🧠 PEHLE BIG PICTURE (1 minute me clarity)

Is poore code ka **sirf ek kaam** hai:

> **“Website se HTML lao → usse data nikaalo → sab pages se data ikattha karo”**

Bas.

OOP sirf itna karta hai ki:

- is kaam ko **tukdon me tod deta hai**
- har tukda ek **method** ban jaata hai
- aur sab tukdon ko ek **Scraper object** ke andar band kar deta hai

Ab line by line chalte hain.

🔷 PART 0: IMPORTS (Tools utha rahe ho)

```python
import requests
from bs4 import BeautifulSoup
```

`import requests`

- `requests` = **Python ka browser**
- Ye kaam karta hai:
  - URL pe request bhejna
  - HTML response lana

👉 Jab tum browser me URL daalte ho, wahi kaam `requests.get()` karta hai.

```
from bs4 import BeautifulSoup
```

- `BeautifulSoup` = **HTML ko samajhne ka tool**
- Raw HTML = ek lamba string
- BeautifulSoup us string ko **tree (parent–child)** me badal deta hai

👉 Browser ke "Elements tab" ka Python version samjho.

🔷 PART 1: CLASS KYUN? (OOP ka root)

```
class QuoteScraper:
```

Ye line kya bol rahi hai?

> "Main ek naya type bana raha hoon jiska naam `QuoteScraper` hai"

Real life analogy:

- `int` , `str` Python ke built-in types hain
- Tum apna **custom type** bana rahe ho = *Scraper*

👉 Ab tum bol sakte ho:

```
scraper = QuoteScraper(...)
```

❓ Tumhe khud kya poochna chahiye?

> "Ye class ka **responsibility** kya hai?"

Answer:

> **Quotes wali website scrape karna**

🔷 PART 2: `__init__` (Scraper ka birth) {#-part-2-**init**-scraper-ka-birth }

```
def __init__(self, base_url):
```

`__init__` ka matlab {#**init**-ka-matlab }

- Jab bhi class ka object banta hai
- Ye function **automatic chal jaata hai**

```
scraper = QuoteScraper("https://quotes.toscrape.com/")
```

👆 Ye likhne ke turant baad `__init__` call hota hai.

```
self.base_url = base_url
```

Iska matlab:

- Jo URL tumne diya
- Usko scraper ke andar **store** kar diya

`self` = **current object**

Real life:

> "Is scraper ko yaad rahega kaunsi website scrape karni hai"

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

Headers kyun?

- Websites bots ko block karti hain
- Headers = tum apni **identity** bata rahe ho

👉 Ye bol raha hai:

> "Main ek normal browser jaisa hoon"

❓ Yahan tum kya question poochte ho?

> "Ye cheez har request me same rahegi ya badlegi?"

Answer:

- Same rahegi
  ➡️ isliye `__init__` me rakhi ✓

🔷 PART 3: `fetch_url` (HTML kaun laayega)

```python
def fetch_url(self, url):
```

Is method ka kaam?

> **Sirf ek kaam: HTML laana**

```python
response = requests.get(url, headers=self.headers)
```

Breakdown:

- `requests.get` → HTTP GET request

- `url` → kaunsi page
- `headers` → kaun ho tum

Return hota hai:

- `response` object (poora packet)

```
return response.text
```

- `.text` = **sirf HTML content**
- Baaki cheezein (status code etc.) ignore ki

👉 Clean rule:

> “Ye method sirf HTML dega, kuch aur nahi”

❓ Tum yahan kya poochte ho?

> “Kya ye method parsing kare?”

Answer:

- ❌ Nahi
- Sirf **fetching**

Ye hi **Single Responsibility Principle** hai (OOP ka core).

🔷 PART 4: `parse_html` (HTML ko tree kaun banayega)

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

Yahan kya ho raha hai?

- Raw HTML string aayi
- BeautifulSoup ne usko **DOM tree** bana diya

`"lxml"` :

- Fast parser
- Stable
- Industry standard

❓ Tum yahan kya sochoge?

> "Parsing ka kaam fetch se alag kyun?"

Answer:

- Kal ko HTML file se parse karna ho
- Ya API response parse karna ho

👉 Flexibility.

🔷 PART 5: `extract_quotes` (DATA nikaalna)

```
def extract_quotes(self, soup):
```

Input:

- `soup` = ek page ka parsed DOM

Output:

- list of dictionaries (quotes)

```
quotes = soup.find_all("div", class_="quote")
```

- `find_all` → multiple records
- `"div"` → tag
- `class_="quote"` → meaningful class

👉 **Parent identification**

```
for quote in quotes:
```

Ab tum **ek-ek record** pe kaam kar rahe ho.

```
text = quote.find("span", class_="text").get_text()
```

Breakdown:

- `quote.find(...)` → parent ke andar search
- `"span"` → tag
- `"text"` → class
- `.get_text()` → HTML hata ke content

Same logic:

```
author = quote.find("small", class_="author").get_text()
tags = quote.find_all("a", class_="tag")
```

```
quotes_data.append({
    "text": text,
    "author": author,
    "tags": tag_texts
})
```

Yahan kya ho raha hai?

- Ek quote = ek dictionary
- Saare quotes = list

👉 **Structured data**

❓ Tum yahan kya poochte ho?

> "Kya ye method pagination handle kare?"

Answer:

- ❌ Nahi
- Sirf **ek page ka data**

🔷 PART 6: `scrape_all_pages` (ORCHESTRATOR)

Ye **sabse important method** hai.

```
current_url = self.base_url
all_quotes = []
```

- Start page
- Final collection

```
while True:
```

- Pages ka count pata nahi
- Isliye infinite loop + break condition

```
html = self.fetch_url(current_url)
soup = self.parse_html(html)
page_quotes = self.extract_quotes(soup)
```

Dekho flow:

1. HTML lao ✓
2. Parse karo ✓
3. Data nikaalo ✓

👉 EXACTLY wahi jo tum script me kar rahe the
Bas ab **clean steps** me.

```
next_button = soup.find("li", class_="next")
if next_button is None:
    break
```

- Pagination stop condition
- Last page detect

```
next_link = next_button.find("a")["href"]
current_url = self.base_url + next_link
```

- Relative URL → absolute URL
- Loop continues

```
return all_quotes
```

- Final data return
- Save / print class ke bahar

❓ Tum yahan kya sochoge?

> "Ye method sabko coordinate kyun karta hai?"

Answer:

- Ye **flow controller** hai
- Isi ko "orchestrator" kehte hain

🔷 PART 7: `if __name__ == "__main__"` {#-part-7-if-name--main }

```
if __name__ == "__main__":
```

Iska matlab:

> "Ye code tab hi chale jab file directly run ho"

```
scraper = QuoteScraper("https://quotes.toscrape.com/")
data = scraper.scrape_all_pages()
```

- Object bana
- Kaam start kiya

```
print(f"Total quotes: {len(data)}")
```

- Result verify

🧠 AB SABSE IMPORTANT ANSWER (PLEASE READ)

Tumne poocha:

> "Main ye structure khud kaise sochunga?"

Sachcha answer:

👉 **Tum ye structure kabhi sochoge hi nahi upfront.**

Tum:

1. Pehle **ugly script** likhoge
2. Jab code lamba lagega
3. Jab repeat dikhega
4. Tab brain bolega:
   > "Isko tod dete hain"

Aur wahi se:

- methods
- class
- structure

Structure is compression of experience.

🔒 EK FINAL FRAMEWORK (SAVE THIS)

Har scraper me ye 5 sawal poochna:

1️⃣ Data kaun laayega? → fetch
2️⃣ Data kaise samjhenge? → parse
3️⃣ Record kya hai? → extract
4️⃣ Flow kaise chalega? → paginate
5️⃣ Sabko kaun sambhalega? → class

Bas.