SAP data analytics master class OCR example

1st April 2026 – 9AM EST

Easy OCR – using python to read images

WHAT WILL WE COVER TODAY?

01

Easy OCR – using python to read images
Applications

02

More than one way to skin a cat!
Different ways to do the same thing

03

Easy OCR – using python to read images
Let´s read an image!


01
Easy OCR – using python to read images

Easy OCR – using python to read images

20260401_01- Why read images

1. How would you rate the usefulness of reading images for audit? (Matching)

A. Automate review of SOX evidence (ITGC screenshots, etc)	A.	Select an option
B. Compare evidence to SAP data (comparison of optical supplier invoices to supplier invoice data)	B.	Select an option
C. Check for evidence entered incorrectly (check for travel and expense entered for non-employees)	C.	Select an option

Easy OCR – T&E example: HR03_14

FIFA Switzerland – Travel Expenses used for non-employees

- Federation Internationale de Football Association
- Expenses charged for third-parties including:
  - Family members
  - Political associates
  - External consultants and intermediaries
- These costs were booked as:
  - Official FIFA travel
  - Committee-related business trips
- Result
  - Criminal investigations
  - Lifetime bans
  - Governance overhaul

HR03_14: Travel & Expense for non-employees

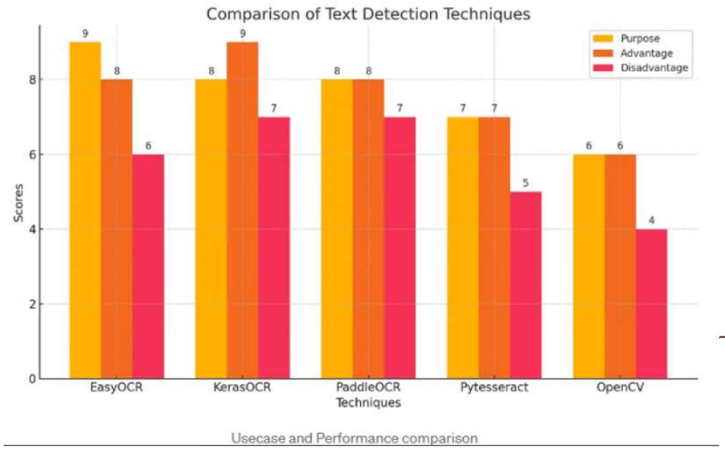
Travel and expense receipt

- Load the receipt into the travel and expense program
- Use the travel and expense program to read the name on the receipt
- Compare the name on the receipt to the name for the employee
  - Use Levenschtein distance or a simple Artificial Intelligence library to check for similar match
- If the name does not match, then you can sample the expenses for review


02
More than one way to skin a cat!

Different ways to do OCR

https://medium.com/@shah.vansh132/comparison-of-text-detection-techniques-easyocr-vs-kerasocr-vs-paddleocr-vs-pytesseract-vs-opencv-44c2bc22b133



Performance Evaluation

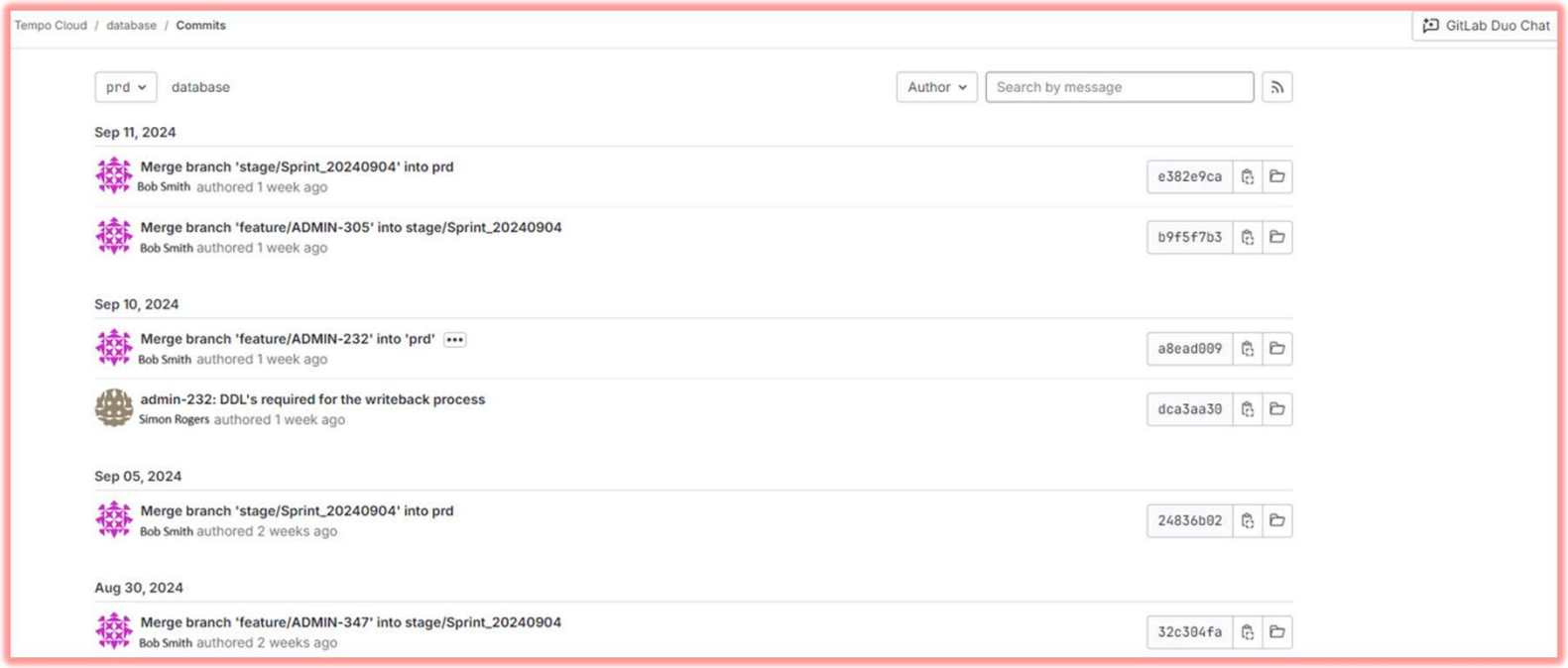
Technique	Purpose	Advantage	Drawback
EasyOCR	General-purpose OCR for detecting and recognizing text in images.	High accuracy with support for multiple languages and complex scripts.	Slower processing compared to other OCR tools due to deep learning models.
KerasOCR	Text detection and recognition using deep learning.	Combines text detection and recognition in a single pipeline with high accuracy.	Requires a deep learning framework and is computationally intensive.
PaddleOCR	High-performance, multilingual OCR based on deep learning.	Supports a wide range of languages and scripts, optimized for accuracy and speed.	Complex setup and requires significant computational resources.
Pytesseract	OCR engine for extracting text from images, using Tesseract.	Easy to use with support for multiple languages, lightweight and fast.	Less accurate with complex backgrounds or low-quality images.
OpenCV	General-purpose computer vision library with basic OCR capabilities.	Fast and lightweight, ideal for preprocessing and simple text detection tasks.	Lower accuracy for text recognition, especially with complex backgrounds.

03

OCR code example

OCR code example – read this file

Image example – for SOX ITGC change management evidence



OCR code example

Install libraries

STEP 0: Install and Import Required Libraries

```
# ========================
# STEP 0: Install and Import Required Libraries
# ========================
# Run these commands in terminal if libraries are not installed:
# pip install easyocr

# PI_OS      -> For file and folder operations
# PI_EASYOCR -> EasyOCR engine for text recognition
# PI_RE      -> Regular expressions for pattern matching
# PI_DATETIME-> Date and time handling


import pandas as PI_PANDAS
import easyocr as PI_EASYOCR
import re as PI_RE
from datetime import datetime as PI_DATETIME
```

Easy OCR – T&E example: HR03_14

Initialize OCR and read image

Easy OCR object

Everything in python is an object.
An object has:
- A type (int, str, list, other)
- Data (the values it holds)
- Methods/ behaviour: things it can do

STEP 1: Initialize OCR and Read Image

```
# ========================
# STEP 1: Initialize OCR and Read Image
# ========================
# PI_OB_READER       -> EasyOCR engine instance
# PI_ST_IMAGE_PATH   -> Path to the image file
# ZV_LI_OCR_RESULTS -> List of OCR results: [(bbox, text, confidence)]

# Step 1.1/ Initialize EasyOCR reader
# 'en' -> English language
# gpu=False -> use CPU (set gpu=True if you have a GPU for faster processing)
PI_OB_READER = PI_EASYOCR.Reader(['en'], gpu=False)
print("[...] EasyOCR reader initialized.")

# Step 1.2/ Set image path (replace with your image file)
PI_ST_IMAGE_PATH = r"Images\Gitlab image1.png"  # raw string to avoid issues w
```

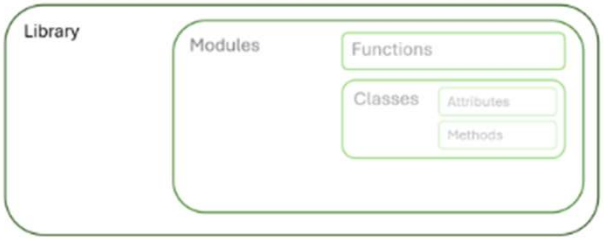
Note: here we don´t put the path at the top of the script or call it ZV_ST_IMAGE_PATH… which is not strictly our naming convention

Pause on objects – added to D01_L04_S4

Tip:

*Library: contains modules; **Module**: contains functions and classes. **Function**(): do an action.*
*Class: create an object. **Object**(): has attributes and methods. **Method**(): act on an object.*

What is a library, module, function, class, attribute and method?



- A library contains modules
- Modules contain functions and classes
- Functions are used to do actions
- Classes are used to create objects
- Objects have attributes and methods
- Methods do actions on objects

Pause on objects – added to D01_L04_S4

Example class, written with 300Framework naming convention

Example of a class:

Create a bank account, that mentions the owner and the balance. Allow the owner to deposit and withdraw money.

```
Class CL_BANK_ACCOUNT:
        def __init__(self, ZVCLI_ST_OWNER, ZVCLI_NU_BALANCE):
                self.ZV_ST_OWNER = ZVCLI_ST_OWNER
                self. ZV_NU_BALANCE = ZVCLI_NU_BALANCE

        def ME_DEPOSIT(self, ZVMEI_NU_AMOUNT):
                self. ZV_NU_BALANCE = self. ZV_NU_BALANCE + ZVMEI_NU_AMOUNT

        def ME_WITHDRAW(self, ZVMEI_NU_AMOUNT):
                if ZVMEI_NU_AMOUNT <= self. ZV_NU_BALANCE:
                        self. ZV_NU_BALANCE = self. ZV_NU_BALANCE – ZVMEI_NU_AMOUNT
                else:
                        print('not enough money')
```

Class CL_BANK_ACCOUNT will create an object (self), that is initiated (__init__) with the two variables ZVCLI_ST_OWNER and ZVCLI_NU_BALANCE. During the initiation, the attributes: ZV_ST_OWNER and ZV_NU_BALANCE will be set. The class has two methods: ME_DEPOSIT and ME_WITHDRAW.

- Naming convention used here:
  - CL: Class
  - ZVCLI: Variable for class incoming
  - ZV_ST_: Variable, type string
  - ZV_NU_: Variable type numeric
  - ME_: Method

Easy OCR – T&E example: HR03_14

Here we use the object to apply the readtext() method to the image. detail =1 means obtain the co-ordinates around each word or group of words.

```
# Step 1/ Run OCR
# detail=1 -> return full info: (bbox, text, confidence)
# mag_ratio=2.0 -> zoom the image by 2x before OCR
# Note You can adjust it down to 1.0 or 1.5 if 2.0 is running too slowly.
#   - This makes small or blurry text easier to recognize
#   - 1.0 = original size, 2.0 = double size
#   - Larger values improve accuracy for small fonts but increase processing time
ZV_LI_OCR_RESULTS = PI_OB_READER.readtext(
    PI_ST_IMAGE_PATH,
    detail=1,
    mag_ratio=2.0
)
```

mag_ratio = 2.0: this is used to increase the size of the image and it makes the reading of the image more accurate

The ZV_LI_OCR_RESUTS will contain a list of Tuples. Inside each Tuple, we see co-ordinates. These are the co-ordinates around the text. Then we see the words. And then we see the score, of how sure the program is that the words it found are accurate (because sometimes the image could be blurry and difficult to make-out).

```
reader.readtext('image.png', detail=0)
# ['GitLab', 'Sign in', 'Username']
```

```
reader.readtext('image.png', detail=1)
# [
# ([[x1,y1],[x2,y2],[x3,y3],[x4,y4]], 'GitLab', 0.99),
# ([[...]], 'Sign in', 0.95)
# ]
```

Easy OCR – T&E example: HR03_14

Here, we are just printing the text of the box, just for information.

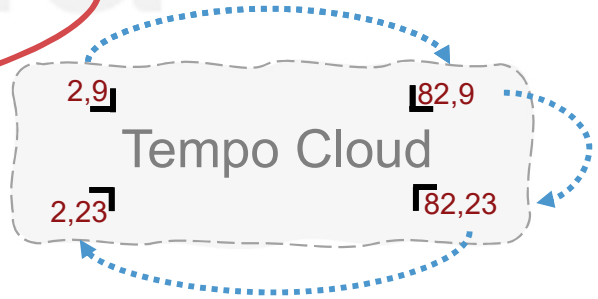
We use the for loop to go around each item in the list and print out the contents.

```
# Step 1.4/ Show raw OCR results
for ZV_DI_ITEM in ZV_LI_OCR_RESULTS:
    ZV_LI_BBOX, ZV_ST_TEXT, ZV_NU_CONF = ZV_DI_ITEM

    # Bounding box (xbox) -> 4 points [top-left, top-right, bottom-right, b
    # Each point is (x, y) coordinate on the image
    # Example: [[100,50],[200,50],[200,100],[100,100]]
    # You can use these coordinates to draw rectangles around text or locat
    #    (100,50)       (200,50)
    #
    #
    #     |   Text    |
    #
    #
    # (100,100)     (200,100)

    print(f"Text: {ZV_ST_TEXT}")              # Original OCR text
    print(f"Bounding Box (XBox): {ZV_LI_BBOX}")  # XBox coordinates
    print(f"Confidence: {ZV_NU_CONF:.2f}")    # OCR confidence (0~1)
    print("-" * 40)
```

```
Using CPU. Note: This module is much faster with a GPU.
✅ EasyOCR reader initialized.
Text: Tempo Cloud
Bounding Box (XBox): [[2, 9], [82, 9], [82, 23], [2, 23]]
Confidence: 0.93
----------------------------------------
Text: database
Bounding Box (XBox): [[98, 9], [155, 9], [155, 23], [98, 23]]
Confidence: 0.92
----------------------------------------
Text: Commits
Bounding Box (XBox): [[172, 9], [227, 9], [227, 23], [172, 23]]
Confidence: 0.89
----------------------------------------
Text: GitLab Duo Chat
Bounding Box (XBox): [[1494, 9], [1607, 9], [1607, 23], [1494, 23]]
Confidence: 0.50
----------------------------------------
Text: prd
Bounding Box (XBox): [[97, 72], [128, 72], [128, 90], [97, 90]]
Confidence: 1.00
----------------------------------------
Text: database
Bounding Box (XBox): [[167, 72], [231, 72], [231, 88], [167, 88]]
Confidence: 1.00
----------------------------------------
...
Text: prd
Bounding Box (XBox): [[435.719631200671, 514.4635574408053], [464.6538530041391,
Confidence: 1.00
----------------------------------------
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



The output to the screen from the above is just for information

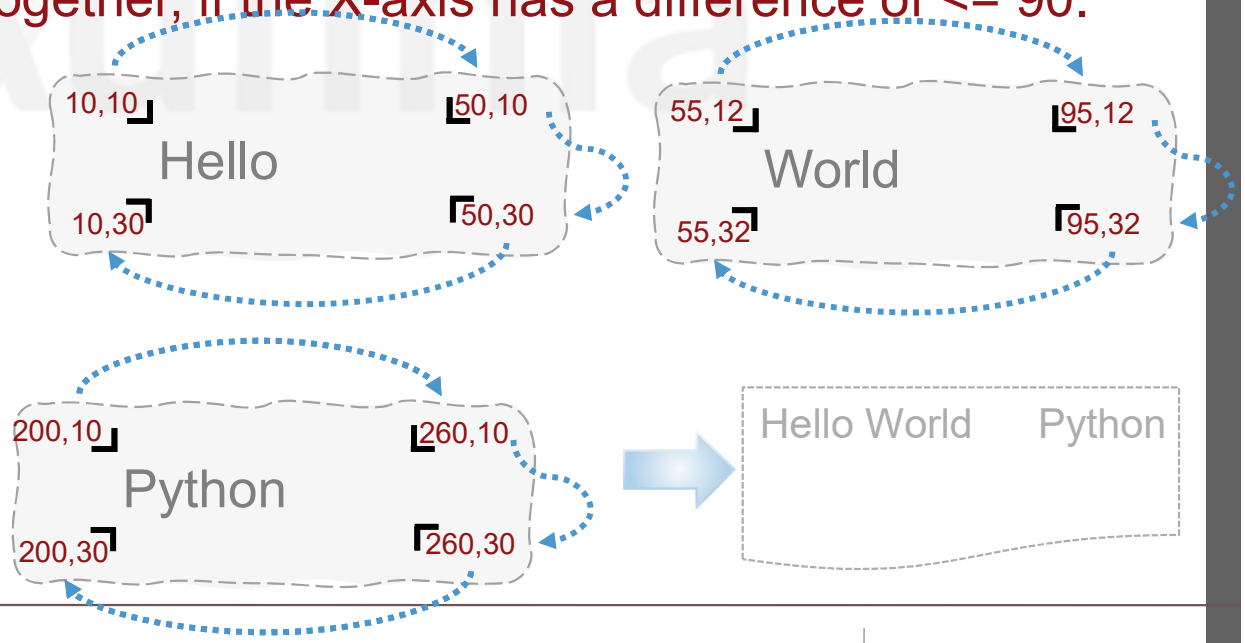
Easy OCR – T&E example: HR03_14

Organizing the boxes into text on lines….

Text is considered to be on the same line if the Y-axis has a difference of <=15.

Text is considered to be read together, if the X-axis has a difference of <= 90.

```
# Suppose OCR outputs three text blocks:
# 1. "Hello"  -> Bbox: [[10, 10], [50, 10], [50, 30], [10, 30]]
# 2. "World"  -> Bbox: [[55, 12], [95, 12], [95, 32], [55, 32]]
# 3. "Python" -> Bbox: [[200, 10], [260, 10], [260, 30], [200, 30]]
#
# STEP 1: VERTICAL GROUPING (Y-axis)
# - "Hello" (Y:10), "World" (Y:12), "Python" (Y:10)
# - All Y-differences <= 15 -> They are all on the SAME LINE.
#
# STEP 2: HORIZONTAL MERGING (X-axis)
# - Gap "Hello" -> "World": 55 (X-left) - 50 (X-right) = 5
#   (5 <= 90 -> MERGE into "Hello World")
#
# - Gap "World" -> "Python": 200 (X-left) - 95 (X-right) = 105
#   (105 > 90 -> NEW SEGMENT)
#
# FINAL STRUCTURE:
# Line 1:
#   - Segment 1: "Hello World" (X: 10 -> 95)
#   - Segment 2: "Python"      (X: 200 -> 260)
# ================================================================
```



Easy OCR – T&E example: HR03_14

Organizing the boxes into text on lines…. continued….

Here we create a list of dictionaries (ZV_LI_LINES). Each dictionary is for one line and contains all the segments that start at a similar y position (top-y).

```
{
    "y_top": 120,
    "items": [
        {"text": "Merge", "x_left": 85, "x_right": 130},
        {"text": "branch", "x_left": 140, "x_right": 210},
        {"text": "'stage/Sprint_20240904'", "x_left": 220, "x_right": 430},
        {"text": "into", "x_left": 440, "x_right": 480},
        {"text": "prd", "x_left": 490, "x_right": 530}
    ]
}
```

```
def FC_OCR_MERGE_LINES_WITH_SEGMENTS(ZVFCI_LI_OCR_RESULTS, ZVFCI_NU_Y_THRESHO
    """
    Merge OCR text blocks into lines and segments based on position.

    - Each OCR block has bounding box, text, confidence.
    - Lines are grouped by vertical alignment (Y coordinate).
    - Segments within a line are merged if horizontal gap is small.
    """

    ZV_LI_LINES = []

    # Step 1: Group text blocks by vertical alignment (Y-top)
    for ZV_LI_BBOX, ZV_ST_TEXT, ZV_NU_CONF in ZVFCI_LI_OCR_RESULTS:
        # Get top-left and top-right positions
        ZV_NU_Y_TOP = min(ZV_LI_POINT[1] for ZV_LI_POINT in ZV_LI_BBOX)
        ZV_NU_X_LEFT = min(ZV_LI_POINT[0] for ZV_LI_POINT in ZV_LI_BBOX)
        ZV_NU_X_RIGHT = max(ZV_LI_POINT[0] for ZV_LI_POINT in ZV_LI_BBOX)

        # Clean text (optional, can remove extra spaces)
        ZV_ST_TEXT = ZV_ST_TEXT.strip()

        # Try to match this block to existing line
        for ZV_DI_LINE in ZV_LI_LINES:
            if abs(ZV_DI_LINE["y_top"] - ZV_NU_Y_TOP) < ZVFCI_NU_Y_THRESHOLD:
                ZV_DI_LINE["items"].append({
                    "text": ZV_ST_TEXT,
                    "x_left": ZV_NU_X_LEFT,
                    "x_right": ZV_NU_X_RIGHT
                })
                break
        else:
```

Easy OCR – T&E example: HR03_14

Organizing the boxes into text on lines…. continued….

ZV_LI_LINES

```
{
    "y_top": 120,
    "items": [
        {"text": "Merge", "x_left": 85, "x_right": 130},
        {"text": "branch", "x_left": 140, "x_right": 210},
        {"text": "'stage/Sprint_20240904'", "x_left": 220, "x_right": 430},
        {"text": "into", "x_left": 440, "x_right": 480},
        {"text": "prd", "x_left": 490, "x_right": 530}
    ]
},
{
    "y_top": 150,
    "items": [
        {"text": "Ed", "x_left": 85, "x_right": 105},
        {"text": "Hillel", "x_left": 110, "x_right": 150},
        {"text": "authored", "x_left": 160, "x_right": 220},
        {"text": "1 week ago", "x_left": 230, "x_right": 310},
        {"text": "e382e9ca", "x_left": 1030, "x_right": 1100}
    ]
}
```

ZV_LI_MERGED

```
[
    {
        "y_top": 120,
        "segments": [
            {
                "text": "Merge branch 'stage/Sprint_20240904' into prd",
                "x_left": 85,
                "x_right": 530
            }
        ]
    },
    {
        "y_top": 150,
        "segments": [
            {
                "text": "Ed Hillel authored 1 week ago",
                "x_left": 85,
                "x_right": 310
            },
            {
```

```python
# Step 2: Sort lines from top to bottom
ZV_LI_LINES.sort(key=lambda ZV_DI_L: ZV_DI_L["y_top"])

ZV_LI_MERGED = []

# Step 3: Merge items horizontally within each line
for ZV_DI_LINE in ZV_LI_LINES:
    # Sort items from left to right
    ZV_LI_ITEMS = sorted(ZV_DI_LINE["items"], key=lambda ZV_DI_I: ZV_DI_I["x_left"])
    ZV_LI_SEGMENTS = []

    if not ZV_LI_ITEMS:
        continue

    ZV_DI_CURRENT = ZV_LI_ITEMS[0]

    # Merge items based on horizontal gap
    for ZV_DI_ITEM in ZV_LI_ITEMS[1:]:
        if ZV_DI_ITEM["x_left"] - ZV_DI_CURRENT["x_right"] > ZVFCI_NU_X_GAP_THRESHOLD:
            # Gap too large -> start new segment
            ZV_LI_SEGMENTS.append(ZV_DI_CURRENT)
            ZV_DI_CURRENT = ZV_DI_ITEM
        else:
            # Merge current segment
            ZV_DI_CURRENT = {
                "text": ZV_DI_CURRENT["text"] + " " + ZV_DI_ITEM["text"],
                "x_left": ZV_DI_CURRENT["x_left"],
                "x_right": ZV_DI_ITEM["x_right"]
            }

    # Add last segment
    ZV_LI_SEGMENTS.append(ZV_DI_CURRENT)

    # Add merged line
    ZV_LI_MERGED.append({
        "y_top": ZV_DI_LINE["y_top"],
        "segments": ZV_LI_SEGMENTS
    })

return ZV_LI_MERGED
```

Easy OCR – T&E example: HR03_14

Organizing the boxes into text on lines…. continued….

Here we have printed out the contents of ZV_LI_MERGED to the terminal… for understanding purposes

```
C:\300FMASTERCLASS\06_PYTHON_MEETING_MC202601\12_PYTHON_MEETING_12\EASY_OCR_EXAMPLE\venv\L
torch\utils\data\dataloader.py:775: UserWarning: 'pin_memory' argument is set as true but
 found, then device pinned memory won't be used. …
Line Y-top: 396
  Segment: admin-232: DDL's required for the writeback process  (X: 134->496)
  Segment: dca3aa3o  (X: 1194->1264)
---------------------------------------
Line Y-top: 418
  Segment: Simon Rogers authored 1 week ag0  (X: 132->351)
---------------------------------------
Line Y-top: 477
  Segment: Sep 05,2024  (X: 86->180)
---------------------------------------
Line Y-top: 512
  Segment: Merge branch 'stagelSprint_20240904 into prd  (X: 134->465.280368799329)
  Segment: 24836b02  (X: 1193->1264)
---------------------------------------
Line Y-top: 534
  Segment: Bob Smith authored 2 weeks ag0  (X: 134->342)
---------------------------------------
Line Y-top: 593
  Segment: Aug 30,2024  (X: 87->182)
---------------------------------------
Line Y-top: 628
  Segment: Merge branch 'feature/ADMIN-347' into stage/Sprint_20240904  (X: 134->576)
  Segment: 32c304fa  (X: 1193->1264)
---------------------------------------
Line Y-top: 650
  Segment: Bob Smith authored 2 weeks ag0  (X: 132->343)
---------------------------------------
PS C:\300FMASTERCLASS\06_PYTHON_MEETING_MC202601\12_PYTHON_MEETING_12\EASY_OCR_EXAMPLE>
```

Easy OCR – T&E example: HR03_14

Extract dates, titles, authors

Here we are obtaining information from the file, because we may want to put it into a structured data set

A regex object for matching date patterns

A string for matching key words

```
# ========================
# Step 3: Extract Dates, Titles, and Authors from Merged Lines
# ========================
# Logic:
# 1. Merge all segments in a line into a single string.
# 2. Check if the line is a metadata line (contains "authored").
#    - If yes, extract author name.
#    - Assign the previous line as the title.
#    - Normalize ADMIN/TEMPO codes in the title.
#    - Assign the nearest date above the title from detected dates.
# 3. If the line is not metadata, search for date strings.
#    - Save detected dates for later matching with titles/authors.
# 4. Append each extracted record (Date / Title / Author) to results.
# 5. Finally, create a DataFrame and extract ADMIN/TEMPO ticket numbers.

# 1/ Date regex to detect lines containing dates
ZV_OB_DATE_REGEX = PI_RE.compile(
    r'(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[^0-9]*\d{1,2}[^0-9]*\
    PI_RE.I
)

# 2/ ADMIN/TEMPO code pattern
ZV_ST_ADMIN_CODE_PATTERN = 'ADMIN|TEMPO'

# ------------------------
# Helper functions
# ------------------------
```

Easy OCR – T&E example: HR03_14

Helper functions for getting bits of information from the image

```python
# ------------------------
# Helper functions
# ------------------------

def FC_TEXT_CHECK_METADATA_LINE(ZVFCI_ST_TEXT):
    """Return True if line contains 'authored' (metadata line with author info
    return 'authored' in ZVFCI_ST_TEXT.lower()

def FC_TEXT_EXTRACT_AUTHOR(ZVFCI_ST_TEXT):
    """Extract the author name from a metadata line (before 'authored')"""
    ZV_ST_CLEANED = ZVFCI_ST_TEXT.strip()
    match = PI_RE.search(r'^(.*?)(?=authored)', ZV_ST_CLEANED, PI_RE.I)
    return match.group(1).strip() if match else "Unknown"

def FC_TEXT_STRIP_UI_INDEX(ZVFCI_ST_TITLE):
    """Remove numbering or UI symbols from the title line"""
    return PI_RE.sub(r'^[\s#\-\–\—_:·\d]+', '', ZVFCI_ST_TITLE).strip()

def FC_TEXT_NORMALIZE_ADMIN_CODE(ZVFCI_ST_TEXT):
    """Normalize ADMIN/TEMPO codes (e.g., 'ADMIN -123' -> 'ADMIN-123')"""
    return PI_RE.sub(
        rf'\b({ZV_ST_ADMIN_CODE_PATTERN})\s*-\s*(\d+)\b',
        r'\1-\2',
        ZVFCI_ST_TEXT,
        flags=PI_RE.I
    )
```

```python
def FC_DATE_NORMALIZE(ZVFCI_ST_TEXT):
    """Convert detected date string into YYYY-MM-DD format"""
    if not ZVFCI_ST_TEXT:
        return None
    match = PI_RE.search(r'([a-zA-Z]{3})[^0-9]*(\d{1,2})[^0-9]*(\d{4})', ZVFC
    if not match:
        return None
    month, day, year = match.group(1).title(), match.group(2).zfill(2), match
    try:
        dt = PI_DATETIME.strptime(f"{month} {day} {year}", "%b %d %Y")
        return dt.strftime("%Y-%m-%d")
    except:
        return None
```

Easy OCR – T&E example: HR03_14

Now we can use our variables and helper functions in order to extract data from the image into a dataframe

```
# ========================
# Process merged lines
# ========================

ZV_LI_DATES_HEADER = []  # Store detected dates with their Y positions
ZV_LI_RESULTS = []       # Store final extracted records

for idx, ZV_DI_LINE in enumerate(ZV_LI_MERGED_LINES):
    # Merge all segments in the line into a single string
    ZV_ST_FULL_LINE = " ".join(seg['text'] for seg in ZV_DI_LINE['segments'])

    # ----------------------------
    # Step 3a: Metadata line (contains author info)
    # ----------------------------
    if FC_TEXT_CHECK_METADATA_LINE(ZV_ST_FULL_LINE):
        ZV_ST_AUTHOR_NAME = FC_TEXT_EXTRACT_AUTHOR(ZV_ST_FULL_LINE)
        ZV_ST_ASSIGNED_DATE = "N/A"
        ZV_ST_CLEAN_TITLE = "N/A"

        # Take the previous line as the title
        if idx > 0:
            ZV_DI_TITLE_LINE = ZV_LI_MERGED_LINES[idx - 1]
            ZV_NU_TITLE_Y = ZV_DI_TITLE_LINE["y_top"]

            # Assign the nearest date above the title
            for date_item in reversed(ZV_LI_DATES_HEADER):
                if date_item["y"] <= ZV_NU_TITLE_Y:
                    ZV_ST_ASSIGNED_DATE = date_item["date"]
                    break

            # Clean and normalize the title
            ZV_ST_RAW_TITLE = ZV_DI_TITLE_LINE["segments"][0]["text"]
            ZV_ST_CLEAN_TITLE = FC_TEXT_NORMALIZE_ADMIN_CODE(
                FC_TEXT_STRIP_UI_INDEX(ZV_ST_RAW_TITLE)
            )
```

```
        # Append final record
        ZV_LI_RESULTS.append({
            "Date": ZV_ST_ASSIGNED_DATE,
            "Title": ZV_ST_CLEAN_TITLE,
            "Author": ZV_ST_AUTHOR_NAME
        })

    # ----------------------------
    # Step 3b: Non-metadata line (check for date)
    # ----------------------------
    else:
        match = ZV_OB_DATE_REGEX.search(ZV_ST_FULL_LINE)
        if match:
            # Save detected date for later matching with titles
            ZV_LI_DATES_HEADER.append({
                "date": FC_DATE_NORMALIZE(match.group(0)),
                "y": ZV_DI_LINE["y_top"]
            })
```

Easy OCR – T&E example: HR03_14

Finally, we can print out the dataframe that we got from the program:

```
# ========================
# Step 4: Create DataFrame
# ========================

ZV_DF_RESULT = PI_PANDAS.DataFrame(ZV_LI_RESULTS)

def FC_EXTRACT_TICKET(ZVFCI_ST_TITLE):
    """Extract ADMIN/TEMPO ticket number from title if exists"""
    match = PI_RE.search(rf'\b({ZV_ST_ADMIN_CODE_PATTERN})-\d+\b', ZVFCI_ST_TITLE, PI_RE.I)
    return match.group(0).upper() if match else None

if not ZV_DF_RESULT.empty:
    ZV_DF_RESULT["Title"] = ZV_DF_RESULT["Title"].astype(str).str.strip()
    ZV_DF_RESULT["ADMIN/TEMPO number"] = ZV_DF_RESULT["Title"].apply(FC_EXTRACT_TICKET)

# Show DataFrame with full content
PI_PANDAS.set_option('display.max_colwidth', None)
print(ZV_DF_RESULT)
```

```
----------------------------------------
        Date                                                        Title        Author ADMIN/TEMPO number
0  2024-09-11                 Merge branch 'stagelSprint_20240904' into prd     Bob Smith                NaN
1  2024-09-11  Merge branch 'feature/ADMIN-305' into stage/Sprint_20240904     Bob Smith          ADMIN-305
2  2024-09-10                   Merge branch 'feature/ADMIN-232' into 'prd'     Bob Smith          ADMIN-232
3  2024-09-10           admin-232: DDL's required for the writeback process  Simon Rogers          ADMIN-232
4  2024-09-05                 Merge branch 'stagelSprint_20240904 into prd     Bob Smith                NaN
5  2024-08-30  Merge branch 'feature/ADMIN-347' into stage/Sprint_20240904     Bob Smith          ADMIN-347
```


Questions?