A tutorial for the Sliderule Activity – part 2

(http://activities.sugarlabs.org/en-US/sugar/addon/4222)

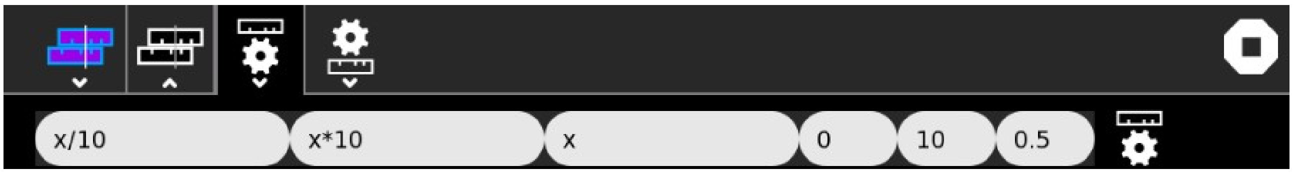
running on the Sugar operating system http://sugarlabs.org/ of the OLPC laptop,

targeted at year 8 to year 12 students

by Tony Forster, licensed under the Creative Commons Attribution-ShareAlike License

The Custom Scales

Start a new Sliderule Activity (don't resume a saved session) to restore default values. Select the add/subtract slide rule, then select a custom toolbar as shown below.



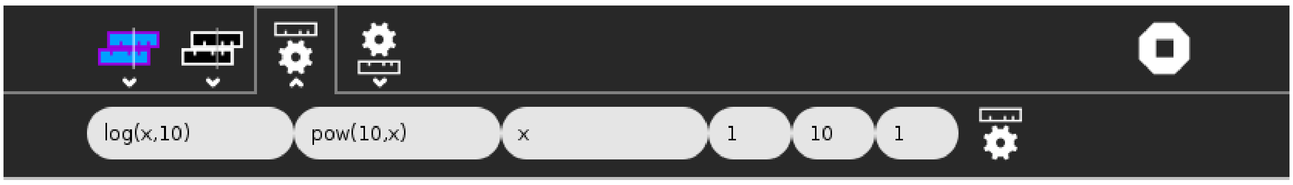
x/10 | x*10 | x | 0 | 10 | 0.5

Fig 1, Custom slide rule showing the formulae that generate the Add/Subtract or Linear scale

The formulae show how the linear slide rule is generated. From left to right, the position along the scale, the label, and function to calculate the number from its position are all linear or proportional to x. The marks and labels are drawn from 1 to 10 in steps of 0.5 in the last 3 fields

You can select a different predefined slide rule scale and view its generating formulae. This behaviour continues till you alter the slide rule by clicking Create Custom Slide, the cog icon on the right. To revert to the initial behaviour, start a new Sliderule (don't resume a saved session).

Select the multiply divide slide rule, then custom scale as shown below.



log(x,10) | pow(10,x) | x | 1 | 10 | 1

Fig 2, Custom slide rule showing the formulae that generate the Multiply/Divide scale

The formulae show how the selected slide rule is generated.

In this example, in the first field – the function position, the position along the scale is proportional to the log of the number where the number is represented by x . The function label in the third field is just the number, x.

To calculate the number at a known distance along the scale, with the distance along the scale, x (that is a different x) calculate 10^x or pow(10,x) the second field.

Finally, the marks and labels are drawn from 1 to 10 in steps of 1 in the last 3 fields.

The fields, from left to right are:

- position function of x used to calculate the horizontal position of each mark along the slide or stator (the range of this function should be 0 to 1)
- result function of x used to read a value from the slide or stator at the cursor (usually the inverse of the position function)
- label function of x used to calculate a label for each mark (often identity)
- the minimum value of x used in the functions domains
- the maximum value of x used in the functions domains
- the step size by which to iterate between minimum and maximum when generating marks

Finally there is the gear-shaped button is used to create the custom slide

The Custom Stator Toolbar is identical in functionality to the Custom Slide Toolbar.

In Fig 2, the illustration of the Custom Toolbar, with a C (log) slide, the values are:

the position is determined by $f(x) = log(x, 10)$
the result (the inverse of $f(x)$) is $f^{-1}(x) = pow(10, x)$
the label is $g(x) = x$
the domain is from 1 to 10
the step size is 1

From http://wiki.sugarlabs.org/go/Activities/Sliderule

Python Programming

Though these fields are mostly common sense, they are actually written in the Python programming language. The Python language is described at http://docs.python.org.

With the exception of the Label Function, all the fields should evaluate to a number, the Label Function can evaluate to a number or a string.

Operators + - * / and to the power of ** are described at http://docs.python.org/library/stdtypes.html string manipulation is also described there.

Many of the functions that you might wish to use are in the built in functions http://docs.python.org/library/functions.html and the math library http://docs.python.org/library/math.html

For example, the first function listed in the math library is ceil(x). Try it starting with a linear scale. Replace the Label Function, x, with ceil(x) and click Create Custom Slide. Now all fractions are rounded up to the next whole integer, The cursor is rounded up. If the step size is set to 0.5, all the 0.5's are rounded up too.

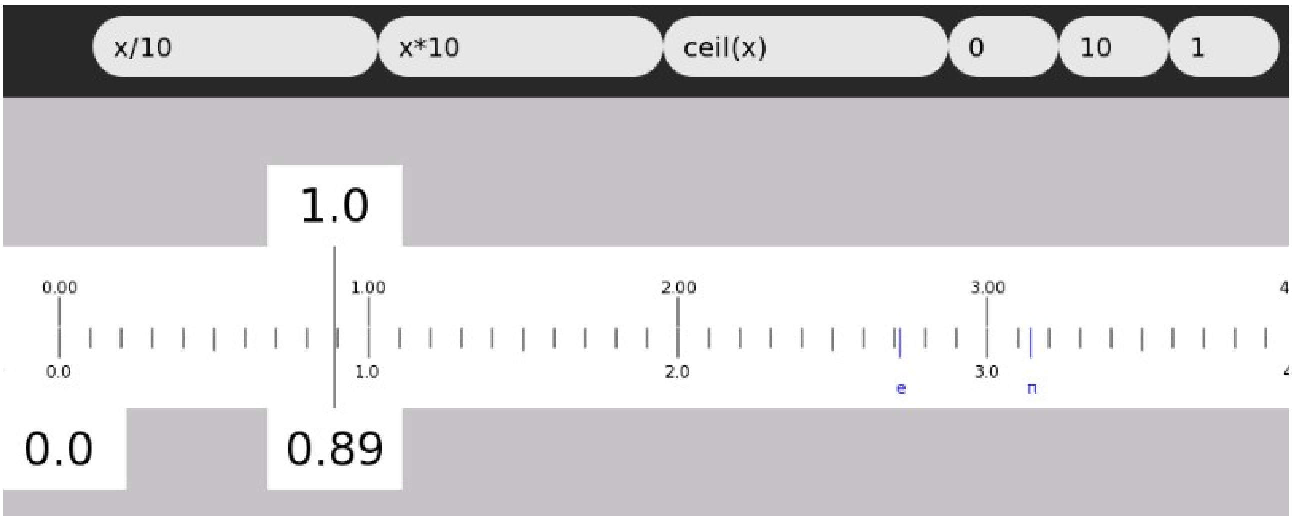


Fig 3, Custom slide rule showing a Linear scale but the labels rounded up

From the built in functions, try int(x), in this case the fractional values are rounded down to the next whole integer.

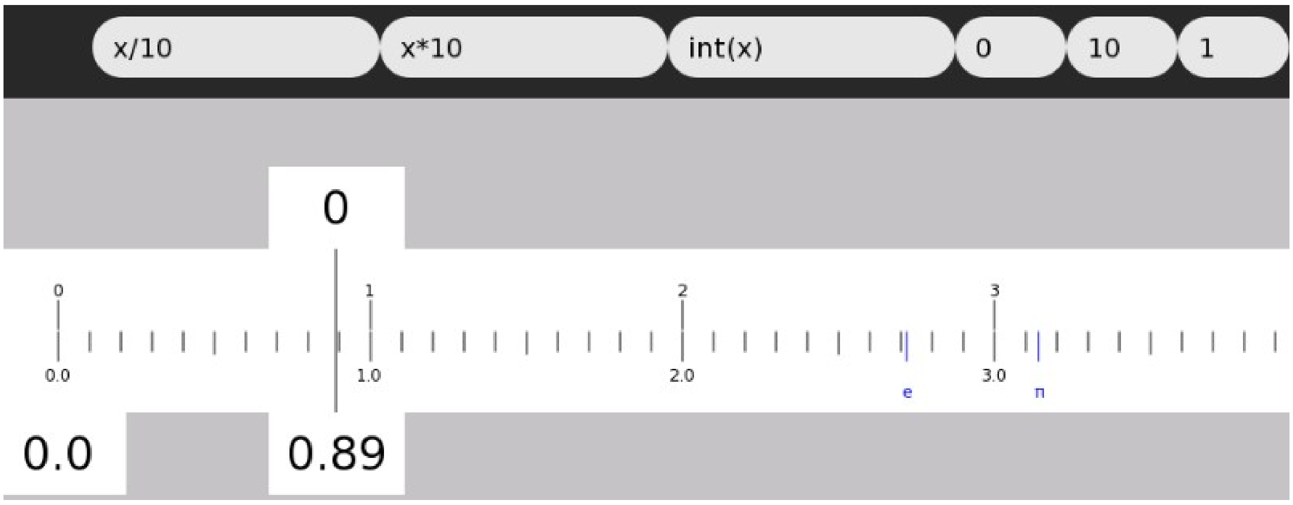


Fig 4, Custom slide rule showing a Linear scale but the labels rounded down

The Label Function also accepts strings, a string constant is denoted by leading and trailing quotes. Try “?” for the Label Function.

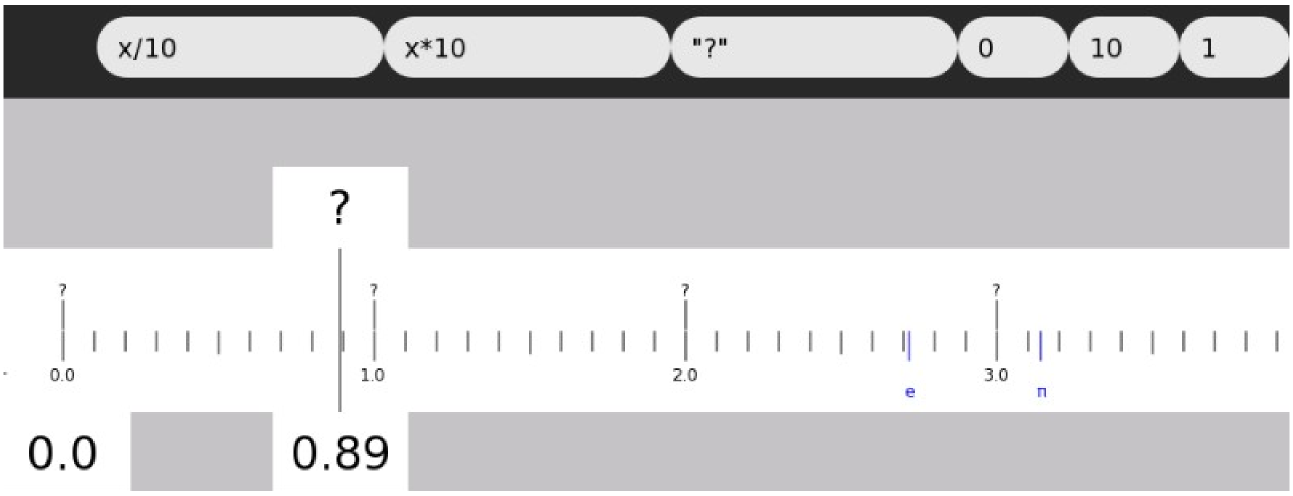


Fig 5, Custom slide rule showing string label

Strings can be concatenated or added together with the + operator. Convert ceil(x) to a string with the str() operator and concatenate with “rounded up”, enter str(ceil(x)) + ” rounded up” into the Label Function.

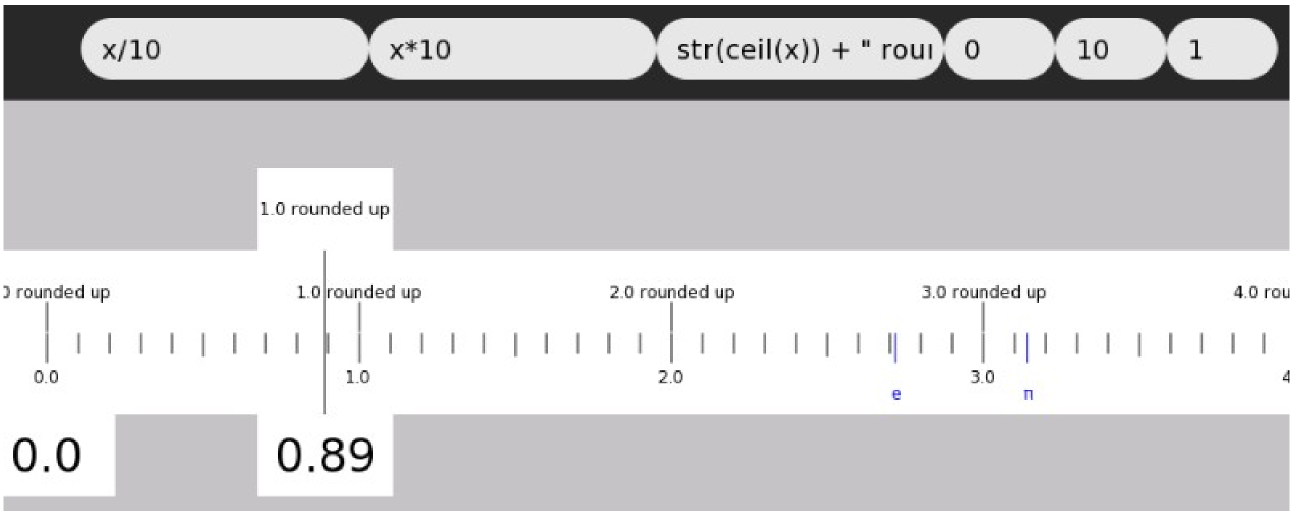
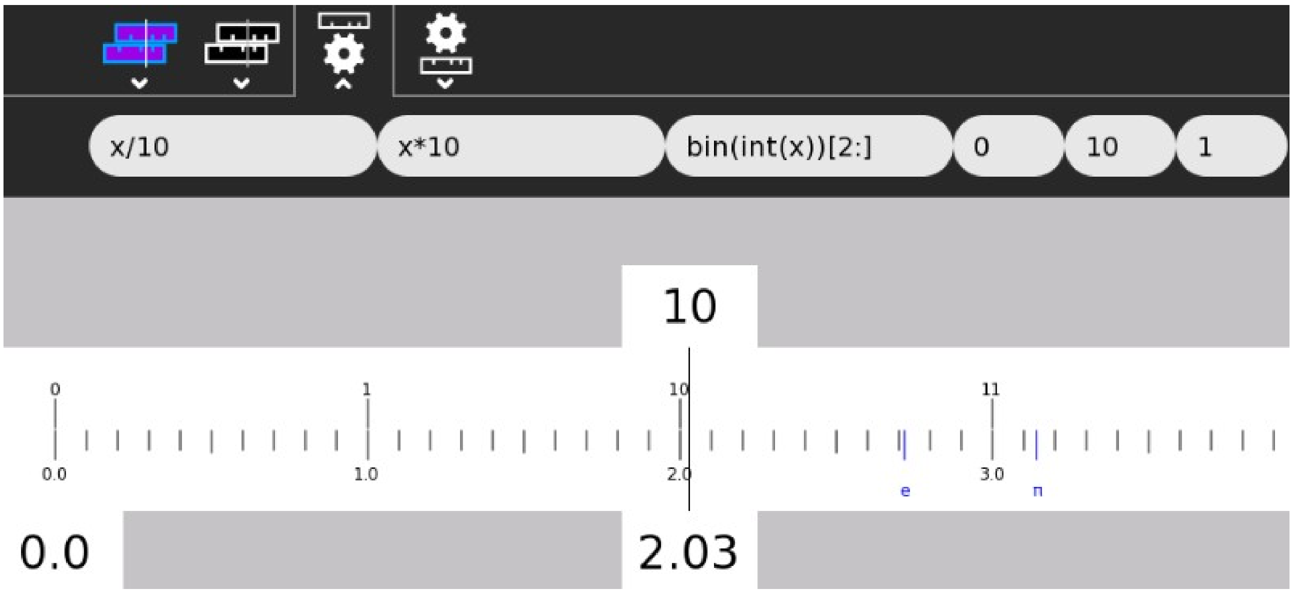


Fig 6, Custom slide rule showing string concatenation

Strings can be trimmed with slice (s[i:j]) , for example try bin(int(x))[2:] as a label with and without the [2:]


x/10 | x*10 | bin(int(x))[2:] | 0 | 10 | 1

10

0.0 | 2.03

Fig 7, Custom slide rule showing string slicing,

the leading 0b which signifies binary is removed

In processing bin(int(x))[2:], the steps are:

int converts x to a whole number eg 4.3 → 4

bin converts it to a binary number 4 → 0b100

[2:] strips the first 2 characters 0b100 → 100

Hexadecimal Add/Subtract Slide Rule

We use a decimal number system, digits 0 to 9 and numbers above 9 represented by place order eg. 10 is 1 in the 10's column and zero in the 1's. Similarly 102 is 1 x 100 +2 x 1.

Hexadecimal numbers have a base 16, they are useful in computers because they convert easily to binary numbers.

0 1 2 3 4 5 6 7 8 9 a b c d e f represent 1-15 decimal

10 hex is decimal 16, 11 hex is decimal 17 etc.

a + a = 14 in hexadecimal (10 +10 decimal)

Q1 Make a linear scale with equal divisions from 0 to 16.

Hint: start with the linear scale and change the 10's to 16's

(See the answers at the end of this document.)

Q2 Use the hex() and int() functions to display hexadecimal labels

Hint: In the label function, first convert x to an int, then convert that to a hexadecimal number

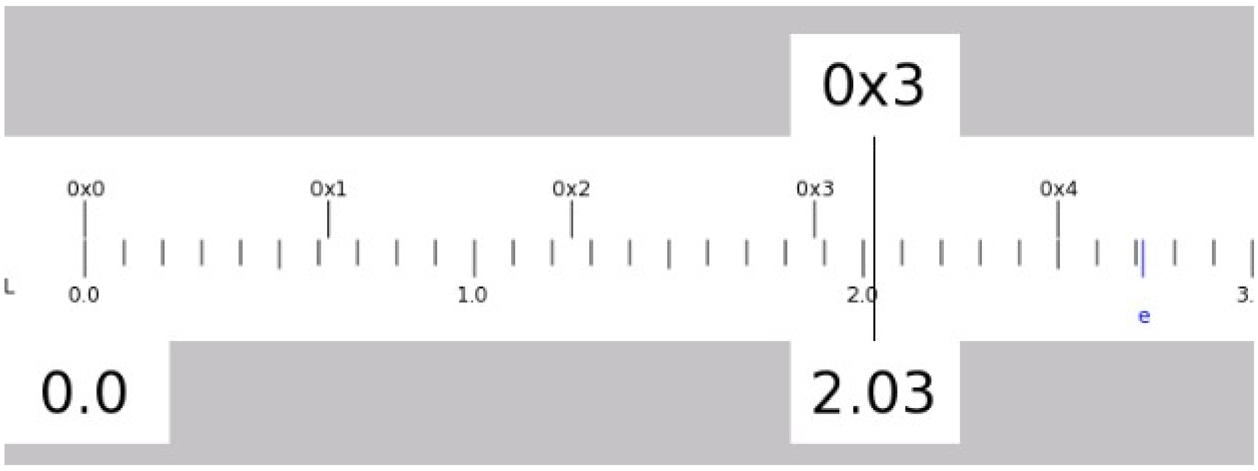


Fig 8, Custom linear slide rule showing hexadecimal labels

Q3 Use (s[i:j]) as previously to strip off the 0x

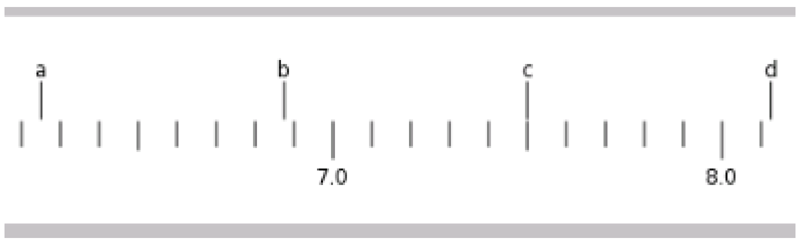


Fig 8, Custom linear slide rule showing hexadecimal labels but with 0x stripped

Extra challenge: Display one hexadecimal (as in decimal) place, eg.   a.a =10 + 10/16

Hexadecimal Multiply/Divide Slide Rule

Q4 Set up a logarithmic scale, convert the labels 1 to 16 to hexadecimal 0 to f. Do this for both scales. Simple multiplication like 2 x 3 = 6 should be easy then

Hint: start with the multiply/divide scale, change the 10's to 16's and use hexadecimal for the label function.

Q5 Now experiment with overflow.

Calculate 0xa x 0x8 , first can you guess the answer?

Hint: Just moving to the right when multiplying overflows the end of the slide rule, in tutorial 1 you saw how you could multiply by moving left by the distance from the right end of the slide to the number. Effectively dividing by the reciprocal. The correct answer will be 0xa x 0x8 = 0x50 (or in decimal 10 x 8 = 80 = 5 x 16)

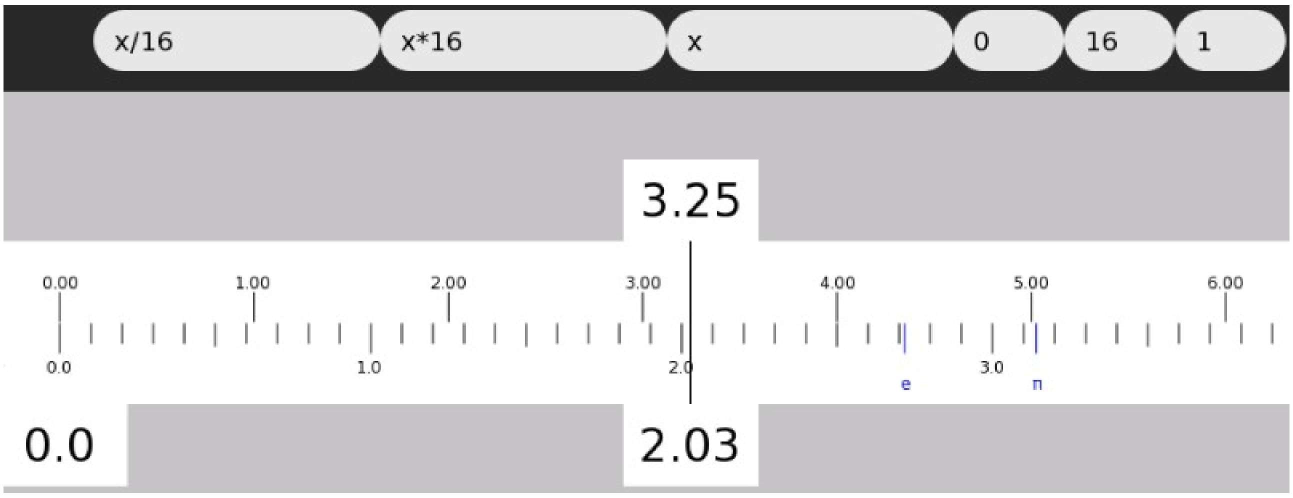
Q6 Can you explain why if the labels are in hexadecimal, the slide rule needs to range from 1 to 10hex or 16 decimal for overflow to be easily handled?

Challenge: Experiment with multiplication and division using the hex slide rule.

Challenge: Do an octal slide rule, octal uses the digits 0 to 7, the next number after 7 is 10.

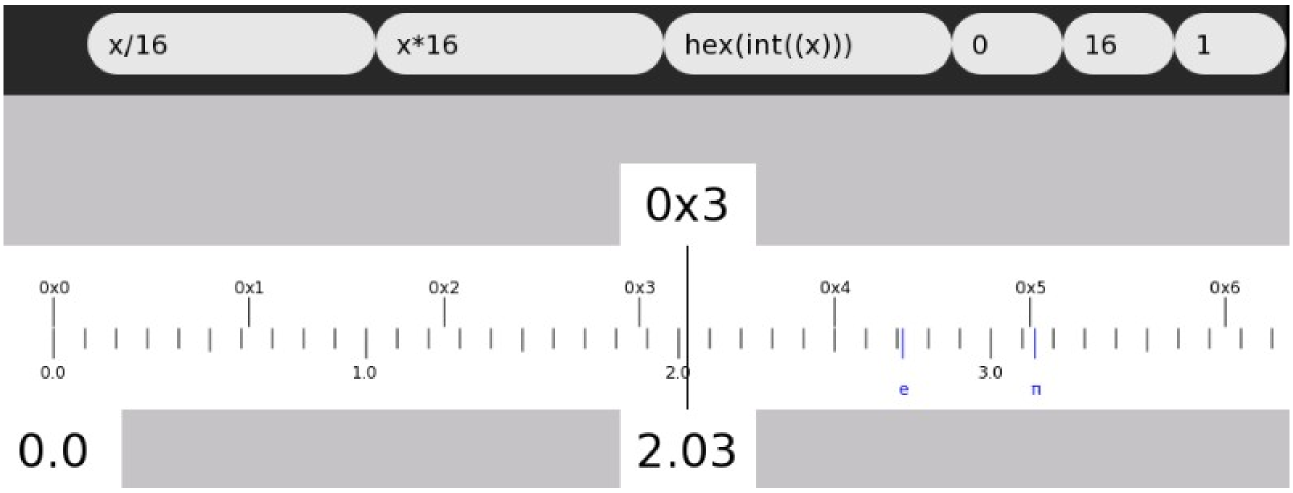
Answers

Q1



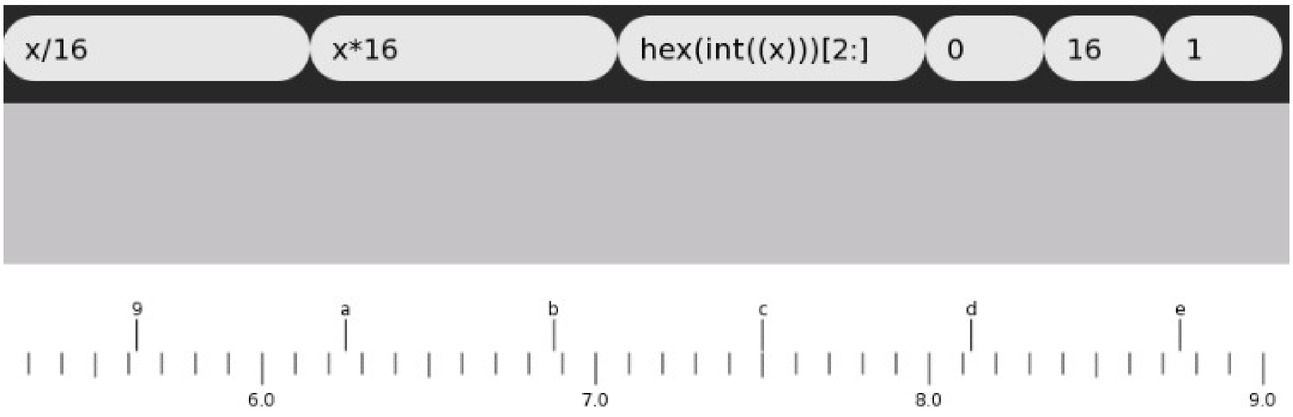
Linear scale 0-16 on the top slider

Q2



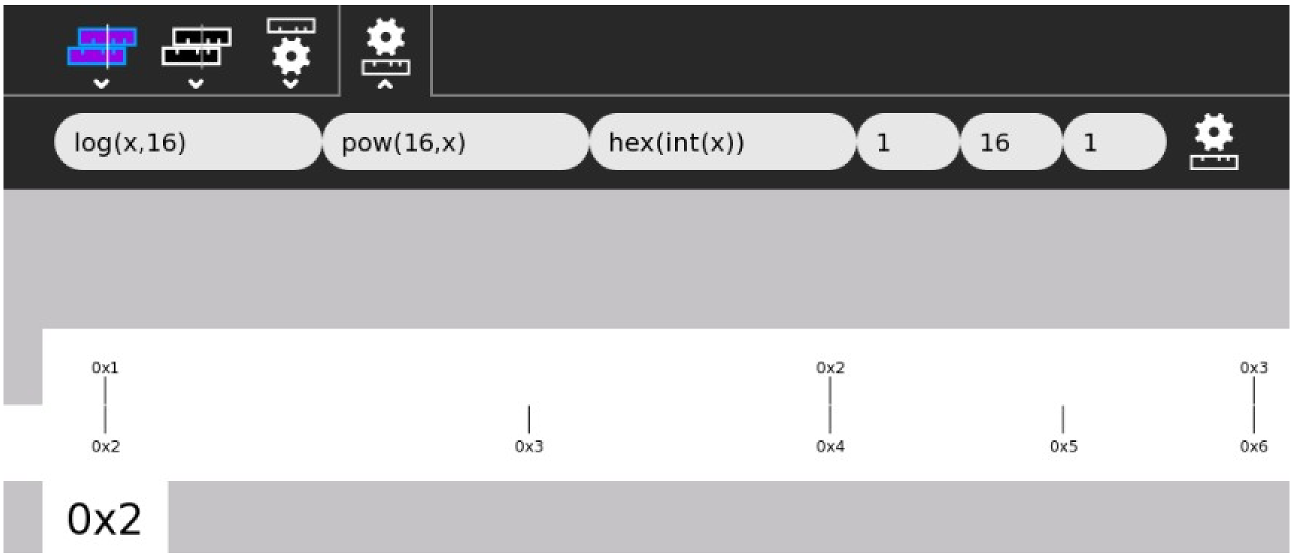
Linear scale 0x0 - 0x10 on the top slider

Q3



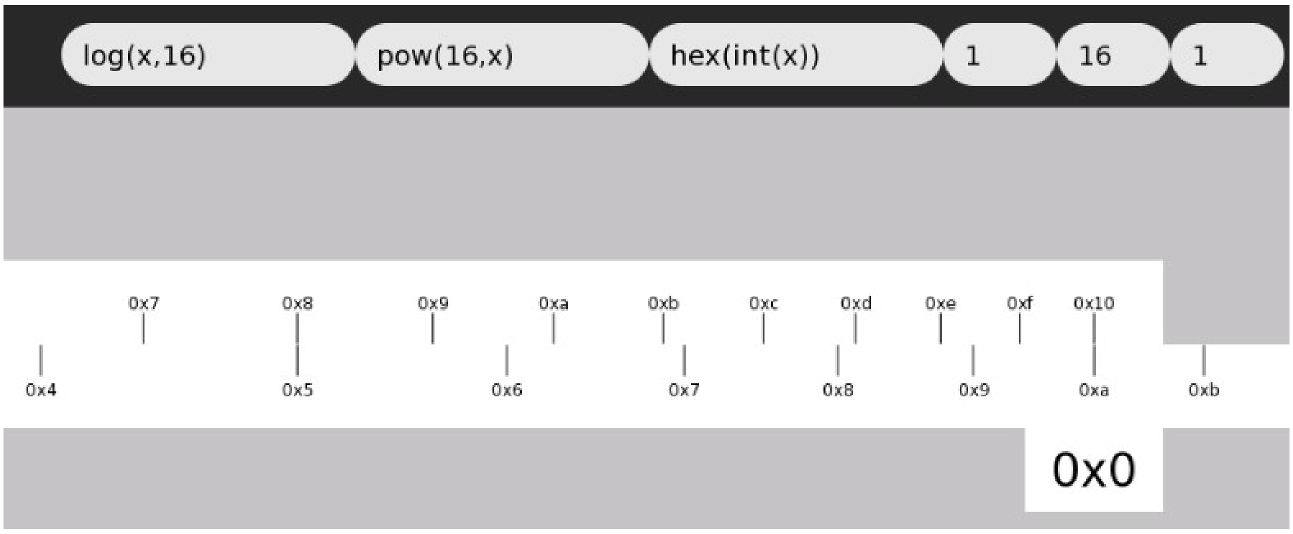
Linear scale 0 - 10 (hexadecimal) on the top slider

Q4



Shown above is 1 x 2 = 1 or 2 x 2 = 4 or 2 x 3 = 6 on log hex scales

Q5



Overflow 0xa x 0x8 = 0x50 or in decimal 10 x 8 =80 = 5 x 16

Q6 In a decimal slide rule you handle overflow by ignoring powers of 10 and fixing up the decimal places later. Similarly with the hexadecimal slide rule, ignore powers of 16 and fix up the hexadecimal places later.

In the previous question, if the distance to the left of 0x8 represents 0x8, the distance to the right represents 0x2 or 0x10/0x8, by going to the left by that distance, you are dividing 0xa by 0x2 to give 0x5, to finish the calculation you estimate the hexadecimal place to give 0x50.

This only works because the scale is 0x10 long and the 'reciprocal' to the right of a number is a reciprocal out by a factor of 0x10 and that error can easily be fixed by moving the hexadecimal place. In this case, we are using 0x10/0x8 and not the reciprocal of 0x8 = 0x1/0x8 in the division to get a result 0x10 times too small which is later fixed by moving the hexadecimal point from 0x5 to 0x50.