cbsoup: Quick Start (OPT Engine Only)

What is `cbsoup`?

cbsoup is an optimized HTTP scraping engine built on a custom `Session` that:

- Reuses connections (keep-alive) and sensible headers.
- Can optionally respect `robots.txt`.
- Plays nicely with BeautifulSoup (`lxml`) for parsing.
- Includes a ready-to-run benchmark/collector that saves HTML & assets.

Install & Run (terminal)

```
# optional: fresh env
python3 -m venv .venv
source .venv/bin/activate # Windows: .venv\Scripts\activate

pip install -U pip
pip install -r requirements.txt

# run the cbsoup-only benchmark/collector (uses OPT engine)
python bench_scrape.py --iters 5 --warmup 1 --timeout 20 --obey-robots
# (omit --obey-robots to ignore robots.txt)
```

Outputs. Files are saved under:

```
res/opt/<domain>/html/index.html
res/opt/<domain>/assets/*.css|*.js
```

Using the `cbsoup` Engine in Your Code

1) Minimal Fetch

```
from pywebcopy.session import Session

opt = Session()
opt.set_follow_robots_txt(False) # or True to obey robots.txt
opt.headers.update({
    "User-Agent": ("Mozilla/5.0␣(X11;␣Linux␣x86_64)␣"
                   "AppleWebKit/537.36␣(KHTML,␣like␣Gecko)␣"
                   "Chrome/124.0.0.0␣Safari/537.36"),
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Language": "en-US,en;q=0.9",
    "Accept-Encoding": "gzip,␣deflate",
```

```
    "Connection": "keep-alive",
})

resp = opt.get("https://example.com", timeout=15, allow_redirects=True)
resp.raise_for_status()
html_bytes = resp.content
```

2) Parse with BeautifulSoup (optional)

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_bytes, "lxml")
title = (soup.title.string.strip() if soup.title and soup.title.string else "")
links = [a["href"] for a in soup.find_all("a", href=True)]
```

3) Discover & Download Assets (CSS/JS)

```
from urllib.parse import urljoin, urlparse
import os, re

CSS_IMPORT_RE = re.compile(r"@import\s+(?:url\()?['\"]?([^'\")\s]+)", re.I)

def discover_assets(soup, base_url):
    css, js = set(), set()
    for l in soup.find_all("link", href=True):
        rel = l.get("rel") or []
        if isinstance(rel, (list, tuple)) and "stylesheet" in [r.lower() for r in rel]:
            css.add(urljoin(base_url, l["href"]))
    for s in soup.find_all("script", src=True):
        js.add(urljoin(base_url, s["src"]))
    for st in soup.find_all("style"):
        if st.string:
            for m in CSS_IMPORT_RE.finditer(st.string):
                css.add(urljoin(base_url, m.group(1)))
    return css, js

def save_bytes(path, data):
    os.makedirs(os.path.dirname(path), exist_ok=True)
    with open(path, "wb") as f: f.write(data)

# Example usage:
base_url = resp.url
css_urls, js_urls = discover_assets(soup, base_url)

assets_dir = os.path.join("res", "opt", urlparse(base_url).netloc, "assets")
for u in css_urls | js_urls:
    r = opt.get(u, timeout=15, allow_redirects=True)
    if r.ok:
        name = os.path.basename(urlparse(r.url).path) or "asset.txt"
        save_bytes(os.path.join(assets_dir, name), r.content)
```

Controlling `robots.txt`

- `opt.set_follow_robots_txt(True)` to obey robots.
- `opt.set_follow_robots_txt(False)` for benchmarking or controlled tests.

CLI Flags You'll Use Most

- `--url <URL>` : target (can repeat).
- `--iters N` : measured iterations per URL.
- `--warmup N` : unmeasured warmups.
- `--timeout SEC` : per-request timeout.
- `--obey-robots` : obey `robots.txt` (off if omitted).
- `--csv <file.csv>` : write summary CSV.
- `--res-root <dir>` : output root (default `res`).

Tips

- Reuse a single `Session()` for many requests (faster TLS + keep-alive).
- Use realistic headers; many sites return lighter/faster variants for known UAs.
- Add light retry/backoff for `429/503`.