Sept 19, 2019 ... birthday of James Alexander of Alexander polynomial fame.

we've seen that we can write functions like

> $f := x \to x^2$

$$f := x \mapsto x^2 \qquad (1)$$

> $f(3)$

$$9 \qquad (2)$$

> $g := n \to [seq([i, i^2], i = 1 .. n)]$

```
Warning, `i` is implicitly declared local to procedure `g`
```

$$g := n \mapsto [seq([i, i^2], i = 1 .. n)] \qquad (3)$$

> $g(3)$

$$[[1, 1], [2, 4], [3, 9]] \qquad (4)$$

I use shift-enter to get a newline without sending command to maple

> $h :=$ **proc**(n)
>   **return**$([seq([i, i^2], i = 1 .. n)]);$
> **end**

```
Warning, `i` is implicitly declared local to procedure `h`
```

$$h := \textbf{proc}(n)\ \textbf{local}\ i;\ \textbf{return}\ [seq([i, i\hat{}2], i = 1 .. n)]\ \textbf{end proc} \qquad (5)$$

> $h(3)$

$$[[1, 1], [2, 4], [3, 9]] \qquad (6)$$

> $h :=$ **proc**(n)
>   **local** i; *# this i only lives inside h*
>   **return**$([seq([i, i^2], i = 1 .. n)]);$
> **end**

$$h := \textbf{proc}(n)\ \textbf{local}\ i;\ \textbf{return}\ [seq([i, i\hat{}2], i = 1 .. n)]\ \textbf{end proc} \qquad (7)$$

Let's write a procedure that, given some data, computes lsq fit and mean squared error, plots the result, and prints out both fit and msq.

```
> lsqpic := proc(data)
      line:=CurveFitting[LeastSquares](data,x)
      return(line)
  end
Error, reserved word `return` unexpected
```

```
> lsqpic := proc(data)
      line:=CurveFitting[LeastSquares](data,x);
      return(line);
  end:
Warning, `line` is implicitly declared local to procedure
`lsqpic`
```

```
> lsqpic := proc(data)
      local line,x;
      line:=CurveFitting[LeastSquares](data,x);
      return(line);
  end:
```

> $lsqpic([[0, 1], [2, 2], [3, -1]])$

$$\frac{3}{2} - \frac{x}{2} \qquad (8)$$

what is diff between return and print?

```
> lsqpic := proc(data)
      local line,x;
      print("using ",nops(data),"points");
      line:=CurveFitting[LeastSquares](data,x);
      return(line);
  end:
```

```
> data := [[0, 1], [2, 2], [3,-1]]:
  lsqpic(data)
```

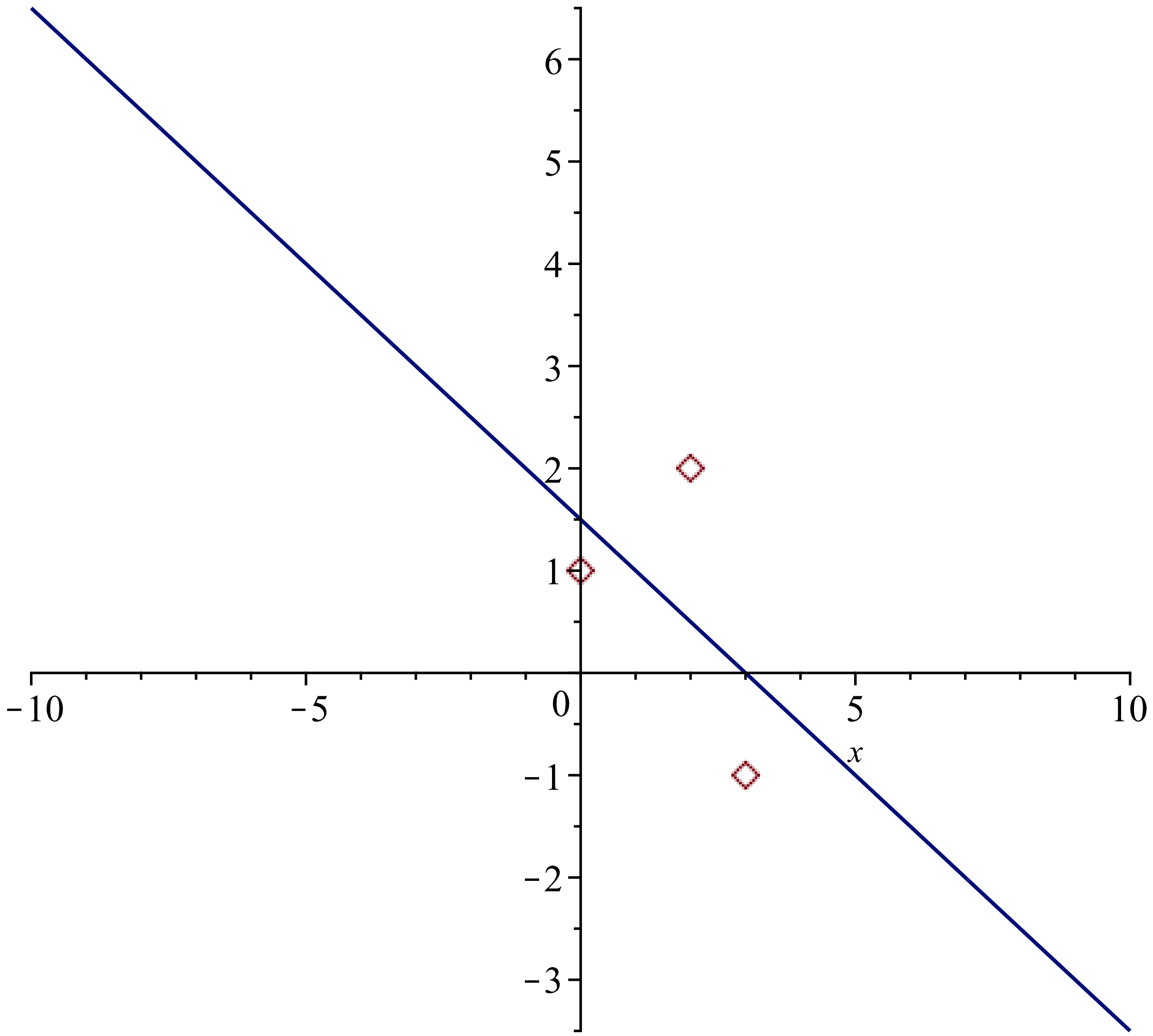
"using ", 3, "points"

$$\frac{3}{2} - \frac{x}{2} \qquad (9)$$

```
> myline := %
```

$$myline := \frac{3}{2} - \frac{x}{2} \qquad (10)$$

```
> plot([data, myline], style = [point, line], symbolsize = 20)
```



Want to pick off the range of the data from given list.

```
> min([1, 3, 17,-8]);
```

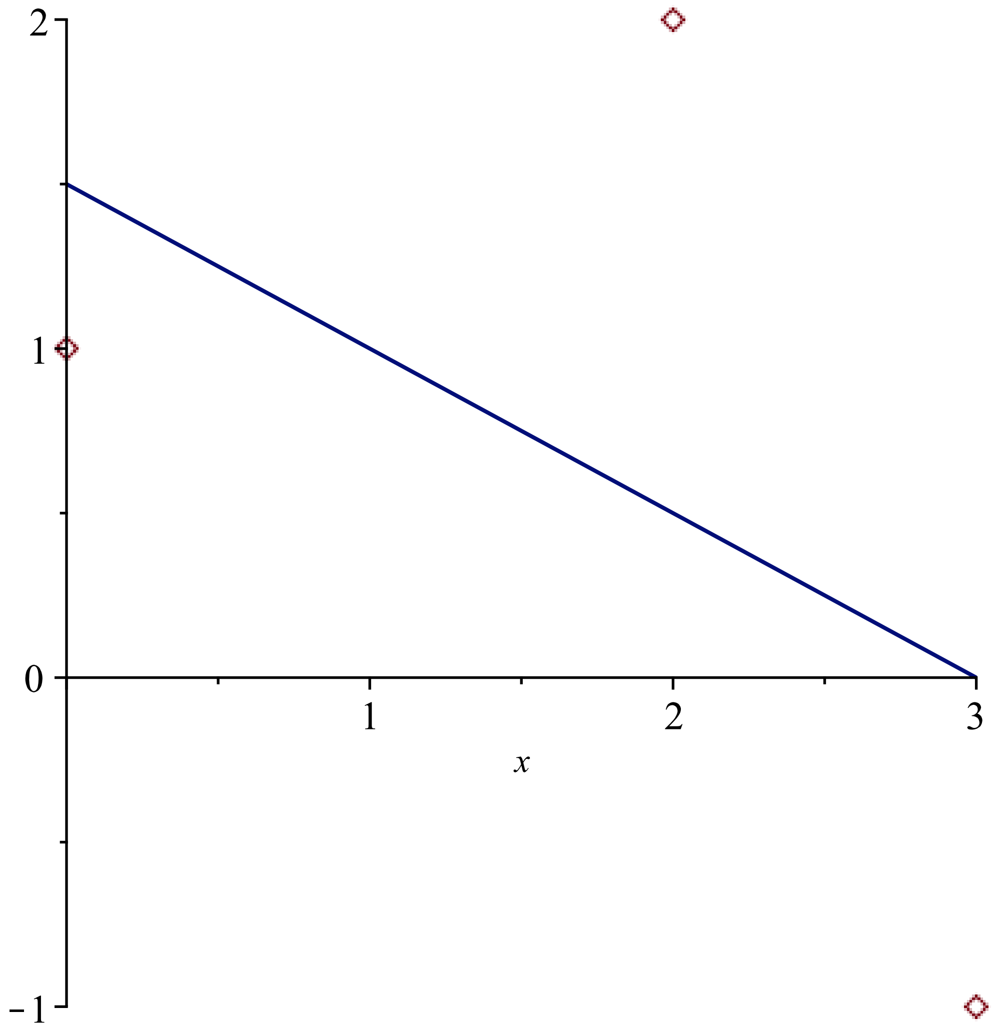
```
   max([1, 3, 17,-8]);
                                      −8
                                      17                                    (11)

> min(data)
                                      −1                                    (12)
> data
                              [[0, 1], [2, 2], [3, −1]]                     (13)
> min(seq(data[i, 1], i = 1 ..nops(data)))
                                       0                                    (14)
> getRange :=proc(data)
    local i, xmin, xmax;
    xmin := min(seq(data[i, 1], i = 1 ..nops(data)));
    xmax := max(seq(data[i, 1], i = 1 ..nops(data)));
    return(xmin .. xmax);
   end:
> getRange(data)
                                     0 ..3                                  (15)
> lsqpic := proc(data)
     local line,x, pic;
     print("using ",nops(data),"points");
     line:=CurveFitting[LeastSquares](data,x);
     pic:=plot([data,line],x=getRange(data),
       style=[point,line], symbolsize=18);
     return(pic);
  end:
> lsqpic(data)
                              "using ", 3, "points"
Error, (in plot) expecting option style to be of type identical
("point", "patch", "patchnogrid", "line", "pointline", "hidden",
"wireframe", "contour", "patchcontour", "polygonoutline",
"polygon", "surface", "surfacecontour", "surfacewireframe",
"wireframeopaque", "default") but received 3/2-(1/2)*x
```

I'm dumb. line means something else

```
> lsqpic := proc(data)
      local fit, x, pic;
     print("using ", nops(data), "points");
      fit := CurveFitting[LeastSquares](data, x);
     pic := plot([data, fit], x = getRange(data),
       style = [point, line], symbolsize = 18);
     return(pic);
   end:

> lsqpic(data);
                              "using ", 3, "points"
```

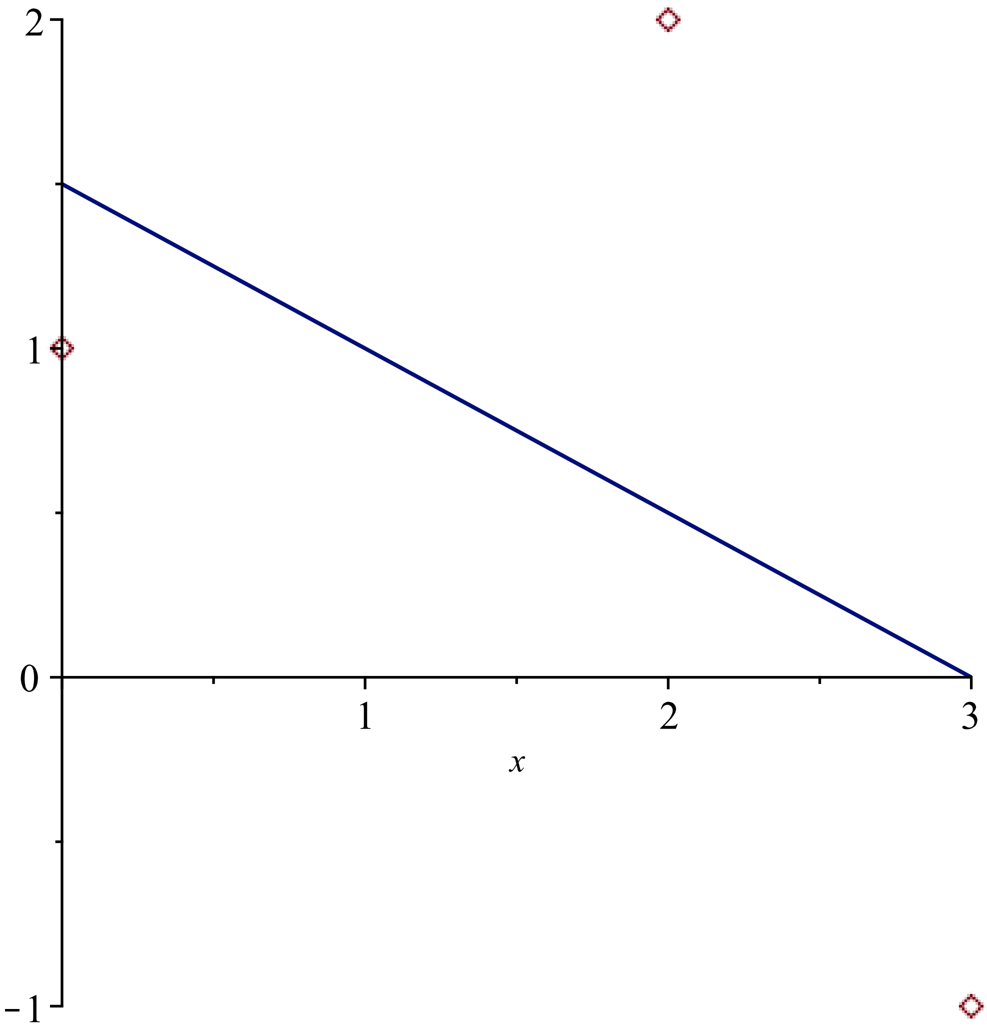


```
> lsqpic ≔ proc(data)
      local fit, x, pic;
      fit ≔ CurveFitting[LeastSquares](data, x);
     pic ≔ plot([data, fit], x = getRange(data),
       style = [point, line], symbolsize = 18);    printf("using %d points\n", nops(data));

     return(pic);
    end:
```

```
> lsqpic ≔ proc(data)
      local fit, x, pic;
      fit ≔ CurveFitting[LeastSquares](data, x);
     pic ≔ plot([data, fit], x = getRange(data),
       style = [point, line], symbolsize = 18);
     printf("with %d points, line is %a\n", nops(points), fit);
     return(pic);
    end:
```

```
> lsqpic(data)
with 1 points, line is 3/2-1/2*x
```



from http://www.math.stonybrook.edu/~scott/mat331.fall19/daily/extras/bdata.txt

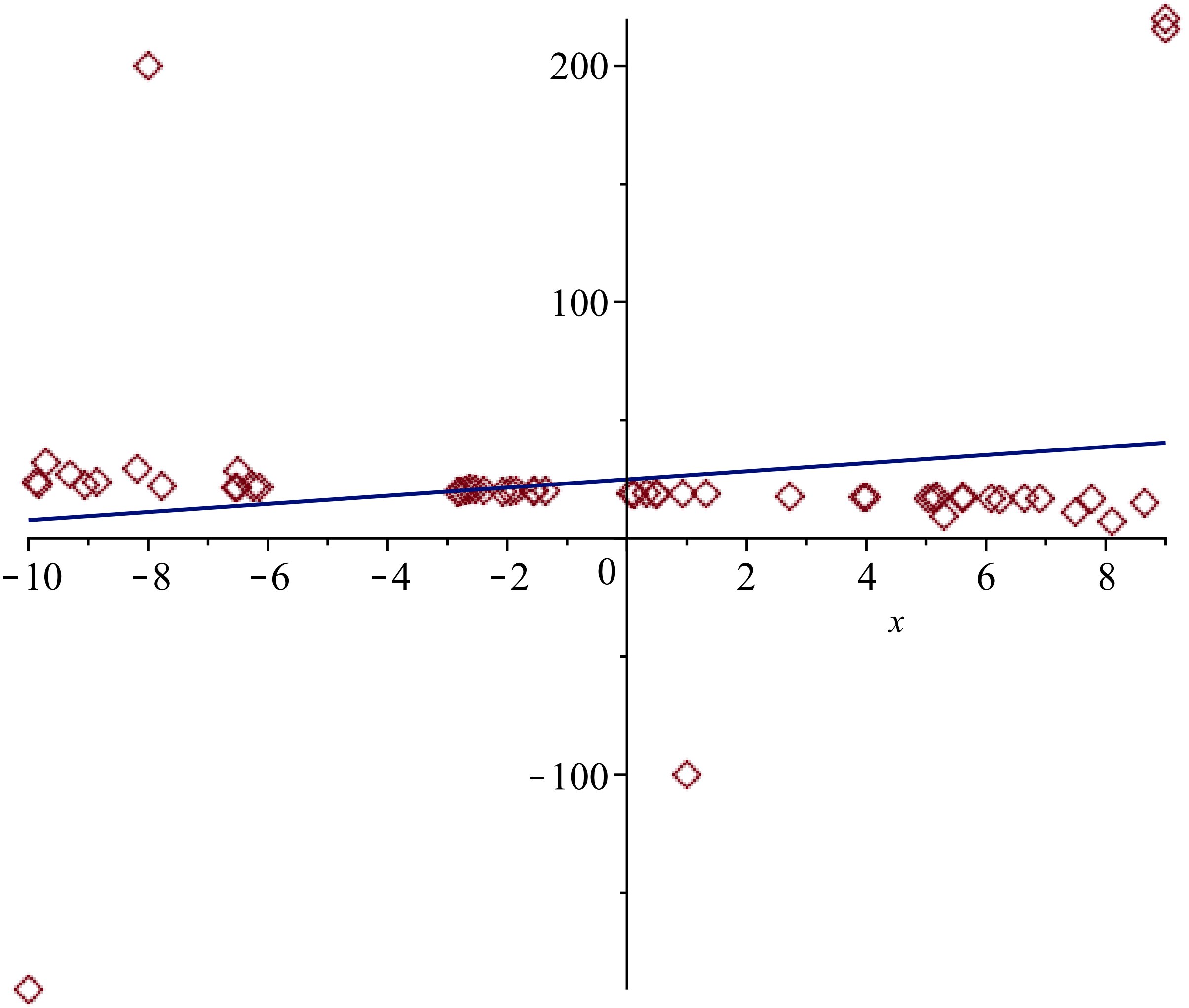
```
> ExecuteFromWeb ≔proc(URL :: string, {printfile :: truefalse ≔ false})
   local n, m, status, webfile, headers;
   status, webfile, headers ≔ HTTP[Get](URL) :
   if ( HTTP[Code](status) ≠ "OK") then
      error(HTTP[Code](status), URL);
   fi;
   # now interpret the maple on the web page
   n ≔ 0 :
   while (n < length(webfile)) do
     m ≔ n;
     parse(webfile, statement, lastread='n', offset=n);
     if (printfile) then printf("%s", webfile[m + 1 ..n]); fi;
   od:
  end:
```

```
> ExecuteFromWeb("http://www.math.stonybrook.edu/~scott/mat331.fall19/daily/extras/bdata.
       txt");
```

```
> nops(bdata);
```

54 **(16)**

```
> lsqpic(bdata);
with 1 points, line is 24.9502982841362+1.72140920411782*x
```



```
>
```