# BẢN PROMT cho file code “p1a\_clean10\_ocr\_bctc\_GPT.py”

- Tôi cần trích xuất file scan báo cáo tài chính ra file txt và meta.json.

- Bạn hãy xem kỹ số liệu sai lêch như nào file txt sinh ra và ảnh scan gốc, đặc biệt các giá trị mã số, thuyết minh, số đầu năm, cuối năm, ngoài ra các tên danh mục và số trước danh mục ví dụ I, II, III, IV, 1,2,3,1.1, 1.2.....

- Bạn xem kỹ file code là có thể tối ưu được để cho số chuẩn xác hơn không.

- ngoài ra có thêm phương án là dùng file YAML để nó đọc trích xuất và chọn số liệu tốt hơn.

- YAML bạn đề xuất nguyên tắc nào để trong YAML để có thể tránh lỗi đọc nhầm bảng báo cáo tài chính, ví dụ dựa vào giá trị số đầu năm và cuối năm nó chỉ chênh lệch tối đa là 50%, hoặc là thông thường các danh mục tên tài sản, dự phòng bồi thường, hay nợ phải trả nó sẽ tương ứng với các mã số tiêu chuẩn chung mà mọi báo cáo tài chính đều giống nhau. Tôi gửi bạn file YAML, bạn xem chỉnh sửa và tối ưu cho tôi.

- Bạn hãy cho tôi trong code các thông tin sau để chỉ cần chạy lênh “python -m src.p1a\_clean10\_ocr\_bctc --start 8 --end 8” nó sẽ in ra trang 8, còn chạy lênh python -m src.p1a\_clean10\_ocr\_bctc thì nó sẽ in tất cả các file ở inputs.

**Luồng tổng thể**

1. **Khởi tạo & đọc cấu hình (CLI / master switch)**
   * Parse tham số: --input-root, --out-root, --ocr-engine, --table-engine, --no-gpt, --gpt-table y|n, --gpt-scope, --dpi, v.v. (hàm build\_argparser()).
   * Áp dụng công tắc đầu file (nếu dùng): GPT\_MASTER\_SWITCH (None=theo CLI; True=ép bật GPT; False=ép tắt GPT).
   * In cấu hình ra terminal:
     + [GPT ] enabled=YES/NO | mode=... | model=... | source=CLI/FORCED-ON/FORCED-OFF
     + [GPT.T ] TABLE=ON/OFF (scope=..., enhancer=READY/MISSING)
     + [YAML ] Trạng thái: ĐANG BẬT/ĐANG TẮT (RAW MODE)
2. **Nạp YAML / hoặc chạy RAW**
   * Nếu P1A\_RAW\_MODE=False: nạp yaml\_table và yaml\_text (file \*.yaml), dùng cho clean/prefilter.
   * Nếu P1A\_RAW\_MODE=True: bỏ qua YAML hoàn toàn (cả TEXT lẫn TABLE) → chạy “thô”.
3. **Quét input**
   * Lấy danh sách file hợp lệ trong --input-root (PDF, DOCX, PNG/JPG/TIFF/BMP).
   * Log số lượng và đường dẫn.
4. **Xử lý từng file** — process\_one\_file(...)
   * Tạo đường dẫn mirror ở --out-root (hàm make\_output\_paths(...)) → chuẩn bị 3 output:
     + \*\_text.txt (nội dung hợp nhất theo trang)
     + \*\_meta.json (metadata theo trang)
     + \*\_vector.jsonl (chunk cho vector store)
   * Lặp từng **trang** (hàm iter\_pages(...) → render về **PIL.Image**):
     + Gọi process\_page(...) (quy trình 1 trang) → nhận:
       - btype = "TEXT" hoặc "TABLE" (và có thể có thêm các khối ("TABLE", ...) nếu ROI)
       - block = nội dung pipe/text đã xử lý
       - meta = metadata trang
       - extra\_blocks = list bảng ROI (nếu mixed-page)
5. **Quy trình 1 trang** — process\_page(...)
   * **OCR sơ bộ cả trang** (lấy text + meta cơ bản: đơn vị, kỳ, ngôn ngữ, công ty):
     + ocr\_engine="tesseract" (mặc định) → ocr\_image\_text(...)
     + hoặc "paddle" → ocr\_image\_text\_paddle(...)
   * **Phát hiện ROI bảng** (OpenCV morphology) → \_find\_table\_rois(...).
   * **Nếu có ROI** (mixed page):
     + **TEXT phần còn lại**: che ROI rồi OCR → clean YAML text (nếu bật).
     + **Mỗi ROI**:
       - Xây bảng var-cols (xây theo thực tế của bảng- ko ép 5 cột) theo 2 route:
         * **TSV route**: build\_table\_tsv(...) (Tesseract TSV + KMeans theo trục X).
         * **Paddle route** (nếu bật): paddle\_table\_to\_pipe(...) (PP-Structure đọc HTML table).
       - **Chọn route tốt hơn** (nhiều dòng hơn) → choose\_better\_pipe(...).
       - **Prefilter** rác header/caption mạnh → \_prefilter\_table\_lines(...) (nếu YAML bật).
       - **(E1) GPT cho ROI** *(nếu bật)*:  
         \_maybe\_gpt\_enhance\_table(pipe, pil\_roi, gpt\_flags...)
         * GPT **đọc ảnh ROI** + **đối chiếu pipe** → chỉnh lại pipe **theo ảnh** (không bịa số, giữ var-cols).
         * Log: [GPT TABLE][ON][ROI] ... hoặc [OFF].
     + Trả về: khối **TEXT** làm block chính + các **TABLE (ROI)** trong extra\_blocks.
   * **Nếu trang là bảng toàn trang** (không chia ROI):
     + Dựng bảng var-cols bằng TSV/Paddle như trên.
     + Prefilter (nếu YAML bật).
     + **(E2) GPT cho full page** *(nếu bật)*:  
       \_maybe\_gpt\_enhance\_table(pipe, pil\_full, ...)
       - Log: [GPT TABLE][ON][FULL] ... hoặc [OFF].
     + Trả về: btype="TABLE", block=pipe (đã chỉnh nếu có GPT).
   * **Nếu không có bảng**:
     + Xử lý **TEXT thuần**: clean YAML text nếu bật → trả về btype="TEXT".
6. **Gộp output theo trang** — ngay trong process\_one\_file(...)
   * Append vào \*\_text.txt theo format:
     + ### [PAGE 01] [TEXT] / ### [PAGE 01] [TABLE]
     + Nếu mixed-page: có thêm ### [PAGE 01] [TABLE] cho từng ROI.
   * **Vector store** (\*\_vector.jsonl): ghi mỗi block (TEXT/TABLE) thành **1 chunk** {"content": ..., "metadata": {...}} (đã normalize).
   * **Meta** (\*\_meta.json): tổng hợp meta theo từng page (đơn vị, kỳ, công ty, engine, route, roi\_area\_ratio, gpt\_used…).
7. **Ghi file & log**
   * Ghi \*\_text.txt, \*\_meta.json, \*\_vector.jsonl.
   * Nếu --split-debug: ghi thêm \*\_TEXT.txt và \*\_TABLE.txt.
   * In preview, cảnh báo, và tổng kết đường dẫn output.

**Vai trò & công tắc GPT**

* **GPT chạy ở bước E1 / E2 (chỉ bảng)**:
  + E1: khi có **ROI** (trang mixed) → GPT xử từng ROI.
  + E2: khi **bảng toàn trang** → GPT xử cả trang.
* **Không dùng GPT cho TEXT** (văn bản thuần).
* Bật/tắt:
  + Toàn cục: --no-gpt hoặc GPT\_MASTER\_SWITCH (ép bật/tắt mọi chỗ GPT).
  + Riêng bước bảng: --gpt-table y|n (và --gpt-scope phải cho phép bảng).
* Terminal hiển thị rõ:  
  enabled=YES/NO, TABLE=ON/OFF, enhancer=READY/MISSING,  
  và khi xử lý: [GPT TABLE][ON/OFF][ROI/FULL] ....

**Tóm tắt đầu vào/đầu ra**

* **Input**: thư mục tài liệu (PDF/DOCX/ảnh), YAML (nếu dùng), tham số CLI.
* **Per page**: PIL.Image → OCR → detect ROI → build table (TSV/Paddle) → (GPT chỉnh TABLE nếu bật) → clean/prefilter → block.
* **Output** (mỗi file):
  + \*\_text.txt: nội dung từng trang (TEXT/TABLE).
  + \*\_meta.json: metadata đầy đủ theo trang/ROI.
  + \*\_vector.jsonl: các chunk chuẩn hoá cho indexing.
  + \*\_TEXT.txt / \*\_TABLE.txt (nếu --split-debug).

- mỗi đoạn code bạn cho tôi thì cần nói rõ vị trí thay thế/thêm vào 🡪 sau dòng nào và trên dòng nào.

Chuẩn — không nên “cứng” 5 cột cho mọi bảng. Mình đề xuất một **schema lõi linh hoạt** + **bộ nhận diện loại bảng** để bạn chuẩn hoá được nhiều dạng (B01, B02, B03, thuyết minh, bảng thường…) mà vẫn QA số ổn định.

**1) Thiết kế schema tổng quát (dùng cho mọi bảng)**

Mỗi dòng (row) xuất ra với **các cột lõi**; cột không áp dụng thì để rỗng:

* TABLE\_ID — định danh bảng (ví dụ: B01-DNPNT, B02-DNLCTT, B03-DNKQKD, NOTE, GENERIC)
* SECTION — tiêu đề/khối con trong bảng (nếu có), ví dụ “A. Tài sản ngắn hạn”
* CODE — mã số chỉ tiêu (nếu có)
* NAME — tên chỉ tiêu
* NOTE — số tham chiếu thuyết minh (nếu có)
* END — Số **kỳ hiện tại** (Số cuối năm/kỳ)
* BEGIN — Số **kỳ so sánh** (Số đầu năm/kỳ)
* UNIT — “VND”, “nghìn VND”, “triệu VND”… (chuẩn hoá thêm UNIT\_MULTIPLIER nếu muốn)
* CURRENCY — ISO nếu nhận diện được (VD: “VND”)
* PERIOD — ngày kỳ báo cáo (VD: 2024-12-31)
* SOURCE\_PAGE — trang gốc (VD: 8)
* RAW — dòng gốc (để audit truy vết)

Với bảng **không có CODE/END/BEGIN** (bảng thường), vẫn dùng schema này, chỉ điền NAME, RAW, có thể thêm các cột phụ COL1..COLN ở phần “GENERIC fallback” phía dưới.

**2) Luồng nhận diện loại bảng**

* Nếu văn bản chứa các từ khoá → gán TABLE\_ID:
  + B01-DNPNT|BẢNG CÂN ĐỐI → B01
  + B02-DNLCTT|LƯU CHUYỂN TIỀN → B02
  + B03-DNKQKD|KẾT QUẢ HOẠT ĐỘNG → B03
  + THUYẾT MINH|THUYẾT → NOTE
  + Mặc định → GENERIC
* Phát hiện UNIT/CURRENCY từ dòng như “Đơn vị tính: …”
* Phát hiện PERIOD từ “ngày 31 tháng 12 năm 2024” (regex ngày).

**3) Drop-in code: bộ chuẩn hoá linh hoạt**

**3.1. Helpers (clean số, nhận diện đơn vị, kỳ, bảng)**