

Igor File Vocabulary

- **Experiment:** an Igor file where you store data and graphs (.pxp) *Let's open a new experiment.*
- **DataFolder:** subdirectory in an experiment
 - Red Arrow in Data Browser shows current DataFolder
- **Notebook:** “file” in an experiment where you can write notes, paste graphs
 - *Windows → New → Notebook* (Formatted Text)
- **Procedures:** functions written by you (or others)
 - Some exist only in the experiment they are in
 - Can be shared with friends/colleagues
 - “Local” procedure window (*ctrl+m*)

More Igor File Vocabulary

- Data Browser
 - Can set Preferences to sort by “Name and Type”
- Help Browser (F1)
 - Look for help on functions
 - Can also get function help by right-clicking on a function name, choosing “Help on *functionName*”
 - Most functions have Examples
 - Many functions have Related Functions

Data is usually stored in Waves

- **Wave:** a vector (array) containing data
 - Exists until you “kill” it (even if you’re not looking at it in a graph or table)
 - Numbers (single precision by default) or text
- Every wave has some number of “points” (length)
- Waves have inherent “x” values = point number (0, 1, 2...)
 - When you **display** a wave, it’s plotted against the x values (unless you tell it to plot vs. something else)

Wave Names have Rules

- Wave names
 - Cannot include spaces, operators (+ - * /), special characters
 - Can use underscore (_)
 - Cannot begin with numbers
 - Must be ≤ 32 characters
 - Are not case sensitive
 - Cannot have the same name as functions, variables, Igor-used terms

Making Waves

- Let's make a really simple waves so that we can practice some simple operations

`Make/N=20 y_vals`

Igor function that creates a wave Flag for make that sets Number of points in wave Name of wave

- Now let's look at the values in the wave in a table:

edit y_vals.id

(special way to see the “x” and “data” values”)

Giving values to y_vals

- Now we can take advantage of the inherent “x” values and make $y_vals = f(x)$

`Y_vals = x`

`Display y_vals`

Fix the axes to have range -10, 40 for x and y

Add “zero lines” (Ticks and Grids tab)

Copy and Paste this plot in your Notebook

Change your line by changing the equation in the command line

*(e.g., `y_vals = 2*x` `y_vals = 2*x + 3`)*

Copy and Paste a new plot in your Notebook

Note that the y_vals.d changes in the table, too!

Kill the table!

- (Kill the wabbit, kill the wabbit...)
- Check for the wave in the Data Browser
 - Waves exist until you kill them, even if you're not looking at the them.
 - Very different from Excel, of course!
- Make table again if you want to watch the data change

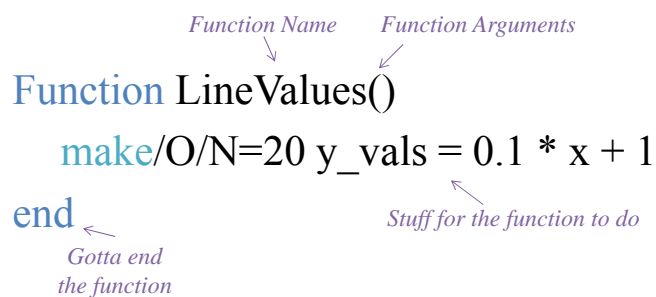
Can you make the line go into the $x < 0$ range?

- Recall from “Getting Started” that you can *change* the inherent x-scaling
Data → Change Wave Scaling
- The command for this change was printed in the History in the Command Window!
 - Useful for using this command in a function, since you can't use the pulldown menu in a function.

Make a function for the line

1. Simple version

Our Sample Function



The diagram shows a MATLAB function definition with three lines of code. Annotations with arrows point to specific parts of the code:

- Function Name* points to `Function` in `Function LineValues()`.
- Function Arguments* points to `LineValues()` in `Function LineValues()`.
- Stuff for the function to do* points to the assignment statement `y_vals = 0.1 * x + 1`.
- Gotta end the function* points to the `end` keyword.

```
Function LineValues()
    make/O/N=20 y_vals = 0.1 * x + 1
end
```

And let's compile!

And let's run it (from the command line).

Change the values. Does the line change?

Make a function for the line

1. Simple version *with variables*

Our Sample Function, with Variables

```
Function LineValues2()  
    variable m = 0.1, b = 1  
    make/O/N=20 y_vals = m * x + b  
end
```

Make a function for the line

1. Simple version with variables
2. Generalize function with inputs

Our Sample Function, with Inputs

```
Function LineValues_input(m,b)
    variable m, b
    make/O/N=20 y_vals = m * x + b
end
```