Installing Anaconda Python 3.6, Wingware Python IDE, and Keras/Tensorflow

Installing Anaconda Python 3.6

You first need to install the Anaconda distribution of Python 3.6. It is important to note that we will be using Python 3.6, not the new Python 3.7 or older versions of Python. Keras/Tensorflow may not work with versions newer or older than Python 3.6.

If you already have Python installed (or think you have Python installed), type the following in the Command Prompt (PC) or Terminal (Mac) to confirm that you are using Python 3.6.x

```
python --version
```

If you do not have Python 3.6 installed, follow these instructions:

- Navigate to https://www.anaconda.com/download.

- The version for your operating system (Windows, MacOS, or Linux) should be selected automatically. You will first install the Anaconda Python 3.7 version and then create a Python 3.6 environment.

- Click on Python 3.7 to download; you will need click on the downloaded file to start the installation if it does not auto-install per the settings on your computer.

- Python 3.7 should now be installed on your computer. Confirm via the Command Prompt (PC) or Terminal (Mac)

```
python --version
```

On a PC, you may want to run the Anaconda3 (64-bit) → Anaconda Prompt to open a Python environment (with paths properly configured). On a Mac, you should be able to run this command right from the Terminal.

- You will now need to create a Python 3.6 environment. The following is from: https://conda.io/docs/user-guide/tasks/manage-environments.html

- You will now need to create a Python 3.6 environment and make it the default. From the Command Prompt (PC) or Terminal (Mac)

```
conda create -n py36 python=3.6 anaconda
```

This command will run for a while as it builds Python 3.6.

- Verify that your new Python 3.6 environment has been created

```
conda info --envs
```

`py36` should be listed

- Activate the Python 3.6 environment.

On Windows

```
activate py36
```

on Mac/Linux

```
source activate py36
```

- Verify that Python 3.6 is now the default

```
python --version
```

Further details on installing and uninstalling Anaconda are available at http://docs.anaconda.com/anaconda/install/.

If you have an older version of Anaconda, you may want to remove that version and install a fresh version, with instructions detailed on this web page.

While we verified that these instructions work on a PC and a Mac, they may not work on every computer. Please let us know if you have problems installing Python.

Installing Wingware Python IDE

For the homework assignments, you are free to turn in .py files or Jupyter Notebooks.

For longer homework assignments, it may be easier to write and debug your Python code in an Integrated Development Environment (IDE); if you want, you can then paste your working code into a Jupyter Notebook (or just turn in the raw .py code). Make sure you include all files needed to run your code (including any files we gave you).

You are free to use whatever IDE you like, whether it be Spyder, PyCharm, Wingware, Eclipse, or any other IDE that works with Python.

We have secured an educational license for Wingware IDE that you can use (installation instructions below). Wingware Python IDE is an IDE that has been specifically designed to work effectively with the Python language. It has a somewhat similar "look-and-feel" to the Matlab IDE, so it may be the easiest to learn for those with experience using Matlab.

Other IDEs may need to be configured to make sure that Python 3.6 (not 3.7 or an earlier version) is being used.

- Navigate to https://wingware.com/downloads. Get (Wing) Pro.

- The web page should auto-detect your operating system. Click on Installer button to the right of the page.

- Wingware IDE will download; you will need click on the downloaded file to start the installation if it does not auto-install per the settings on your computer.

- Complete the installation. For MacOS, you will need to drag the WingIDE app to the Applications folder, as instructed. For Windows, simply open and run the executable (you may need to run as Administrator). Refer to the installation instructions for additional information (https://wingware.com/doc/install/installing); Linux users should note the additional installation instructions (https://wingware.com/doc/install/linux-installation-detail).

- Wingware IDE should now be available for use.

- Run WingIDE. You will be warned that "Wing is running without a valid license". Select "Install and activate a permanent license. Enter license id:"
`6N31V-XB9HV-DQX74-ER84M`

You will need to be connected to the internet to activate the license. Note that providing your name and other contact information is optional.

- Before using WingIDE, you will need to complete the next step, which ensures that the IDE is using the correct Anaconda Python 3.6 distribution.

Setting up Wingware IDE for Python 3.6

The following is based on these instructions:
https://docs.anaconda.com/anaconda/user-guide/tasks/integration/wing/
The Anaconda web site should have information for setting up other Python IDEs.

- Find the path for your Python 3.6 by typing the following in the Command Prompt (Windows) or Terminal (Mac)
`which python`
There should be a path that includes something like
`Anaconda3\envs\py36\python.exe` (Windows)
or
`anaconda3/envs/py36/bin/python` (Mac)
Make sure you type in the full path below (including elements of the path before `Anaconda3`).

- Open WingIDE.

- On the menu, go to Project → Project Properties…

- Choose the Custom radio button to the right of Python Executable.

- In the text box, enter in the Python 3.6 path you noted above.

- Choose the Custom radio button to the right of Python Path.

- In the text box, enter in the Python 3.6 path you noted above (not including the `python.exe`).

- Quit Wing and rerun. The Python Shell should now show 3.6.x as the current version (rather than 3.7.x or another version)

We encourage you to read through some of the online manual and tutorial to understand the WingIDE environment in more detail:
Tutorial: https://wingware.com/doc/intro/tutorial
Manual: https://wingware.com/doc/manual

Installing Keras and Tensorflow

We will use Keras (https://keras.io) and TensorFlow (https://www.tensorflow.org) for some assignments. Keras runs on top of TensorFlow which both run within Python.

- Before installing Keras, you need to install TensorFlow
https://www.tensorflow.org/install/

You should be able to just run the following to install TensorFlow
`conda install tensorflow`
If that does not work, see the link above.

Unless you know you have a CUDA-enabled GPU card on your computer, we recommend you install the CPU-only version of TensorFlow. This installation will take a while to complete. On a Windows machine, make sure you are installing within the Anaconda Prompt and that you have the Python 3.6 environment enabled.

Note that Mac users need to be running macOS 10.12.6 (Sierra) or later. If you cannot install a recent version of the macOS on your hardware, we will look into other options for running Keras/Tensorflow code (such as running on ACCRE).

- Next, install Keras
https://keras.io/#installation

You should be able to just run the follow to install Keras
`conda install -c conda-forge keras`
If that does not work, see the link above.

- Now try to run the autoencoder.py program uploaded to Brightspace. It trains a neural network called an autoencoder to reproduce hand-written digits. If the program works, it should show the results of training in the Python Debug I/O window (if using Wing IDE) and end by displaying images of digits in a figure window (the upper row are input test images, the lower row are reproductions made by the trained autoencoder).

Here is the code in autoencoder.py:

```
# autoencoder.py
# uses conventions from Keras functional API
# from https://blog.keras.io/building-autoencoders-in-keras.html

from keras.layers import Input, Dense
from keras.models import Model

# this is the size of our encoded representations
encoding_dim = 32
# 32 floats -> compression of factor 24.5, assuming the input is 784 floats

# this is our input placeholder
input_img = Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = Dense(784, activation='sigmoid')(encoded)

# this model maps an input to its reconstruction
autoencoder = Model(input_img, decoded)

# this model maps an input to its encoded representation
encoder = Model(input_img, encoded)
```

```
# create a placeholder for an encoded (32-dimensional) input
encoded_input = Input(shape=(encoding_dim,))
# retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# create the decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))

# compile model
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')

# prepare inputs (from MNIST databse)
from keras.datasets import mnist
import numpy as np
(x_train, _), (x_test, _) = mnist.load_data()

# normalize images
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
print(x_train.shape)
print(x_test.shape)

# train autoencoder
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

# encode and decode some digits
# note that we take them from the *test* set
encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)

# use Matplotlib
import matplotlib.pyplot as plt

n = 10  # how many digits we will display
plt.figure(figsize=(20, 4))
for i in range(n):
    # display original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # display reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```