PyPlot Basics

Our sample data set for this course is the NLDL Data set from F. Kubke.

NLDL stands for "Nucleus Laminaris Delay Lines". The data in each file describes the voltage over time at a specific location in the brain.

In simple terms an electrical stimulation was applied at the point where sound enters the brain, and then electrodes measured the voltage over time at a range of locations.

By measuring the time delay from stimulation to peak voltage for each data file, we can obtain secondary data which can be combined to determine whether the stimulation came from the left ear or right ear.

Our goal here is to plot the voltage over time as recorded in a single file (for a single location).

The resulting chart will be a time series showing the voltage waveform.

Loading the Data

We're using an external module so it's best to enable autoreload.

This means if we make any changes to the module, the script here will be automatically updated.

To use the Jupyter autoreload functionality, we need to **load the extension** and then **initialise autoreload**.

From the autoreload documentation (https://ipython.org/ipython-doc/3/config/extensions/autoreload.html) we can see that the "2" option means "always reload all modules":

> `%autoreload`
> Reload all modules (except those excluded by %aimport) automatically now.
>
> `%autoreload 0`
> Disable automatic reloading.
>
> `%autoreload 1`
> Reload all modules imported with %aimport every time before executing the Python code typed.
>
> `%autoreload 2`
> Reload all modules (except those excluded by %aimport) every time before executing the Python code typed.

In [ ]:

```
%load_ext autoreload
%autoreload 2
```

Next you need to import the NLDL parser module, which is in the **libs** folder.

This module can take one or more NLDL files and convert them to Python objects for you.

Also import pprint (pretty print) to make the printed data easier to inspect.

In [ ]:

```
from libs.nldl_parser import *
from pprint import *
```

Set the source file. Let's just start with one.

In [ ]:

```
source_file = 'data/TEK0000.CSV'
```

Create a new `parser`. Pass it the `source_file` and receive back the `data` object.

In [ ]:

```
parser = NLDLParser()
data = parser.parse_file(source_file)
```

In [ ]:

```
print("File Data:")
pprint(data)
```

Generating a Chart

Before we start, let's make it so that any charts we generate display on this page rather than opening in a new window.

We use the `inline` command to plot inside the notebook rather than having a new window open.

Importing `Image` from IPython allows us to show images here in our notebook.

In [ ]:

```
%matplotlib inline
from IPython.display import Image
```

We then need to import the relevant libraries. At this point, all we need is **pyplot** for creating the charts and **numpy** for working with numbers.

In [ ]:

```
import matplotlib.pyplot as pyplot
import numpy
```

A basic line chart is made of x and y values.

To generate a chart, we need (at a bare minimum) the following steps:

- Create separate lists of all the values for the **x** axis and all the values for the **y** axis.
- Pass the x and y lists to **pyplot.plot()**.
- Use **pyplot.show()** to display the result.

In [ ]:

```
x_data = [0,1,2,3,4,5,6,7,8,9]
y_data = [1,2,4,5,7,9,10,7,5,2]

pyplot.plot(x_data, y_data)
pyplot.show()
```

Plotting the NLDL Data

In our NLDL data, we want to plot the values from the **readings** list.

The **x** axis will be **time**. The **y** axis will be **voltage**.

Have another look at the printed NLDL data we created earlier and identify where you can find the **time** and **voltage** for each reading.

We need to:

- Create lists for the x values and y values.
- Look at each reading from the list of readings.
- Extract the reading's time and save it to the x values.
- Extract the reading's voltage and save it to the y values.
- Plot the data.

Extract the X and Y Data

Here we need to look at each reading individually and extract the x value (time) and the y value (voltage).

Each value will be saved to the relevant list.

In [ ]:

```
x_data = []
y_data = []

for reading in data['readings']:

    time = reading['time']
    time = float(time) * 1000
    x_data.append(time)

    voltage = reading['voltage']
    y_data.append(voltage)
```

We should also check that our data looks reasonable before we begin plotting it...

In [ ]:

```
pprint(x_data)
```

In [ ]:

```
pprint(y_data)
```

Create a Basic Plot

Once we have the data, creating a chart is a matter of just two lines.

In [ ]:

```
pyplot.plot(x_data, y_data)
pyplot.show()
```

We can also add some styling to make it look nicer.

The title and axis labels can be added using the relevant plot functions.

We can also add annotations such as arrows and lines. Here we've added a vertical line to show where **time = 0**, which is the point at which the stimulation was applied. The time before zero is a recording buffer.

In [ ]:

```
pyplot.style.use('ggplot')

pyplot.title('NLDL Data')
pyplot.xlabel('Time (ms)')
pyplot.ylabel('Voltage')

pyplot.axvline(x=0, color='blue')

pyplot.plot(x_data, y_data)
pyplot.show()
```

To save your chart as a file, change the function **show()** to **savefig()** instead.

Jupyter Notebooks has some limitations around savefig!
Note that if you try to savefig from a different cell than the one where you generated the figure, **the output image may be blank**.

Also, in Jupyter Notebooks, using **plot()** will display the resulting figure automatically. In a script, using **show()** or **savefig()** is required.

In [ ]:

```
pyplot.style.use('ggplot')

pyplot.title('NLDL Data')
pyplot.xlabel('Time (ms)')
pyplot.ylabel('Voltage')

pyplot.axvline(x=0, color='blue')

pyplot.plot(x_data, y_data)
pyplot.savefig('TEK0000.png')
```