SAP data analytics master class OCR example

1st July 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

20260701: why read images?

1. **Reading images is useful for automating the review of SOX evidence (ITGC, screenshots, etc)** (Rating scale)

0: Not useful, 10: Very useful

0 1 2 3 4 5 6 7 8 9 10

2. **Reading images is useful to compare evidence to SAP data (comparison of optical supplier invoices to supplier invoice data)** (Rating scale)

0: Not useful, 10: Very useful

0 1 2 3 4 5 6 7 8 9 10

3. **Reading images is useful to check for evidence entered incorrectly (check for travel and expense entered for non-employees)** (Rating scale)

0: Not useful, 10: Very useful

0 1 2 3 4 5 6 7 8 9 10

4. **Reading images is useful to categorize supplier locations (residential, office, tect) after grabbing the location image from Google Maps** (Rating scale)

Note: to categorize something as a building/ residential... we would rather use the Transformers library with an image model such as Microsoft/resnet-50 or a specific model from HuggingFace for categorizing images.. Transformers automatically gets these from HuggingFace for you

EasyOCR is more about getting text out of images

OCR: Optical Character Recognition

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

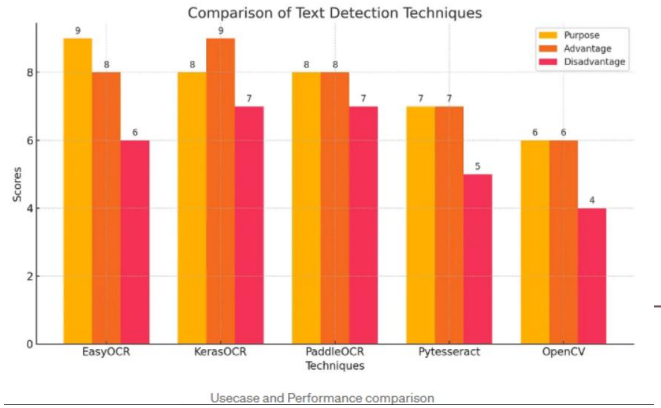
- Load the receipt into the travel and expense program
- Use the travel and expense program to read the name on the receipt
- Compare the name on the receipt to the name for the employee
  - Use Levenschtein distance or a simple Artificial Intelligence library to check for similar match
- If the name 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

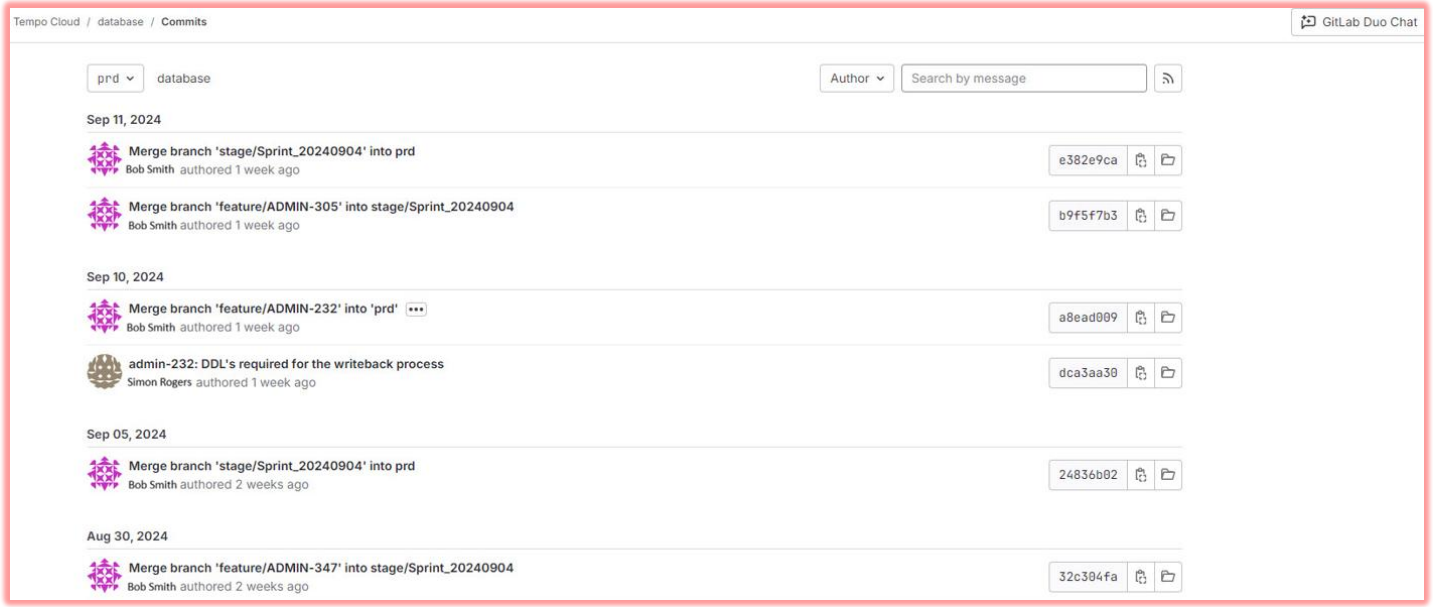
Technique	Purpose	Advantage	Drawback
EasyOCR	General-purpose OCR for detecting and recognizing text in images.	High accuracy with support for multiple languages and complex scripts.	Slower processing compared to other OCR tools due to deep learning models.
KerasOCR	Text detection and recognition using 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



What does the README.md say?

300Academy / PYTHON_MEETING_12

https://github.com/300Academy/PYTHON_MEETING_12

Create the venv... 3.12

README

Extract text from images. Typically, this can be useful when reviewing travel and expense data.

Instructions

Please note: this program automatically finds the requirements file and runs it. To use this functionality, ensure that you are in python version 3.12 or lower.

This program does not require you to input the root folder address. The program will infer the folder address. However, you need to make sure that you respect the folder structure as mentioned below for this inference to work.

1/ Create Virtual Environment (if you are running on a Windows machine and not on Kaggle)

For this project, we need to use the 3.12 version of Python, because this is compatible with PI_SUBPROCESS, that we use to automatically install the Python libraries.

Ensure that you have Python version 3.12 on your machine. If you do not have it, download it from https://www.python.org/downloads.

In Visual Studio Code, open a PowerShell terminal (File-> Terminal) at the location: (you can use cd (which means change directory) to navigate to the 02_PROGRAMMES folder: cd .. means go up cd 02_PROGRAMMES means change to 02_PROGRAMMES folder)

```
PS C...\02_PROGRAMMES
```

Ensure that the address in the terminal matches C:\YourProjectRootFolder\02_PROGRAMMES. The rest of the instructions are commands that we will run in the Powershell terminal at this location.

```
py -3.12 -m venv venv
```

After running, check that you have a \venv folder created inside 02_PROGRAMMES and that it contains python.exe inside \venv\Scripts and that you see pip3.12.exe

What does the README.md say?

300Academy / PYTHON_MEETING_12

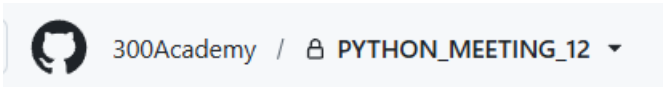
Activate the venv

2/ Activate Virtual Environment using the Powershell activation program

```
.\venv\Scripts\Activate.ps1
```

After activation, open a Python File from the 02_PROGRAMMES folder and check that the virtual environment, mentioned in the bottom-right corner of VSC, points to the python.exe inside your \venv\Scripts folder. If it does not, click on the venv mentioned in the bottom-right of VSC and browse to choose the correct python.exe from the \venv\Scripts folder.

What does the README.md say?



Create .env (empty in this case)

4/ Create a .env file from the .env.example

This program does not use anything from the .env. However, please make sure that you create an empty .env from the .env.example.

What does the README.md say?

300Academy / PYTHON_MEETING_12

Double-check the folder structure:

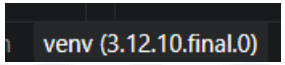
Project Structure

```
ROOT_FOLDER/
│
├── 01_SOURCES/
│   ├── Gitlab image1.png
│   └── AM_VARIABLES.txt
│
├── 02_PROGRAMMES/
│   ├── HR03_11_EXTRACT_TEXT_IMAGES
│   ├── .env.example: - change to .env
│   ├── pyproject.toml
│   └── requirements.txt
│   └── Z_SHARED_FUNCTIONS/
│       └── FC_CREATE_DI_VARIABLES.py
│       └── FC_EXPORT.py
│       └── FC_INSTALL_REQUIREMENTS.py
│
└── 03_RESULTS/
    └── EasyOcrGitLabImage1Result.xlsx

# SAP Source Files

The analysis does not work on SAP data. This analysis is based on non-structured data.
```

Double-check the venv: (pointing to the right location)



What does the README.md say?

300Academy / PYTHON_MEETING_12

Run the script:

5/ Run the script

With the virtual environment corretly activated, as mentioned above, open the file HR03_11_EXTRACT_TEXT_IMAGES.py (that should be in 02_PROGRAMMES) in Visual Studio Code and click on the button Run Python File, or write the following in the terminal:

```
python HR03_11_EXTRACT_TEXT_IMAGES.py
```

If your VSC is picking up the wrong python.exe from the wrong virtual environment - ensure that your venv folder is in the 02_PROGRAMMES folder and that the terminal is in the 02_PROGRAMMES folder and then run the script with direct reference to the python.exe in the virtual environment:

```
.\venv\Scripts\python.exe R03_11_EXTRACT_TEXT_IMAGES.py
```

OCR code example

Our function:

FC_INSTALL_REQUIREMENTS()

automatically installs all the requirements for us.

```python
from pathlib import Path as PI_PATH
import subprocess as PI_SUBPROCESS
import sys as PI_SYS
import os as PI_OS

def FC_INSTALL_REQUIREMENTS(*,ZVFCI_ST_02PROG_FOLDER):

    ZV_OB_02PROG_FOLDER = PI_PATH(ZVFCI_ST_02PROG_FOLDER)

    # 1/ Find requirements.txt
    ZV_LI_OB_PATH_REQ_FILES = list(
        ZV_OB_02PROG_FOLDER.rglob('requirements.txt')
    )
    if not ZV_LI_OB_PATH_REQ_FILES:
        raise FileNotFoundError(f'No requirements.txt file found under
        {ZV_OB_02PROG_FOLDER}')

    ZV_OB_PATH_REQUIREMENTS_FILE = ZV_LI_OB_PATH_REQ_FILES[0]

    # 2/ Install python requirements
    PI_SUBPROCESS.run(
        [
            PI_SYS.executable,
            '-m',
            'pip',
            'install',
            '-r',
            str(ZV_OB_PATH_REQUIREMENTS_FILE)
        ],
        check=True
    )
```

OCR code example

Get the variables from AM_VARIABLES.txt:

```
39  # 2/ Variables
40  ZV_LI_EASY_OCR_LANGUAGES = (ZV_DI_VARIABLES.get('ZV_ST_EASY_OCR_LANGUAGES')).split(',')
41  ZV_BO_EASY_OCR_USE_GPU=(ZV_DI_VARIABLES.get('ZV_ST_EASY_OCR_USE_GPU')).upper()=='TRUE'
42  ZV_ST_IMAGE_FILENAME=ZV_DI_VARIABLES.get('ZV_ST_IMAGE_FILENAME')
43  ZV_ST_RESULTS_FILENAME=ZV_DI_VARIABLES.get('ZV_ST_RESULTS_FILENAME')
44  ZV_ST_REGEX_PATTERN_DATE=ZV_DI_VARIABLES.get('ZV_ST_REGEX_PATTERN_DATE')
45  ZV_ST_REGEX_PATTERN_TEXT1=ZV_DI_VARIABLES.get('ZV_ST_REGEX_PATTERN_TEXT1')
46  ZV_ST_REGEX_PATTERN_TEXT2=ZV_DI_VARIABLES.get('ZV_ST_REGEX_PATTERN_TEXT2')
47  ZV_ST_SOURCES_FOLDER = PI_OS.path.join(
48      ZV_ST_ROOT_FOLDER,
49      '01_SOURCES'
50  )
51  ZV_ST_RESULTS_FOLDER = PI_OS.path.join(
52      ZV_ST_ROOT_FOLDER,
53      '03_RESULTS'
54  )
55
56  ZV_OB_REGEX_PATTERN_DATE = PI_RE.compile(
57      ZV_ST_REGEX_PATTERN_DATE
58  )
```

Easy OCR – T&E example: HR03_14

Initialize OCR and read image

Easy OCR object

```
60    # 3/ Create OCR reader object
61    ZV_OB_READER = API_EASYOCR.Reader(ZV_LI_EASY_OCR_LANGUAGES, gpu=ZV_BO_EASY_OCR_USE_GPU)
62    print("✅ EasyOCR object created.")
63
```

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

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

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.

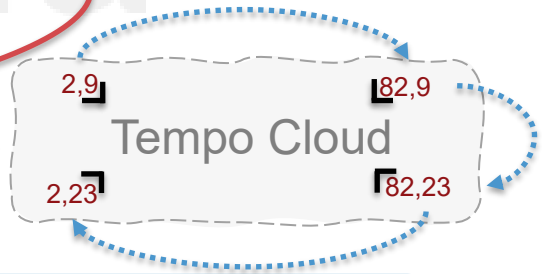
We use the for loop to go around each item in the list and print out the contents.

```
# 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

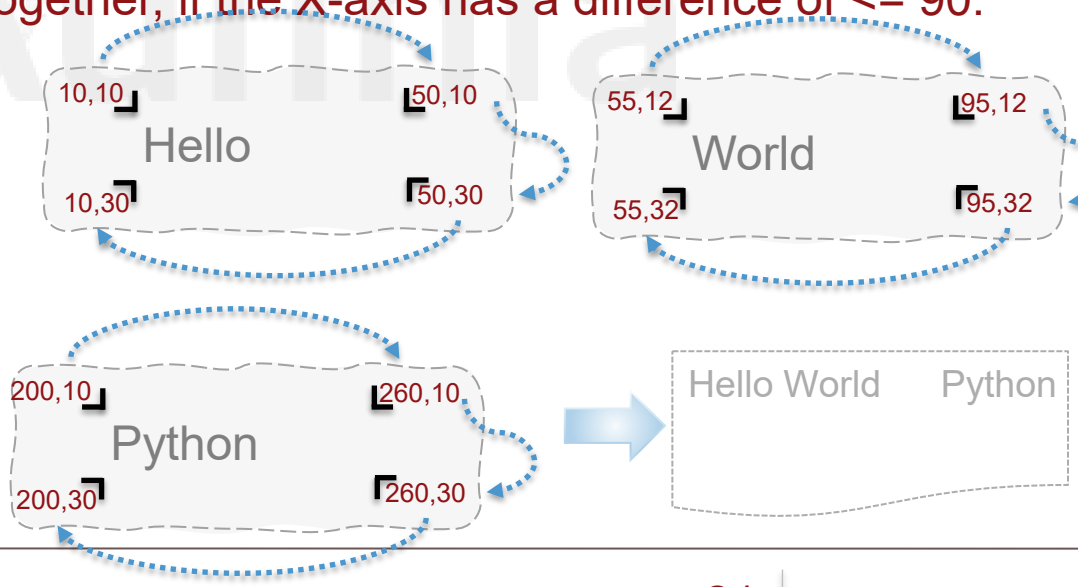
Easy OCR – T&E example: HR03_14

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

Helper functions to check the line type extract text, dates, format text dates

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

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

def FC_TEXT_CHECK_METADATA_LINE(ZVFCI_ST_TEXT):
    """Return True if line contains 'authored' (metadata line with author info)"""
    return ZV_ST_REGEX_PATTERN_TEXT2 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()

    ZV_OB_REGEX_MATCH_WORD = PI_RE.search(
        rf'^(.*?)(?={ZV_ST_REGEX_PATTERN_TEXT2})',
        ZV_ST_CLEANED,
        PI_RE.I
    )
    ZV_OB_REGEX_MATCH_WORD = PI_RE.search(r'^(.*?)(?=authored)', ZV_ST_CLEANED, PI_RE.I)
    return ZV_OB_REGEX_MATCH_WORD.group(1).strip() if ZV_OB_REGEX_MATCH_WORD 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_REGEX_PATTERN_TEXT1})\s*-\s*(\d+)\b',
        r'\1-\2',
        ZVFCI_ST_TEXT,
        flags=PI_RE.I
    )

def FC_DATE_NORMALIZE(ZVFCI_ST_TEXT):
    """Convert detected date string into YYYY-MM-DD format"""
    if not ZVFCI_ST_TEXT:
        return None
    ZV_OB_REGEX_MONTHDDYYYY = PI_RE.search(r'([a-zA-Z]{3})[^0-9]*(\d{1,2})[^0-9]*(\d{4})',
    ZVFCI_ST_TEXT)
    if not ZV_OB_REGEX_MONTHDDYYYY:
        return None
    ZV_ST_MONTH, ZV_ST_DAY, ZV_ST_YEAR = ZV_OB_REGEX_MONTHDDYYYY.group(1).title(),
    ZV_OB_REGEX_MONTHDDYYYY.group(2).zfill(2), ZV_OB_REGEX_MONTHDDYYYY.group(3)
    try:
        ZV_DT_DATE = PI_DATETIME.strptime(f"{ZV_ST_MONTH} {ZV_ST_DAY} {ZV_ST_YEAR}", "%b %d %Y")
        return ZV_DT_DATE.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

```
    # ----------------------------
    # Step 3b: Non-metadata line (check for date)
    # ----------------------------

    else:
        ZV_OB_REGEX_MATCH_DATE = ZV_OB_REGEX_PATTERN_DATE.search(ZV_ST_FULL_LINE)
        if ZV_OB_REGEX_MATCH_DATE:
            # Save detected date for later matching with titles
            ZV_LI_DI_DATES.append({
                "date": FC_DATE_NORMALIZE(ZV_OB_REGEX_MATCH_DATE.group(0)),
                "y": ZV_DI_LINE["y_top"]
            })
```

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

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

for ZV_NU_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(ZV_DI_SEG['text'] for ZV_DI_SEG in ZV_DI_LINE['segments']).
    replace('.', '')

    # ----------------------------
    # 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 ZV_NU_IDX > 0:
            ZV_DI_TITLE_LINE = ZV_LI_MERGED_LINES[ZV_NU_IDX - 1]
            ZV_NU_TITLE_Y = ZV_DI_TITLE_LINE["y_top"]

            # Assign the nearest date above the title
            for ZV_DI_DATE in reversed(ZV_LI_DI_DATES):
                if ZV_DI_DATE["y"] <= ZV_NU_TITLE_Y:
                    ZV_ST_ASSIGNED_DATE = ZV_DI_DATE["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
        })
```

Easy OCR – T&E example: HR03_14

Finally, we can print out the DataFrame that we got from the program and also export it to Excel.

	A	B	C	D
1	Date	Title	Author	ADMIN/TEMPO number
2	2024-09-11	Merge branch 'stagelSprint_20240904' into prd	Bob Smith	
3	2024-09-11	Merge branch 'feature/ADMIN-305' into stage/Sprint_20240904	Bob Smith	ADMIN-305
4	2024-09-10	Merge branch 'feature/ADMIN-232' into 'prd'	Bob Smith	ADMIN-232
5	2024-09-10	admin-232: DDL's required for the writeback process	Simon Rogers	ADMIN-232
6	2024-09-05	Merge branch 'stagelSprint_20240904 into prd	Bob Smith	
7	2024-08-30	Merge branch 'featurelADMIN-347' into stage/Sprint_20240904	Bob Smith	
8				

```
357  # ========================
358  # Step 4: Create DataFrame
359  # ========================
360
361  ZV_DF_RESULT = PI_PANDAS.DataFrame(ZV_LI_RESULTS)
362
363  def FC_EXTRACT_TICKET(ZVFCI_ST_TITLE):
364      """Extract ADMIN/TEMPO ticket number from title if exists"""
365      ZV_OB_REGEX_MATCH_ADMINCODE = PI_RE.search(rf'\b({ZV_ST_REGEX_PATTERN_TEXT1})-\d+\b',
         ZVFCI_ST_TITLE, PI_RE.I)
366      return ZV_OB_REGEX_MATCH_ADMINCODE.group(0).upper() if ZV_OB_REGEX_MATCH_ADMINCODE else None
367
368  if not ZV_DF_RESULT.empty:
369      ZV_DF_RESULT["Title"] = ZV_DF_RESULT["Title"].astype(str).str.strip()
370      ZV_DF_RESULT["ADMIN/TEMPO number"] = ZV_DF_RESULT["Title"].apply(FC_EXTRACT_TICKET)
371
372  # Show DataFrame with full content
373  PI_PANDAS.set_option('display.max_colwidth', None)
374  print(ZV_DF_RESULT)
375
376  ZV_DF_RESULT = PI_POLARS.from_pandas(
377      ZV_DF_RESULT
378  )
379  # Export to Excel
380  FC_EXPORT_EXCEL_POLARS(
381          ZV_DF_RESULT,
382          ZV_ST_RESULTS_FOLDER,
383          ZV_ST_RESULTS_FILENAME
384      )
```

```
----------------------------------------
        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?