



**UTM**  
UNIVERSITI TEKNOLOGI MALAYSIA

**SMJE4383**

**ADVANCED PROGRAMMING (ADVANCED  
PROGRAMMING)**

**Assignment 2**

**Screen Scraping & OCR Text Recognition using Python**

**SECTION: 1**

- 1. Muhammad Danish Danial bin Idjarmizuan    A19MJ3036**
- 2. Muhammad Haziq Aiman bin Kamarulzaman    A19MJ3041**

**Githublink :** <https://github.com/hhaziqaimann/ASSIGNMENT-2-SMJE-4383-adv-Programming>

## INTRODUCTION

There are many tools for Python, including BeautifulSoup, Scrapy, and Selenium. Screen scraping is a technique for retrieving data from a web page.

A package called BeautifulSoup makes it easy to retrieve data from websites, providing Python methods to access data in HTML and XML files, which are placed on top of an HTML or XML parser.

Scrapy for Python is a free and open source web crawler framework. It provides methods to manage the extraction, storage, and output of data and to define the structure of the scraping process.

The process of converting a scanned photo or PDF document into editable and searchable text is called OCR (Optical Character Recognition) Python has a number of libraries for OCR text recognition.

- Tesseract: Google's OCR engine that recognizes over 100 languages; available in Python using the pytesseract library.
- OCRopus is a free and open source OCR engine written in Python that recognizes text using the concept of structured prediction.
- pyOCR is an OCR engine wrapper for Tesseract written in Python that supports multiple OCR engines and has an intuitive user interface.

These libraries can be used to extract text from scanned documents and photos, and can be trained to recognize text in specific font styles, sizes, and languages for increased accuracy.

Tesseract is a popular OCR engine that can be used in Python through the pytesseract library.

the Image library from the Python Imaging Library (PIL) is used to open the image file, and the pytesseract library is used to recognize the text using Tesseract. The recognized text is then printed using the print function.

need to install both the pytesseract and Pillow libraries if they are not already installed on your system. You may also need to install Tesseract OCR on your system, which can be done by following the instructions on the Tesseract GitHub page.

## OBJECTIVES

The objectives for this assignment are to perform the entire process of OCR text recognition by screen scraping using Python scripting was selected to be performed. This assignment was selected as an industry-academia collaborative project by MJIIT. The primary objective is to solve an industry-based problem.

## METHODOLOGY

First, we need to install Tesseract OCR on our PC system, by following follow the steps below, which are specific to your operating system:

Windows

1. Download the Tesseract installer from the following URL: <https://github.com/UB-Mannheim/tesseract/wiki>
2. Run the installer and follow the on-screen instructions to install Tesseract on your system.
3. Set the environment variable `TESSDATA_PREFIX` to the directory where Tesseract is installed, for example: `C:\Program Files (x86)\Tesseract-OCR.`

After installing Tesseract OCR, can use the `pytesseract` library in Python to recognize text from images, as described in my previous answer.

OpenCV (Open Source Computer Vision) is a programming library primarily for real-time computer vision. It helps to apply. In this tutorial, you will learn how to detect text in an image using contour lines and save it to a text file.

Installation is required by pip install in terminal

```
pip install opencv-python
```

```
pip install pytesseract
```

To perform image processing on an image.

## **Applying image processing for the image:**

First, the color space of the image is changed and stored in a variable. The function `cv2.cvtColor (input image, flag)` is used for the color transformation. The type of conversion is determined by the second flags parameter. `cv2.COLOR_BGR2GRAY` and `cv2.COLOR_BGR2HSV` can be selected from `cv2.COLOR_BGR2GRAY` converts an RGB image to a grayscale image, `cv2.COLOR_BGR2HSV` converts an RGB image to a HSV (hue, saturation, value) color space image. `COLOR_BGR2GRAY` is used here. `cv2.threshold` function is used to apply a threshold to the converted image.

There are three types of thresholding methods

1. simple thresholding
2. Otsu binarization
3. adaptive thresholding

## **To get a rectangular structure:**

Use the `cv2.getStructuringElement()` method to define structures such as ellipses, circles, and rectangles. Here, a rectangular structuring element (`cv2.MORPH_RECT`) is used. To do so, size must be added to the kernel parameter of `cv2.getStructuringElement`. Larger kernels group larger text blocks. `cv2.dilate` method expands the image after selecting the appropriate kernel. The expansion enlarges (dilates) the text blocks, thus improving the accuracy of text cluster detection.

## **Confirmation of contours**

`CV2.findContours()` To extract contours from an enhanced image, use the `contours()` function. `cv.find` has three parameters. the `contours()` function contains the source image, the contour extraction mode, and the contour approximation method.

hierarchy and edges returned by the function. The Python list of all contours in the image is called contours. Each contour is a Numpy array of (x,y) coordinates of the boundary points of the object. Contours are often used to identify white objects from black backgrounds. The above image processing methods can be used to identify the boundaries of text blocks in an image with outlines. Refresh the text file after opening it in write mode. The text created by the program is saved in this text file.

## Using the OCR

Use the `cv2.boundingRect` method to walk around each contour and measure its width, height, and `x,y ()` coordinates. Then, using the obtained `x,y,width,height` coordinates and the `cv2.rectangle()` function, draw a rectangle on the image. `cv2.rectangle()` function has 5 parameters. The first argument specifies the input image, followed by the rectangle start coordinates `x` and `y`, the rectangle end coordinates `(x+w, y+h)`, the RGB value of the rectangle border color, and the border size. That's right. To extract text from the image, a rectangular region is cut out and sent to the tesseract.

Then we open the created text file in append mode to append the obtained text and close the file.

Sample image used for the code:

```
import cv2
import pytesseract

# Mention the installed location of Tesseract-OCR in your system
pytesseract.pytesseract.tesseract_cmd = 'System_path_to_tesseract.exe'

# Read image from which text needs to be extracted
img = cv2.imread("sample.jpg")

# Preprocessing the image starts

# Convert the image to gray scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Performing OTSU threshold
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU |
cv2.THRESH_BINARY_INV)

# Specify structure shape and kernel size.
# Kernel size increases or decreases the area
# of the rectangle to be detected.
# A smaller value like (10, 10) will detect
# each word instead of a sentence.
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (18, 18))

# Applying dilation on the threshold image
dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)
```

```
# Finding contours
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL,
                                       cv2.CHAIN_APPROX_NONE)

# Creating a copy of image
im2 = img.copy()

# A text file is created and flushed
file = open("recognized.txt", "w+")
file.write("")
file.close()

# Looping through the identified contours
# Then rectangular part is cropped and passed on
# to pytesseract for extracting text from it
# Extracted text is then written into the text file
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)

    # Drawing a rectangle on copied image
    rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Cropping the text block for giving input to OCR
    cropped = im2[y:y + h, x:x + w]

    # Open the file in append mode
    file = open("recognized.txt", "a")

    # Apply OCR on the cropped image
    text = pytesseract.image_to_string(cropped)

    # Appending the text into file
    file.write(text)
    file.write("\n")

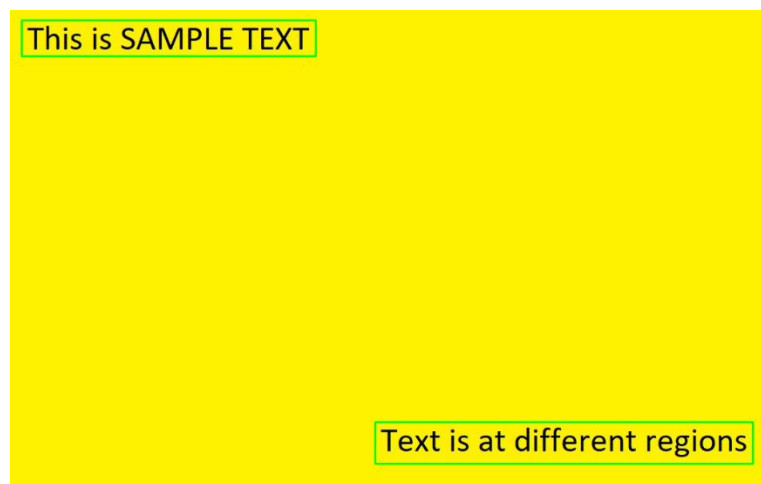
# Close the file
file.close
```

## RESULT

The result of an OCR process using Tesseract in Python is typically a string of text that represents the recognized text in the input image or document.




the recognized text is stored in the text variable, Note that the quality of the OCR result may depend on several factors, such as the resolution and quality of the input image, the font style and size of the text, and the presence of noise or other distracting elements in the image. To improve the accuracy of the OCR process, you may need to pre-process the input image, for example by cropping or resizing the image, or by enhancing the contrast or removing noise.

The sample text that the program scrapping:



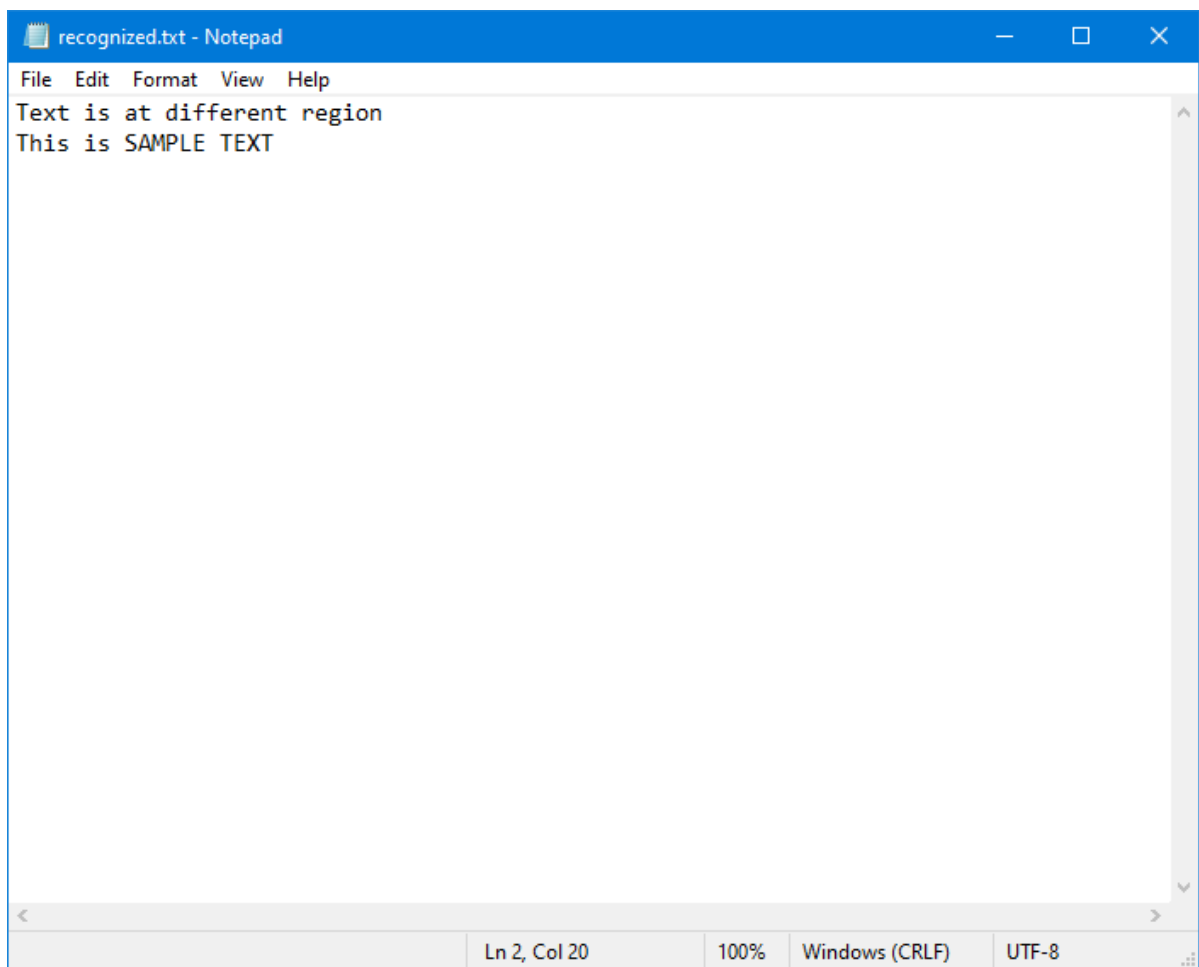
**Figure 1: sample text**

## OUTPUT

 ass2.py	6/2/2023 4:18 PM	Python Source File
 recognized.txt	6/2/2023 4:19 PM	Text Document
 sample.jpg	6/2/2023 4:19 PM	JPG File

**Figure2: Output in file**

The program automatically create a txt file



**Figure3: Output txt file**



## CONCLUSION

In conclusion, web scraping (or screen scraping) is a technique for extracting data from websites and converting it into a structured format for analysis and processing. Python provides several libraries for web scraping, including BeautifulSoup, Scrapy, and Selenium, that make it easy to extract information from websites.

However, it's important to note that web scraping can be a complex process, as websites can use different technologies and structures, and may change over time. To ensure that the web scraping process is reliable and efficient, it's important to have a good understanding of the underlying technologies and structure of the websites you are scraping, as well as to be familiar with the libraries and tools that you are using.

In any case, web scraping can be a valuable tool for collecting data from websites and automating the process of data collection. However, it's important to follow ethical and legal guidelines when scraping websites, and to respect the terms of use and privacy policies of the websites you are scraping.

OCR (Optical Character Recognition) is a technology that allows you to convert scanned images or PDF documents into editable and searchable text. Python provides several libraries for OCR text recognition, including Tesseract and its wrapper pytesseract. By using these libraries, you can extract text from images or documents, and process the text for various purposes, such as text analysis, information extraction, and content management.

However, it's important to note that the accuracy of the OCR process may vary depending on several factors, such as the quality of the input image or document, the font style and size of the text, and the presence of noise or other distracting elements. To improve the accuracy of the OCR process, you may need to pre-process the input image or document, and adjust the parameters of the OCR engine as needed.

In any case, OCR technology is a powerful tool for automating the process of text recognition, and can save significant amounts of time and effort compared to manual data entry.

## REFERENCES

- "Extracting text from images using OCR on Android"*. June 27, 2015. Archived from [\*the original\*](#) on March 15, 2016.
- OnDemand, HPE Haven. *"OCR Document"*. Archived from [\*the original\*](#) on April 15, 2016.
- Tappert, C. C.; Suen, C. Y.; Wakahara, T. (1990). *"The state of the art in online handwriting recognition"*. IEEE Transactions on Pattern Analysis and Machine Intelligence. **12** (8): 787.
- Zeng, Qing-An (2015). *Wireless Communications, Networking and Applications: Proceedings of WCNA 2014*. Springer. [\*ISBN 978-81-322-2580-5\*](#).