SmpChrt: (NEW)

Function table. Stolen from http://www.hpcalc.org/hp48/docs/faq/48faq-5.html because it was smaller and more useful than my old one. Runs on Unix-beard magic.

Input

5: Function

4: Variable (usually 'X')

3: Start

2: End

1: Step

Code:

```
« 4 DUPN 4 PICK 4
ROLLD SEQ OBJ→ COL→
6 ROLLD SEQ OBJ→
COL→ 2 COL→ »
```

fσ:

Finds standard deviation of list

Input

1: List of numbers in {}

Code

```
« DUP AVG - SQ AVG √ »
40 bytes.
```

AVG:

Averages a list of numbers.

Input

1: List of numbers in {}

Code

```
« DUP ΣLIST SWAP SIZE / »
33 bytes.
```

LOGX:

Solves for log base *n*, which confusingly is not a native calculator feature.

Input

2: Number

1: Base

Code

```
« SWAP LOG SWAP LOG / »
31 bytes.
```

SmpChrt: (OLD)

Takes an algebraic equation and a range of x values and solves it for every x value in that range.

Consider it a replacement for the TABLE mode in the Casio FX-300ES or whatever.

Input?

3: X-Min

2: X-Max

1: Equation to solve

Code

```
« SWAP DUP 'X' STO
  SWAP 0 → C B A
     « C B FOR A
       DUP EVAL
       'X' RCL 1 + 'X' STO SWAP
       NEXT
     »
»
120 bytes.
```

D2F2:

Lite version of D2F. In theory it should work great, in practice it fails a significant quantity of the time.

I have no idea how I came up with this. Was written in a notebook somewhere. Only supports improper fractions. I'd seriously consider using the built-in →Q function instead.

Input

1: Real number to convert into fraction.

Code

```
« 7 RND
  DUP 1 LCM DUP ROT / SWAP
  "" + "/" + SWAP +
»
66 bytes.
```

Distance:

The distance formula, plain and simple.

Input

X1

Y1 – Coordinates of first point

X2

Y2 – Coordinates of second point

Code

```
« ROT SWAP - SQ 3 ROLLD - SQ + √ »
48 bytes.
```

GenSQ:

Makes a list of perfect squares or cubes or whatever.

Input

A = Number to generate perfect list from

Code

```
« 0 → B A
     « {} 2 12
     FOR A A B ^ +
     NEXT
     »
»
```

```
80 bytes.
```

Example

Input: 2

Output: {4 9 16 25 36 49 64 81 100 121 144}

VtxF

Equivalent to -b/2a maybe. Used to find Vertex Forms.

Input

A: First variable for equation

B: Second variable for equation

Code

```
« NEG SWAP 2 * / »
31 bytes.
```

Example

Input: 2 5

Output:

LCM

Least Common Multiple. Stolen from some guide on the internet.

Input

[??]

Code

```
« DUP2 GCD / * »
Relies on GCD existing on system.
32 bytes.
```

Example

[??]

GCD

Greatest Common Denominator. From same book.

Input

[??]

Code

```
« WHILE OVER MOD DUP
     REPEAT SWAP
     END DROP
»
38 bytes.
```

Example

```
[??]
```

%Err

Finds percent error between two calculations.

Equivalent to | [your value - real value] / real value | x 100%

Input

1: Your measurement [?]

2: Real measurement [?]

Code

```
« OVER - SWAP / ABS 100 * »
46 bytes.
```

Example

[??]

D2F

Turns a decimal to a fraction. Hideously inefficient and shouldn't be used if possible. Made by me. That might explain it.

Input

1: A real number to turn into a fraction. Hopefully.

Code (new)

```
« ABS DUP IP → I F
```

```
     « 2 99 FOR M
          I FP M * 5 RND → W
               « W 1 MOD
               IF 0 == THEN
               W "/" M + + F " " + SWAP + KILL
               END
               »
          NEXT
     »
     "Nope." MSGBOX
»
179 bytes.
```

Code (OLD-ISH)

```
« ABS DUP → I
« FLOOR → F
« 'I-F' EVAL → D
     « 2 999 FOR M D
          M * 5 RND → W
          « W 1 MOD → R
               « IF R 0 == THEN
                    W "/" M + + F " " + SWAP + KILL
               END
          »
          »
     NEXT
»
»
» "Nope." MSGBOX
»

231 bytes.
```

Example

Input: 5.6666667

Output: “5 2/3”

CircleVol

Volume of a circle. Very simple program. My first one, actually.

Equivalent to π*r^2.

Input

1: Radius of circle

Code

```
« 2 ^ π * »
34 bytes.
```

Example

[??]