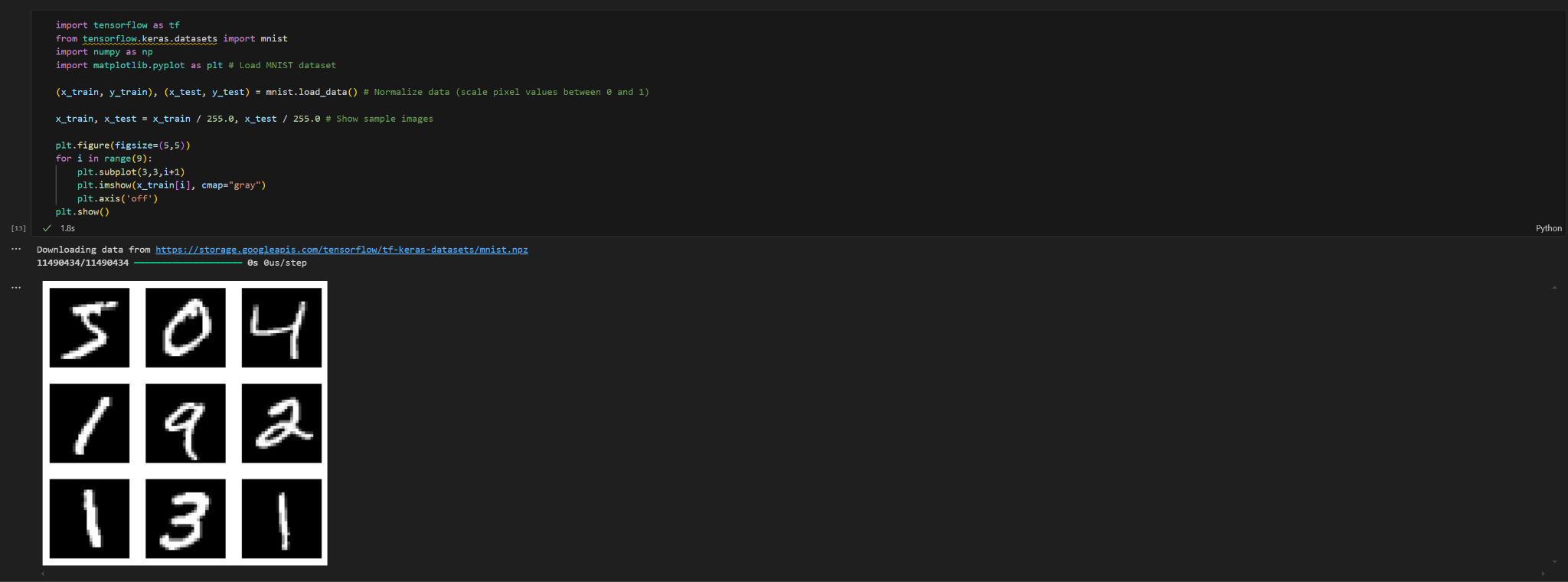
### **1. MNIST Data Loading and Visualization**

**Purpose:** Load the MNIST handwritten digit dataset, normalize the image data, and visually display sample images.

**What it does:**

* Loads training and testing data from tensorflow.keras.datasets.mnist.
* Scales pixel values to a range of 0–1 for neural network compatibility.
* Uses matplotlib to plot the first 9 images from the training set.



### 

### 

### 

### 

### 

### 

### 

### 

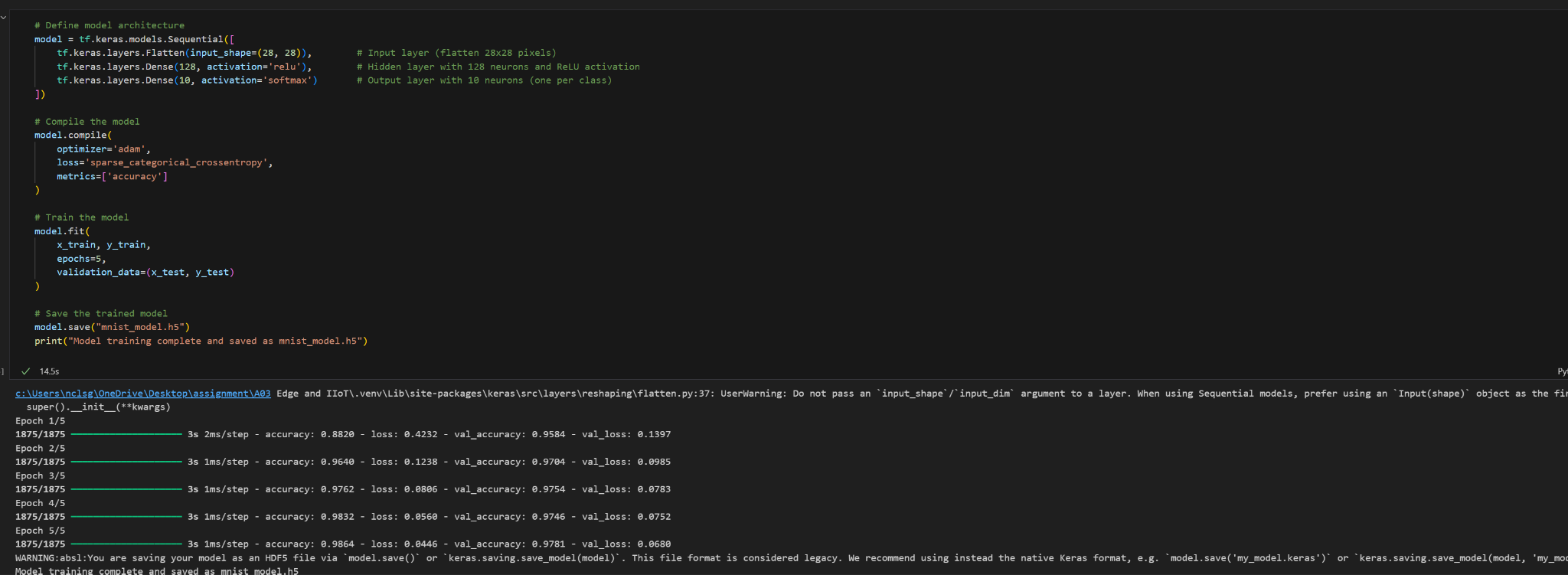
### 

### **2. Define, Train, and Save a Keras Model**

**Purpose:** Create a simple neural network using Keras to classify handwritten digits and save the trained model.

**What it does:**

* Defines a 3-layer sequential model: flatten input, dense ReLU layer, and softmax output.
* Compiles the model using the Adam optimizer and cross-entropy loss.
* Trains for 5 epochs using training/validation data.
* Saves the trained model to a .h5 file for later use.



### 

### 

### 

### 

### 

### 

### 

### 

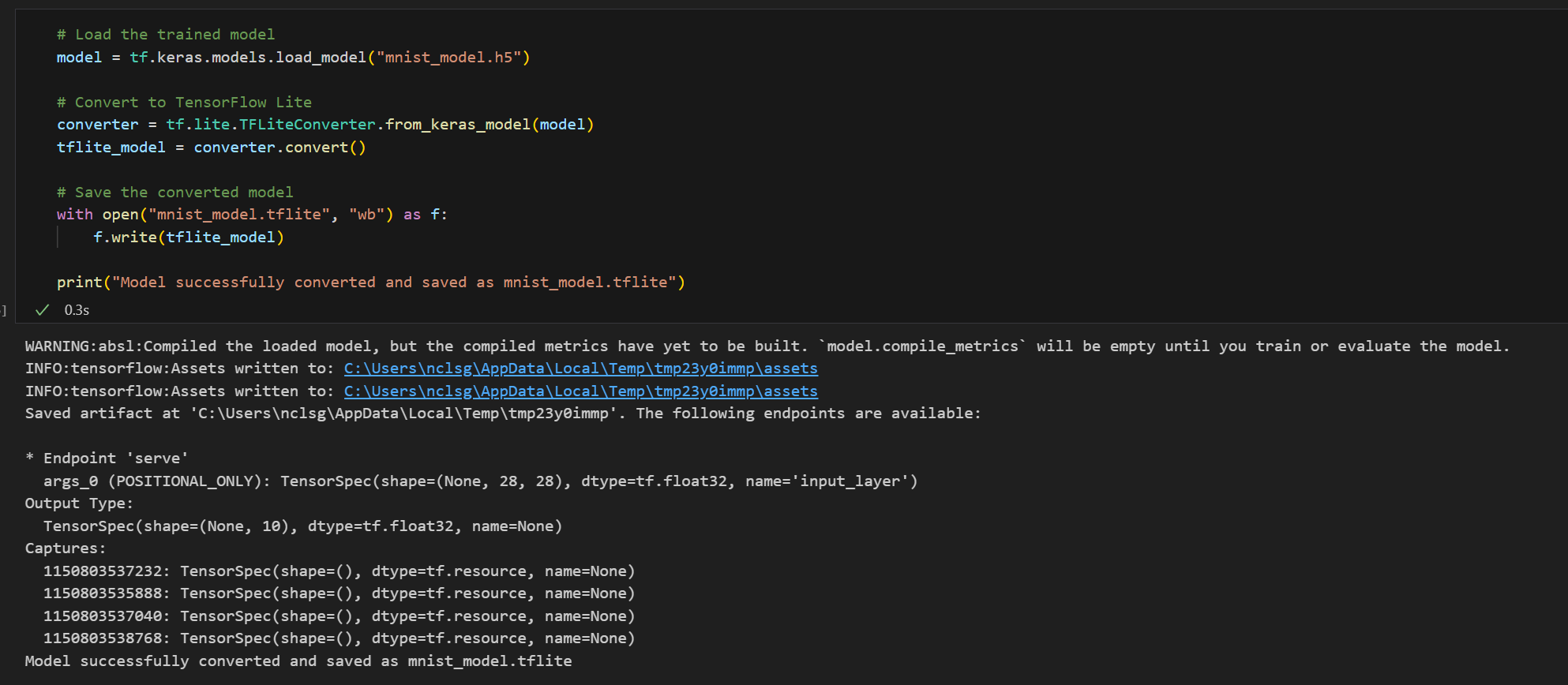
### 

### **3. Convert Keras Model to TensorFlow Lite**

**Purpose:** Convert the trained Keras model into a lightweight format for use on mobile or embedded devices.

**What it does:**

* Loads the saved .h5 model.
* Uses TFLiteConverter to convert it to a .tflite model.
* Saves the .tflite file to disk for deployment on edge devices.

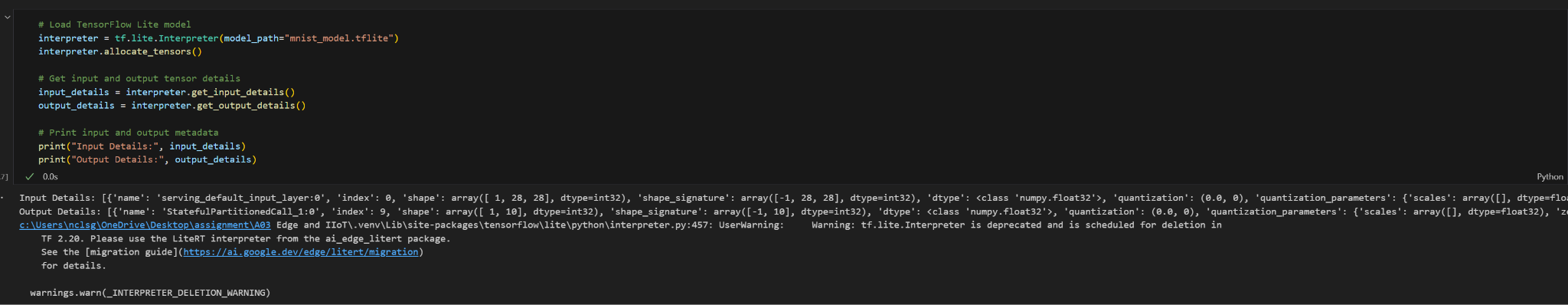


### **4. Load TensorFlow Lite Model and View Input/Output Info**

**Purpose:** Load the .tflite model into a TensorFlow Lite interpreter and examine its input and output structure.

**What it does:**

* Initializes the TFLite interpreter and allocates memory for tensors.
* Extracts and prints input and output tensor details such as shape, index, and data type.



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **5. Run Inference on a Test Image Using TensorFlow Lite**

**Purpose:** Use the loaded TFLite model to predict the digit in a single MNIST test image.

**What it does:**

* Selects the first image from the test set and formats it correctly for the model.
* Sets it as input to the interpreter and runs inference.
* Retrieves prediction output, determines the predicted class, and displays the image alongside the predicted and actual labels.

