Akshay Srivatsan's Calculator

Important: This program runs on Python 2.7 (the default on Mac OS X). It does not run on Python 3.

Table of Contents:

1. Standard Operation

Running

To run this program, double-click `calculator.pyc` in Finder/Windows Explorer, or type `python calculator.pyc` at Terminal or Command Prompt. Make sure that your computer is set to use **Python 2** by default, **not Python 3**.

User Input

My calculator is designed to accept input in the most traditional text-based form (the form that is understood by Google Search). This means that I had to make minor changes to normal Python math syntax: for example, I replaced `**` with `^` for exponentiation. The regular Python syntax will still work in most cases. Functions from Python's `math` module (like `log` and `sin`) will, for the most part, work in my calculator.

The commands `quit` and `exit` will end the program.

I have programmed this calculator to interpret user input in the most logical way (using Python's `eval` function to determine the order of operations), but properly parenthesized input has a much higher chance of being interpreted correctly than unparenthesized input. For example, `-10^2` evaluates to `-100`, since it is parsed as `-(10^2)`. To get `100`, enter `(-10)^2`.

Fractions can be entered as `X/Y`, while mixed numbers can be entered as `W X/Y`. For the best results, surround fractions and mixed numbers with parentheses. If the numerator or denominator of a fraction contains a float or a parenthesized expression, it will be interpreted as division. If any part of a mixed number contains a float, it will be corrected into a proper mixed number.

I have used Python's `readline` library to allow more user-friendly input (including working arrow keys).

The parser is designed to accept decimal, but will also accept binary, octal, and hexadecimal using Python's `eval` function. These are the ways you can enter a number (in this case `42`) in different bases:

- `0b101010` (Binary - Base 2)
- `052` (Octal - Base 8)
- `0x2A` (Hexadecimal - Base 16)

Input in bases other than 10 is not guaranteed to work in all scenarios. Again, there is a greater chance of your input being interpreted correctly if it is properly parenthesized.

Output

Output will typically be in the same form as the input: fractions will be printed as `(X/Y)`, and mixed numbers as `(W X/Y)`. If the input is invalid, the program will print an error message (possibly accompanied by the location of the error). *(The location of the error is approximate, and may not be accurate for very complex inputs.)*

Output will always be in base 10, but can be converted to other bases using the functions `bin`, `oct`, and `hex`.

Variables and Aliases

This calculator also supports named variables and aliases. The difference between these is that variables are numbers which are parsed when stored, while aliases are strings which are expanded when they are used. Custom variables and aliases must have names that begin with `_`. This is to avoid the variable name from clashing with a function name.

The syntax for assigning variables is `_x = 10`. This sets the variable `_x` to `10`.

The syntax for assigning aliases is `_3x := 3*`. This sets the alias `_3x` to the string `3*`

Built in variables and aliases include:

- `_pi = 3.141592653589793`
- `_e = 2.718281828459045`
- `E := *10^`

In addition, `_ans` always evaluates to the answer to the previous expression.

Variables can be deleted by setting them to nothing (`x =`). The command `variables` will print out all currently defined variables.

2. Special Features

I have added some extra features to this calculator. A full list can be found by typing `help` at the prompt. They include:

- `table`: prints out a table of values.
- `graph`: graphs the values (as best as it can, since this is on the command-line).
- `simple`: puts the graph into simple mode.
- `complex`: puts the graph into complex mode.
- `variables`: prints out all currently defined variables.
- `derivative`: takes the derivative of a function at a point.

Table

This function prompts for:

- An expression
- The independent variable in the expression (this variable name does not have to begin with an underscore)
- The starting value
- The ending value
- The step size

It prints out a table of values for the expression with the specified characteristics.

An example of its use is:

```
Enter an expression or command: table
------------- Table Generation -------------
Enter an expression: x^2
Select a variable to change: x
Select a starting x value: -10
Select an ending x value: 10
Select a step size: 5

 x          | y
------------|-------
-10         | 100
-5          | 25
0           | 0
5           | 25
10          | 100
```

Graph, Simple and Complex Modes

This function is very similar to `table` in terms of inputs and outputs. It prompts for:

- An expression using `x` as the independent variable.
- The starting value for `x`

- The ending value for `x`
- The width of the graph (in x-units)

It then draws a rudimentary graph based on the expression.

This is a graph of `x^2` in `complex` mode:

```
Enter an expression or command: graph
------------- Graph Generation -------------
** WARNING: Axes may not be to scale. **
Enter an expression, with variable 'x': x^2
Select a starting 'x' value: -10
Select an ending 'x' value: 10
How wide should the graph be printed?: 6

100.000|\                                            /
 80.800|
 61.600|
 42.400|
 23.200|         \                          /
  4.000|                  -        -
       -------------------------------------------------------
        -10.00    -6.00    -2.00     2.00     6.00    10.00
```

This is the same graph in `simple` mode:

```
Enter an expression or command: graph
------------- Graph Generation -------------
** WARNING: Axes may not be to scale. **
Enter an expression, with variable 'x': x^2
Select a starting 'x' value: -10
Select an ending 'x' value: 10
How wide should the graph be printed?: 6

100.000|*                                            *
 80.800|
 61.600|
 42.400|
 23.200|         *                          *
  4.000|                  *        *
       -------------------------------------------------------
        -10.00    -6.00    -2.00     2.00     6.00    10.00
```

Derivative

This function accepts the following parameters:

- An expression with independent variable `x`
- An `x` value at which to take the derivative

It then calculates the slope at that point using the limit definition of the derivative, where `h =`

`0.0001`. Since it is not actually taking the limit, but rather calculating the value at a certain `h`, this will usually be a little bit inaccurate. Usually, the inaccuracy is small enough to be disregarded.

This example demonstrates taking the derivative of `_e^x` at `x=0`.

```
Enter an expression or command: derivative
------------- Derivative Calculator -------------
** WARNING: This derivative is approximated using h=0.0001. **
Enter an expression (with variable 'x'): _e^x
Select an 'x' value: 0
1.00000000042
```