Writing Neural Network Code: Introduction to TensorFlow, Hands-On

Class 4: TensorFlow Hands-On Part 2: Defining and Building Your Network

May 14, 2020

Charles J. Lord, PE
President, Consultant, Trainer
Blue Ridge Advanced Design and Automation

Presented by:

This Week's Agenda

5/11 A Brief History of Artificial Neural Networks

5/12 Neural Network Simulation and Programming

5/13 TensorFlow Hands-On Part 1: Hello World!

5/14 TensorFlow Hands-On Part 2: Defining and Building Your Network

5/15 TensorFlow Hands-On Part 3: Teaching and Testing and Conclusion

This Week's Agenda

5/11 A Brief History of Artificial Neural Networks

5/12 Neural Network Simulation and Programming

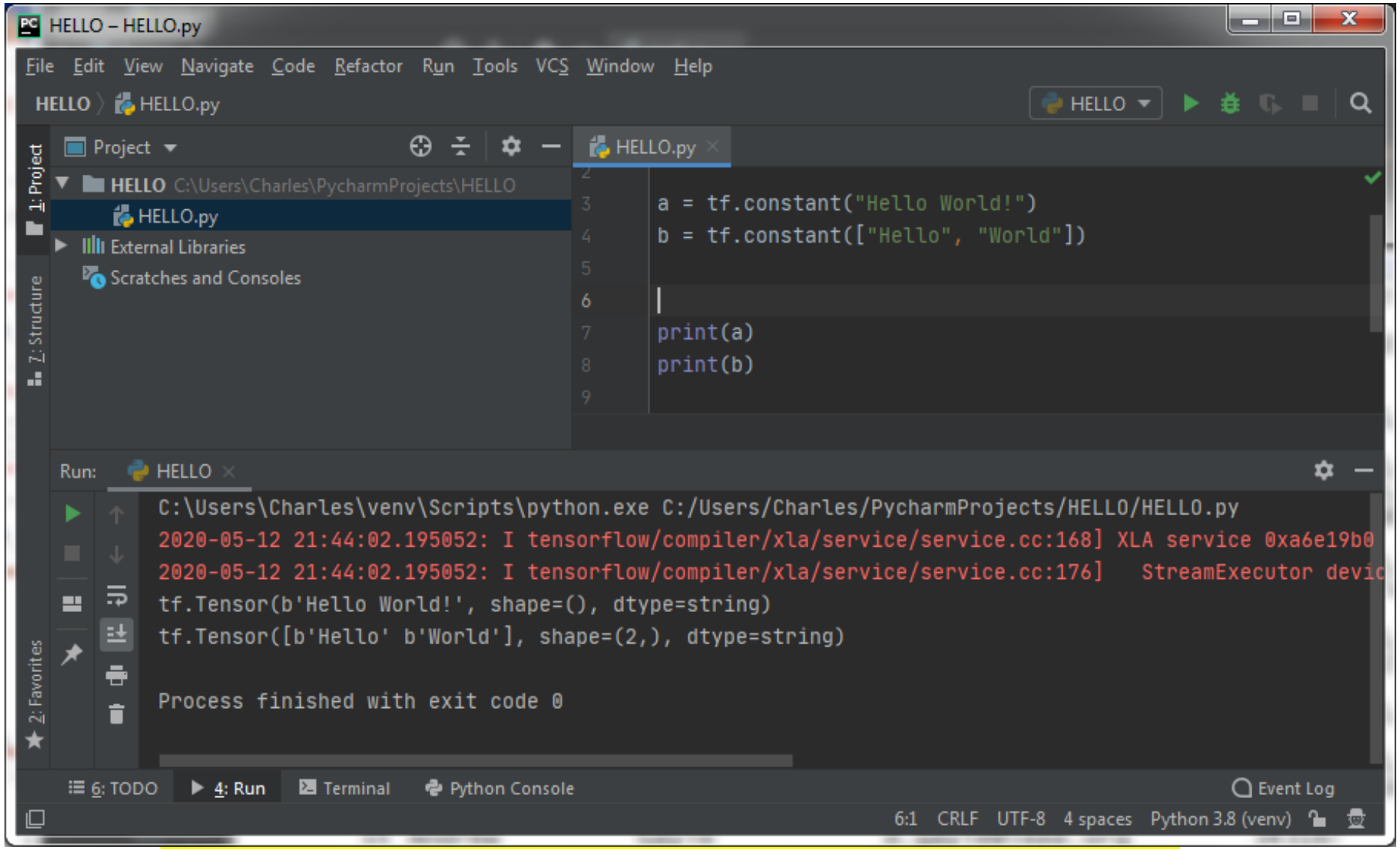
5/13 TensorFlow Hands-On Part 1: Hello World!

5/14 TensorFlow Hands-On Part 2: Defining and Building Your Network

5/15 TensorFlow Hands-On Part 3: Teaching and Testing and Conclusion

Hello World in TensorFlow

- The traditional “Hello World” in computerese is print(“Hello World”);
- The traditional “Hello World” in embedded is <clr GPIO3.2>
- The traditional “Hello World” in neural networks is to train, test, and execute the MNIST data set.



Question 1 – What is the difference between the two outputs?

What IS the MNIST

- Modified National Institute of Standards and Technology database
- Original database was half census workers' writing and half HS students
- MNIST is a mixture of both

60,000 training images
10,000 testing images

28x28 pixels, 256 bits each

http://yann.lecun.com/exdb/mnist/

Want to add characters? EMNIST

- Cohen et al at Western Sydney 2017
- Alphanumeric database
- Upper and Lower case
- 240,000 training
- 40,000 testing
- Over 500 writers
- Can be sorted by writer, page, field
- https://www.westernsydney.edu.au/bens/home/reproducible_research/emnist

Models

- MNIST is the typical demonstration dataset but there are many many more.
- Many of these are available directly through TensorFlow
- Not part of the code – downloaded as needed
- Audio, Image, Image classification, Object detection, text, etc
- Current list always at tfds.list_builders()

https://colab.research.google.com/github/tensorflow/datasets/blob/master/docs/overview.ipynb

Find available datasets

All dataset builders are subclass of `tfds.core.DatasetBuilder`. To get the list of available builders, uses `tfds.list_builders()` or look at our catalog.

```
tfds.list_builders()
```

```
'math_dataset',
'mnist',
'mnist_corrupted',
'movie_rationales',
'moving_mnist',
'multi_news',
'multi_nli',
'multi_nli_mismatch',
'natural_questions',
'newsroom',
'nsynth',
'omniglot',
'open_images_challenge2019_detection',
'open_images_v4',
'opinosis',
'oxford_flowers102',
'oxford_iiit_pet',
'para_crawl',
'patch_camelyon',
'pet_finder',
'places365_small',
'plant_leaves',
'plant_village',
'plantae_k',
```

To load a training database

- Remember that TF2 includes many keras extensions
- Keras.datasets both loads the datasets but includes the operations for loading sections of the database, whether for training, testing, or other uses
- We load the datasets into tensors

Loading the MNIST

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# we just converted the original data from integer to floating point
```

In Jupyter (Colab)

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

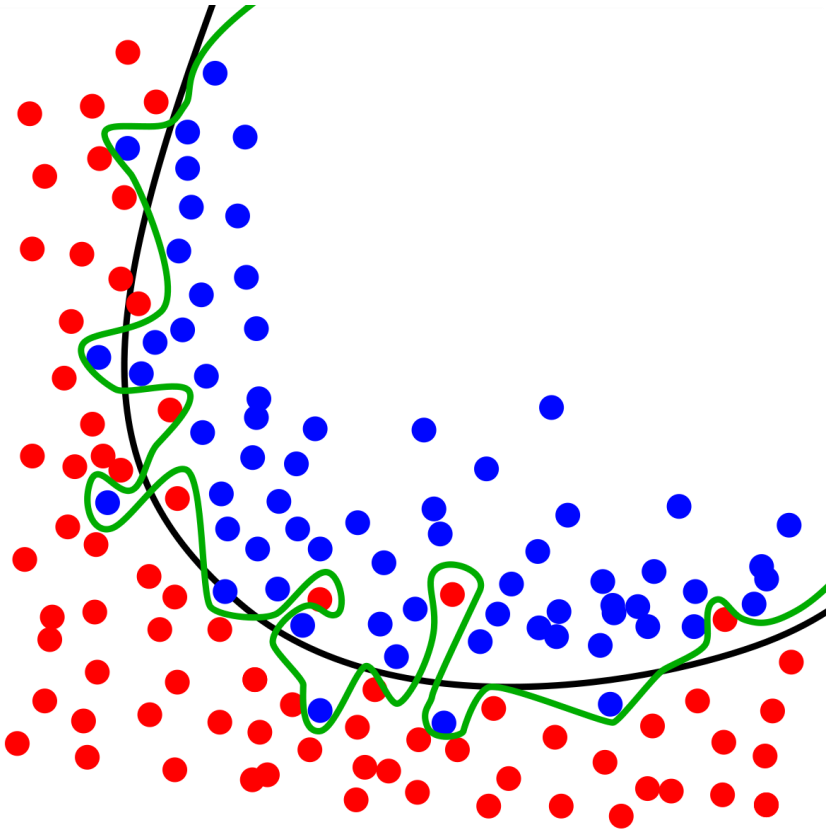
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

Building a Network

- Keras also adds extremely powerful tools for building a neural network
- To process the MNIST database we need the following:
  - Inputs from a 28x28 matrix
  - 10 outputs to signify the probability of each digit
  - At least one hidden layer

Overfitting

- An important consideration when training a network is the concept of overfitting
- In regression, this is making a model that fits one example too perfectly, making its fit worse for other models
- In training a NN, we typically introduce random ‘dropouts’ or noise



How Big a Hidden Layer(s)?

- There are few ‘hard and fast’ rules to the size and number of hidden layers
- The more complex and varied, the more you may need additional layers
- For a single hidden layer, N should be between the size of the input and the output
- The larger the N, the longer calculations take
- We will pick 128 for now (784 > 128 > 10)

Question 2 – Can we do without a hidden layer? When?

keras.layers

- The models.Sequential method stacks a series of network layers into a single model
- Each layer can be one of many types – we will use three in our model:
  - Flatten takes a 2-D array (such as an image) and ‘flattens’ it out into a vector
  - Dense creates a densely connected (traditional) layer of nodes
  - Dropout is used to create the random dropouts to help prevent overfitting

Our Initial Code

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

We Need Output!

- Keras has powerful training and model API
- Uses numpy arrays

```
predictions=model(x_train[:1]).numpy()
```

- gives us the raw predicted output ('logits')

- We can then feed these into Softmax to create our 'probabilities' for each output

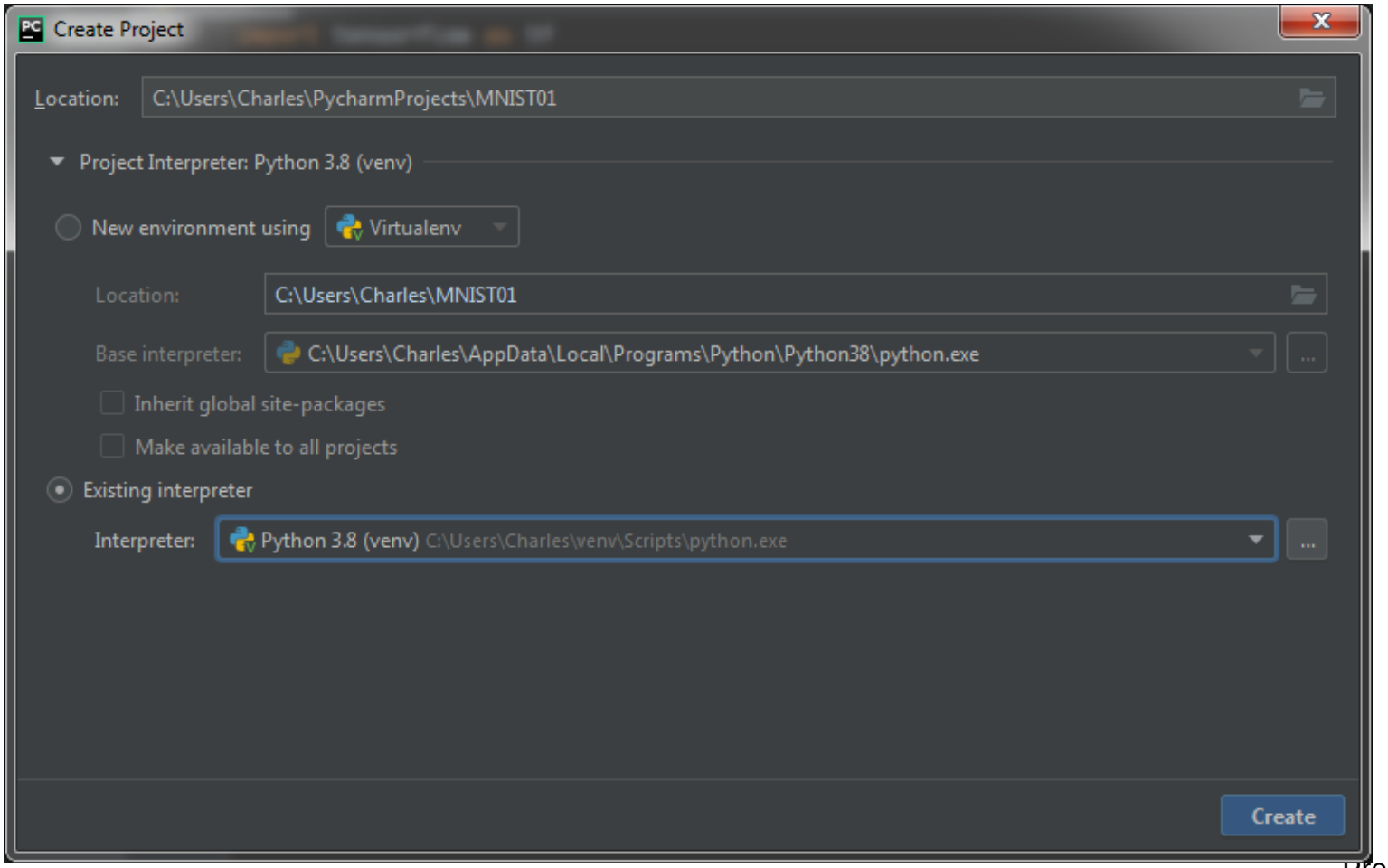
```
tf.nn.softmax(predictions).numpy()
```

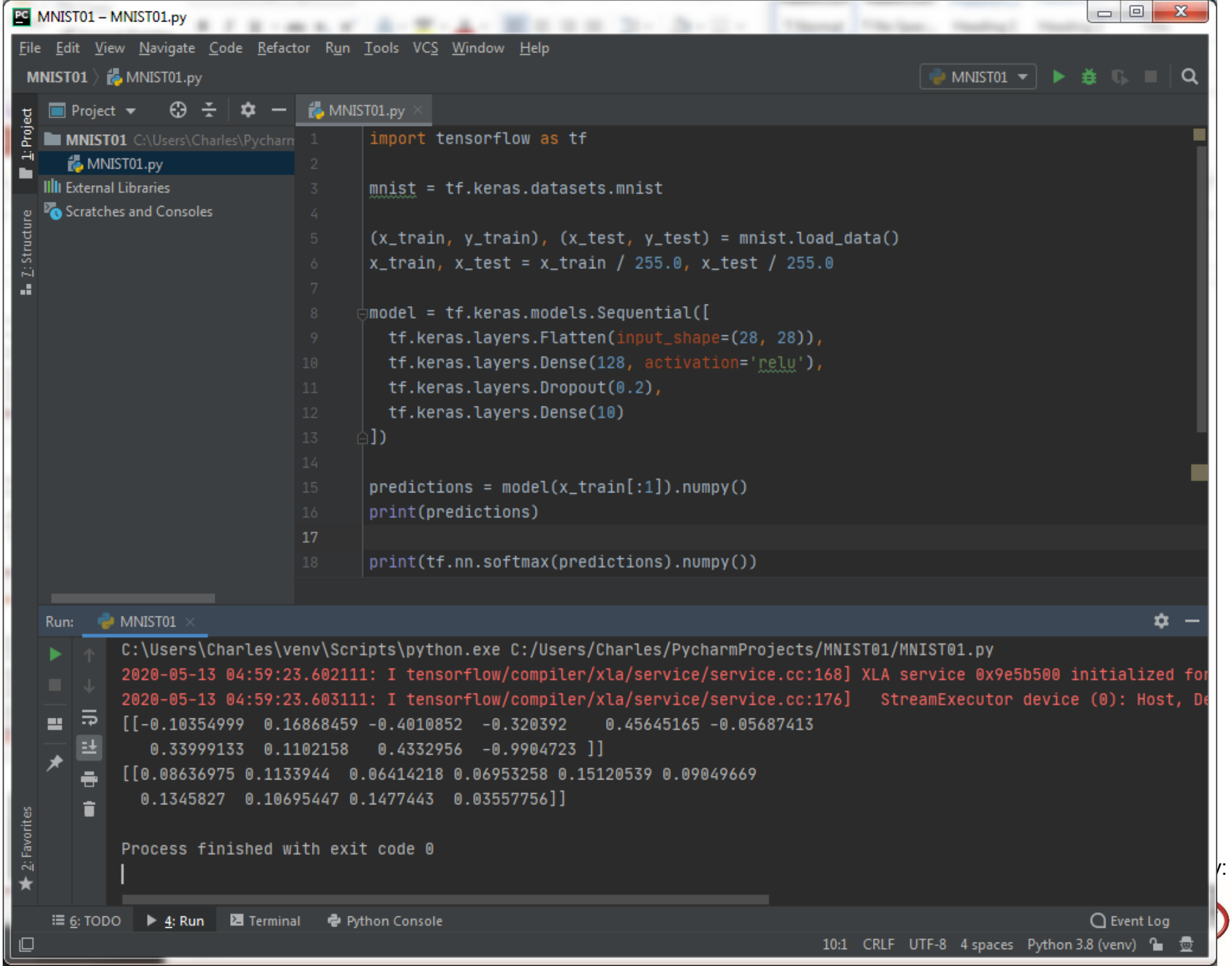
Our code so far

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
predictions = model(x_train[:1]).numpy()
print(predictions)
print(tf.nn.softmax(predictions).numpy())
```

PyCharm – Use our venv




MNIST01 – MNIST01.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
MNIST01 〉 MNIST01.py
MNIST01
Project
MNIST01 C:\Users\Charles\Pycharm
MNIST01.py
External Libraries
Scratches and Consoles
MNIST01.py
1 import tensorflow as tf
2
3 mnist = tf.keras.datasets.mnist
4
5 (x_train, y_train), (x_test, y_test) = mnist.load_data()
6 x_train, x_test = x_train / 255.0, x_test / 255.0
7
8 model = tf.keras.models.Sequential([
9 tf.keras.layers.Flatten(input_shape=(28, 28)),
10 tf.keras.layers.Dense(128, activation='relu'),
11 tf.keras.layers.Dropout(0.2),
12 tf.keras.layers.Dense(10)
13])
14
15 predictions = model(x_train[:1]).numpy()
16 print(predictions)
17
18 print(tf.nn.softmax(predictions).numpy())
Run: MNIST01
C:\Users\Charles\venv\Scripts\python.exe C:/Users/Charles/PycharmProjects/MNIST01/MNIST01.py
2020-05-13 04:59:23.602111: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x9e5b500 initialized for
2020-05-13 04:59:23.603111: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, De
[[-0.10354999 0.16868459 -0.4010852 -0.320392 0.45645165 -0.05687413
0.33999133 0.1102158 0.4332956 -0.9904723]]
[[0.08636975 0.1133944 0.06414218 0.06953258 0.15120539 0.09049669
0.1345827 0.10695447 0.1477443 0.03557756]]
Process finished with exit code 0
6: TODO
4: Run
Terminal
Python Console
Event Log
10:1 CRLF UTF-8 4 spaces Python 3.8 (venv)

The Output

```
[[0.05303392 0.0802143
0.15128678 0.16979037
0.15091117 0.08144777
0.11463938 0.07014603
0.04761226 0.08091807]]
```

- The values are all over the place, but average 0.1 (1/10)
- The model is not trained, of course!
- Weights and biases are set to normalized random values

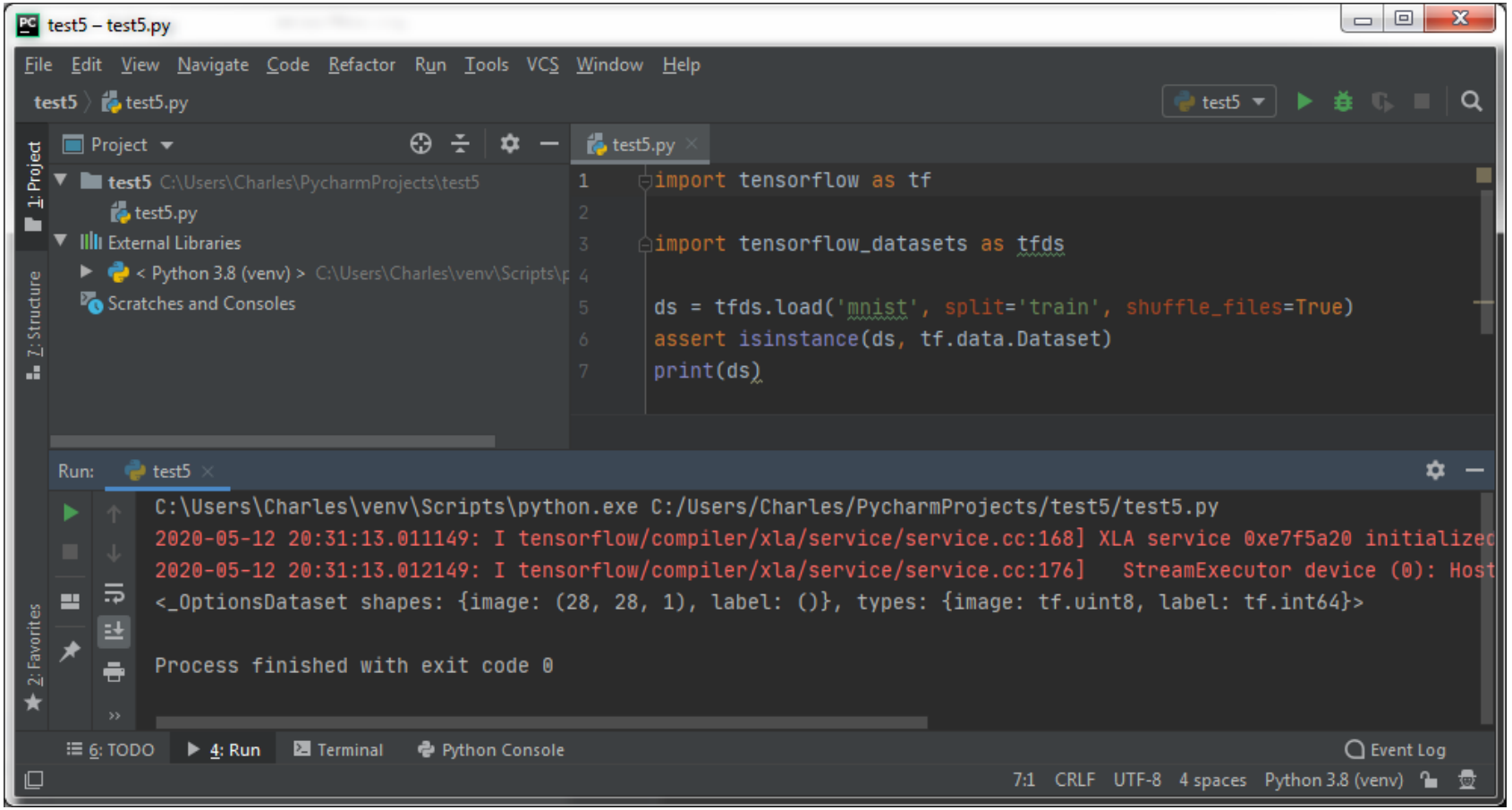
Looking at MNIST Record Structure

```
import tensorflow as tf

import tensorflow_datasets as tfds

ds = tfds.load('mnist', split='train', shuffle_files=True)
assert isinstance(ds, tf.data.Dataset)
print(ds)
```

Results



<_OptionsDataset shapes: {image: (28, 28, 1), label: ()}, types: {image: tf.uint8, label: tf.int64}>

Tomorrow!

- We built our neural network – now to train and test it!
- We will look at some basics of speeding the network up
- A bit on optimization
- Porting TF to other processors

Question 3 – What applications are you [considering] building?

This Week's Agenda

5/11 A Brief History of Artificial Neural Networks

5/12 Neural Network Simulation and Programming

5/13 TensorFlow Hands-On Part 1: Hello World!

5/14 TensorFlow Hands-On Part 2: Defining and Building Your Network

5/15 TensorFlow Hands-On Part 3: Teaching and Testing and Conclusion

Please stick around as I answer your questions!

- Please give me a moment to scroll back through the chat window to find your questions
- I will stay on chat as long as it takes to answer!
- I am available to answer simple questions or to consult (or offer in-house training for your company)
  c.j.lord@ieee.org
  http://www.blueridgetechnc.com
  http://www.linkedin.com/in/charleslord
  Twitter: @charleslord
  https://www.github.com/bradatraining