Plik HTML (samochody)

Zawiera: 4 auta (C100..C400), ceny w różnych formatach (62 900, 134,500, 179.990), promocję (old + promo), różne stany dostępności (liczba w nawiasie, out), warianty/wyposażenie (ul.trims), specyfikację (.specs), inline JSON (dla C300), rating gwiazdkowy w klasach, obrazki z data-src lub src, breadcrumbs i paginację.

---

4 mini-ćwiczenia (samodzielne)

Ćw. 1 — Start: podstawy i nawigacja

Cel: zbudować soup, policzyć auta, zebrać kluczowe pola.
Zadania:

1. Wczytaj training_cars.html, zrób soup.
2. Policz li.car.
3. Dla każdego auta wypisz: SKU, Nazwa, Kategoria/segment (data-seg), URL (połącz z bazą https://automarket.example.com).
4. Wypisz breadcrumbs jako listę tekstów.

Wskazówki: soup.select("ul.car-list > li.car"), li.get("data-sku"), li.select_one(".name a")["href"], urllib.parse.urljoin.

Ćw. 2 — Ceny, formaty i dostępność

Cel: poprawnie sparsować ceny i policzyć statystyki.
Zadania:

1. Wyciągnij cenę końcową:
    - jeśli .price.promo — weź promo,
    - w przeciwnym razie .price (pomiń .old).
2. Zamień cenę tekstową na float (obsłuż spacje tysięcy, kropki i przecinki).
3. Wyciągnij dostępność: liczba z nawiasu (5) → int; dla .availability.out lub tekstu „Chwilowo niedostępne” → 0; jeśli brak — None.
4. Policz:
    - średnią cenę aut z dostępnością > 0,
    - ile aut ma stock == 0.

Wskazówki: re.sub żeby usunąć spacje, potem re.search(r"\d+(?:[.,]\d+)?"), zamień , na ..

Ćw. 3 — Promocje i % rabatu

Cel: zidentyfikować auta w promocji i policzyć rabaty.
Zadania:

1. Znajdź li.car:has(.price.promo) (lub iteracyjnie sprawdź obecność .price.promo).
2. Dla każdej karty pobierz old i promo, zamień na float.
3. Policz % rabatu = 100 * (old - promo) / old, zaokrąglij do 1 miejsca.
4. Posortuj malejąco po rabacie i wypisz: SKU | Nazwa | Stara | Promo | Rabat%.

Wskazówki: Uważaj na formaty: 119 900 vs 109 900.

Ćw. 4 — Trims, specs, inline JSON i rating

Cel: zbudować pełny rekord auta z wielu sekcji.
Zadania:

1. Z ul.trims > li zrób listę wariantów: name (tekst), trim (data-trim), code, vin.
2. Z tabeli .specs zrób dict {"Silnik":"...","Skrzynia":"...",...} (może być puste).
3. Z script[type="application/json"].data wczytaj JSON (np. short, img, safety) — jeśli brak, None.
4. Rating: z .star-rating mapuj klasy One..Five na liczby 1..5.
5. Obrazek: preferuj data-src, fallback src; zbuduj pełny URL do obrazka urljoin.
6. Zbuduj finalny rekord: sku, name, url, seg, rating, price_float, price_raw, stock, image, featured (czy li ma klasę featured), trims, specs, meta_json.
7. Zapisz listę rekordów do .json lub .csv i wypisz krótkie podsumowanie (np. liczba aut per segment, średnia cena per segment).

Wskazówki:

- Inline JSON: tag = li.select_one('script[type="application/json"].data'), potem json.loads(tag.string).
- urljoin("https://automarket.example.com", href_or_src) dla linków i obrazków.