🧠 UNIVERSAL STATIC WEBSITE SCRAPER

(ENGINEER-LEVEL STRUCTURE)

🟢 STEP 0 — DIMAG ME EK RULE FIX KAR LO

> **Scraper = 5 roles ka system**

1. Network (HTML laana)
2. Parser (HTML → DOM)
3. Extractor (DOM → data)
4. Flow (pagination)
5. Output (save / return)

Ab code isi order me likha jaata hai.

🟢 STEP 1 — BASIC IMPORTS (TOOLS)

```python
import requests
from bs4 import BeautifulSoup
import json
from pathlib import Path
```

Kyun?

- `requests` → browser ka kaam
- `BeautifulSoup` → Elements tab ka kaam
- `json` → data store
- `Path` → file handling clean

🟢 STEP 2 — SCRAPER CLASS (SYSTEM KA NAAM)

```
class UniversalScraper:
```

Kyun class?

- Taaki scraper **reusable** ho
- Taaki code **spread na ho**
- Taaki tum bole sako:

```
scraper = UniversalScraper(...)
```

🟢 STEP 3 — `__init__` (SCRAPER KA DNA) {#-step-3--init-scraper-ka-dna }

```
class UniversalScraper:
    def __init__(self, base_url):
        self.base_url = base_url
```

Hinglish meaning:

- `base_url` = website ka starting point
- Scraper ko yaad rahega **kahan se start karna hai**

Headers (IDENTITY)

```
        self.headers = {
            "User-Agent": "Mozilla/5.0",
            "Accept-Language": "en-US,en;q=0.9",
        }
```

Kyun?

- Website ko bolo: *“Main browser hoon, bot nahi”*
- 403 block se bachav

Data store (memory)

```
        self.data = []
```

Kyun?

- Har page ka data yahin add hoga
- End me yahin se file banegi

🟢 STEP 4 — NETWORK METHOD (HTML LAANA)

```
    def fetch_html(self, url):
        response = requests.get(url, headers=self.headers)
        return response.text
```

Is method ka rule:

❌ parse nahi
❌ extract nahi
❌ pagination nahi

✅ **sirf HTML laana**

> Ye method **browser ka replacement** hai.

🟢 STEP 5 — PARSER METHOD (HTML → TREE)

```
    def parse_html(self, html):
        return BeautifulSoup(html, "lxml")
```

Hinglish:

- HTML ek lamba string hota hai
- BeautifulSoup usko **DOM tree** bana deta hai
- Jaise browser ka *Elements tab*

🟢 STEP 6 — EXTRACTOR METHOD (SABSE IMPORTANT)

```
    def extract_records(self, soup):
        records = soup.find_all("article", class_="product_pod")
```

YAHAN SABSE PEHLE KYA HUA?

- Tumne **parent record identify** kar liya
- 1 parent = 1 item (book, product, quote)

> **Parent hamesha repeat hota hai**

LOOP (RECORD BY RECORD)

```
        for record in records:
```

Ab tum **ek single item** ke andar ho.

🔹 TITLE EXTRACTION (ATTRIBUTE vs TEXT)

```
title = record.find("h3").find("a")["title"]
```

Kyun aise?

HTML:

```
<a title="Full Book Name">Short Name</a>
```

- `.text` → short / truncated
- `["title"]` → **full real data**

> Rule:
> **Important data aksar attribute me hota hai**

🔹 STAR RATING (CLASS KE ANDAR DATA)

```
rating_tag = record.find("p", class_="star-rating")
rating = rating_tag["class"][-1]
```

HTML:

```
<p class="star-rating Three"></p>
```

BeautifulSoup:

```
["star-rating", "Three"]
```

- `[-1]` → actual rating

> Rule:
> **Kabhi kabhi data text me nahi, class ke naam me hota hai**

🔷 PRICE

```
price = record.find("p", class_="price_color").text
```

Simple case:

- Visible text
- `.text` best

🔷 AVAILABILITY (DIRTY TEXT CLEANING)

```
availability = record.find(
    "p", class_="instock availability"
).text.strip()
```

- `.strip()` → extra spaces / newline hatao

🔷 DATA STRUCTURE (ENGINEER WAY)

```
item_data = {
    "title": title,
    "rating": rating,
    "price": price,
    "availability": availability,
}
```

Kyun dict?

- JSON ready
- DB ready
- API ready

🔹 STORE DATA

```
            self.data.append(item_data)
```

Extractor ka kaam yahin khatam.

🟢 STEP 7 — FLOW / PAGINATION METHOD (PROCESS BRAIN)

```
def scrape_all_pages(self):
    current_url = self.base_url
```

Meaning:

- Start yahin se hoga

LOOP (UNKNOWN PAGES)

```
        while True:
```

Kyuki:

- Page count pata nahi
- Last page ka indicator hota hai

FLOW KE STEPS (FIX ORDER)

```
            html = self.fetch_html(current_url)
            soup = self.parse_html(html)
            self.extract_records(soup)
```

> **Golden order (never change):**

1. fetch
2. parse
3. extract

NEXT PAGE CHECK

```
next_button = soup.find("li", class_="next")
if not next_button:
    break
```

Hinglish:

- Agar “Next” nahi mila
- Matlab last page
- Loop band

NEXT URL BUILD

```
next_link = next_button.find("a")["href"]
current_url = self.base_url.rsplit("/", 1)[0] + "/" + next_link
```

Important concept:

- Website relative URL deti hai
- Tumhe full URL banana padta hai

🟢 STEP 8 — OUTPUT METHOD (DATA SAVE)

```
def save_to_json(self, filename="data.json"):
    path = Path(filename)
    with open(path, "w", encoding="utf-8") as f:
        json.dump(self.data, f, indent=4, ensure_ascii=False)
```

Kyun alag method?

- Kal JSON → CSV / DB
- Extractor + flow unchanged

🟢 STEP 9 — SCRAPER RUN KARNA

```python
if __name__ == "__main__":
    scraper = UniversalScraper("https://books.toscrape.com/catalogue/page-1.html")
    scraper.scrape_all_pages()
    scraper.save_to_json("books.json")
```

🧠 AB IS STRUCTURE KO DEKH KE KYA AATA HAI?

Tum ye confidently bol sakte ho:

- ✓ Main static website analyze kar sakta hoon
- ✓ Parent–child identify kar sakta hoon
- ✓ Attribute vs text samajhta hoon
- ✓ Pagination ka flow bana sakta hoon
- ✓ Clean scraper structure likh sakta hoon
- ✓ AI ka code review kar sakta hoon