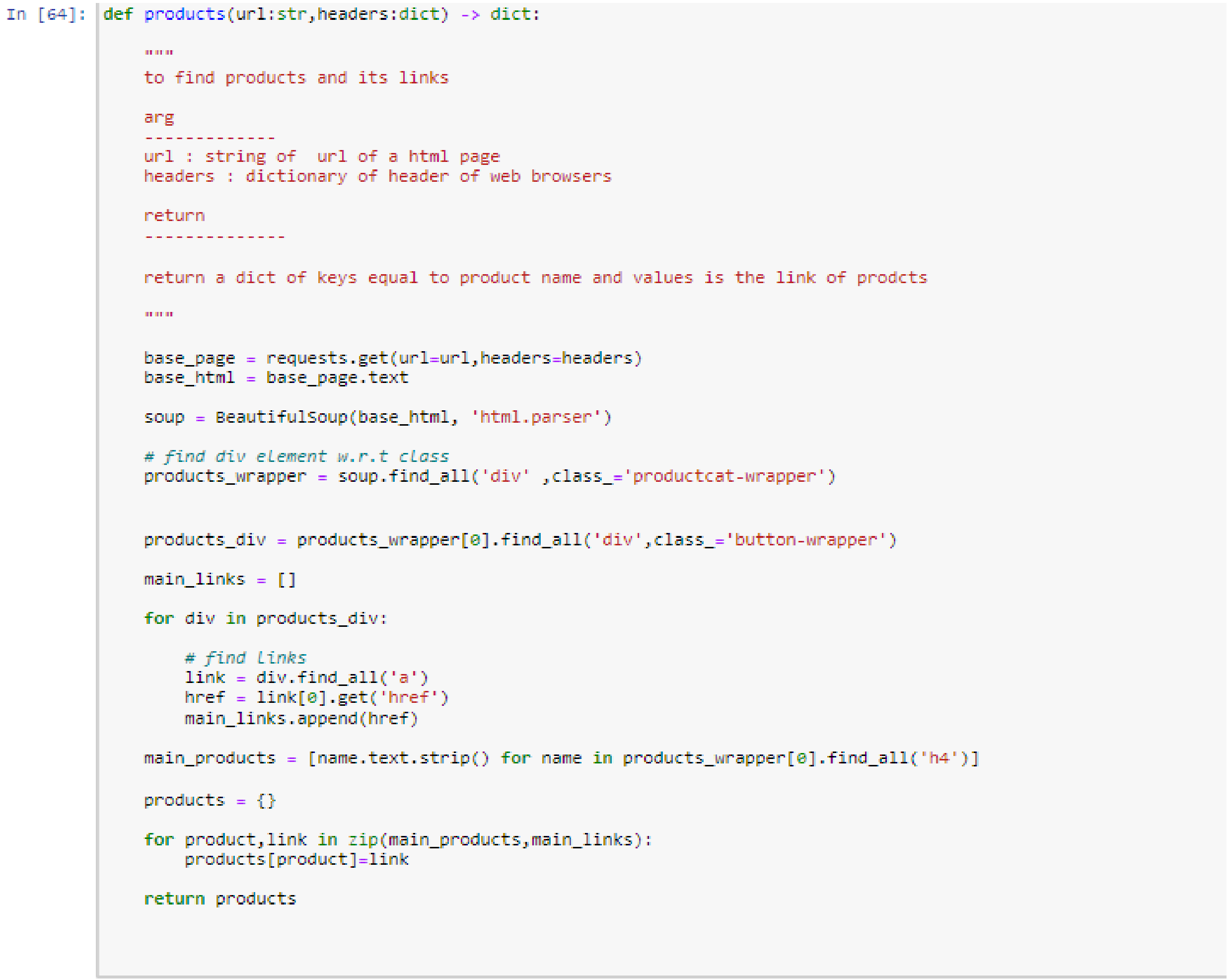
create functions to make easier and readable the program

2.3.1 create function for taking link of milling product:

the base page 5 products to get the link of this product page we locate to details Button by using find_all function and class name of this button . then we looking <a> element of the button and find the ‘href’ of the <a> for link



```
▼<div class="container">
  ▼<div class="row"> flex
    ▼<div class="col-md-4 mb-4 mb-md-0 only_sub_shw">
      ▼<div class="box-wrapper">
        ▶<div class="img-wrapper">…</div>
        ▶<div class="content slide">…</div>
        ▼<div class="button-wrapper">
          ▼<a href="https://www.acemicromatic.net/product_cat/drill-tap-machining-centers/"> flex  == $0
              "DETAILS"
              <img src="/wp-content/themes/ace/images/arrow-red.svg" class="img-fluid" alt>
            </a>
        </div>
      </div>
    </div>
```

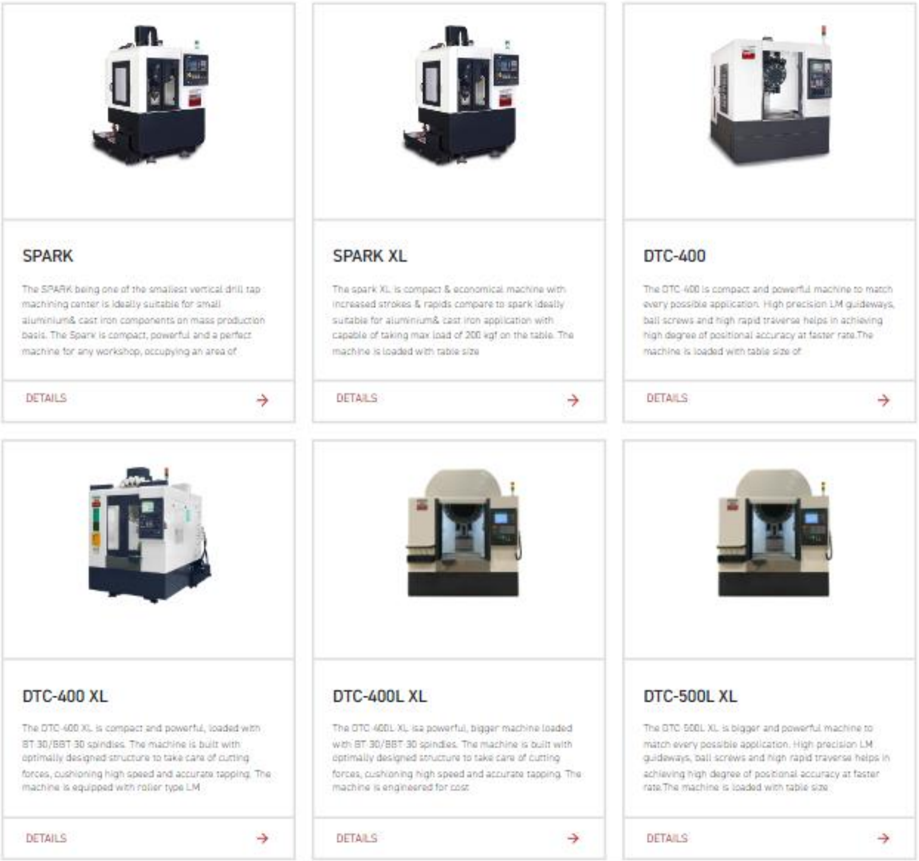


In [64]:

```
def products(url:str,headers:dict) -> dict:

    """
    to find products and its links

    arg
    -------------
    url : string of  url of a html page
    headers : dictionary of header of web browsers

    return
    --------------

    return a dict of keys equal to product name and values is the link of prodcts

    """

    base_page = requests.get(url=url,headers=headers)
    base_html = base_page.text

    soup = BeautifulSoup(base_html, 'html.parser')

    # find div element w.r.t class
    products_wrapper = soup.find_all('div' ,class_='productcat-wrapper')


    products_div = products_wrapper[0].find_all('div',class_='button-wrapper')

    main_links = []

    for div in products_div:

        # find links
        link = div.find_all('a')
        href = link[0].get('href')
        main_links.append(href)

    main_products = [name.text.strip() for name in products_wrapper[0].find_all('h4')]

    products = {}

    for product,link in zip(main_products,main_links):
        products[product]=link

    return products
```

2.3.2 create function for taking link for sub categories :

similar to above function we use same logic in this function



SPARK

The SPARK being one of the smallest vertical drill tap machining center is ideally suitable for small aluminium& cast iron components on mass production basis. The Spark is compact, powerful and a perfect machine for any workshop, occupying an area of

DETAILS →

SPARK XL

The spark XL is compact & economical machine with increased strokes & rapids compare to spark ideally suitable for aluminium& cast iron application with capable of taking max load of 200 kgf on the table. The machine is loaded with table size

DETAILS →

DTC-400

The DTC-400 is compact and powerful machine to match every possible application. High precision LM guideways, ball screws and high rapid traverse helps in achieving high degree of positional accuracy at faster rate.The machine is loaded with table size of

DETAILS →

DTC-400 XL

The DTC-400 XL is compact and powerful, loaded with BT 30/BBT 30 spindles. The machine is built with optimally designed structure to take care of cutting forces, cushioning high speed and accurate tapping. The machine is equipped with roller type LM

DETAILS →

DTC-400L XL

The DTC-400L XL isa powerful, bigger machine loaded with BT 30/BBT 30 spindles. The machine is built with optimally designed structure to take care of cutting forces, cushioning high speed and accurate tapping. The machine is engineered for cost

DETAILS →

DTC-500L XL

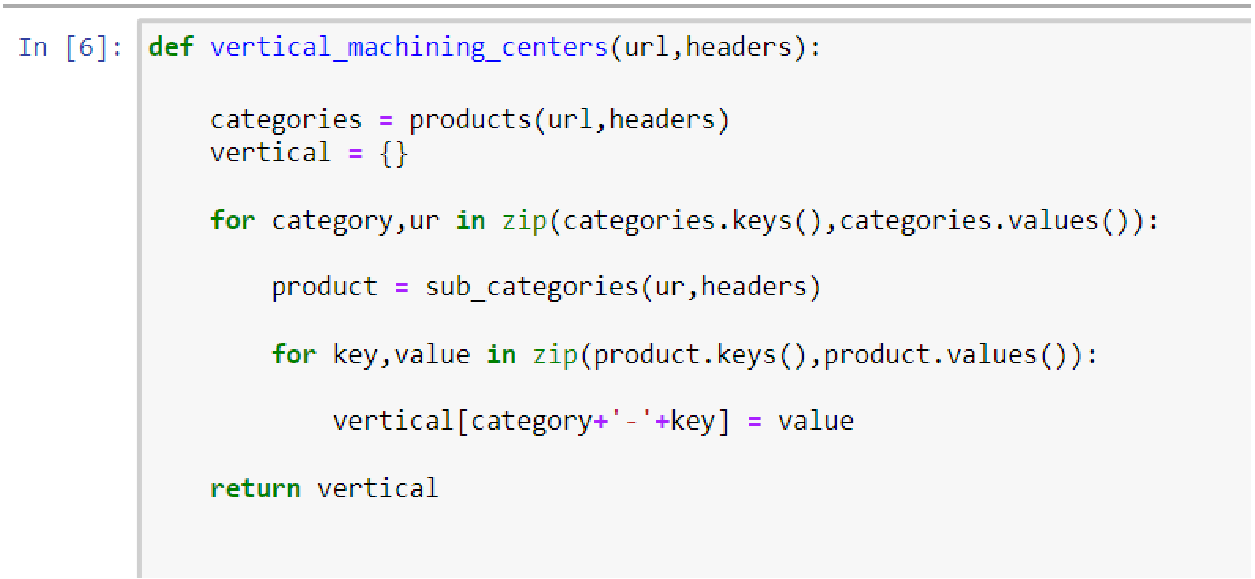
The DTC-500L XL is bigger and powerful machine to match every possible application. High precision LM guideways, ball screws and high rapid traverse helps in achieving high degree of positional accuracy at faster rate.The machine is loaded with table size

DETAILS →

```
In [5]: def sub_categories(url,headers):

            product_page= requests.get(url,headers=headers)
            product_page_html = product_page.text
            soup = BeautifulSoup(product_page_html, 'html.parser')
            products_wrapper = soup.find_all('div' ,class_='section-block common-block wow fadeIn')
            products_div = products_wrapper[0].find_all('div',class_='button-wrapper')

            main_links = []

            for div in products_div:

                link = div.find_all('a')
                href = link[0].get('href')
                main_links.append(href)

            main_products = [name.text.strip() for name in products_wrapper[0].find_all('h4')]
            product= {}
            for categories,link in zip(main_products,main_links):
                product[categories]=link

            return product
```

creating special function to extract data from ‘vertical_machining_centers’:

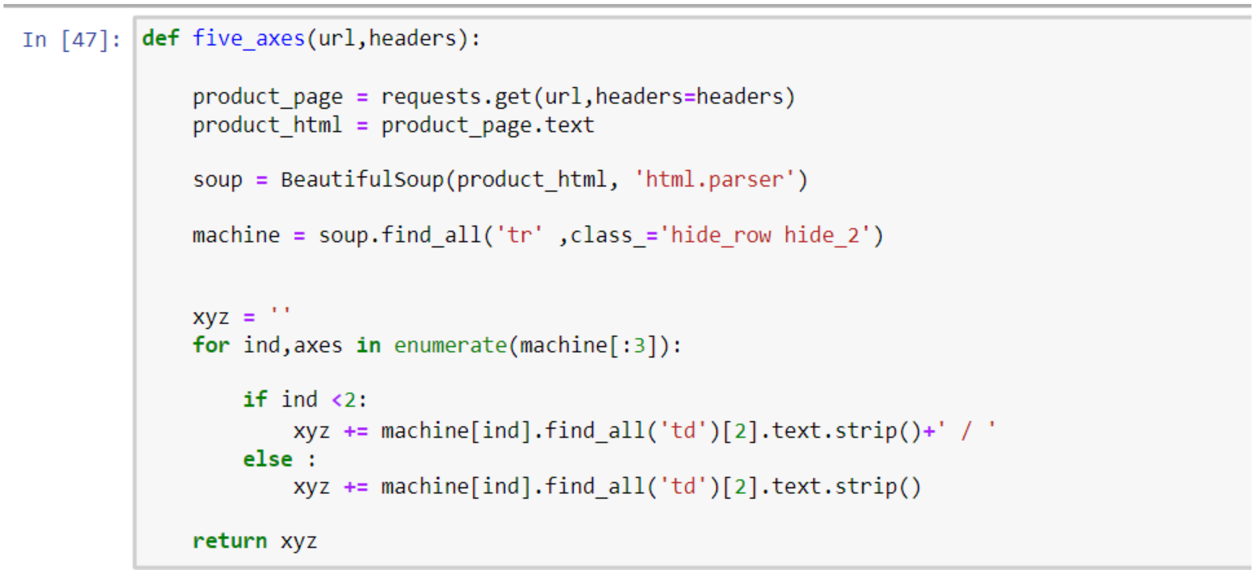
for ‘vertical_machining_centers’ mechine has again sub categories so we want to find sub categories twice.



```
In [6]: def vertical_machining_centers(url,headers):

            categories = products(url,headers)
            vertical = {}

            for category,ur in zip(categories.keys(),categories.values()):

                product = sub_categories(ur,headers)

                for key,value in zip(product.keys(),product.values()):

                    vertical[category+'-'+key] = value

            return vertical
```

2.3.3 creating a function for ‘five_axes’ milling machine:

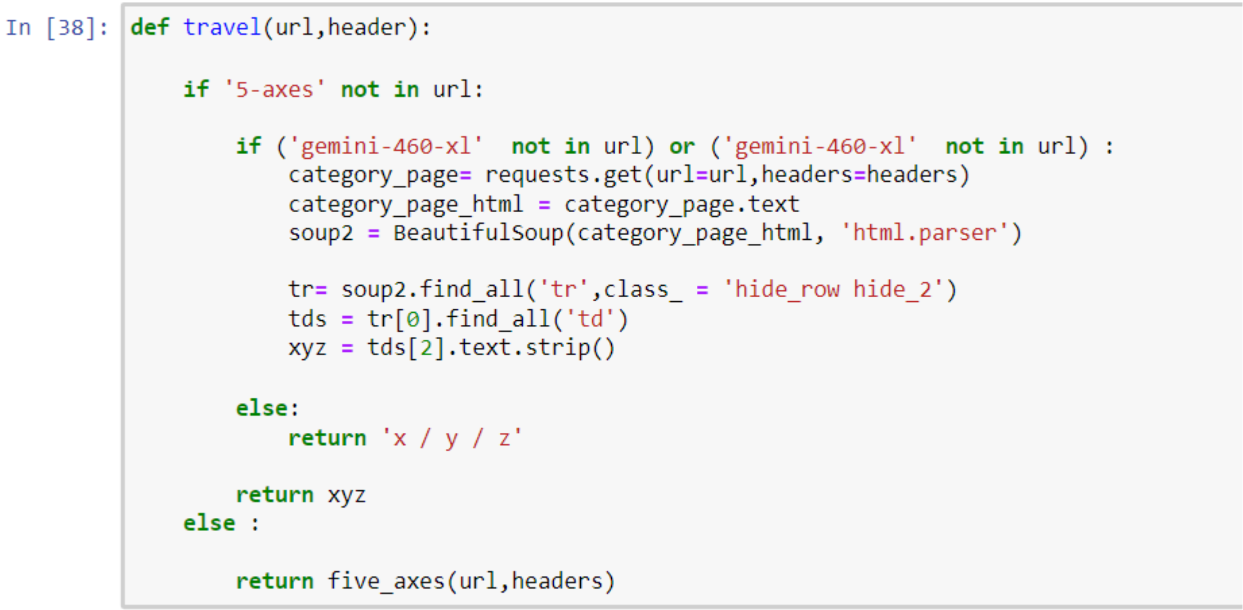
for five axes milling machine the travel is not in the form of (x/y/z). we has separate fields for each of the axes so, this function will find travel in the form of (x/y/z)



```
In [47]: def five_axes(url,headers):

             product_page = requests.get(url,headers=headers)
             product_html = product_page.text

             soup = BeautifulSoup(product_html, 'html.parser')

             machine = soup.find_all('tr' ,class_='hide_row hide_2')


             xyz = ''
             for ind,axes in enumerate(machine[:3]):

                 if ind <2:
                     xyz += machine[ind].find_all('td')[2].text.strip()+' / '
                 else :
                     xyz += machine[ind].find_all('td')[2].text.strip()

             return xyz
```

2.3.4 creating a function to extract travels :

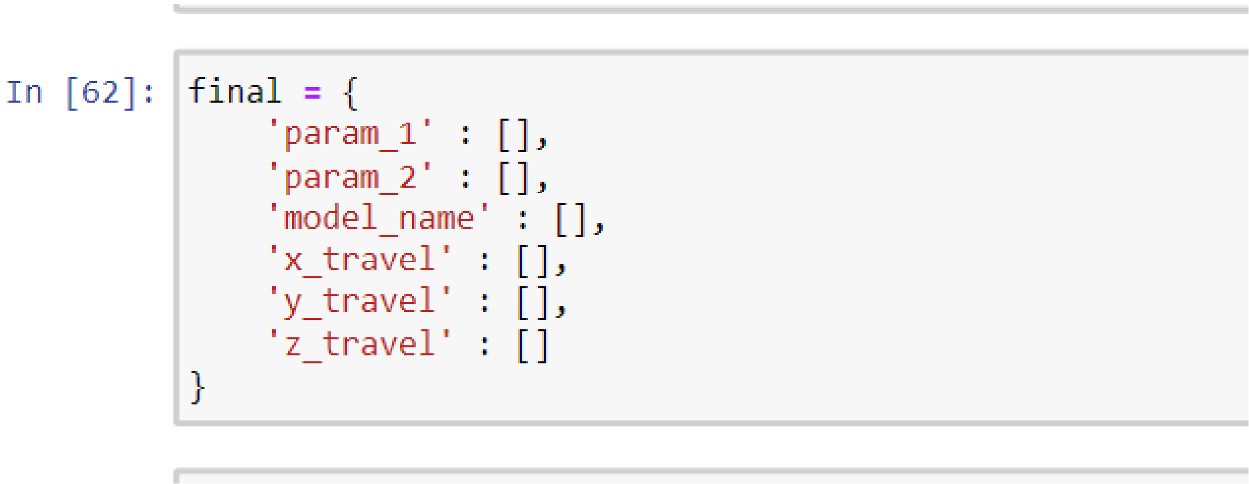
this function find the travel of the each products



```
In [38]: def travel(url,header):

             if '5-axes' not in url:

                 if ('gemini-460-xl'  not in url) or ('gemini-460-xl'  not in url) :
                     category_page= requests.get(url=url,headers=headers)
                     category_page_html = category_page.text
                     soup2 = BeautifulSoup(category_page_html, 'html.parser')

                     tr= soup2.find_all('tr',class_ = 'hide_row hide_2')
                     tds = tr[0].find_all('td')
                     xyz = tds[2].text.strip()

                 else:
                     return 'x / y / z'

                 return xyz
             else :

                 return five_axes(url,headers)
```

2.4 creating dictionary for store this data:

we created a dictionary with keys are same as the headers of the output csv file.



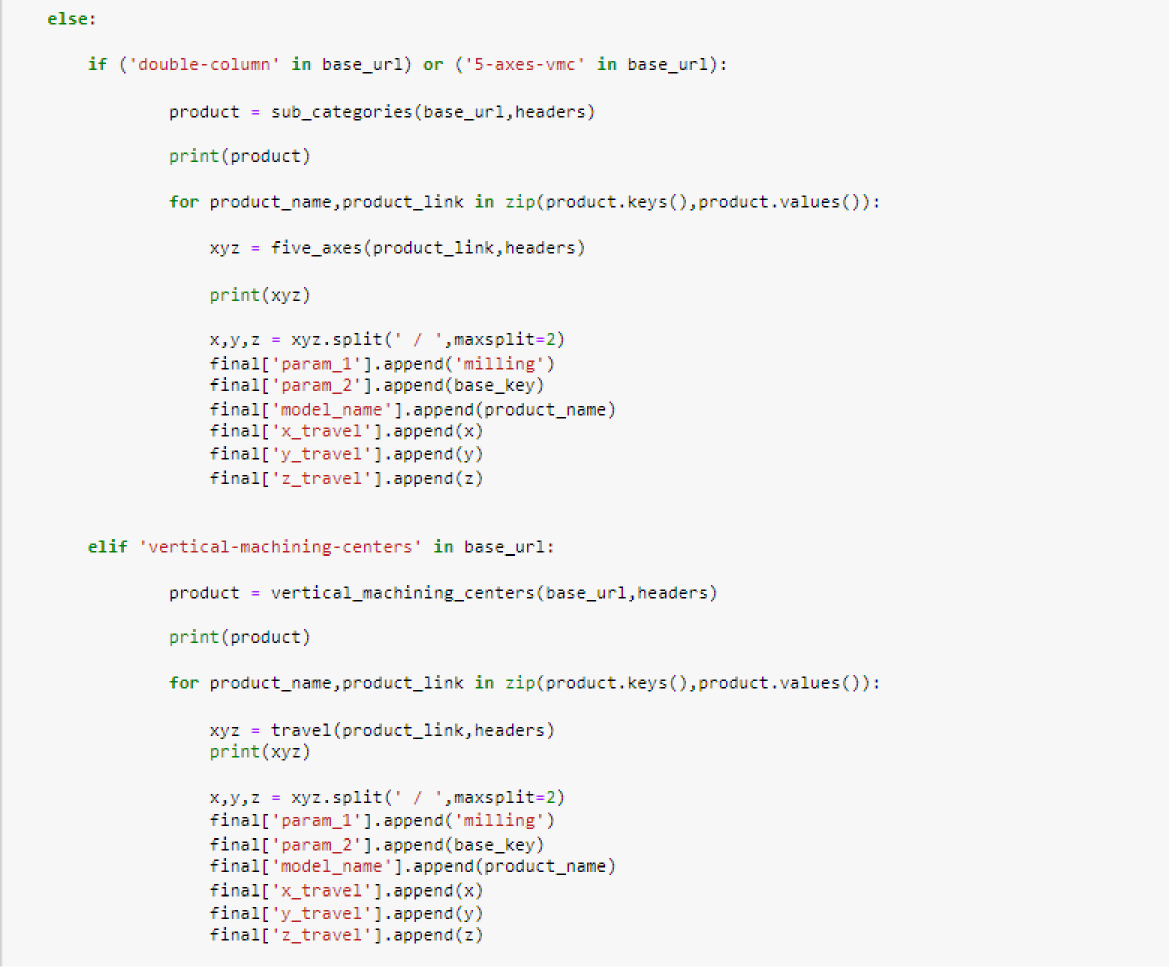
```
In [62]: final = {
                 'param_1' : [],
                 'param_2' : [],
                 'model_name' : [],
                 'x_travel' : [],
                 'y_travel' : [],
                 'z_travel' : []
         }
```

2.5 Loop through all base link and append value to final dictionary:

This loop will extract the data from entire product data from website

In [63]:

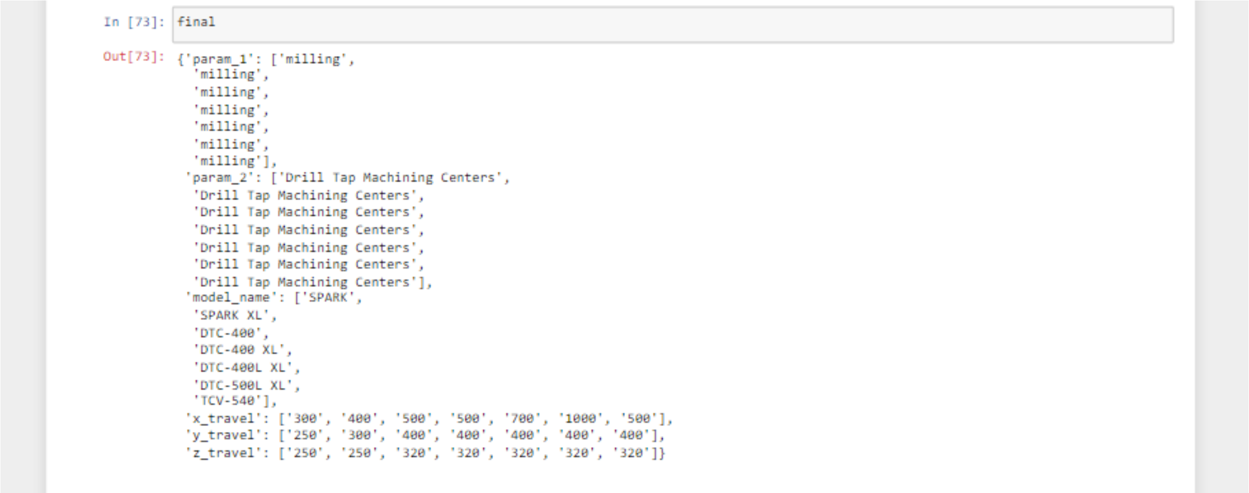
```
base_products = products('https://www.acemicromatic.net/product_cat/milling/',headers)

for base_key,base_url in zip(base_products.keys(),base_products.values()):


    if ('vertical-machining-centers' not in base_url) and ('double-column' not in base_url) and ('5-axes-vmc' not in base_url):

        product = sub_categories(base_url,headers)

        print(product)

        for product_name,product_link in zip(product.keys(),product.values()):

            xyz = travel(product_link,headers)

            x,y,z = xyz.split(' / ',maxsplit=2)

            final['param_1'].append('milling')
            final['param_2'].append(base_key)
            final['model_name'].append(product_name)
            final['x_travel'].append(x)
            final['y_travel'].append(y)
            final['z_travel'].append(z)
```



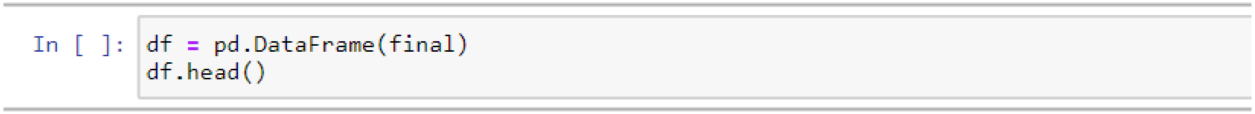
```
    else:

        if ('double-column' in base_url) or ('5-axes-vmc' in base_url):

                product = sub_categories(base_url,headers)

                print(product)

                for product_name,product_link in zip(product.keys(),product.values()):

                    xyz = five_axes(product_link,headers)

                    print(xyz)

                    x,y,z = xyz.split(' / ',maxsplit=2)
                    final['param_1'].append('milling')
                    final['param_2'].append(base_key)
                    final['model_name'].append(product_name)
                    final['x_travel'].append(x)
                    final['y_travel'].append(y)
                    final['z_travel'].append(z)


        elif 'vertical-machining-centers' in base_url:

                product = vertical_machining_centers(base_url,headers)

                print(product)

                for product_name,product_link in zip(product.keys(),product.values()):

                    xyz = travel(product_link,headers)
                    print(xyz)

                    x,y,z = xyz.split(' / ',maxsplit=2)
                    final['param_1'].append('milling')
                    final['param_2'].append(base_key)
                    final['model_name'].append(product_name)
                    final['x_travel'].append(x)
                    final['y_travel'].append(y)
                    final['z_travel'].append(z)
```

2.6 Convert data into DataFrame :

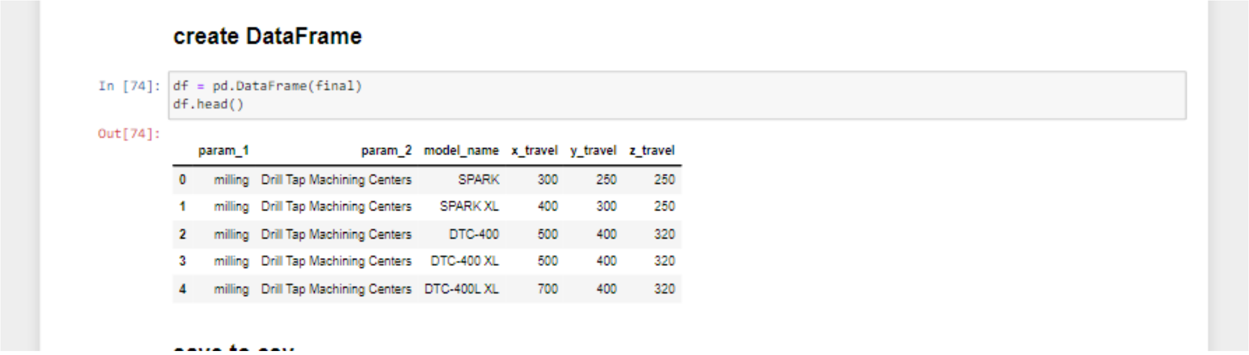
Create a DataFrame from final dictionary .



```
In [73]: final

Out[73]: {'param_1': ['milling',
           'milling',
           'milling',
           'milling',
           'milling',
           'milling',
           'milling'],
          'param_2': ['Drill Tap Machining Centers',
           'Drill Tap Machining Centers',
           'Drill Tap Machining Centers',
           'Drill Tap Machining Centers',
           'Drill Tap Machining Centers',
           'Drill Tap Machining Centers',
           'Drill Tap Machining Centers'],
          'model_name': ['SPARK',
           'SPARK XL',
           'DTC-400',
           'DTC-400 XL',
           'DTC-400L XL',
           'DTC-500L XL',
           'TCV-540'],
          'x_travel': ['300', '400', '500', '500', '700', '1000', '500'],
          'y_travel': ['250', '300', '400', '400', '400', '400', '400'],
          'z_travel': ['250', '250', '320', '320', '320', '320', '320']}
```



```
In [ ]: df = pd.DataFrame(final)
        df.head()
```



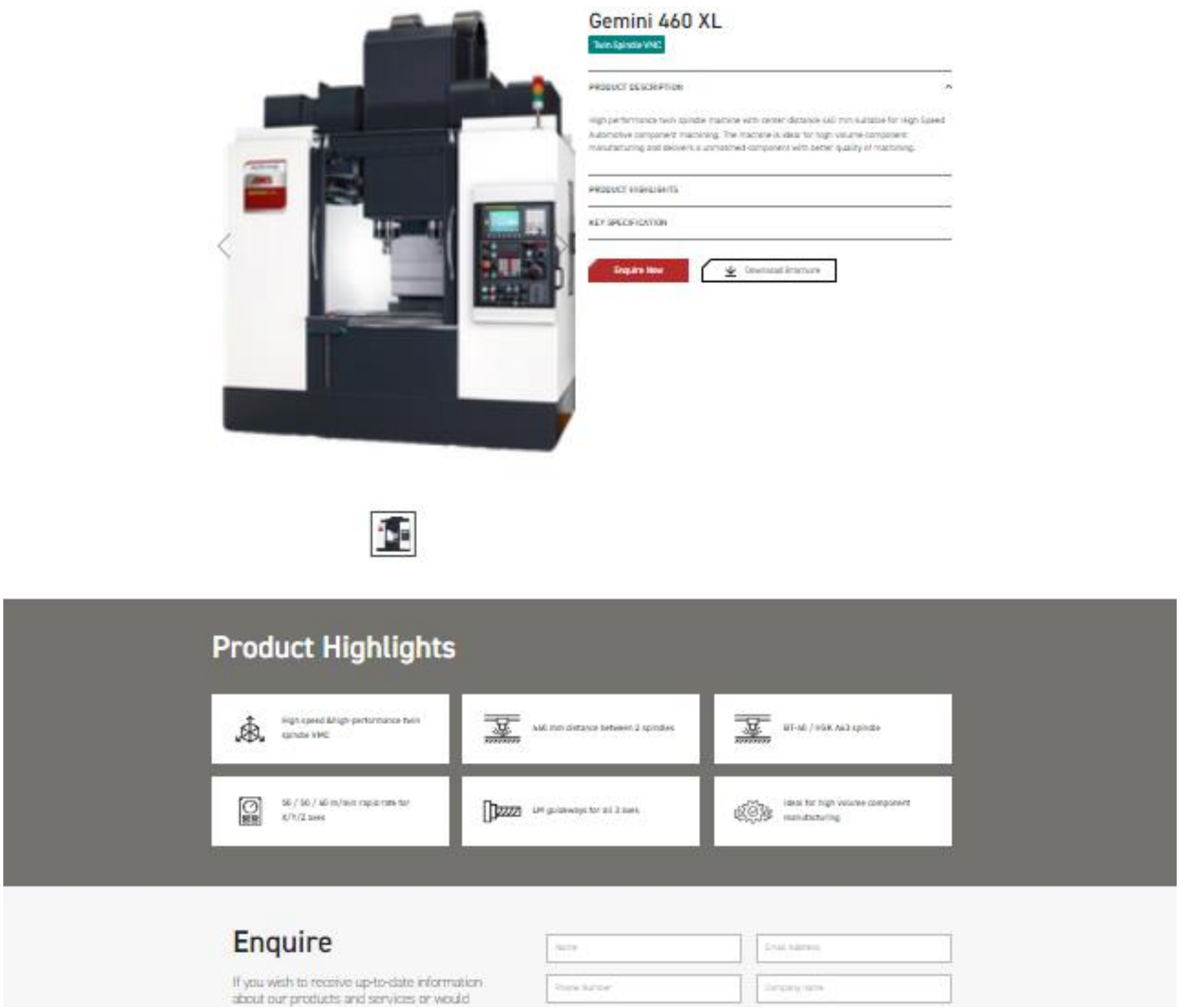
create DataFrame

```
In [74]: df = pd.DataFrame(final)
         df.head()

Out[74]:
```

	param_1	param_2	model_name	x_travel	y_travel	z_travel
0	milling	Drill Tap Machining Centers	SPARK	300	250	250
1	milling	Drill Tap Machining Centers	SPARK XL	400	300	250
2	milling	Drill Tap Machining Centers	DTC-400	500	400	320
3	milling	Drill Tap Machining Centers	DTC-400 XL	500	400	320
4	milling	Drill Tap Machining Centers	DTC-400L XL	700	400	320

save to csv

2.7 Replace x, y, z with None:



Gemini 460 XL

Twin Spindle VMC

PRODUCT DESCRIPTION

High performance twin spindle machine with center distance 460 mm suitable for High Speed Automotive component machining. The machine is ideal for high volume component manufacturing and delivers a unmatched component with better quality of machining.

PRODUCT HIGHLIGHTS

KEY SPECIFICATION

Enquire Now

Download Brochure

Product Highlights

- High speed &high performance twin spindle VMC
- 460 mm distance between 2 spindles
- BT-40 / HSK A63 spindle
- 50 / 50 / 40 m/min rapid rate for X/Y/Z axes
- LM guideways for all 3 axes
- Ideal for high volume component manufacturing

Enquire

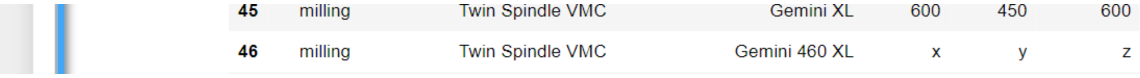
If you wish to receive up-to-date information about our products and services or would

Name

Email Address

Phone Number

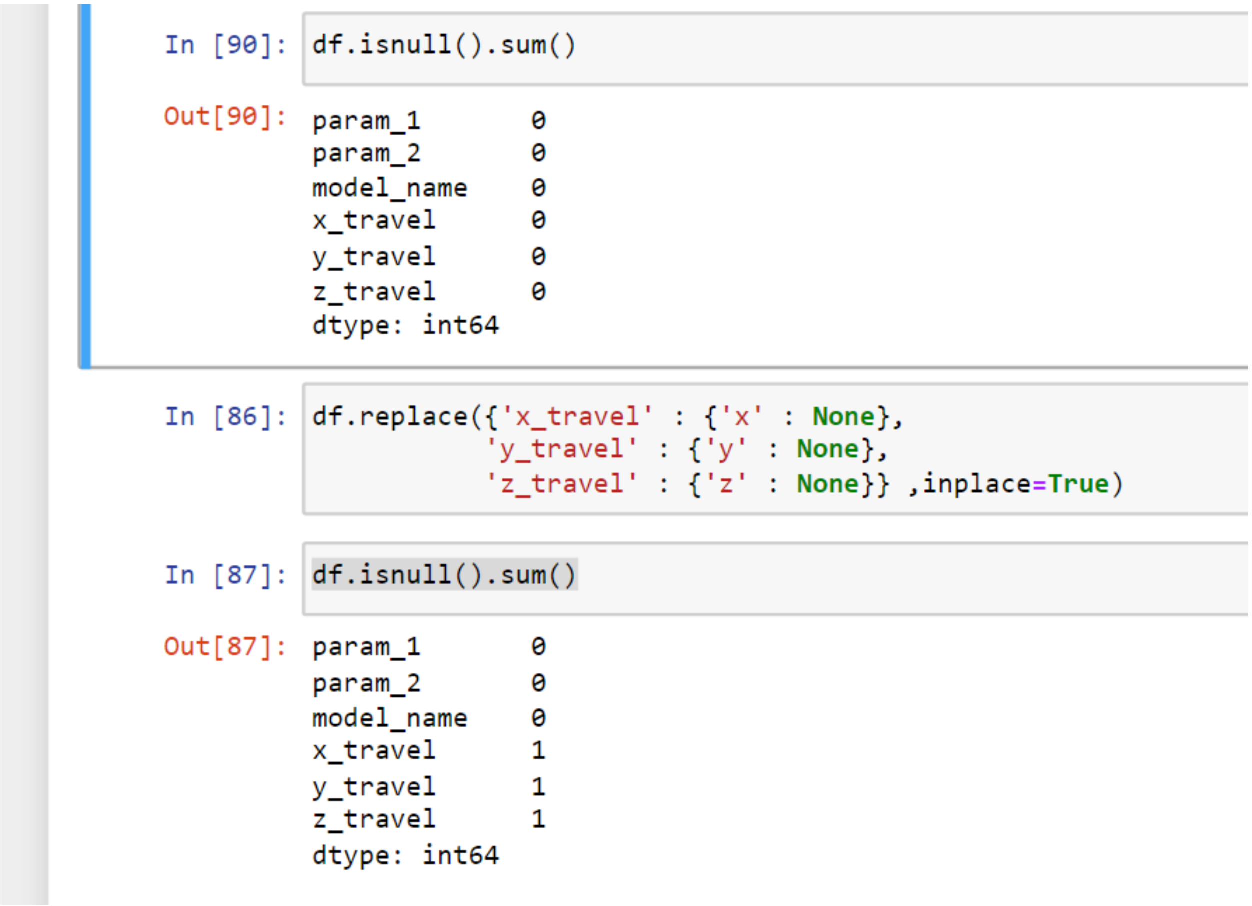
Company name



45	milling	Twin Spindle VMC	Gemini XL	600	450	600
46	milling	Twin Spindle VMC	Gemini 460 XL	x	y	z

For a Gemini 460 xl milling has no x, y ,z value available in website so we fill with x, y, z this time we replace this x,y,z to None .

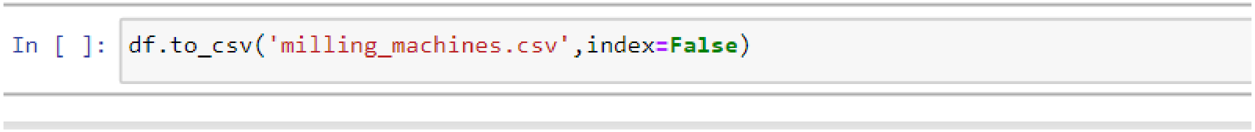
it use full for further analysis . we easily identifies **this** row using df.isnull().sum()



```
In [90]: df.isnull().sum()

Out[90]: param_1       0
         param_2       0
         model_name    0
         x_travel      0
         y_travel      0
         z_travel      0
         dtype: int64

In [86]: df.replace({'x_travel' : {'x' : None},
                     'y_travel' : {'y' : None},
                     'z_travel' : {'z' : None}} ,inplace=True)

In [87]: df.isnull().sum()

Out[87]: param_1       0
         param_2       0
         model_name    0
         x_travel      1
         y_travel      1
         z_travel      1
         dtype: int64
```

2.8 Save DataFrame to csv format:

Convert the DataFrame into csv **and** save into base directory



```
In [ ]: df.to_csv('milling_machines.csv',index=False)
```