(3) Exercise:

« One Method to load them all »

Loading/Saving

Setting Aliases, Signal, Axes...

What you need: load/save

Install the package:

```
>> addpath(genpath('/path/to/iFit'))
```

Loading a file into an object:

```
>> a = iData('filename')
>> a = iData('')                    % opens a file selector
>> a = iData(variable)              % convert a Matlab variable
```

Saving an object into a file:

```
>> saveas(a, 'filename.ext')
>> saveas(a, 'filename', 'format')
>> saveas(a, 'filename', 'gui')     % selects file format from a list
```

If you want to use example **data files** from iFit, they are in

/path/to/iFit/Data

Or you can use your own.

List of supported file formats: doc(iData)

File formats

format (load/saveas)	**Read**	**Write**
Any free format text, including ILL data, SPEC (ESRF), McStas data files, INX, ISIS/SPE, CSV (comma separated values)	Yes	Yes
HDF 4, HDF 5, NeXus and all deviratives, NetCDF	Yes	Yes
Matlab Mat-file (MAT), figure (FIG) and variables	Yes	Yes
Excel spreadsheet (XLS)	Yes	Yes
GIF BMP PNG TIFF JPEG ICO Images	Yes	Yes
ESRF Data format (EDF)	Yes	Yes
XML description file, Lotus 1-2-3, FITS astronomical image, NeXT/SUN (.au) sound, Microsoft WAVE sound (WAV), Audio/Video Interleaved multimedia container (AVI)	Yes	
Distant URL (ftp:// http:// https://) and compressed files (ZIP, GZip, TAR, Z)	Yes	
PostScript (PS,EPS), Adobe PDF and Illustrator, Virtual Reality world (VRML), Scalable Vector Graphics (SVG)		Yes

More file formats may be added in the future.

What you need: aliases

First inquire the object structure:

```
>> get(a)
>> a.Data       % available stuff from the imported data
```

Set an alias:

```
>> setalias(a, 'NewAlias', 'link' or numerical value)
>> a.NewAlias = numerical value
>> set(a, 'NewAlias', numerical value)
```

Get an alias value or definition:

```
>> get(a, 'NewAlias')          % value
>> a.NewAlias
>> getalias(a, 'NewAlias')     % definition (link)
```

Remove an alias:

```
>> rmalias(a, 'NewAlias')
```

What you need: signal

Check the actual Signal value:

```
>> get(a)
>> a.Signal
```

Identify what is the signal from the Data property:

```
>> a.Data
```

Set the Signal definition:

```
>> set(a, 'Signal', 'link' or numerical value)
>> a.Signal = numerical value
```

Get the Signal value or definition:

```
>> getalias(a, 'Signal')          % definition
>> getalias(a, 'Signal' or '0')   % idem
>> a.Signal                       % Signal value
>> getaxis(a, 0)                  % Signal/Monitor
>> a{0}                           % idem
```

What you need: axes, monitor and error bars

Monitor and Error bars are set the same way as other aliases:

```
>> a.Monitor = 'link' or numeric;      % sets the Monitor
>> a.Error = 'link' or numeric;
```

Setting empty values ([] or '') reset default definitions (1 and *sqrt*).

Axes should make use of existing aliases:

```
>> setaxis(a, 1, 'link' or value)
>> a{1} = 'link' or value
```

Hard work ahead

- Choose a data file (your own or *iFit/Data*).
- Have a look at it, if it is editable (text or dedicated viewer).
- Import it with Matlab/iFit.
- Have a look at the loaded object structure.
- Create a *Temperature* alias, which value you choose.
- Create, if needed, aliases for the axes.
- Assign the axes to their aliases.
- Save the object in formats *HDF, mat, m*, and experiment others.
- Look at the final file size.