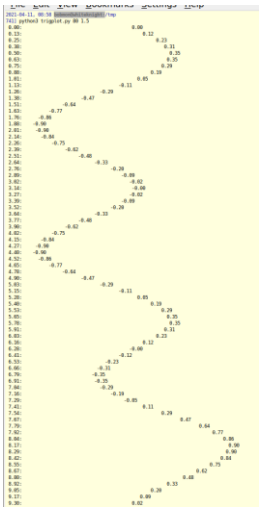
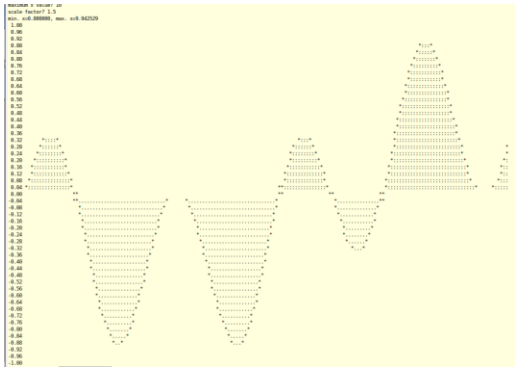
## *Making a Horizontal Graph On a Terminal*

## *The Problem*

- We have:



- We want:
  - a horizontal version of the graph



## *Converting Data Space to Display Space*

- Range $x_{min}$ to $x_{max}$ maps to column space
  - x in *e.g.* (0 .. 6π), or (-10 .. +10)
  - 0 to n_cols (*e.g.* 0 to 80)
- Domain $y_{min}$ to $y_{max}$ maps to row space
  - y in *e.g.* (-1.0 .. +1.0) for trigonometric functions
  - 0 to n_rows (*e.g.* 0 to 24)
- Linear mappings – we need:
  - f(col) = x → $f(0) = x_{min}$, $f(n\_cols) = x_{max}$
  - g(row) = y → $g(0) = y_{max}$, $g(n\_rows) = y_{min}$

## *Calculate Column Parameters a and b*

$x = a \bullet col + b$

$x_{min} = a \bullet 0 + b$ , $x_{max} = a \bullet n\_cols + b$

**$b = x_{min}$**

$x_{max} = a \bullet n\_cols + x_{min}$

**$a = (x_{max} - x_{min}) / n\_cols$**

**$x = (x_{max} - x_{min}) / n\_cols \bullet col + x_{min}$**

## *Calculate Row Parameters c and d*

$y = c \bullet row + d$

$y_{min} = c \bullet 0 + d$ , $y_{max} = c \bullet n\_rows + d$

**$d = y_{min}$**

$y_{max} = c \bullet n\_rows + y_{min}$

**$c = (y_{max} - y_{min}) / n\_rows$**

**$y = (y_{max} - y_{min}) / n\_rows \bullet row + y_{min}$**

## *Where Does the Data Come From?*

- A couple of possible data sources:
  1) a list of arbitrary x-y pairs (perhaps from a file)
  2) a generating function
- First case is very general
  - *interpolate* x- and y- values for each column, row
- Second case is fairly easy
  - Calculate x-value for each column
  - Calculate corresponding y-value
- Convert y-value to row
  - Put mark at (row, column) position

## Horizontal Plotting With a List of Data

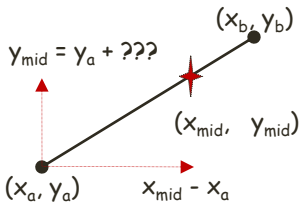
```
def draw_data(x_vals, n_cols, y_vals, n_rows):
    '''Draw y(x) for lists of x- and y- values.'''
    xmax, xmin = max(x_vals), min(x_vals)
    ymax, ymin = max(y_vals), min(y_vals)
    y_rows = []
    for c in range(n_cols):
        x_c = xmin + (xmax-xmin) * c/n_cols
        for i, x in enumerate(x_vals):
            if x >= x_c:
                y_rows.append(y_vals[i])
                break
    for r in range(n_rows, 0, -1):
        y_r =  interpolate( r, n_rows, 0, ymin, ymax )
        for c in range(n_cols):
            if 0 < y_r <= y_rows[c] or y_rows[c] < y_r <= 0:
                print('*', end='')
            else:
                print(' ', end='')
        print()
#-------
```

## Interpolating

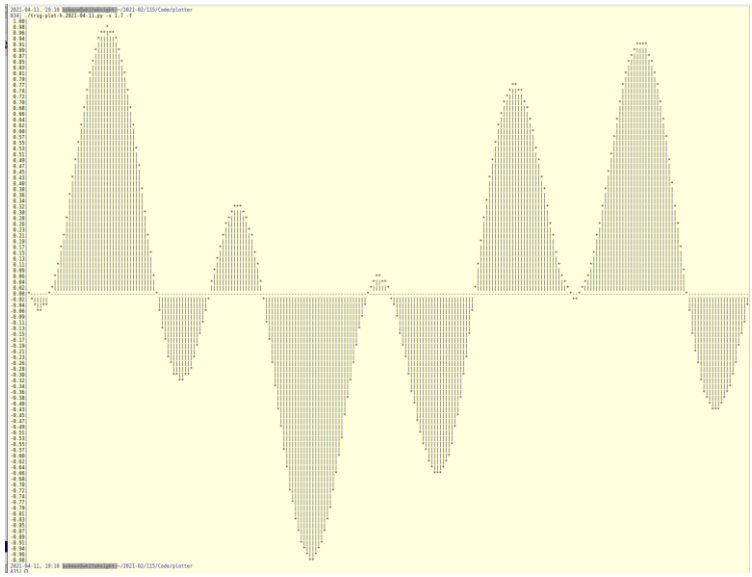
- Given two points $(x_a, y_a)$ and $(x_b, y_b)$, and an x-value $x_{mid}$ between $x_a$ and $x_b$, find the corresponding $y_{mid}$ value that lies between $y_a$ and $y_b$

$$y_{mid} = y_a + (y_b - y_a)*(x_{mid} - x_a)/(x_b - x_a)$$

```
# Interpolate the y-value corresponding to an x-value.
def interpolate(xmid, xa, xb, ya, yb):
    ymid = (yb - ya) * (xmid - xa) / (xb - xa)
    return ya + ymid
#-------
```



## *y = sin(x) * cos( 1.7 * x)*



## *Horizontal Plotting With a Y-Function*

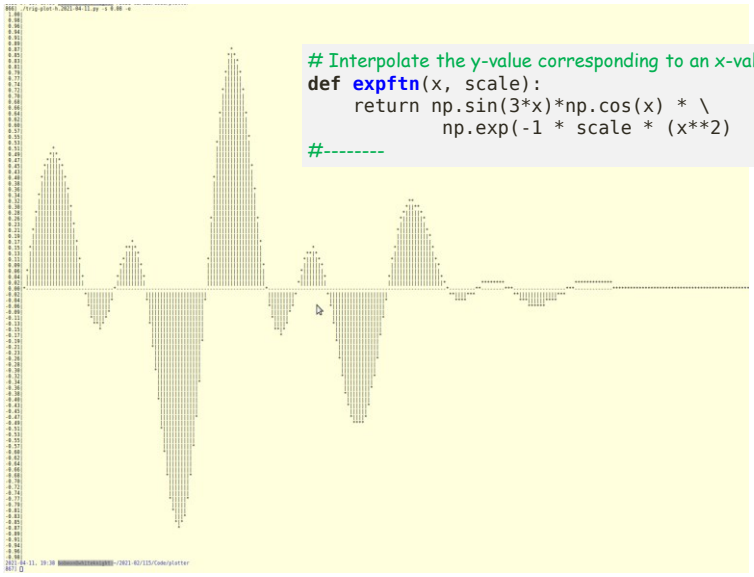
```
def draw_ftn(xmax, xmin, n_cols, ymax, ymin, n_rows,  y_ftn):
    '''Draw y_ftn(x) for x from xmin to xmax.'''
    x_cols = [interpolate(x, xmin,xmax, 0,n_cols)
                for x in range(n_cols)]

    y_vals = [y_ftn(x) for x in x_cols]

    for r in range(n_rows, 0, -1):
        y_r =  interpolate( r, n_rows, 0, ymin, ymax )
        for c in range(n_cols):
            if 0 < y_r <= y_rows[c] or y_rows[c] < y_r <= 0:
                print('*', end='')
            else:
                print(' ', end='')
        print()
#-------
```

You can pass a function
as an argument to another function !

$$y = \sin(3x) * \cos(x) * \exp(-0.08x^2)$$



```
# Interpolate the y-value corresponding to an x-value.
def expftn(x, scale):
    return np.sin(3*x)*np.cos(x) * \
            np.exp(-1 * scale * (x**2)
#--------
```