SymPy: A Python Solution for Symbolic Computation

SymPy is an open source Computer Algebra System (CAS) that allows the use of symbolic computation.

To explain what SymPy does, when it becomes useful, and why it was developed, take the following example comparing SymPy and Python's math.py library (SymPy Development Team 2018).

Python comes with the builtin math.py library, which can be imported into any shell to perform mathematical calculations beyond simple arithmetic (Python Software Foundation 2019). A common example is the square root function, or sqrt():

In [15]:
```
import math
math.sqrt(25)
# this results in a perfect square
```

Out[15]:
```
5.0
```

The square root of 25 is a rational number, 5.0. However, in the case of irrational numbers such as `pi`, `e`, and the square root of 8, `math.sqrt()` returns an irrational number which can only be displayed as an approximation:

In [13]:
```
math.sqrt(8)
# this results in an approximation of an irrational number
```

Out[13]:
```
2.8284271247461903
```

If we are doing math that only requires an approximate decimal of the square root of 8, we can leave this equation only approximately evaluated. However, for instances which require an exact value, SymPy leaves imperfect square roots unevaluated and simplified:

In [14]:
```
import sympy
sympy.sqrt(8)
# this results in an algebraic, or symbolic computation
```

Out[14]:
```
2*sqrt(2)
```

which translates into √8 = √4·2 = 2√2

This final value, 2√2, is an exact equivalent of the approximate value `2.8284271247461903`, but is left unevaluated, thereby increasing the speed of the program and reducing the chance of errors in a code.

Python's builtin math.py library is not equipped to handle the advanced calculations and printing needs required in many science, engineering, and mathematics professions. That's where SymPy comes in. SymPy allows the use of symbolic computation, as well as the use of unicode and pretty pringting to print out equations in a more legible format, rather than the default Python equation formatting (SymPy Development Team 2018).

Makes sense, so where did SymPy come from?

SymPy is an open source library released under the BSD Licence (https://opensource.org/licenses/BSD-2-Clause (https://opensource.org/licenses/BSD-2-Clause)), and is developed and maintained by Aaron Meurer. Its first version, v0.5.13-hg, was released on April 1, 2008, with its latest release of v1.13 on September 14, 2018. The library is programmed purely with Python, and can be used by all Python versions 2.7 and above. It can also be used across all operating systems (PyPi 2018), which distinguishes it from other CAS libraries such as Axiom (Dhingra, G. 2017).

Since its first release, SymPy has become highly reknowned and popular in scientific and mathematics computing communities. Many resources have been developed by the team behind SymPy as well as external users and delevopers. A Github file called "SymPy Papers", located at https://github.com/sympy/sympy/wiki/SymPy-Papers (https://github.com/sympy/sympy/wiki/SymPy-Papers), tracks Zotero publications which cite the library. Another Github file in the SymPy Github wiki, https://github.com/sympy/sympy/wiki/Presentations (https://github.com/sympy/sympy/wiki/Presentations), lists SymPy presentations from conferences across the globe. The wiki also provides a list of cookbooks for SymPy: https://github.com/sympy/sympy/wiki/Cookbook (https://github.com/sympy/sympy/wiki/Cookbook), and countless tutorials have been written both by the maintenance team and by external developers and users: https://github.com/sympy/sympy/wiki/External-SymPy-Media%2C-Tutorials%2C-and-Presentations (https://github.com/sympy/sympy/wiki/External-SymPy-Media%2C-Tutorials%2C-and-Presentations).

Who uses SymPy and What can you do with it?

SymPy is designed for use in mathematics, engineering, and physics (PyPi 2018). It can be used in functions, in various areas of calculus, and in linear algebra: for example, simplifying expressions; computing differentials, integrals, and limits; working with matrices; solving equations. In a nutshell, SymPy can help you pass MATH1310 and MATH1410. For those who aren't York University math and science students, but who happen to be professionals in a STEM field, SymPy can greatly reduce the time and error margin involved in solving complex problems.

Direct from the main SymPy website, SymPy can be used for: "plotting, printing (like 2D pretty printed output of math formulas, or LATEX), code generation, physics, statistics, combinatorics, number theory, geometry, logic, and more" (SymPy Development Team 2018).

If that doesn't get you excited to do some Python math, I don't know what will.

There are other CAS libraries, so why choose SymPy?

Excellent question. That all depends on your mathematical needs. Axiom (http://axiom-developer.org/ (http://axiom-developer.org/)) is an excellent open source alternative, as a general computer algebra system. It is mostly used by mathematical researchers, whereas SymPy is more often used for scientific research as well as mathematics. Additionally, SymPy is available on more operating systems than Axiom, since it is a pure Python program and requires only Python 2.7 and above. Overall, both are good options, but SymPy is more accessible due to its only dependability being Python 2.7 or higher, and it's ease of use: if you're already using Python, you can use SymPy, as its language and syntax are the same (Dhingra, G 2017).

Magma (http://magma.maths.usyd.edu.au/magma/ (http://magma.maths.usyd.edu.au/magma/)) is another CAS very similar to SymPy and can handle many of the same types of expressions and mathematics. However, Magma does not have the ability to perform plotting, while SymPy, with the addition of other libraries such as Matplotlib, can perform sophisticated plots. Magma is also proprietary software, and while a free trial is available, any prolonged use requires purchase of a licence of its latest version, v2.18-2 (Guarín-Zapata, N. 2016).

Additional comparisons between SymPy and similar CAS libraries can be found at https://github.com/sympy/sympy/wiki#documentation (https://github.com/sympy/sympy/wiki#documentation). Many of these are proprietary and require purchase, often upwards of $1000 USD (SymPy Development Team 2018). SymPy, on the other hand, is not only free, but its source code can be altered and sold for profit, according to its BSD licence (Open Source Initiative ND).

SymPy is a highly user-friendly, easy to download and install CAS library with a lot of online support. Depending on your particular mathematical needs, it serves as an ideal choice for symbolic computing (SymPy Development Team 2018).

SymPy in Action:

In the introduction to this notebook there are a few minor examples of how SymPy works differently than other mathematics libraries. Below are some more advanced functions and expressions that SymPy can handle and how they make scientific and mathematical computing easier.

The basic code for first two examples come from the SymPy docs tutorials: https://docs.sympy.org/latest/tutorial/intro.html#the-power-of-symbolic-computation (https://docs.sympy.org/latest/tutorial/intro.html#the-power-of-symbolic-computation) (SymPy Development Team 2018). The basic code for last example comes from a user, Pedro Jorge De Los Santos, from Stack Overflow: https://stackoverflow.com/questions/40747474/sympy-and-plotting (https://stackoverflow.com/questions/40747474/sympy-and-plotting)

The most important feature to remember about SymPy is that variables **must be defined**. Take the expression x+2y:

In [16]:
```
from sympy import symbols
x, y = symbols('x y')
expr = x + 2*y
expr
```

Out[16]: x + 2*y

Normally, Python will evaluate the expression `x + 2*y`, but SymPy returns the variable `expr` as the expression `x + 2*y`. Now we can work with this expression as a variable:

In [21]:
```
expr + 1
```

Out[21]: x + 2*y + 1

In [23]:
```
expr - x
```

Out[23]: 2*y

In the case above, SymPy simplifies the second expression, which automatically cancels x out. This is similar to how the expression `sympy.sqrt(8)` returns `2*sqrt(2)`.

However, SymPy does not always simplify an expression, and this is most often the case when the form of the expression is not obvious, as in when there is a factored and expanded form of an equation. By default, SymPy will give the factored result when not explicitly told which form to give:

In [24]:
```
x*expr
```

Out[24]: x*(x + 2*y)

Based on the result from `expr - x = 2*y`, we might expect SymPy to return x^2+2xy from `x*expr`.

In order to simplify the result, we can tell SymPy to either factor or expand the result:

In [25]:
```
# to return an expanded result:
from sympy import expand, factor
expandedExpr = expand(x*expr)
expandedExpr
```

Out[25]: x**2 + 2*x*y

In [27]:
```
# to return a factored result:
factor(expandedExpr)
```

Out[27]: x*(x + 2*y)

Let's look at some calculus with SymPy

First, to print mathematical expressions in unicode, we use this script:

In [5]:
```
from sympy import *
init_printing(use_unicode=True)
```

This makes our expressions easier for us to read. Other print types can be used with the same function call: init_printing(); init_printing(pretty_print), init_printing(use_unicode=False), etc. However, some shells cannot print in certain types and the result will print in whichever type it can (SymPy Development Team 2018).

Now we will take the derivitive of `sin(x)*e**x`. First, we define any variables we will be using, followed by the expression:

In [6]:
```
x, t, z, nu = symbols('x t z nu')
diff(sin(x)*exp(x), x)
```

Out[6]: $e^{x} \sin\left(x\right) + e^{x} \cos\left(x\right)$

Another example:

Compute ∫(e^xsin(x)+e^xcos(x))d^x:

In [7]:
```
integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
```

Out[7]: $e^{x} \sin\left(x\right)$

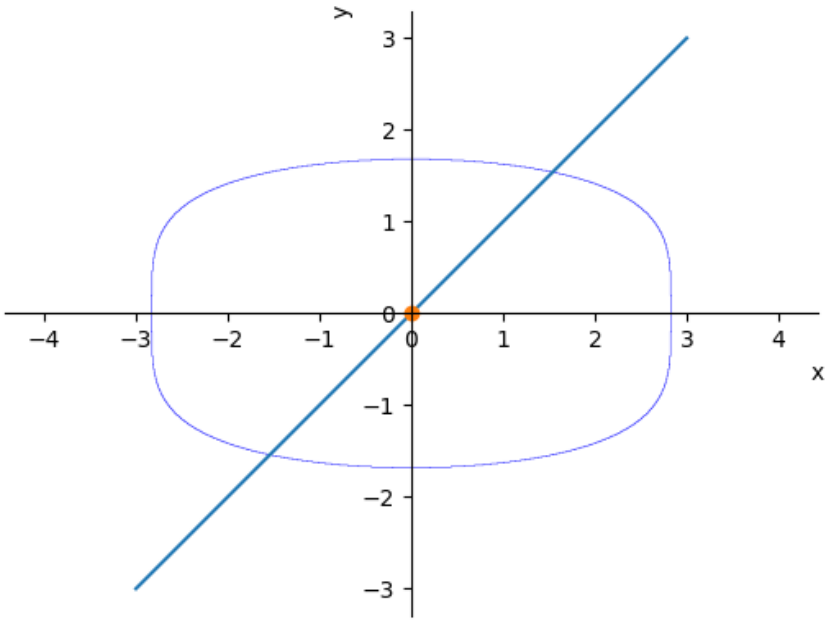
Whether you understand how to do calculus and algebra or not, the above examples illustrate the ability of SymPy not only to handle complex mathemetics, but to do so easily and symbolically, making it much faster and clearer to work with. The language and syntax is easy to learn as it uses the Python language (SymPy Development Team 2018).

This also means that SymPy calculations can integrate easily with other Python code and modules. When developing software that requires a plotted graph or other data, you can simply import SymPy into your existing code and add the script for your data, or create a separate .py file and import it with SymPy.

Plotting with SymPy and other modules:

Let's take a look at a final example program in which we create two plots on one interactive graph, integrated with the standard Python library as well as a few other libraries: Numpy and Matplotlib.pyplot.

In [1]:
```
# this program creates a graph with two plots: a red linear equation and a blue
curve.

import matplotlib.pyplot as plt
import numpy
%matplotlib notebook
from sympy import *

# define SymPy variables:
x, y = symbols("x y")

# use any integers you need to plot:
hp = plot_implicit(Eq(x**2 + y**4, 8), (x, -3, 5), (y, -5, 3))
fig = hp._backend.fig
ax = hp._backend.ax

# create list of points:
xx = yy = numpy.linspace(-3,3)

# plot functions from matplotlib:
ax.plot(xx,yy) # y = x
ax.plot([0],[0],'o') # Point (0,0)
ax.set_aspect('equal','datalim')
# show plot:
plt.show()
```



In addition to displaying two plots, this program also has a mouse-over function that displays the coordinates of the cursor location in the bottom-right corner. It also features an "off-switch" next to the plot title to end this mouse-over function (De Los Santos 2016).

This plot graph can be saved as a .py file and imported into other Python programs. It can be altered by changing the variable integer values.

Conclusion

SymPy is a highly versitile library in its ability to function alongside other modules and languages, as well as on various operating systems. Because it is written entirely in Python and depends only on having an installation of Python 2.7 or newer, it can be used by anyone with even a basic understanding of the Python language. It works well beside other mathematics and science modules, such as Numpy and Scipy, and can be used with Matplotlib's Pyplot module to plot graphs. It features the ability to print results in pretty print, LaTeX, unicode, or the standard Python mathematics syntax, which provides options for improving human readability or increasing processing speed.

Whether you're struggling through a calculus class, or you're a geomatics researcher and need to plot different wavelengths of ocean topography, SymPy will simplify your experience.

Citations

De Los Santos, P. J. (2016 November 22). Sympy and Plotting. *Stack Overflow*. Retrieved from https://stackoverflow.com/questions/40747474/sympy-and-plotting (https://stackoverflow.com/questions/40747474/sympy-and-plotting)

Dhingra, G. (2017 May 17). SymPy vs. Axiom. *Github*. Retrieved from https://github.com/sympy/sympy/wiki/SymPy-vs.-Axiom (https://github.com/sympy/sympy/wiki/SymPy-vs.-Axiom)

Guarín-Zapata, N. (2016 July 15). SymPy vs. Magma. *Github.* Retrieved from https://github.com/sympy/sympy/wiki/SymPy-vs.-Magma (https://github.com/sympy/sympy/wiki/SymPy-vs.-Magma)

Meurer, A. (2018 August 17). SymPy Wiki. *Github*. Retrieved from https://github.com/sympy/sympy/wiki (https://github.com/sympy/sympy/wiki)

Open Source Initiative. (No Date). The 2-Clause BSD License. *The Open Source Initiative.* Retrieved from https://opensource.org/licenses/BSD-2-Clause (https://opensource.org/licenses/BSD-2-Clause)

PyPi. (2018 September 14). Sympy. *PyPi*. Retrieved from https://pypi.org/project/sympy/ (https://pypi.org/project/sympy/)

Python Software Foundation. (2019 January 28). math - Mathematical Functions. *Python 3.7.2 Documentation.* Retrieved from https://docs.python.org/3/library/math.html (https://docs.python.org/3/library/math.html)

SymPy Development Team. (2018 September 14). Intro. *SymPy*. Retrieved from https://docs.sympy.org/latest/tutorial/intro.html#the-power-of-symbolic-computation (https://docs.sympy.org/latest/tutorial/intro.html#the-power-of-symbolic-computation)

SymPy Development Team. (2018 September 14). Preliminaries. *SymPy*. Retrieved from https://docs.sympy.org/latest/tutorial/preliminaries.html (https://docs.sympy.org/latest/tutorial/preliminaries.html)

SymPy Development Team. (2018 September 14). Preliminaries. *SymPy*. Retrieved from https://docs.sympy.org/latest/tutorial/printing.html (https://docs.sympy.org/latest/tutorial/printing.html)

SymPy Development Team. (2018 August 24). SympPy Gamma. Retrieved from https://www.sympygamma.com/ (https://www.sympygamma.com/)