TensorFlow/Keras

Python

Python Basic Syntax

TensorFlow & Keras

Keras Basic Syntax

Contents

Python

- An open-source programming language
- MATLAB-like syntax
- Faster than MATLAB
- Simpler than C/C++
- Supported by a vast community

Python

- Two main versions: 2.x and 3.x (recommended)
- Can be installed via python.org, Anaconda, etc.

1. Download Anaconda 2019.03 with Python 3.7
   - https://repo.anaconda.com/archive/Anaconda3-2019.03-Windows-x86_64.exe
   - https://repo.anaconda.com/archive/Anaconda3-2019.03-MacOSX-x86_64.pkg
2. Follow the instructions to install Python 3.7

Python

- Python IDEs: **Jupyter**, PyCharm, etc.

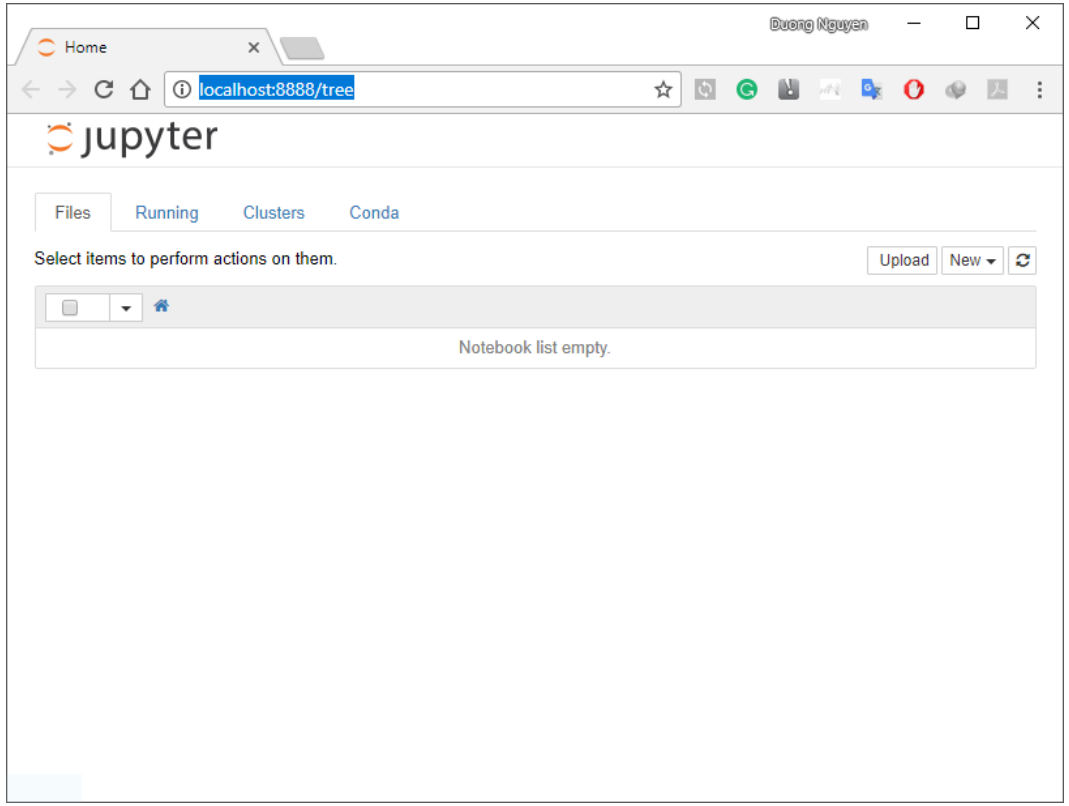
- Jupyter:
  - Installed by default with Anaconda
  - Low resource requirement
  - Interactive, visualizable environment

Python

Working with Jupyter

1. Open Command Prompt
2. Enter `jupyter notebook --notebook-dir="project_dir"`

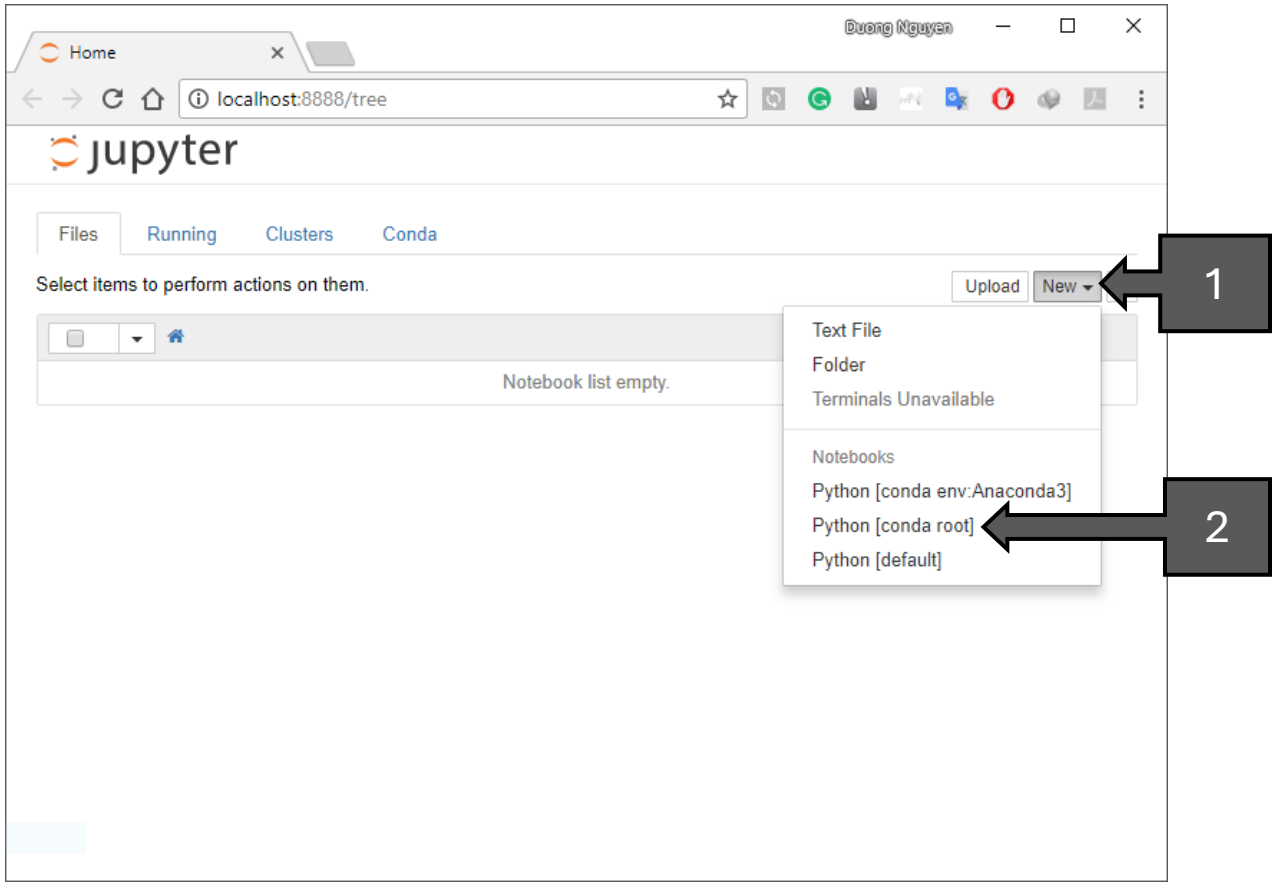
For example: "D:\DL_projs"



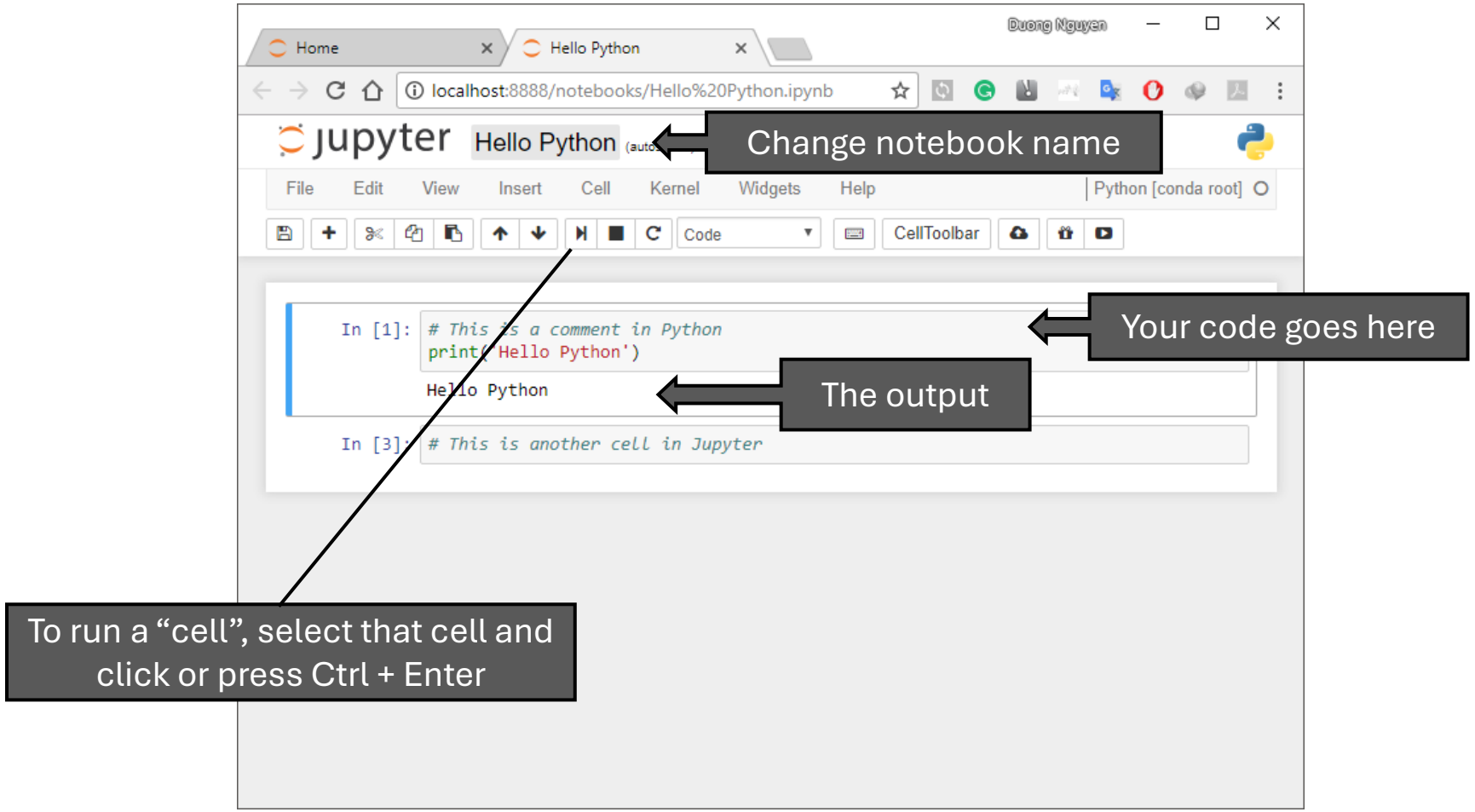
Python

Working with Jupyter

3. Select New > Python [conda root]



Python


Home
Hello Python
Duong Nguyen
localhost:8888/notebooks/Hello%20Python.ipynb
jupyter
Hello Python
Change notebook name
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Python [conda root]
Code
CellToolbar
In [1]:
# This is a comment in Python
print('Hello Python')
Hello Python
Your code goes here
The output
In [3]:
This is another cell in Jupyter
To run a “cell”, select that cell and click or press Ctrl + Enter

Python Basic Syntax

- Basic functions and structures

```
In [4]: print('text')
        text
```

```
In [7]: print('%s%d, %s%.3f' % ('an integer = ', 7, 'a real number = ', 17.2))
        an integer = 7, a real number = 17.200
```

```
In [8]: for i in range(7):
            if i % 2 == 0:
                print(i)
        0
        2
        4
        6
```

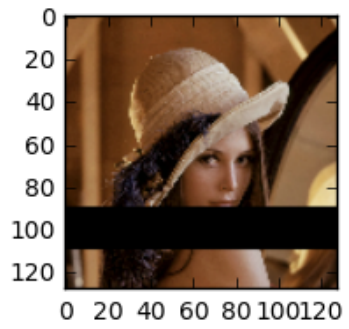
Python Basic Syntax

- Matrix and image

```
In [20]: import numpy as np
         import cv2
         import matplotlib.pyplot as plt
```

To install cv2 (OpenCV):
- Open Command Prompt as admin
- Enter `conda install -c anaconda opencv`

```
In [27]: img = cv2.imread('lena.jpg')
         img = cv2.resize(img, (128, 128))
         img[90:110, :] = 0

         plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
         plt.show()
```



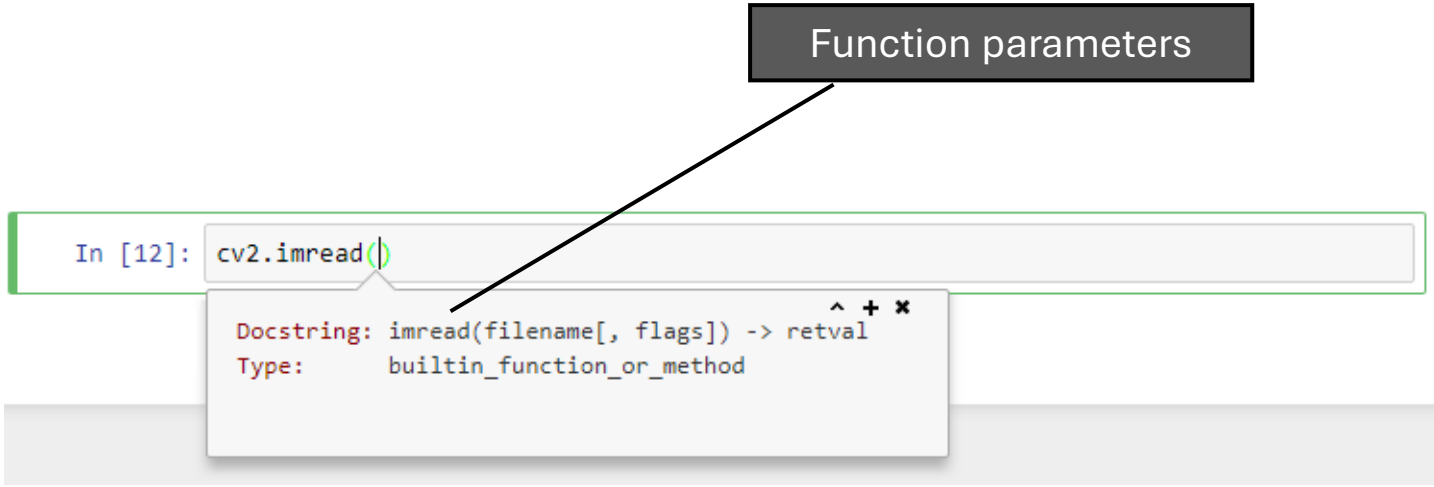
For more information, visit http://www.numpy.org/

Python Basic Syntax

- Jupyter **tips: Authocomplete(press Tab)**

Python Basic Syntax

- Jupyter tips: Function hint(press Shift+Tab)



TensorFlow & Keras

- TensorFlow:
  - A computing library by Google
  - Building, training, and evaluating deep learning models

- Keras:
  - Created by a Google engineer
  - High-level API by for TensorFlow
  - Simple and fast way to use TensorFlow
  - Easier to understand
  - Does not require graph definition, which is considered as the most difficult concept in TensorFlow

TensorFlow & Keras

References

[1] TensorFlow, "TensorFlow," [Online]. Available: https://www.tensorflow.org/

[2] Keras, "Keras," [Online]. Available: https://keras.io/

TensorFlow & Keras

TensorFlow

```
# A convolutional layer
W = tf.Variable(tf.random_normal([3, 3, 1, 32]))
b = tf.Variable(tf.random_normal([32]))
x = tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
x = tf.nn.bias_add(x, b)
x = tf.nn.relu(x)
```

5 lines for such a simple convolutional layer!

Keras

```
# A convolutional layer
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
```

Only 1 line of code

I am going to define a Conv layer

x = Conv2D(.)


```
W = tf.Variable(.)
b = tf.Variable(.)
x = tf.nn.conv2d(.)
x = tf.nn.bias_add(.)
x = tf.nn.relu(.)
```

TensorFlow & Keras

- Working Environment:
  - Windows x64
  - Python 3.5 or later
  - TensorFlow
  - CUDA 8.0 or later (optional, GPU only)
  - cuDNN v5 or later (optional, GPU only)

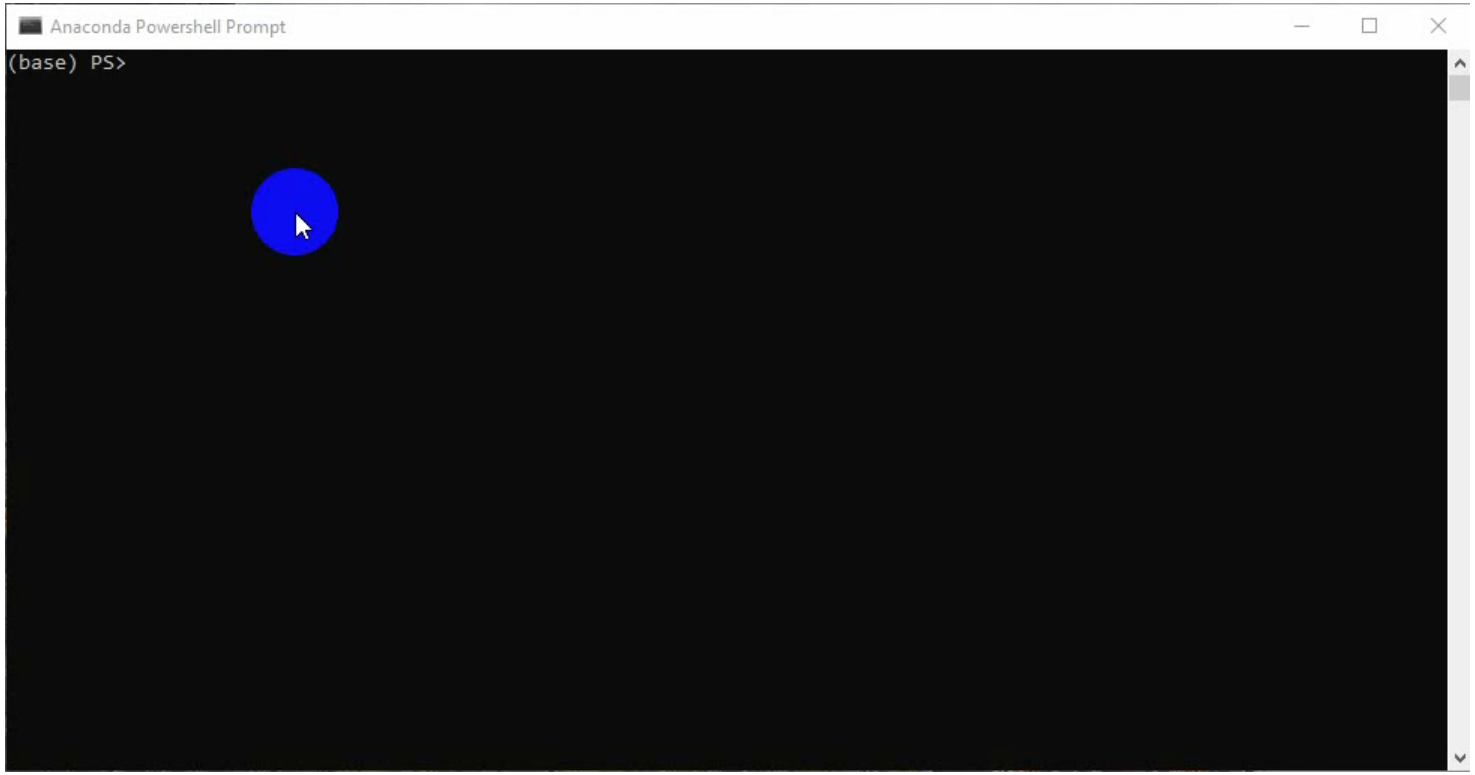
Anaconda will install automatically when we install keras-gpu/tensorflow-gpu

- To install TensorFlow/Keras, please visit
  - https://github.com/nhduong/intro_deep

TensorFlow & Keras

Environment Setup 1 (with GPUs)

1. Download and install [Anaconda 2019.03 with Python 3.7]
2. Open Command Prompt as Administrator
3. Install `TensorFlow-GPU` by entering `conda install -c anaconda tensorflow-gpu`
4. Enter `conda install -c anaconda keras-gpu` to install Keras-GPU



TensorFlow & Keras

Environment Setup 2 (with GPU Tesla K80 + Google Colab)

A ***free of charge*** way to experience training deep models with high-performance GPU!

1. Visit Google Colaboratory
2. Sign in with you personal Google account
3. Menu `File` > `New Python 3 notebook`
4. Menu `Runtime` > `Change runtime type` > `Hardware acceleration` > `GPU` > `Save`
   (Note that python source codes will be saved in your Google Drive, and a work shift lasts for 12 hours)
5. Let's get started with this Jupyter Notebook (how to use GPU, execute Linux commands, install Python packages, and read data from Google Drive)

Environment Setup 3 (with CPUs)

1. Download and install Anaconda 2019.03 with Python 3.7
2. Open Command Prompt as Administrator
3. Install `TensorFlow` by entering `conda install -c conda-forge tensorflow`
4. Enter `conda install -c conda-forge keras` to install Keras

Keras Basic Syntax

- Check Keras version and list all available GPUs in your PC

```
In [2]: import keras
```

```
In [3]: print(keras.__version__)
        print(keras.backend.tensorflow_backend._get_available_gpus())

        2.0.9
        ['/device:GPU:0']
```

Keras Basic Syntax

Fundamental layers in deep neural networks:

- Input: how the input will be, e.g., an image, a video, etc.
- Dense: fully connected layer in MLP and CNN
- Convolutional layer
- Max pooling layer
- Dropout: randomly dropping input units
- LSTM layer
- Flatten: vectorize a tensor (image, video, feature maps, etc.)

Keras Basic Syntax

Fundamental layers

- Input layer clarifies input size, e.g., a 28x28 grayscale image

```
In [7]: from keras.layers import Input
        x = Input(shape=(28, 28, 1), name='input_layer')
```

-

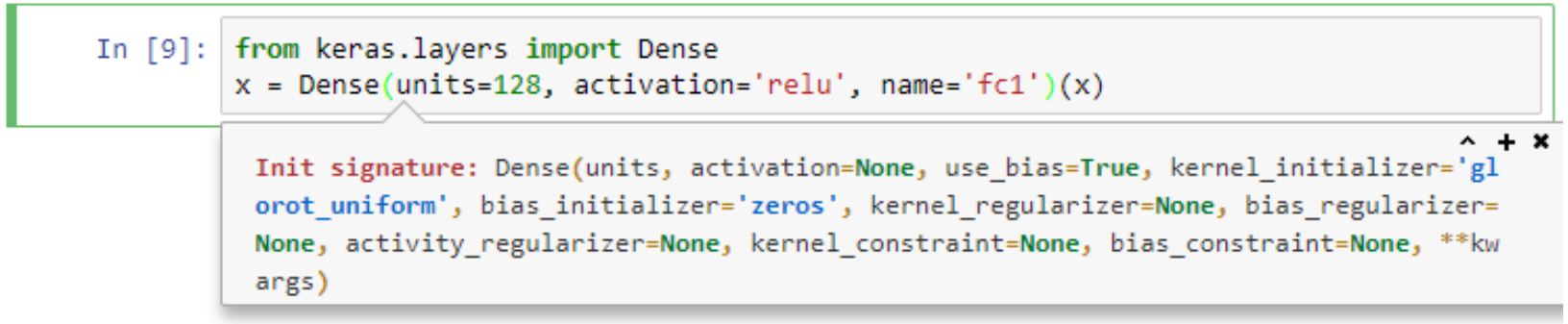
```
In [9]: from keras.layers import Dense
        x = Dense(units=128, activation='relu', name='fc1')(x)
```

The activation function can be a sigmoid or softmax for classification

Keras Basic Syntax

Reminder!

Use *function hint* in Jupyter (press Shift + Tab)



or head over to https://keras.io/ to study more about a function

Keras Basic Syntax

Fundamental layers

- Conv2D declares a 2D convolutional layer

```
In [11]: from keras.layers import Conv2D
         x = Conv2D(filters=128, kernel_size=(3, 3), activation='relu', name='conv1')(x)
```

- Use MaxPooling2D to define a max pooling layer

```
In [13]: from keras.layers import MaxPooling2D
         x = MaxPooling2D(pool_size=(2, 2), name='maxpool1')(x)
```

Keras Basic Syntax

Fundamental layers

- Dropout layer is worth avoiding overfitting

```
In [14]: from keras.layers import Dropout
         x = Dropout(0.75, name='drop1')(x)
```

75% of units in x will be set to 0 **randomly**

Keras Basic Syntax

Fundamental layers

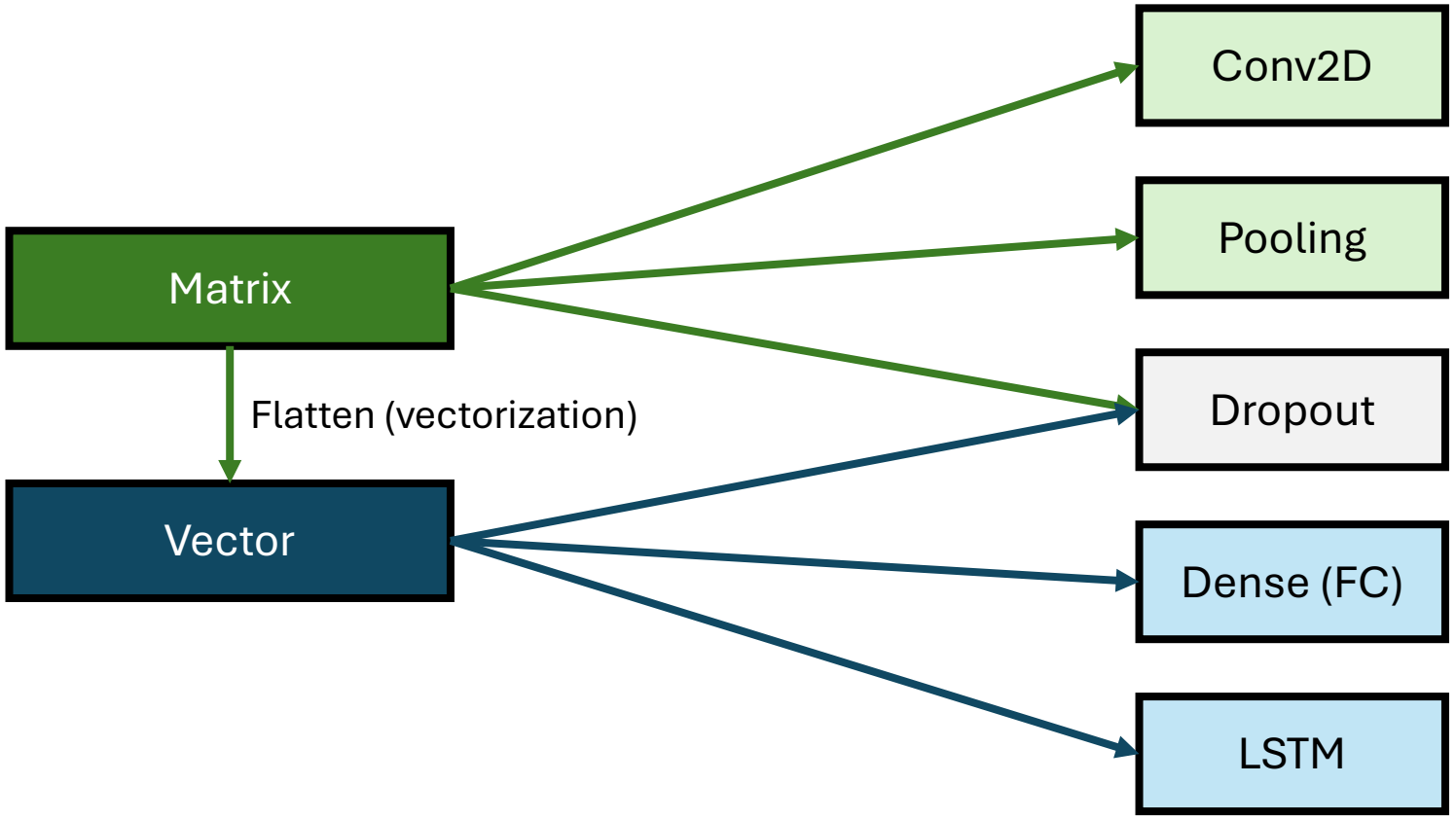
- LSTM layer is useful for temporal data

```
In [ ]: from keras.layers import LSTM
        x = LSTM(units=64, input_shape=(timesteps, feature_dim), name='lstm1')(x)
```

- Before using LSTM/FC, vectorize feature maps using Flatten

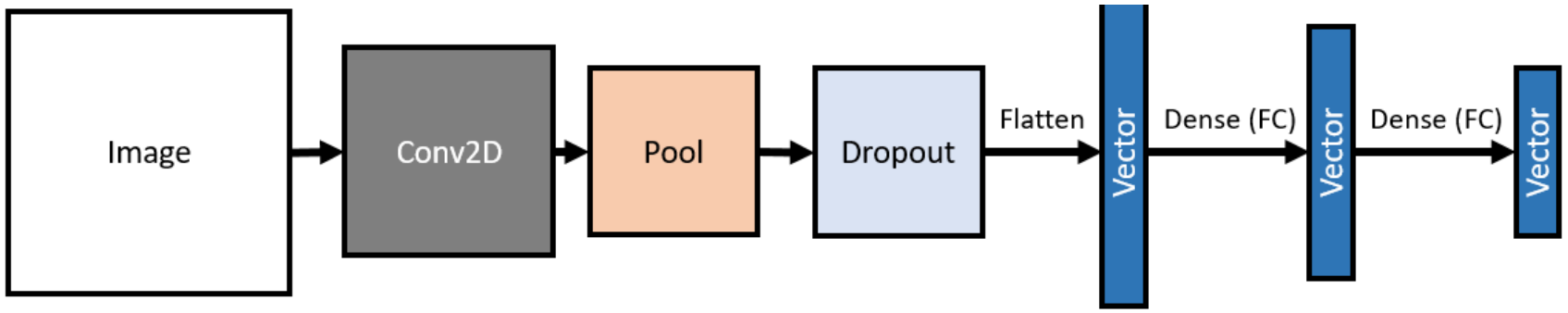
```
In [18]: from keras.layers import Flatten
         x = Flatten(name='flatten')(x)
```

Keras Basic Syntax



Keras Basic Syntax

General deep neural network models



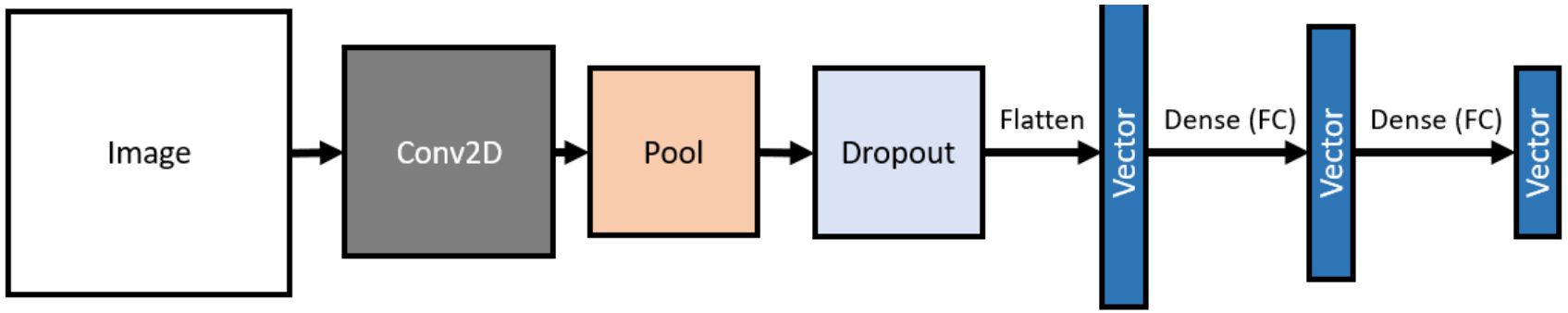
Keras Basic Syntax

A basic Keras model: a sequence of layers!

In [25]:
```
from keras import Model
from keras.layers import Input, Conv2D, MaxPooling2D, Dropout, Flatten, Dense

img = Input(shape=(28, 28, 1), name='input_layer')

x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='conv')(img)
x = MaxPooling2D(pool_size=(2, 2), name='maxpool')(x)
x = Dropout(0.5, name='drop')(x)

x = Flatten(name='flatten')(x)

x = Dense(units=128, activation='relu', name='fc')(x)
class_label = Dense(units=10, activation='softmax', name='softmax')(x)

model = Model(inputs=img, outputs=class_label, name='simple_cnn')
```

Keras Basic Syntax

In [25]:
```
from keras import Model
from keras.layers import Input, Conv2D, MaxPooling2D, Dropout, Flatten, Dense

img = Input(shape=(28, 28, 1), name='input_layer')

x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='conv')(img)
x = MaxPooling2D(pool_size=(2, 2), name='maxpool')(x)
x = Dropout(0.5, name='drop')(x)

x = Flatten(name='flatten')(x)

x = Dense(units=128, activation='relu', name='fc')(x)
class_label = Dense(units=10, activation='softmax', name='softmax')(x)

model = Model(inputs=img, outputs=class_label, name='simple_cnn')
```



Keras Basic Syntax

View model summary using summary(.) function

```
In [26]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 28, 28, 1)         0
_________________________________________________________________
conv (Conv2D)                (None, 26, 26, 64)        640
_________________________________________________________________
maxpool (MaxPooling2D)       (None, 13, 13, 64)        0
_________________________________________________________________
drop (Dropout)               (None, 13, 13, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 10816)             0
_________________________________________________________________
fc (Dense)                   (None, 128)               1384576
_________________________________________________________________
softmax (Dense)              (None, 10)                1290
=================================================================
Total params: 1,386,506
Trainable params: 1,386,506
Non-trainable params: 0
_________________________________________________________________
```

Keras Basic Syntax

Compiling a model by defining:

- Loss function, e.g., cross entropy, mean squared error, etc.
- Optimization algorithm, e.g., stochastic gradient descent, Adam, etc.
- Learning rate, e.g., 0.01, 0.001, etc.

```
In [27]: from keras import optimizers

         sgd = optimizers.SGD(lr=0.01) # Stochastic gradient descent, learning rate = .01
         model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

https://keras.io/optimizers/

Keras Basic Syntax

Train a model using fit(.) function which contains parameters:

- Training data, validation data
- Batch size
- Number of epochs

```
In [ ]: model.fit(X_train, y_train,
                  validation_data=(X_val, y_val),
                  batch_size=64,
                  epochs=100)
```

- Return: a trained model

https://keras.io/models/sequential/

Keras Basic Syntax

Evaluate a model with evaluate(.)

```
In [ ]: evaluate(X_test, y_test)
```

- Return: error/accuracy

THANK YOU!