Price Stock

raw data is astring so we are not able to work with it so we use:

- **import json**
- **data = json.loads(raw_data)**
- **data**
- **to save data:**
- **with open(r'c:\temp.key.json' ,'w') as f:**
- **f.write(raw_data)**

-in json file (offers is a list so to use it):

- **for item in data["offers"]:**
- **print(item["name"])**

just for example: when we use print instead of yield:

- **we use the coMmand : scrapy filename.py -L WARN ----> L is limit and also is log level**

after importing :

- **from scrapy.crawler import CrawlerProcess**
- **and we write this : 1- process = CrawlerProcess(). 2- process.crawl(class name). 3-process.start()**
- **after that in command we can use the command : python filename.py ---> directly without run**
- **instead of saving as : python filename.py -o newFileName.csv, we will make avariable in our python file.**
- **to check that if the file is created in the directory type: dir filename***
- **to checkout the content: type filename .csv**

```
import scrapy

import json

import os

from scrapy.crawler import CrawlerProcess
```

```
import pandas as pd

import scraper_helper as sh
```

-The script imports the necessary libraries for web scraping, data processing, and email sending. The last import statement is for a custom module `scraper_helper` that is not included in the code snippet.-

```
CSV_FILE = 'k4US.csv'

# Set Debug to True to get mails even if No stock

DEBUG = False

FULL_SIZE_ONLY = True
```

-These are constants used throughout the script. `CSV_FILE` is the name of the CSV file where the scraped data is saved. `DEBUG` is a flag that determines whether the script sends an email notification even if no products are in stock. `FULL_SIZE_ONLY` is a flag that determines whether to only scrape data for full-size keyboards.

```
class KeychronSpider(scrapy.Spider):

  custom_settings = {

    'FEEDS': {

      CSV_FILE: {

        'format': 'csv',

        'encoding': 'utf-8',

        'overwrite': True

      },

    },

    'LOG_LEVEL': 'DEBUG' if DEBUG else 'WARN'

  }

  name = 'keychron'
```

```
    start_urls = [
        # 'https://www.keychron.com/products/keychron-k4-wireless-mechanical-keyboard-version-2',
        # 'https://www.keychron.com/products/keychron-k2-hot-swappable-wireless-mechanical-keyboard',
        # 'https://www.keychron.com/products/keychron-k2-wireless-mechanical-keyboard',
        # 'https://www.keychron.com/products/keychron-k8-tenkeyless-wireless-mechanical-keyboard',
        'https://www.keychron.com/collections/keyboard/products/keychron-k1-wireless-mechanical-keyboard'

    ]

    def parse(self, response):

        data = json.loads(response.xpath(
            '//script[@type="application/ld+json"]/text()').get()
        )

        for var in data['offers']:
            if FULL_SIZE_ONLY:
                if not '104-key' in var['name']:
                    continue
            available = True if 'InStock' in var['availability'] else False
            if not available:
                continue
            item = {
```

```
            'name': var['name'],
            'available': available,
            'price': var['price'],
            'url': response.url
        }
        yield item
```

-This is a Scrapy spider that defines the web scraping behavior. It first sets some custom settings for the spider, including the output format and log level. The `name` attribute is the name of the spider. The `start_urls` attribute is a list of URLs to scrape. The `parse` method is the main method for scraping data. It extracts data from JSON-LD format on the website and filters out products that are not in stock or not full-size if `FULL_SIZE_ONLY` flag is set to `True`. Finally, it yields an item that contains the name, availability, price, and URL of the product.

```
def get_body_subject():
    try:
        df = pd.read_csv(CSV_FILE)
        if df["available"].any():
            subject = "✔ Keychron(US) - Available"
            body = df[df["available"]].to_html()
        else:
            subject = "✖ Keychron(US) - Out of Stock" if DEBUG else None
            body = df.to_html() if DEBUG else None
    except:
        return None, None

    return body, subject
```

```
def send_mail():
    import smtplib
    from email.message import EmailMessage
    try:
        msg = EmailMessage()
        MAIL_USER = 'asdsfsdfs@gmail.com'
        MAIL_PASS = 'asdsfsdfs'
        MAIL_TO = 'asdsfsdfs@gmail.com'
        body, subject = get_body_subject()
        if not (body and subject) and not DEBUG:
            print("Nothing in stock, exiting...")
            return
        msg["Subject"] = subject
        msg['From'] = MAIL_USER
        msg['To'] = MAIL_TO
        msg['Cc'] = MAIL_USER

        msg.set_content(body, subtype='html')

        with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
            smtp.starttls()
            smtp.login(MAIL_USER, MAIL_PASS)
            smtp.send_message(msg)

    except Exception as e:
```

```
        print("Error in Sending Mail\n", str(e))
    else:
        print("Mail Sent!")
```

- **These functions are used to send an email notification with the scraped data. `get_body_subject` reads the CSV file and generates the email subject and body based on the availability of products. `send_mail` sends the email using Gmail's SMTP server.**

```
def main():
    sh.run_spider(KeychronSpider)
    send_mail()


if __name__ == '__main__':
    main()
```

- **This is the main function that executes the script. It first runs the Scrapy spider to scrape data and save it to a CSV file. Then it sends an email notification with the scraped data if products are in stock.**

`import smtplib`
`from config import from_email, password, to`
`from email.message import EmailMessage`
	importing the necessary modules

`def send_mail():`
`msg = EmailMessage()`
`subject = "This is HTML mail 6"`
	The send_mail() function creates an instance of the EmailMessage class
	It sets the subject, sender (From), and recipient (To) of the email using

html_body = ''' This is colorful body. ''' msg["Subject"] = subject msg['From'] = from_email msg['To'] = to	**the variables imported from the config.py file**
msg.set_content(html_body, subtype='html')	**It then sets the content of the email using the set_content() method. The first argument is the HTML code for the email body, and the second argument specifies that the email content is in HTML format.**
# FALLBACK	
msg.add_alternative("This is plain text", subtype='text')	**The add_alternative() method is used to provide a fallback plain-text version of the email for email clients that do not support HTML. The first argument is the plain-text version of the email, and the second argument specifies that it is in plain-text format.**
with smtplib.SMTP("smtp.gmail.com", 587) as smtp:	**connects to SMTP() server of gmail using smtplib.SMTP() method**
smtp.starttls()	**starting secure connection**
smtp.login(from_email, password)	**login with my email and password to access**
smtp.send_message(msg)	**sends the email**

print("sent")	prints a message to the console indicating that the email has been sent.
if __name__ == '__main__': send_mail()	block that calls the send_mail() function when the script is run as the main program.

```
import scrapy

import json

import os

from scrapy.crawler import CrawlerProcess

import pandas as pd

import scraper_helper as sh


CSV_FILE = 'k4US.csv'

# Set Debug to True to get mails even if No stock

DEBUG = False

FULL_SIZE_ONLY = True



class KeychronSpider(scrapy.Spider):

   custom_settings = {

      'FEEDS': {

         CSV_FILE: {

            'format': 'csv',
```

```
            'encoding': 'utf-8',
            'overwrite': True
        },
    },
    'LOG_LEVEL': 'DEBUG' if DEBUG else 'WARN'
}
name = 'keychron'
start_urls = [
    # 'https://www.keychron.com/products/keychron-k4-wireless-mechanical-keyboard-version-2',
    # 'https://www.keychron.com/products/keychron-k2-hot-swappable-wireless-mechanical-keyboard',
    # 'https://www.keychron.com/products/keychron-k2-wireless-mechanical-keyboard',
    # 'https://www.keychron.com/products/keychron-k8-tenkeyless-wireless-mechanical-keyboard',
    'https://www.keychron.com/collections/keyboard/products/keychron-k1-wireless-mechanical-keyboard'

]

def parse(self, response):

    data = json.loads(response.xpath(
        '//script[@type="application/ld+json"]/text()').get()
    )
```

```
        for var in data['offers']:
            if FULL_SIZE_ONLY:
                if not '104-key' in var['name']:
                    continue
            available = True if 'InStock' in var['availability'] else False
            if not available:
                continue
            item = {
                'name': var['name'],
                'available': available,
                'price': var['price'],
                'url': response.url
            }
            yield item



def get_body_subject():
    try:
        df = pd.read_csv(CSV_FILE)
        if df["available"].any():
            subject = "✔ Keychron(US) - Available"
            body = df[df["available"]].to_html()
        else:
            subject = "✖ Keychron(US) - Out of Stock" if DEBUG else None
```

```
            body = df.to_html() if DEBUG else None
        except:
            return None, None

        return body, subject



def send_mail():
    import smtplib
    from email.message import EmailMessage
    try:
        msg = EmailMessage()
        MAIL_USER = 'asdsfsdfs@gmail.com'
        MAIL_PASS = 'asdsfsdfs'
        MAIL_TO = 'asdsfsdfs@gmail.com'
        body, subject = get_body_subject()
        if not (body and subject) and not DEBUG:
            print("Nothing in stock, exiting...")
            return
        msg["Subject"] = subject
        msg['From'] = MAIL_USER
        msg['To'] = MAIL_TO
        msg['Cc'] = MAIL_USER

        msg.set_content(body, subtype='html')
```

```
        with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
            smtp.starttls()
            smtp.login(MAIL_USER, MAIL_PASS)
            smtp.send_message(msg)

    except Exception as e:
        print("Error in Sending Mail\n", str(e))
    else:
        print("Mail Sent!")


def main():
    sh.run_spider(KeychronSpider)
    send_mail()


if __name__ == '__main__':
    main()
```