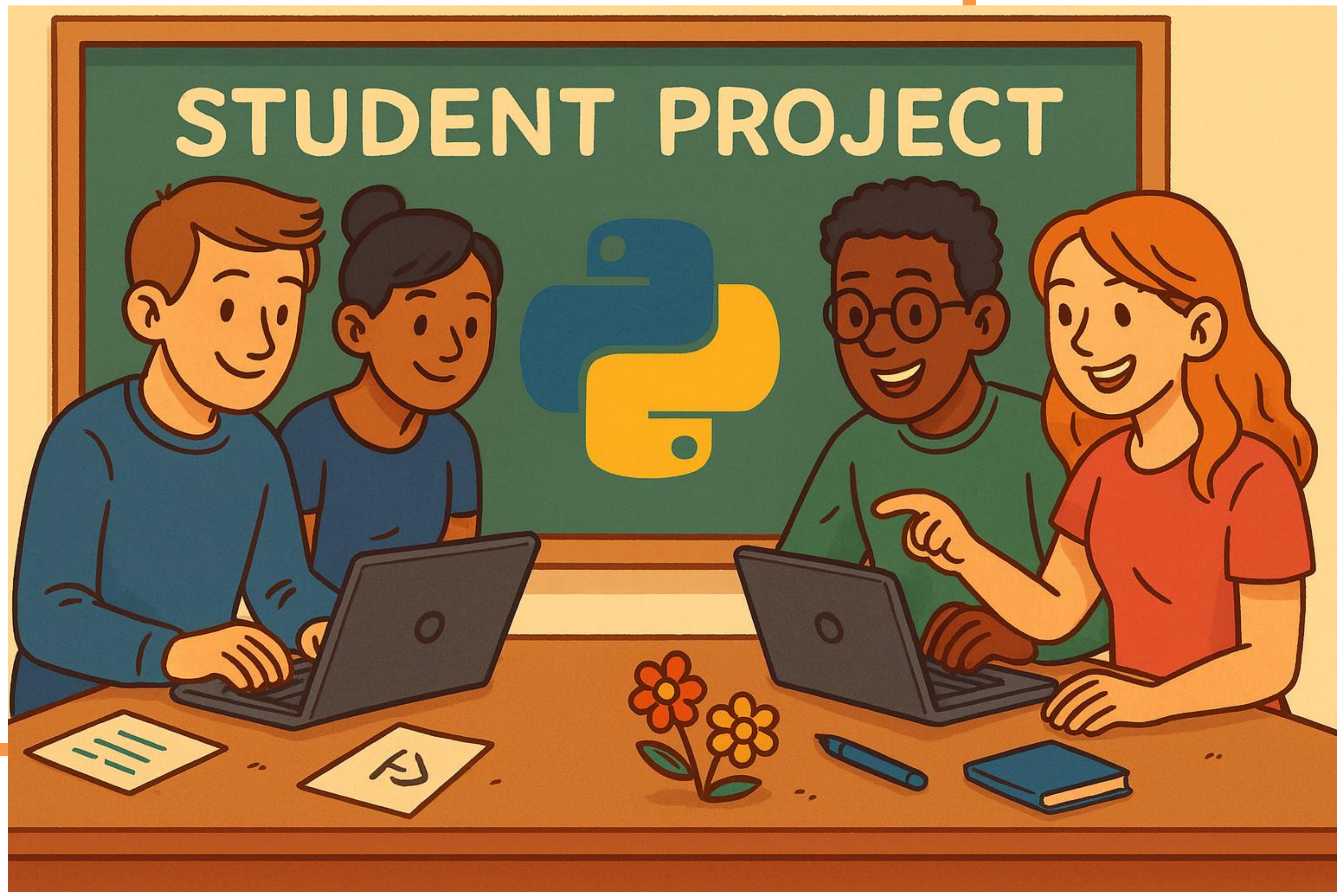
# SW10 – Work 2

**Handwritten digit recognition - with Python**

Handwritten digit recognition is a classic problem in machine learning and computer vision. It involves recognizing handwritten digits from images or scanned documents. This task is widely used as a benchmark for evaluating machine learning models especially neural networks due to its simplicity and real-world applications.

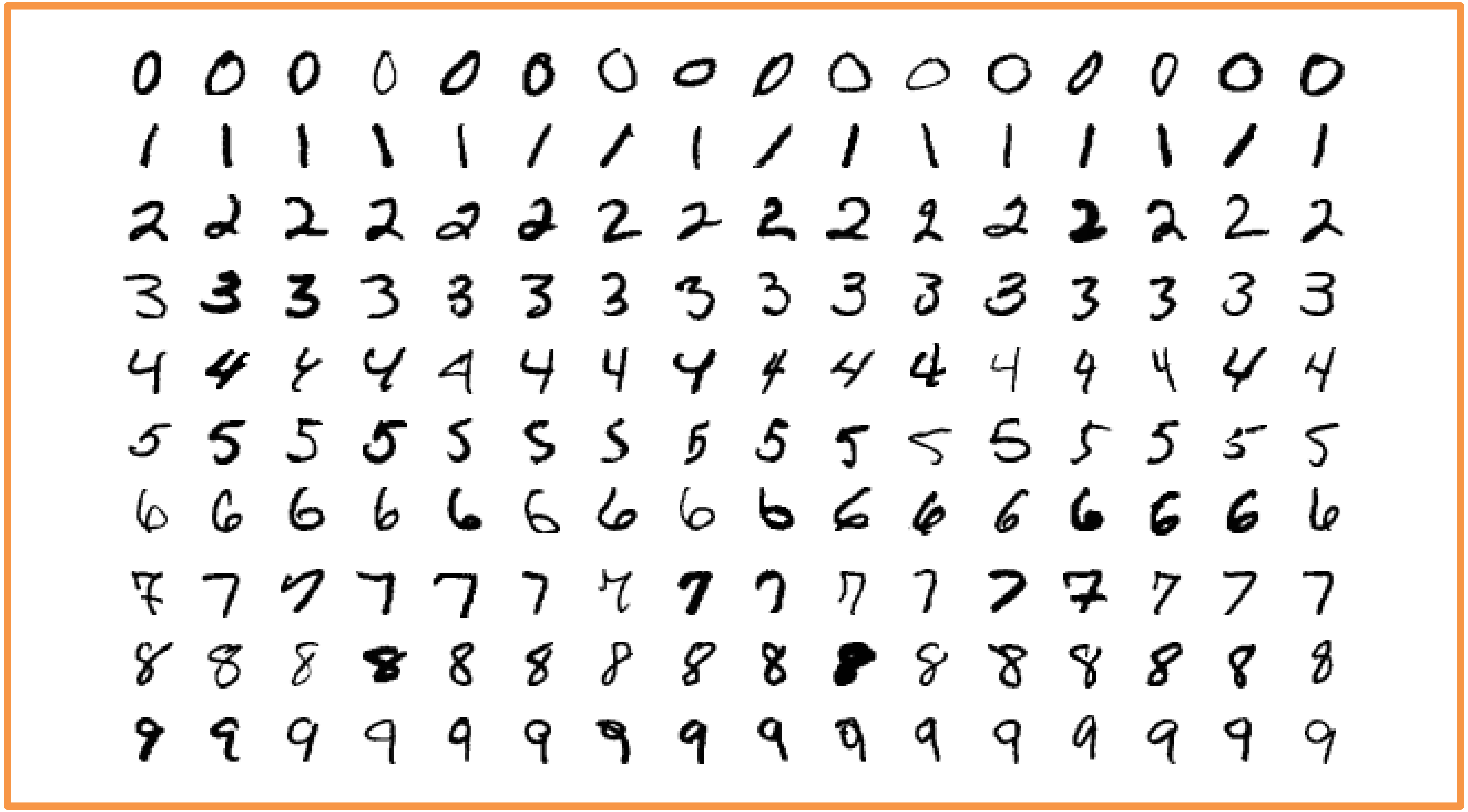


# SW10 – Work 2

## What is the MNIST Handwritten Dataset?

- The MNIST handwritten dataset is one of the most well-known and widely used datasets in the field of machine learning and deep learning.
- MNIST stands for Modified National Institute of Standards and Technology.
- It is a dataset of grayscale images of handwritten digits from 0 to 9.
- Each image in the dataset is 28x28 pixels, resulting in a total of 784 pixels per image. These pixels are represented as numerical values indicating the grayscale intensity, ranging from 0 (black) to 255 (white).
- The dataset consists of 70,000 images:
- 60,000 labeled training images used for training models.
- 10,000 test images used for evaluating the performance of models.

# SW10 – Work 2



# SW10 – Work 2

**Todo**

- Run the code and check what is being done to train the neural network.
- Try to answer the following questions (next slide).

💾 IN_SYS_HS2025_SW10_Work2.ipynb

# SW10 – Work 2

**Questions**

- Why do the image data need to be normalized (e.g., divided by 255.0)?
- Why must we split the data into training and test sets?
- The MNIST dataset contains grayscale images with a resolution of 28 × 28 pixels. If the model uses a fully connected (Dense) input layer, how many input values must be provided to the model for each image?
- What is the purpose of a Dense layer with 10 neurons in the model?
- What is the difference between a perceptron and a Dense layer?
- Why is training accuracy usually higher than test accuracy?
- What does a confusion matrix tell you about the model?
- Why is np.argmax() used when predicting digits?
- What happens if you feed an image with incorrect dimensions (e.g., 28×28×3 instead of 28×28×1)?

# SW10 – Work 2

## Question

- Why do the image data need to be normalized (e.g., divided by 255.0)?

## Answer

- Normalization scales pixel values to the range 0–1. This stabilizes training, makes gradients smaller and smoother, and helps the model converge faster and more reliably.

# SW10 – Work 2

**Question**

- Why must we split the data into training and test sets?

**Answer**

- To measure generalization. The test set contains unseen data that shows if the model learned patterns instead of memorizing.

# SW10 – Work 2

## Question

- The MNIST dataset contains grayscale images with a resolution of 28 × 28 pixels. If the model uses a fully connected (Dense) input layer, how many input values must be provided to the model for each image?

## Answer

- 784 (28 x 28 x 1)

# SW10 – Work 2

**Question**

- What is the purpose of a Dense layer with 10 neurons in the model?

**Answer**

- It outputs one value for each of the 10 digit classes (0–9), enabling classification across all possible MNIST digits.

# SW10 – Work 2

**Question**

- What is the difference between a perceptron and a Dense layer?

**Answer**

- A perceptron is a single neuron with weights and a bias. A Dense layer is a collection of perceptrons, all fully connected to previous layers.

# SW10 – Work 2

## Question

- Why is training accuracy usually higher than test accuracy?

## Answer

- Because the model has seen the training data many times but has never seen the test data.

# SW10 – Work 2

**Question**

- What does a confusion matrix tell you about the model?

**Answer**

- It shows how often each true class is predicted as each possible class, helping identify systematic misclassifications.

# SW10 – Work 2

**Question**

- Why is np.argmax() used when predicting digits?

**Answer**

- It selects the class index with the highest predicted probability.

# SW10 – Work 2

## Question

- What happens if you feed an image with incorrect dimensions (e.g., 28×28×3 instead of 28×28×1)?

## Answer

- The model may throw an error or produce completely incorrect predictions.

```
pic = x_test[0].reshape(28, 28)
plt.imshow(pic, cmap='gray')
pic_input_1 = pic.reshape(1, 28 * 28)
print(pic_input_1.size)
pred_1 = model.predict(pic_input_1)
pic_pred_1 = np.argmax(pred_1, axis=1)
print(pic_pred_1)
```

```
784
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step
[7]
```