❖ **Assignment No:** 2

❖ **Problem Statement:** Implement a program to generate & verify CAPTCHA image

❖ **Objectives:**

- To generate a random CAPTCHA image containing letters and numbers to prevent automated access.
- To display the CAPTCHA to the user and verify the input against the generated text.
- To ensure secure verification by making the CAPTCHA difficult for bots to read.

❖ **Hardware Requirements:** Intel Core i3 or higher Processor, 4 GB RAM, 250 GB Hard Disk

❖ **Software Requirements:** Windows 10 / Linux / macOS Operating System, PyCharm, VS Code, or any Python IDE

❖ **Theory:**

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a security mechanism used to differentiate between human users and automated bots. CAPTCHAs are widely used on websites for login forms, online registrations, and other areas where automated attacks or spamming must be prevented.

1. Purpose of CAPTCHA

- Prevent Bots: CAPTCHAs stop automated scripts from abusing online services.
- Security Enhancement: They provide an additional layer of security to forms and authentication processes.
- Human Verification: Ensures that only humans can perform certain actions like login, sign-up, or submitting forms.

2. Structure of CAPTCHA

A typical CAPTCHA contains:

- Random Characters: A combination of letters, numbers, or symbols.
- Noise/Distortions: Lines, dots, or background patterns to make it difficult for automated systems to recognize the text.
- Image Format: Usually generated as an image (PNG, JPEG) for human readability.

3. CAPTCHA Generation Process

1. Random Text Generation: Create a string of random letters and numbers.
2. Image Creation: Render the random text on an image canvas.
3. Add Noise & Distortion: Draw lines, curves, or dots and apply filters to make automated recognition difficult.
4. Display to User: Show the image to the user for input.

4. CAPTCHA Verification

- The user reads the text from the image and enters it in an input field.
- The program compares the input with the originally generated text.
- If the input matches, access is granted; otherwise, the user is prompted to try again.

5. Applications

- Login and registration forms on websites.
- Preventing spam in online surveys or comment sections.
- Protecting sensitive operations in web applications from automated attacks.

❖ **Conclusion:**

The CAPTCHA Generator & Verifier project demonstrates how automated systems can be prevented from accessing secure forms or services. By generating random, distorted text images and verifying user input, the system ensures that only humans can complete actions like login or registration. This practical highlights the importance of CAPTCHAs in enhancing security and protecting applications from bots and spam.

❖ **Code:**

```
import random
import string
from PIL import Image, ImageDraw, ImageFont, ImageFilter
from IPython.display import display  # for inline display

# Function to generate random CAPTCHA text
def generate_captcha_text(length=6):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

# Function to create CAPTCHA image
def create_captcha_image(captcha_text):
    width = 200
    height = 80
    image = Image.new('RGB', (width, height), 'white')
    draw = ImageDraw.Draw(image)

    try:
        font = ImageFont.truetype("arial.ttf", 40)
    except:
        font = ImageFont.load_default()

    for i, char in enumerate(captcha_text):
        x = 20 + i * 30 + random.randint(-5,5)
        y = 20 + random.randint(-5,5)
        draw.text((x, y), char, font=font, fill=(random.randint(0,150), random.randint(0,150),
random.randint(0,150)))

    for _ in range(5):
        x1 = random.randint(0, width)
        y1 = random.randint(0, height)
        x2 = random.randint(0, width)
        y2 = random.randint(0, height)
```

```
        draw.line(((x1, y1), (x2, y2)), fill=(random.randint(0,150), random.randint(0,150),
    random.randint(0,150)), width=2)

        image = image.filter(ImageFilter.EDGE_ENHANCE_MORE)

        # Display inline in notebook
        display(image)
        return image

    # Main
    captcha_text = generate_captcha_text(6)
    create_captcha_image(captcha_text)

    user_input = input("Enter CAPTCHA text: ")

    if user_input == captcha_text:
        print("CAPTCHA Verified Successfully!")
    else:
        print("Incorrect CAPTCHA. Verification Failed.")
```

❖ **Output:**

```
Enter CAPTCHA text:  eYMIoA
CAPTCHA Verified Successfully!
```