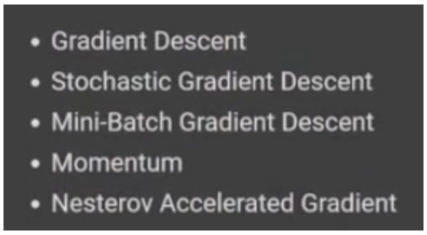
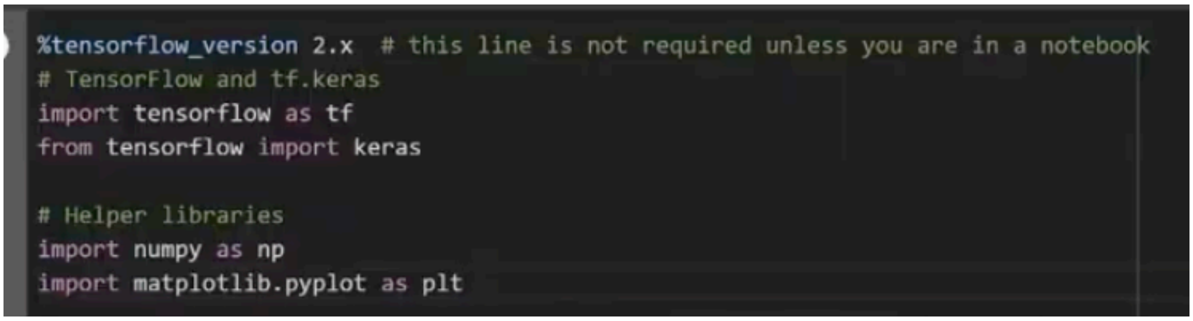
- ***Optimisers***
  - -> these are the algorithms which do the gradient descent and back propagation in the neural networks



- Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- Momentum
- Nesterov Accelerated Gradient

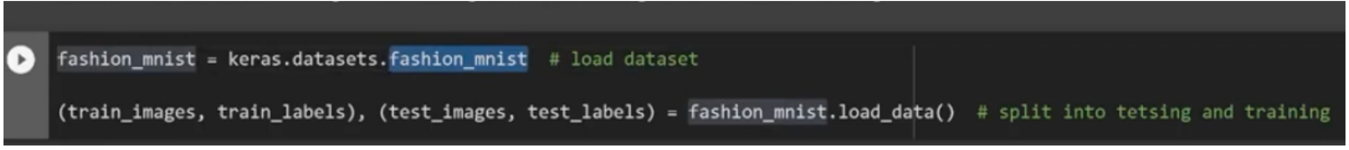
  - <- these are all of the different options for these optimisers

- ***Neural network example***
  - -> he imports the code numpy, tensor flow, matplot lib



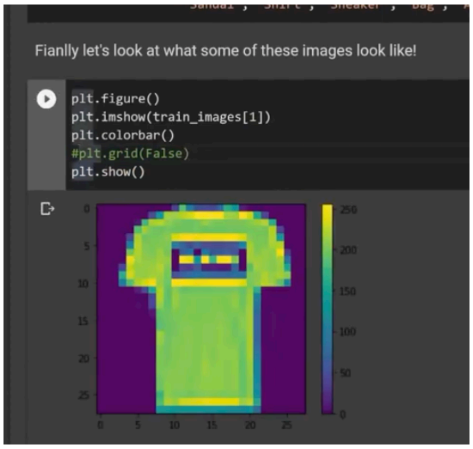
```
%tensorflow_version 2.x  # this line is not required unless you are in a notebook
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
```

  - ***-> the dataset***
    - -> it's a fashion dataset
    - -> images of clothes
    - -> <u>loading it in using keras</u>



```
fashion_mnist = keras.datasets.fashion_mnist  # load dataset

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()  # split into tetsing and training
```

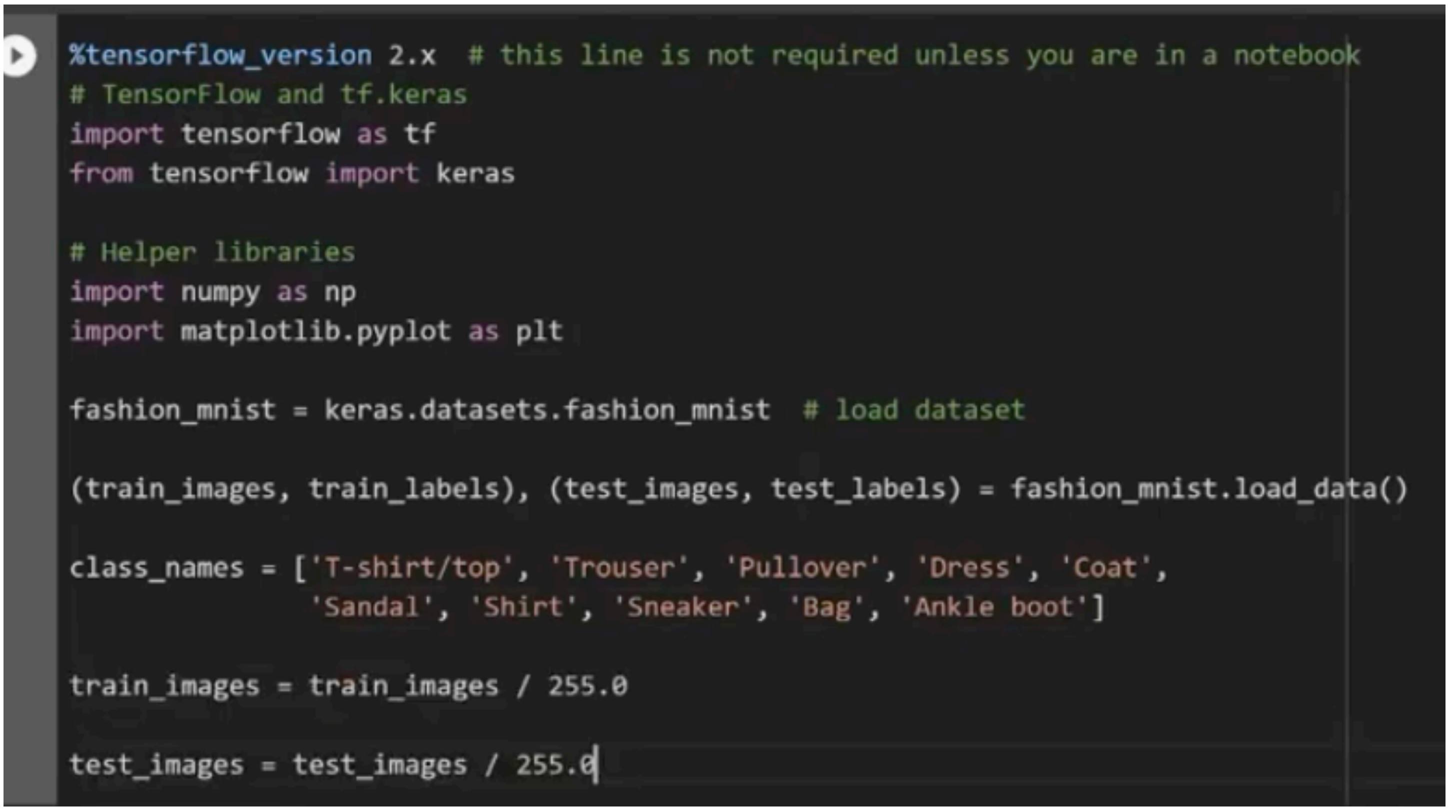
      - -> this splits the data into the sets which we need
      - -> then stores it (second line on in the cell above) into tuples
    - ***-> then he prints the shape of the images -> train_images.shape <- this returns the number of pixels***
      - -> the pixels are stored in an numpy array
      - <u>-> he's printing out different entries -> the images are matrices</u>
        - <u>-> you store images as mathematical objects by putting their pixels into arrays / matrices</u>
        - -> you can also have rgb values -> these are grayscale
    - ***-> then the training labels***
      - -> there are 10 different clothing items in the dataset
      - -> <u>in other words the labels in a dataset are the different possible outcomes / things which the image (in this example) could be of</u>
      - -> he's printing out different images in the dataset



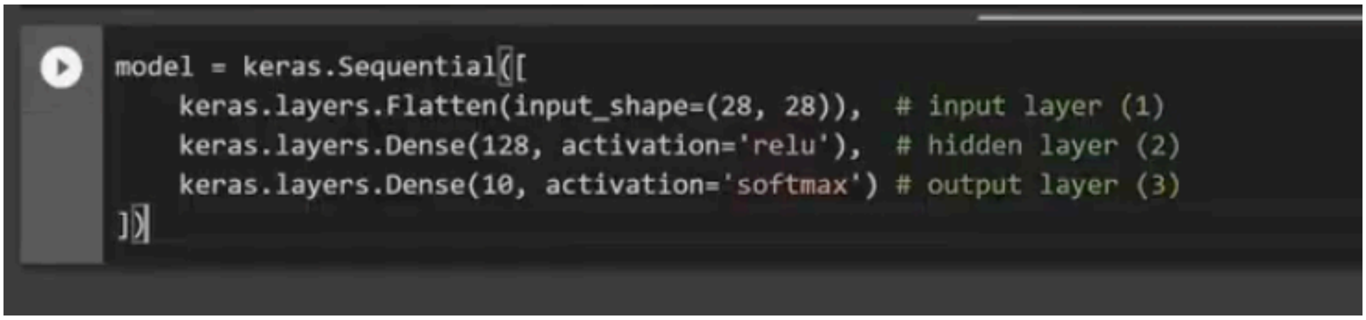
Fianlly let's look at what some of these images look like!

```
plt.figure()
plt.imshow(train_images[1])
plt.colorbar()
#plt.grid(False)
plt.show()
```

        - ***-> using an activation function to normalise the values in the dataset***
            - -> he's using a tanh activation function to move them in between -1 and 1
            - -> this reduces the spread of the data and makes it easier for the algorithm to update the data
            - -> in this case he's just divided it by the number of training images to normalise to
- ***This is all of the code so far***
    - <u>-> optimiser functions are functions which do gradient descent and back propagation algorithms -> i.e reducing the failure rate of the model's predictions and which goes back into the layers of the neural network and updates the weights and constants (biases) in that model</u>



```
%tensorflow_version 2.x  # this line is not required unless you are in a notebook
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist  # load dataset

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

train_images = train_images / 255.0

test_images = test_images / 255.0
```



```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),  # input layer (1)
    keras.layers.Dense(128, activation='relu'),  # hidden layer (2)
    keras.layers.Dense(10, activation='softmax') # output layer (3)
])
```