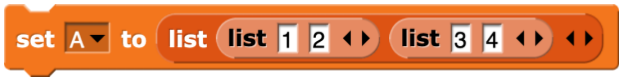
NetsBlox Lesson: Multi-Dimensional Lists and Plotting

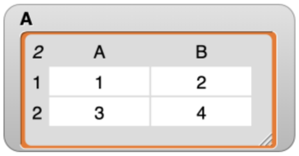
Multi-dimensional lists

In NetsBlox, lists can contain other lists as elements. For example, we can create a 2-by-2 matrix that is a list with two list elements:



```
set A to list (list 1 2) (list 3 4)
```

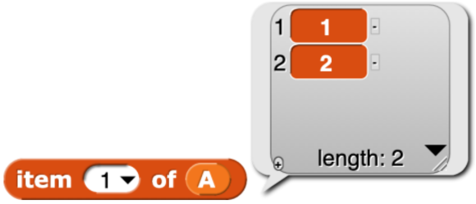
This block will create this data structure:



A

2	A	B
1	1	2
2	3	4

How do we access elements of this? Well, item 1 is going to be the first list from above or you can think of it as the first row of the matrix:

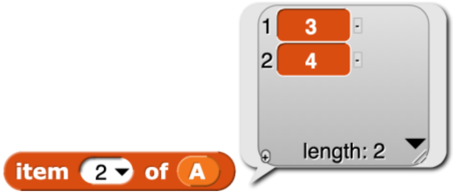


```
item 1 of A
```

1	1
2	2

length: 2

Similarly, item 2 is going to be the second list:

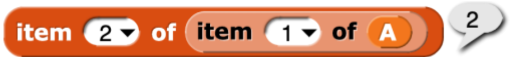


```
item 2 of A
```

1	3
2	4

length: 2

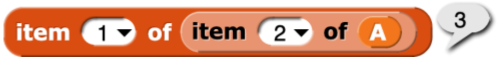
If we want to get to the individual numbers (elements or cells of the matrix), we need to use two item blocks:



```
item 2 of item 1 of A
```

2

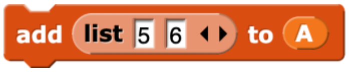
The first block, item 1 of A, returns the first list (or first row) and then we take item 2 of that, so we get 2. Here is another example:



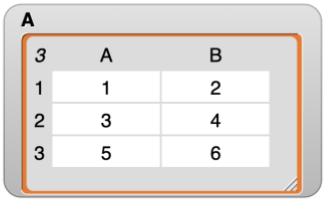
```
item 1 of item 2 of A
```

3

All other blocks related to lists work with lists of lists as well:



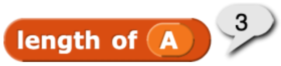
```
add list 5 6 to A
```



A

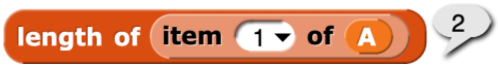
3	A	B
1	1	2
2	3	4
3	5	6

And then:



```
length of A
```

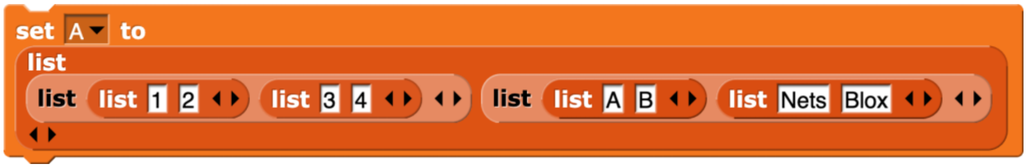
3



```
length of item 1 of A
```

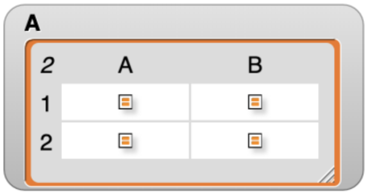
2

Finally, we can create even lists of lists of lists to any depth. In other words, NetsBlox supports multi-dimensional arrays too:



```
set A to
list
  list list 1 2   list 3 4     list list A B   list Nets Blox
```

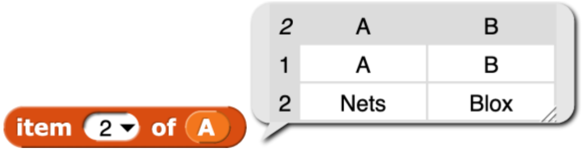
As you can see, they do not even have to be of the same data type. We can mix numbers, text and other types as well. The display, however, is not going to be as easy:



A

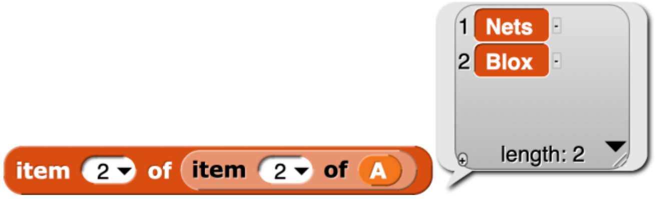
2	A	B
1	▤	▤
2	▤	▤

However, we can get to elements the usual way:



```
item 2 of A
```

2	A	B
1	A	B
2	Nets	Blox

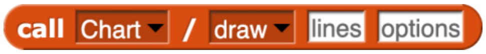


```
item 2 of item 2 of A
```

1 Nets
2 Blox
length: 2

Plotting

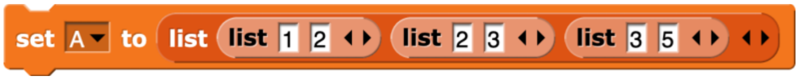
NetsBlox supports plotting using a service called Chart. It is using a library called gnuplot on the server. The major RPC of the Chart service is called draw:



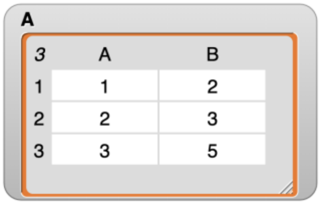
```
call Chart / draw lines options
```

The first input called lines is the data to be plotted. The RPC expects a list of lists of x and y coordinates. In other words, it expects a matrix with as many rows as many points we want to plot and two columns, the first for x coordinates and the second for y.

In this example, we have three points: (1, 2), (2, 3) and (3, 5):



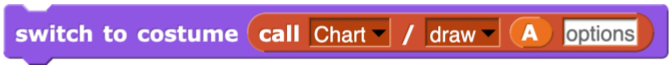
```
set A to list (list 1 2) (list 2 3) (list 3 5)
```



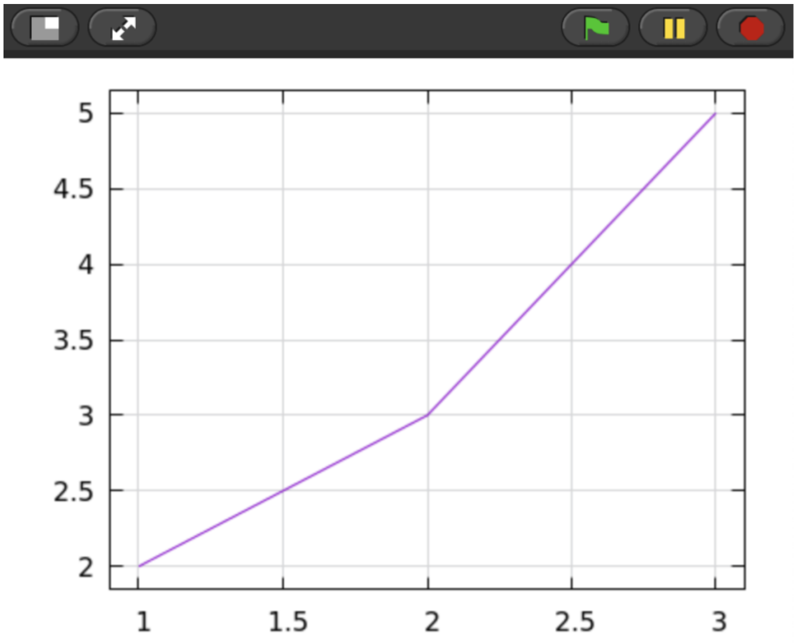
A

3	A	B
1	1	2
2	2	3
3	3	5

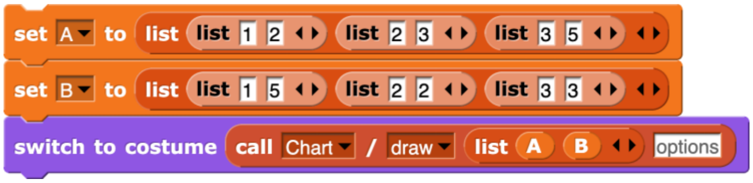
Let us plot them:



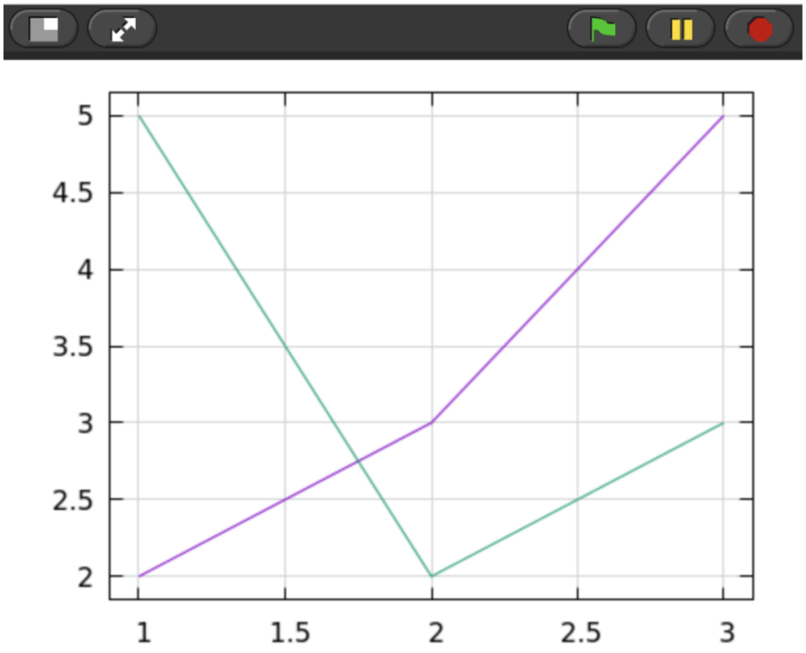
```
switch to costume call Chart / draw A options
```



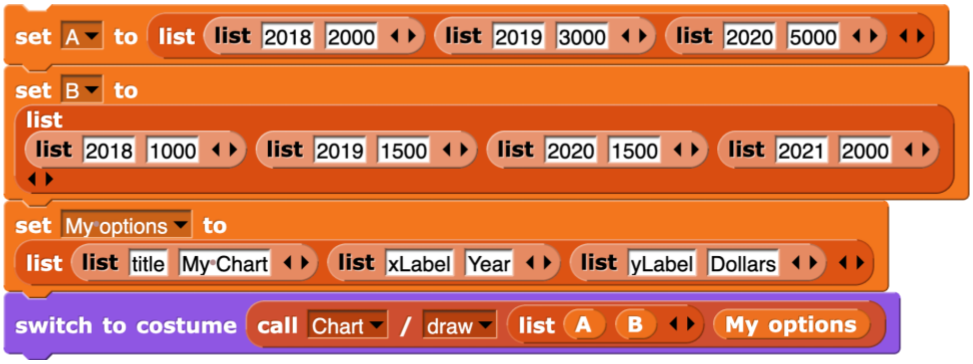
The RPC also supports multiple lines. Simply provide a list of data item like this:



```
set A to list (list 1 2) (list 2 3) (list 3 5)
set B to list (list 1 5) (list 2 2) (list 3 3)
switch to costume (call Chart / draw (list A B) options)
```

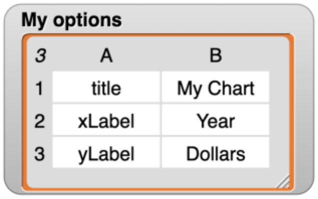


So what about the options input to the draw RPC? That is where we can customize our plot with things like title, labels, etc. It is also a list of lists: the first item in a list specifies what attribute we want to adjust and the second is the corresponding value. Here is an example:



```
set A to list list 2018 2000 ◂▸ list 2019 3000 ◂▸ list 2020 5000 ◂▸ ◂▸
set B to
  list
    list 2018 1000 ◂▸ list 2019 1500 ◂▸ list 2020 1500 ◂▸ list 2021 2000 ◂▸
  ◂▸
set My options to
  list list title My Chart ◂▸ list xLabel Year ◂▸ list yLabel Dollars ◂▸ ◂▸
switch to costume call Chart / draw list A B ◂▸ My options
```

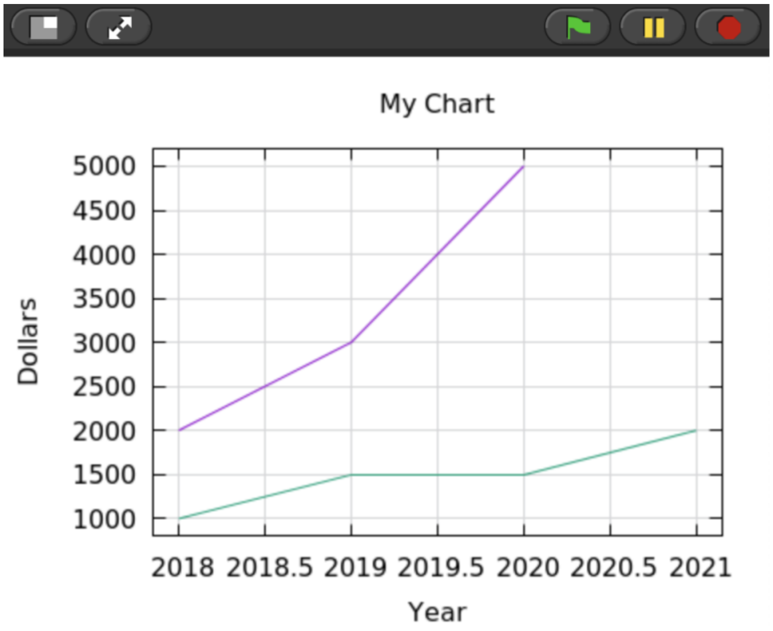
My options will look like this:



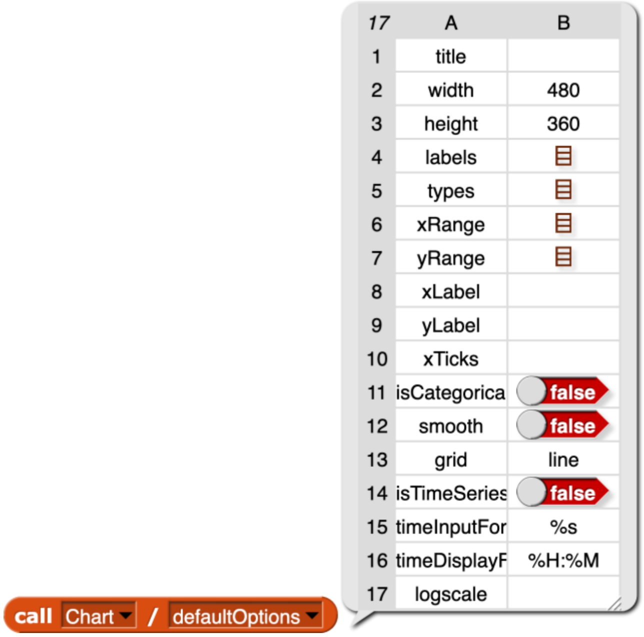
My options

3	A	B
1	title	My Chart
2	xLabel	Year
3	yLabel	Dollars

And the plot:



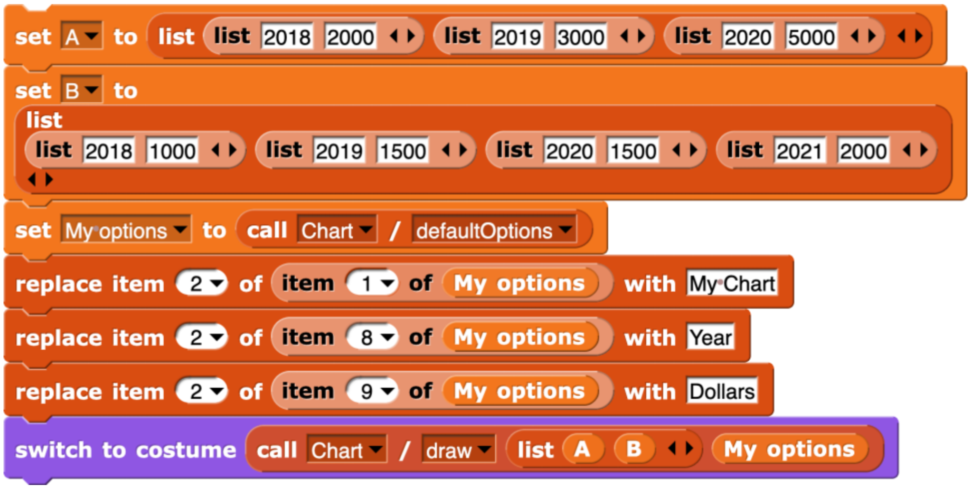
How do we know what options are even available for the plot? Well, that is what the second RPC of the Chart service provides:



```
call Chart / defaultOptions
```

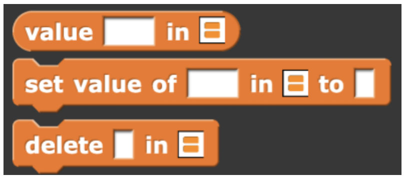
17	A	B
1	title	
2	width	480
3	height	360
4	labels	☰
5	types	☰
6	xRange	☰
7	yRange	☰
8	xLabel	
9	yLabel	
10	xTicks	
11	isCategorica	false
12	smooth	false
13	grid	line
14	isTimeSeries	false
15	timeInputFor	%s
16	timeDisplayF	%H:%M
17	logscale	

That is helpful, but if we use it, it will get a bit clunky. We will need to remember which attribute is which item. This will generate the same chart as before:



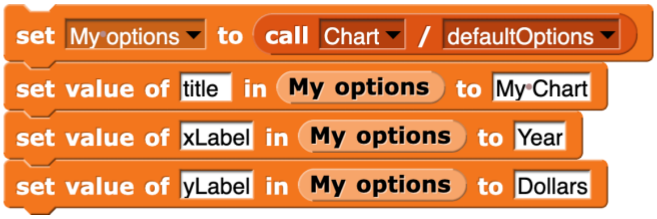
```
set A to list (list 2018 2000) (list 2019 3000) (list 2020 5000)
set B to
  list
    (list 2018 1000) (list 2019 1500) (list 2020 1500) (list 2021 2000)
set My options to call Chart / defaultOptions
replace item 2 of (item 1 of My options) with My Chart
replace item 2 of (item 8 of My options) with Year
replace item 2 of (item 9 of My options) with Dollars
switch to costume (call Chart / draw (list A B) My options)
```

There must be a better way! There is. Select the File Menu, Libraries and then Structured Data. It will load a set of new custom blocks that you can find in the Custom palette:



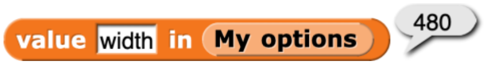
```
value [ ] in ☰
set value of [ ] in ☰ to [ ]
delete [ ] in ☰
```

These blocks were designed to work with the exact kind of data the options of the draw RPC needs. They take a list of lists where the first item of each such list is a key (a name) and the second is a value. Now we can manipulate the option using the names! Here we go:



```
set My options to call Chart / defaultOptions
set value of title in My options to My Chart
set value of xLabel in My options to Year
set value of yLabel in My options to Dollars
```

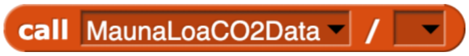
We can also get the current values:



```
value width in My options
```

Data Services

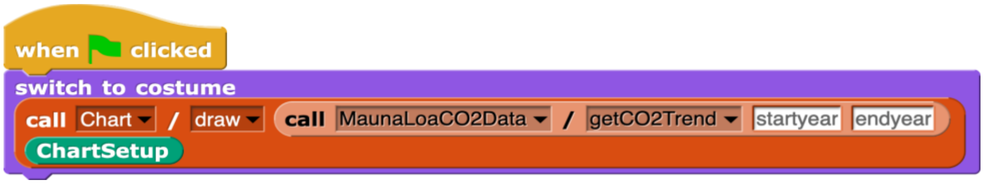
It is time now to plot some real data. NetsBlox comes with a number of services that provide interesting real-world data. Let's start by plotting the atmospheric carbon dioxide concentrations being measured by the Mauna Loa Observatory in Hawaii since 1958. The service is under the Climate heading in the menu:



```
call MaunaLoaCO2Data /
```

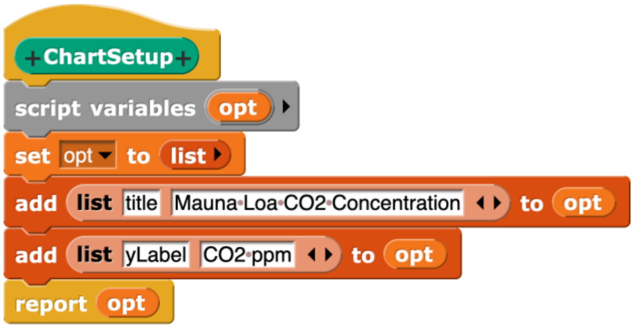
It has two RPCs, getCO2Trend and getRawCO2. The raw data is one measurement per month. Since the CO2 concentration has an annual cycle, the trend is just an annual average value, again once a month.

Here is the code that plots the data:



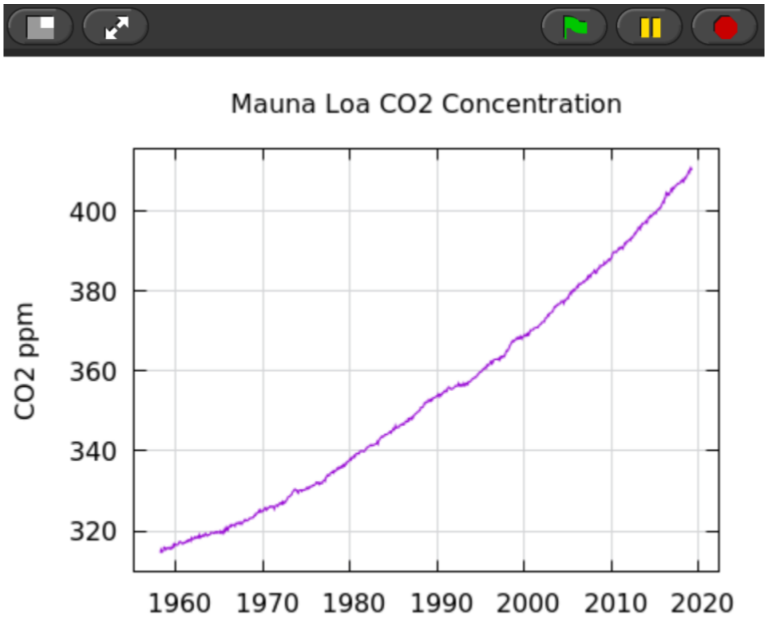
```
when clicked
switch to costume
  call Chart / draw (call MaunaLoaCO2Data / getCO2Trend startyear endyear)
  ChartSetup
```

If the start and stop year are not specified, the RPC returns the data for the entire date range available. The ChartSetup custom block should look familiar:



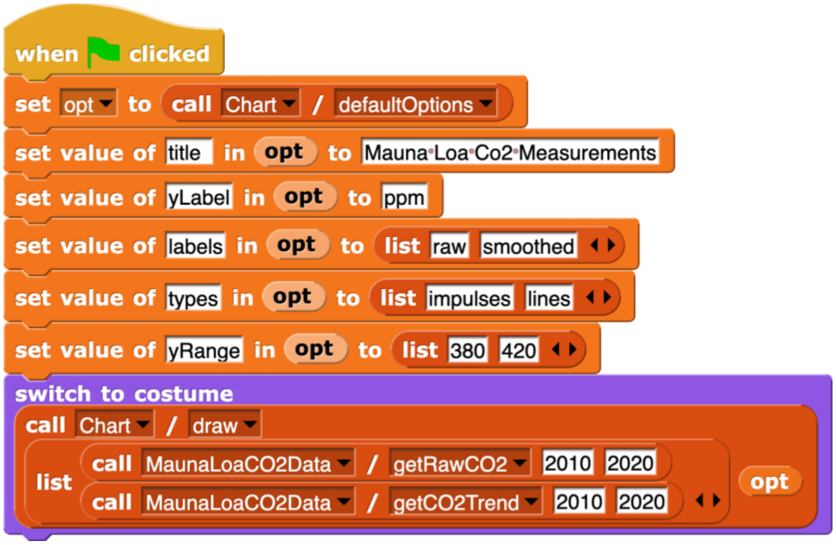
```
+ChartSetup+
script variables opt
set opt to list
add list title Mauna·Loa·CO2·Concentration to opt
add list yLabel CO2·ppm to opt
report opt
```

And the result:



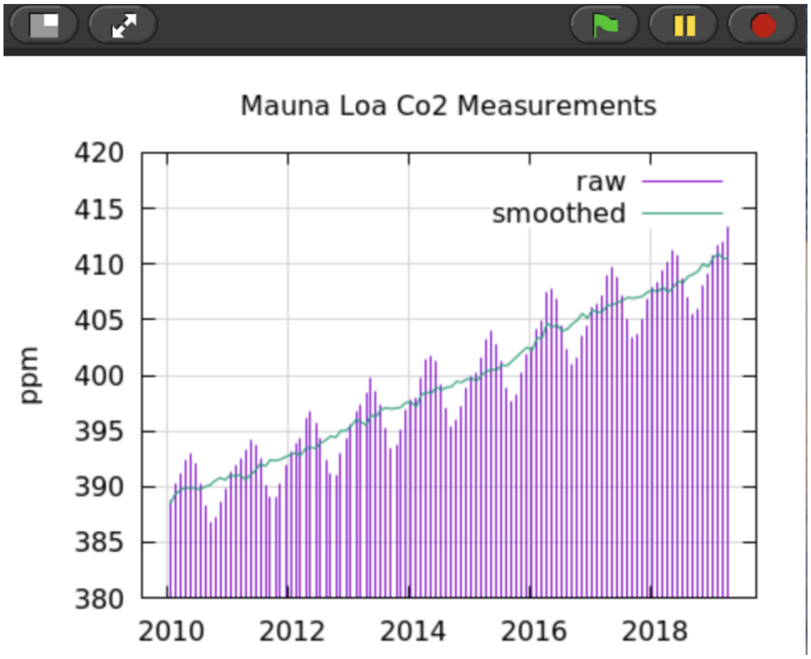
The project is available here:
https://editor.netsblox.org/?action=present&Username=ledeczi&ProjectName=Mauna%20Loa%20v2

Let’s tweak the project to see a few other options we can play with:



```
when 🏁 clicked
set opt to call Chart / defaultOptions
set value of title in opt to Mauna Loa Co2 Measurements
set value of yLabel in opt to ppm
set value of labels in opt to list raw smoothed
set value of types in opt to list impulses lines
set value of yRange in opt to list 380 420
switch to costume
    call Chart / draw
        list
            call MaunaLoaCO2Data / getRawCO2 2010 2020
            call MaunaLoaCO2Data / getCO2Trend 2010 2020
        opt
```

The types option will plot the data using different styles. Options available are lines, dots, points, circles, impulses, boxes. You can provide a legend using the labels option. And you can also set the ranges. Notice how some options take a list of values as opposed to a single one. Here is the result:



https://editor.netsblox.org/?action=present&Username=ledeczi&ProjectName=Mauna%20Loa%20v3