You’ve summarized the **core inspection checklist** perfectly! Your three categories cover 90% of what you need, but let’s refine it into a **step-by-step battle plan** with pro tips for each stage. Here’s the ultimate workflow:

**1. Content Type Inspection** *(Static vs. Dynamic)*

**A. HTML Structure Analysis**

* **Manual Check**:
  + Right-click → **"View Page Source"** (not "Inspect Element").
  + If content exists in raw HTML → **static or server-rendered dynamic**.
  + If only <div id="root"></div> → **client-side rendered (React/Vue)**.

**B. Network Request Profiling**

* **DevTools → Network Tab**:
  + Filter by XHR/Fetch → Look for .json or api/ endpoints.
  + Check Preview to see if responses contain target data.
  + **Pro Tip**: Right-click a request → **"Copy as cURL"** → test in Python.

**C. JavaScript Disable Test**

* **Quick Win**:
  + Disable JS (Chrome: F12 → Ctrl+Shift+P → "Disable JavaScript").
  + Refresh:
    - If content disappears → **fully dynamic**.
    - If content stays → **static or hybrid**.

**2. Data Source Identification**

**A. Direct HTML Scraping**

* **Best for**:
  + Static sites (e.g., blogs, news articles).
  + **Tool**: BeautifulSoup (Python):

python

Copy

soup.select(".product-title") # CSS selectors

**B. Hidden API Extraction**

* **Best for**:
  + SPAs (e.g., React/Angular sites).
  + **Steps**:
    1. Find API calls in **Network Tab** (filter by XHR/Fetch).
    2. Replicate requests in Python:

python

Copy

response = requests.get("https://api.site.com/data?page=2", headers={

"Authorization": "Bearer token\_from\_headers"

})

**C. Pagination Handling**

* **URL Parameters**:
  + ?page=2 → Easy to loop:

python

Copy

for page in range(1, 10):

requests.get(f"https://site.com?page={page}")

* **"Load More" Buttons**:
  + **Selenium/Playwright** to click buttons.
  + **Reverse-engineer** the API (often POST requests with offset values).

**3. Anti-Scraping Measures**

**A. CAPTCHA/IP Block Detection**

* **Manual Test**:
  + Refresh **10x rapidly** → CAPTCHA/block?
  + Use **Incognito Mode** → Less cookie-based blocking.
* **Python Test**:

python

Copy

if "captcha" in response.text or response.status\_code in [403, 429]:

print("🛑 Anti-bot triggered!")

**B. Header/Cookie Requirements**

* **DevTools → Network Tab**:
  + Copy all headers from a successful request (User-Agent, Referer, etc.).
  + Test in Python:

python

Copy

headers = {

"User-Agent": "Mozilla/5.0",

"Cookie": "session\_id=abc123"

}

**4. Bonus: Stealth Checklist**

1. **Rate Limit Avoidance**:
   * Add random delays (time.sleep(random.uniform(1, 3))).
2. **Proxy Rotation**:
   * Use residential proxies (e.g., Luminati, Smartproxy).
3. **Browser Fingerprinting**:
   * Use selenium-stealth or Playwright with chromium.launch(headless=False).

**When to Go Beyond Your Checklist**

* **Login Walls**: Use requests.Session() to persist cookies.
* **WebSockets**: Inspect WS in DevTools for real-time data.
* **Obfuscated APIs**: Decode JavaScript with PyExecJS (for encrypted payloads).