Common Mistakes with Sage

Aaron Tresham

February 16, 2018

1 Common Error Messages

Many (but not all) mistakes that you make in Sage will result in an error message. This is usually several lines of output all in red. You can ignore most of the message; just look down at the last line, which tells you the type of error.

1.1 SyntaxError: invalid syntax

This is one of the broader errors, and many mistakes can lead to this error. Here are some examples:

- Missing Explicit Multiplication

  Every multiplication in Sage must be explicitly typed.

  For example, to define the function $f(x) = 5x^2+3x-2$, you must type a multiplication after the 5 and 3: `f(x)=5*x^2+3*x-2`. If you try `f(x)=5x^2+3x-2`, you will get this error.

- Extra End Parenthesis

  Every set of parentheses must have a beginning and an end: $(\cdots)$. If you have an extra end parenthesis, you will get this error. For example, try `f(x)=sin(x))`.

- Unmarked Comment

  Sometimes you may want to write a note in Sage. You can do this by putting a pound sign (#) at the beginning of the line. This tells Sage that the rest of the line is a comment and should be ignored when processing. If you leave of the comment marker, Sage will try to intepret your comment as a command, often resulting in a syntax error (may depend on exactly what's in your comment).

1.2 SyntaxError: unexpected EOF while parsing

I have seen two kinds of mistakes that lead to this message:

- Missing End Parenthesis

  This error usually results from missing an end parenthesis (or, equivalently, having an extra beginning parenthesis).

  For example, `f(x)=sin(x` will result in this error.

  This mistake is more common than having an extra end parenthesis. If you see "unexpected EOF," then check for unmatched parentheses.

- Adding a Space after a %

  Another mistake that results in this message is having a space after the % sign in a % decorator. For example, `% var` (note the space after %) instead of `%var`.

1.3 TypeError: _____ object is not callable

I have seen two kinds of mistakes result in this error:

- Missing Multiplication

  Every multiplication must be explicitly typed (with a *), and missing multiplications may result in this error. For example, try `3(5+2)`. The reason is that Sage confuses this with function notation, such as `f(5+2)`. When you type `3(5+2)`, Sage wants to call a function named "3," but of course 3 is not a function, so it is "not callable." What you want to type is `3*(5+2)`, with an explicit multiplication.

- Overwriting a Sage Command

  This error may also result if you try to use a Sage command that you have accidentally overwritten. For example, if you type `cos=5`, the cos function will be overwritten with the number 5. Then if you try `cos(2)`, you will get a "object is not callable" error (Sage tries to do 5(2), as if 5 were a function).

1.4 TypeError: _____ takes at least 1 argument (0 given)

If you fail to give a Sage command the number of arguments it expects, you will get an error like this (the numbers may be different).

For example, if you try to run `plot()` with no function to graph, you will see

"`TypeError: plot() takes at least 1 argument (0 given).`"

If you try `solve(x^2==4)`, then you will see

"`TypeError: solve() takes at least 1 positional argument (0 given).`"

This is because you forgot to specify the variable to solve for. You should have typed `solve(x^2==4,x)`.

1.5 TypeError: unable to simplify to float approximation

There may be various ways for this error to arise.

For example, if you try `find_root(x^2==4,x,0,5)`, you will get this error. The problem is the "x" – the correct syntax would be `find_root(x^2==4,0,5)` (note: you need the extra "x" when using the `solve` command, since `solve` can handle equations with more than one variable).

1.6 TypeError: invalid integration algorithm

You may see this error when using the `numerical_integral` command. For example, $\int_1^2 \frac{1}{x}\,dx \approx$ `numerical_integral(1/x,1,2)`. Notice that the regular integral command likes to have the variable of integration specified: `integral(1/x,x,1,2)`. The "x" specifies the variable. However, `numerical_integral` does not need the variable specified, since there can be only one variable involved (so the answer will be a number). If you try to specify the variable anyway, e.g., `numerical_integral(1/x,x,1,2)`, then you will get this `TypeError`.

1.7 NameError: name '__' is not defined

There are a few different situations in which a "NameError" may arise.

- Failing to Declare a Variable

  The variable x is automatically declared, but any other variable must be explicitly declared using `%var`.

  For example, you may want to find $\frac{d}{dx}ax^2+bx+c$, so you type

  `derivative(a*x^2+b*x+c,x).`

  When you run this, you will get "NameError: name 'a' is not defined."

  Instead, you need to do this:

  ```
  %var a,b,c
  derivative(a*x^2+b*x+c,x)
  ```

  Note: There are certain contexts that do not require a variable declaration. For example, if you type `y=10`, you do not have to declare y. In this case, y is not really a variable, it's just another name for 10.

- Misspelling a Command Name

  This error can also result from a simple typo. For example, if you try `f(x)=son(x)` instead of `f(x)=sin(x)` you will get "NameError: name 'son' is not defined."

- Forgetting Quote Marks

  Some options, such as the plot options `color` and `linestyle`, will require quote marks.

For example, if you want to change the plot color to black, you can add `color='black'` inside the plot command. If you leave off the quote marks, you will get "NameError: name 'black' is not defined."

1.8 RuntimeError: Error in line(): option '__' not valid

Many commands have optional arguments. For example, you can set the plot range using the xmin and xmax options: `plot(f(x),xmin=-10,xmax=10)`. If you misspell one of these options, you may get this error.

For example, if you misspell "xmax" and type `plot(x^2,xmin=0,xman=5)`, you will get "RuntimeError: Error in line(): option 'xman' not valid."

1.9 RuntimeError: f appears to have no zero on the interval

This error occurs when you try to solve an equation using the `find_root` command, but the equation has no solutions in the interval you specified.

For example, if you try `find_root(cos(x)==x,-1,0)`, you will get this error. This is because the equation $\cos(x) = x$ has only one solution ($x \approx 0.74$), and this solution is not in the interval from -1 to 0 that you gave to `find_root`.

1.10 RuntimeError: ECL says: Error executing code in Maxima

Sage uses another program called Maxima to perform many mathematical calculations; however, there are some calculations that Maxima is not able to do, which can result in this error.

For example, if you try to compute $\int \sqrt{1+\sin(x)^2}\,dx$ by typing

`integral(sqrt(1+sin(x)^2),x)`, you will get this error, since Maxima is not able to evaluate this antiderivative. If you get this error from a definite integral, then you can use the `numerical_integral` command to get a decimal approximation of the answer.

1.11 ValueError: Assumption is inconsistent

This error arises when you set assumptions about variables (we do this when dealing with improper integrals). If your assumptions are contradictory, you will get this error. For example, if you try: `assume(x<-1); assume(x>1)`, you will get this `ValueError`.

A common way to get this error is failing to "forget" previous assumptions. You may use `assume(x<-1)` for one problem, and then when you move on to a different problem you may use `assume(x>1)`. After you finish the first problem, you must run the command `forget()`, which will clear the assumptions.

1.12 ValueError: Assumption is redundant

This is similar to the last error, except you have two assumptions that are redundant rather than contradictory. For example, if you try: `assume(x>5); assume(x>1)`, you will get this `ValueError`. This can also result from failing to use `forget()`.

You must remember to forget()!

1.13 ValueError: Integral is divergent

Mathematically, this is really no error at all. This error may result when you try to calculate a divergent improper integral.

For example, try to find $\int_0^1 \frac{1}{x}\,dx$ by typing: `integral(1/x,x,0,1)`.

1.14 ValueError: Sum is divergent

This is similar to the last error, but this message results when you try to calculate the sum of a divergent infinite series.

For example, try to find $\sum_{n=1}^{\infty} \frac{1}{n}$ by typing: `sum(1/n,n,1,+Infinity)`

(note: you must have `%var n` before this line).

Some divergent series will not give you this error. Here are some examples:

`sum(n*(-2)^n,n,1,+Infinity)` will output `Infinity`.

`sum(n*(-1)^n,n,1,+Infinity)` will output `und`.

`sum(n*2^n,n,1,+Infinity)` will output `+Infinity`.

1.15 DeprecationWarning

This is not an error, but it will result in a block of red in the output. When you get a deprecation warning, Sage will output the answer you expect. However, a future version of Sage may eliminate the syntax you used to get that output.

For example, if you type `derivative(x^2,x)(3)`, then you expect to get $\left.\frac{d}{dx}x^2\right|_{x=3} = 6$. You do indeed get 6, but you also get this warning message:

"DeprecationWarning: Substitution using function-call syntax and unnamed arguments is deprecated and will be removed from a future release of Sage; you can use named arguments instead, like EXPR(x=..., y=...)"

Sage is telling you to do this: `derivative(x^2,x)(x=3)`. Sage wants you to explicitly indicate that $x = 3$. Right now, this is optional (although preferred). In a future version of Sage, $x = 3$ may be necessary.

1.16 verbose 0 (3757: plot.py, generate_plot_points) WARNING: When plotting, failed to evaluate function at 100 points.

This message appears in black rather than red. You will get this warning before a graph if your plot window includes input values not in the domain of the function you are plotting.

For example, if you try `plot(ln(x))`, then the plot will use $-1 \leq x \leq 1$ by default. However, logarithms are not defined for $x \leq 0$ (if you want the output to be a real number). Sage will produce a graph (with $0 \leq x \leq 1$), but it will warn you that it has ignored $-1 \leq x \leq 0$.

2 Other Common Mistakes

Some mistakes commonly made when using Sage *do not* result in an error message, but you don't get the right answer. You must be very careful about these!

2.1 Missing Explicit Multiplication

Sometimes missing multiplications will result in an error message (see above). However, here is an example that does not: To define the function $f(x) = (x - 3)(x + 2)$, you must type `f(x)=(x-3)*(x+2)`, with an explicit multiplication between the two factors. If you leave off this multiplication, you will *not* get an error message, but you *will* get the wrong function!

2.2 Missing Parentheses

There are many situations when parentheses are not optional.

For example, to write the fraction $\dfrac{x+y}{5}$ in Sage, you must type `(x+y)/5`. If you leave the parentheses off, you will not get an error message, you'll just get the wrong fraction: $x + y/5 = x + \dfrac{y}{5} \neq \dfrac{x+y}{5}$.

Here's an example involving powers: `e^2*x` $= e^2 \cdot x \neq e^{2x} =$ `e^(2*x)`.

2.3 Scientific Notation

If the output from Sage is `2.801232340e9`, you may think this is a number close to 2.8. However, this is actually a very large number (about 2.8 *billion*). You have to watch out for scientific notation in Sage, which is indicated by the little "e" in the output above. This output is actually $2.801232340 \times 10^9 = 2,801,232,340$.

Here is another example: `3.456400000e-12` is not close to 3.5, it is actually close to 0:

$3.456400000 \times 10^{-12} = 0.0000000000034564 \approx 0.$

2.4 Derivatives

There are two kinds of derivative questions: (1) the derivative function, such as $f'(x)$, and (2) the derivative at a particular value, such as $f'(3)$.

The derivative command in Sage answers the first question. If you want $f'(x)$, then type `derivative(f,x)`.

To answer the second question, you should use a two step process. First, assign the derivative function a name. I like to use "df" for "derivative of f", but you can use any name you want. Type `df(x)=derivative(f,x)`.

Now that you have the derivative function, the second step is to plug the particluar value into this. For example, `df(3)` is $f'(3)$, and `df(-1)` is $f'(-1)$.

If you want the second derivative, then you add a 2 to the derivative command:

`derivative(f,x,2)` is $f''(x)$.

Similarly, `derivative(f,x,10)` is the 10th derivative, $f^{(10)}(x)$.

A common mistake is to use `derivative(f,x,2)` when you want $f'(2)$. Of course, $f''(x)$ is very different from $f'(2)$.

Help me build this list! Tell me any other common mistakes you come across. Thanks.