Table of contents (../toc.ipynb)

SymPy

- SymPy is a symbolic mathematics library for Python.
- It is a very powerful computer algebra system, which is easy to include in your Python scripts.
- Please find the documentation and a tutorial here https://www.sympy.org/en/index.html (https://www.sympy.org/en/index.html)

SymPy live

There is a very nice SymPy live shell in [https://live.sympy.org/ (https://live.sympy.org/)](https://live.sympy.org/), where you can try SymPy without installation.

In [125]:
```
IFrame(src='https://live.sympy.org/', width=1000, height=600)
```

Out[125]:

SymPy

⋮

- Main Page
- Download
- Documentation
- Support
- Development
- Donate
- Online Shell

```
Python console for SymPy 1.5.1 (Python 2.7.12)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)

Warning: this shell runs with SymPy 1.5.1 and so examples pulled from
other documentation may provide unexpected results.
Documentation can be found at http://docs.sympy.org/1.5.1.
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

SymPy import and basics

In [126]:

```
import sympy as sp

%matplotlib inline
```

Symbols can be defined with `sympy.symbols` like:

```
In [127]: x, y, t = sp.symbols('x, y, t')
```

These symbols and later equations are rendered with LaTeX, which makes pretty prints.

In [128]:
```
display(x)
```

x

Expressions can be easily defined, and equations with left and right hand side are defined with `sympy.Eq` function.

In [129]:
```
expr = x**2
expr
```

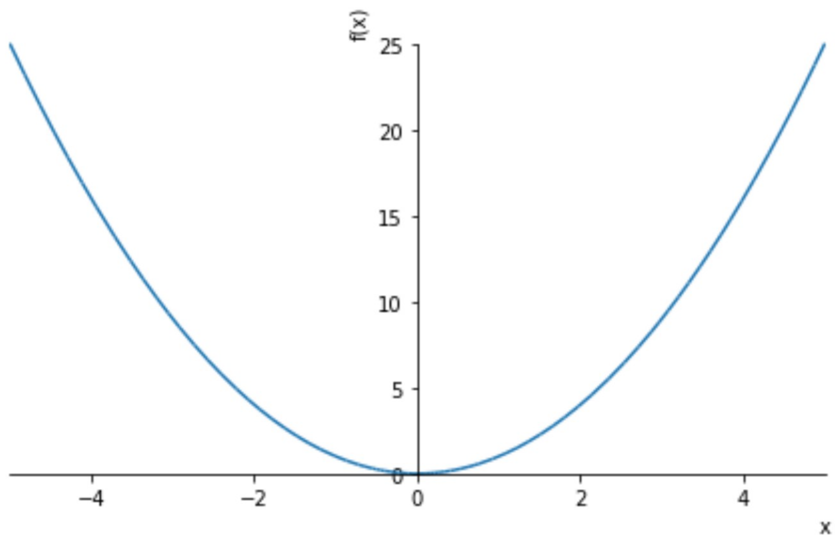
Out[129]: x^2

In [130]:
```
eq = sp.Eq(3*x, -10)
eq
```

Out[130]: $$3x = -10$$

Plots are done with `sympy.plot` and the value range can be adjusted.

```
In [131]: sp.plot(expr, (x, -5, 5))
```




```
Out[131]: <sympy.plotting.plot.Plot at 0x7fa3389a8b10>
```

Why should you consider symbolic math at all?

The power of symbolic computation is their precision. Just compare these two results.

In [132]:
```
import math
math.sqrt(8)
```

Out[132]: 2.8284271247461903

In [133]:
```
sp.sqrt(8)
```

Out[133]: $2\sqrt{2}$

You can simplify expressions and equations and also expand them.

In [134]:
```
eq = sp.sin(x)**2 + sp.cos(x)**2
eq
```

Out[134]: $\sin^2(x) + \cos^2(x)$

In [135]:
```
sp.simplify(eq)
```

Out[135]: 1

In [136]:
```
eq = x*(x + y)
eq
```

Out[136]: $x(x+y)$

In [137]:
```
sp.expand(eq)
```

Out[137]: $x^2 + xy$

In [138]:
```
sp.factor(eq)
```

Out[138]: $x(x+y)$

Differentiation and integration are built in of course.

In [139]:
```
eq = sp.sin(x) * sp.exp(x)
eq
```

Out[139]: $e^{x} \sin{\left(x \right)}$

In [140]:
```
sp.diff(eq, x)
```

Out[140]: $e^{x} \sin{\left(x \right)} + e^{x} \cos{\left(x \right)}$

In [141]:
```
sp.integrate(eq, x)
```

Out[141]: $$\frac{e^{x} \sin{\left(x \right)}}{2} - \frac{e^{x} \cos{\left(x \right)}}{2}$$

Or define an explicit interval for the integration.

In [142]:
```
sp.integrate(eq, (x, -10, 10))
```

Out[142]:
$$\frac{e^{10}\sin(10)}{2} + \frac{\cos(10)}{2e^{10}} + \frac{\sin(10)}{2e^{10}} - \frac{e^{10}\cos(10)}{2}$$

We can also easily substitute one variable of an expression.

In [143]:
```
eq.subs(x, 2)
```

Out[143]: $e^2 \sin{\left(2 \right)}$

Solve one equation. $x^2 + \sin(x) = -10.$

In [144]:

```
sp.solve(x**2 + 3*x - 10, x)
```

Out[144]:

```
[-5, 2]
```

More advanced topics

Here, we will solve a linear system of equations.

In [145]:
```
e1 = sp.Eq(3*x + 4*y, -20)
e2 = sp.Eq(4*y, -3)

system_of_eq = [e1, e2]

from sympy.solvers.solveset import linsolve

linsolve(system_of_eq, (x, y))
```

Out[145]:

$$\left\{\left(-\frac{17}{3},\ -\frac{3}{4}\right)\right\}$$

Also differential equations can be used. Let us solve $y'' - y = e^t$ for instance.

In [146]:
```
y = sp.Function('y')
sp.dsolve(sp.Eq(y(t).diff(t, t) - y(t), sp.exp(t)), y(t))
```

Out[146]:

$$y(t) = C_2 e^{-t} + \left(C_1 + \frac{t}{2}\right) e^t$$

Finally, we will have a short look at matrices.

In [147]:
```
A = sp.Matrix([[0, 1],
               [1, 0]])
A
```

Out[147]:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

In [148]:
```
A = sp.eye(3)
A
```

Out[148]:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In [149]:
```
A = sp.zeros(2, 3)
A
```

Out[149]:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Inversion of a matrix is done with `**-1` or better readable with `.inv()`.

In [150]:
```
A = sp.eye(2) * 4
A.inv()
```

In [153]:
```
A[-2] = x
A
```

Out[153]:

$$\begin{bmatrix} 4 & 0 \\ x & 4 \end{bmatrix}$$

To sum up

- SymPy is a very powerful computer algebra package!
- It is light, small, and easy to install through pip and conda.
- Simple to integrate in your Python project.