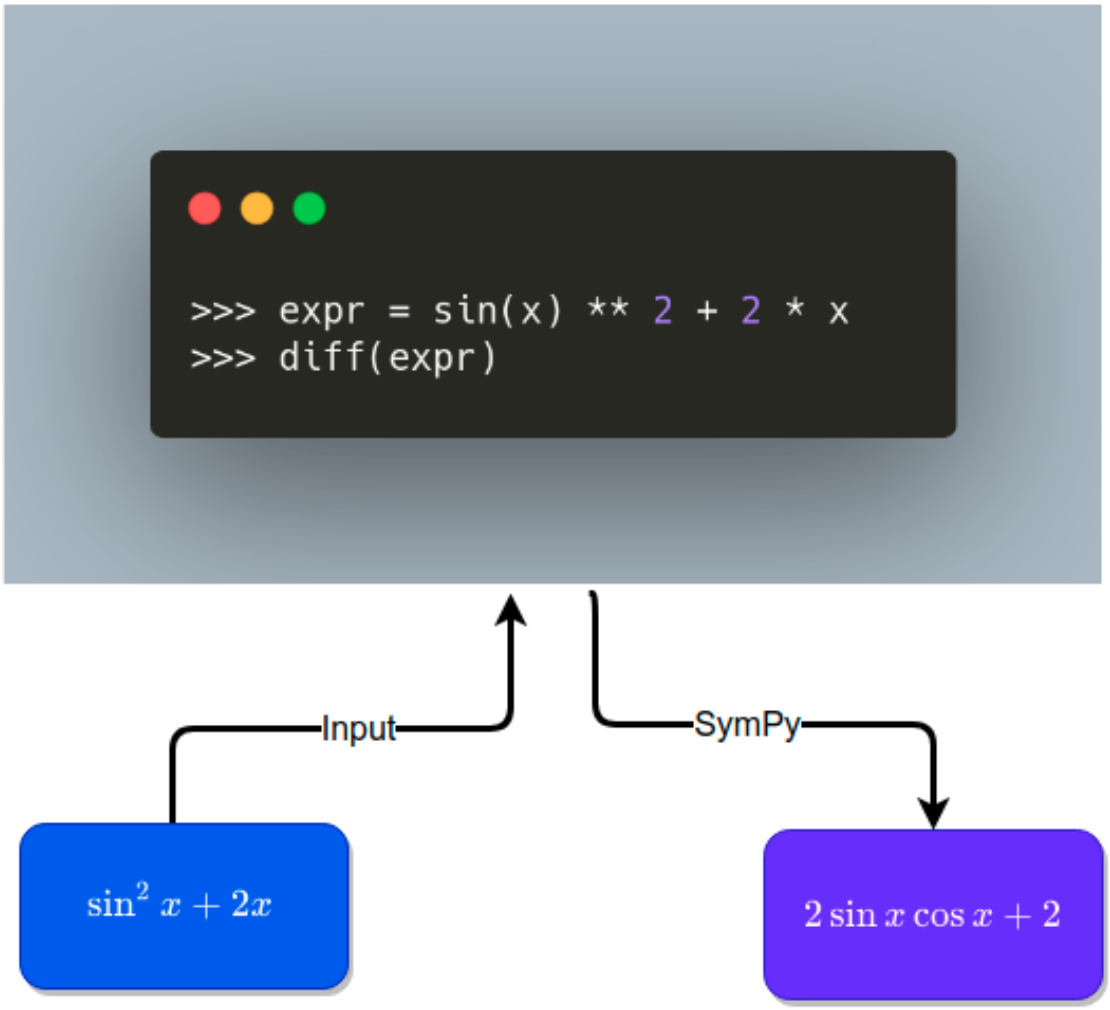
SymPy: Symbolic Computation in Python



Motivation

Have you ever wished to solve a math equation in Python? Wouldn't it be nice if we could solve an algebraic equation like below in one line of code

```
[-1/2, 0]
```

…or simply work with math symbols instead of boring Python code?

```
expr = 3 * x + y
expr
```

executed in 26ms, finished 19:14:00 2021-07-01

$3x + y$

```
expr + 2
```

executed in 19ms, finished 19:14:10 2021-07-01

$3x + y + 2$

That is when SymPy comes in handy.

What is SymPy?

SymPy is a Python library that allows you to compute mathematical objects symbolically.

To install SymPy, type:

```
pip install sympy
```

Now let's go over some of the amazing things that SymPy can do!

Start with importing all methods provided by SymPy

```
from sympy import *
```

Basic Operations

Normally, when computing a square root, we get a decimal:

```
18 ** (1 / 2)
```

executed in 5ms, finished 19:13:30 2021-07-01

```
4.242640687119285
```

But with SymPy, we can get a simplified version of the square root instead:

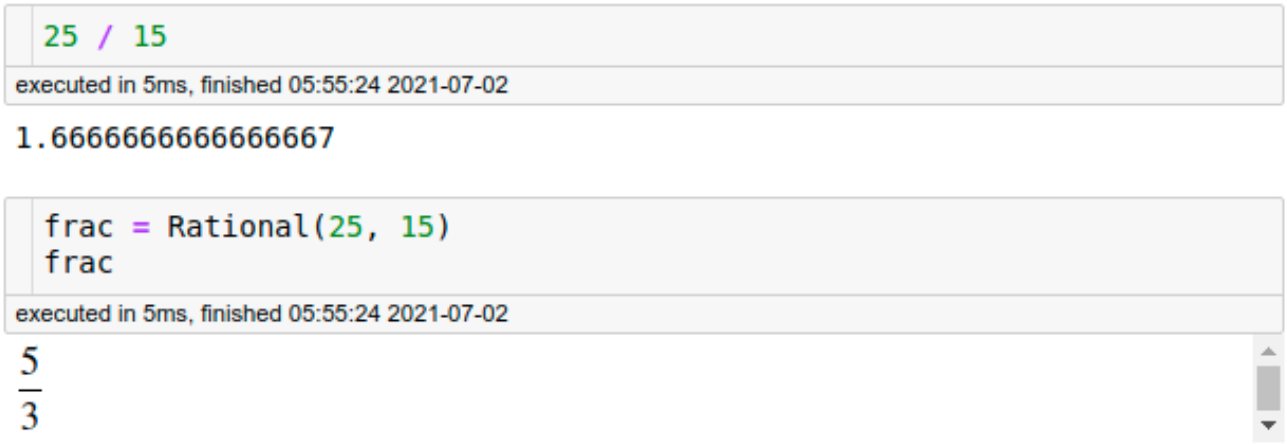
```
sqrt(18)
```

executed in 186ms, finished 19:13:30 2021-07-01

$$3\sqrt{2}$$

It is because SymPy tries to represent mathematical objects exactly instead of approximately.

Thus, we will get a fraction instead of a decimal when dividing 2 numbers using SymPy.



Symbols

The real power of SymPy is its ability to work with symbols. To create symbols, use the method `symbols()` :

```
x, y = symbols("x y")
expr = 3 * x + y
expr
```

executed in 29ms, finished 19:13:42 2021-07-01

$3x + y$

Cool! We can create an expression in terms of x and y. What happens if we add a number to this expression?

```
expr + 2
```

executed in 14ms, finished 19:37:20 2021-07-01

$3x + y + 2$

Aha! + 2 is added to the expression and the expression remains unevaluated.

Why would working with symbols be useful? Because we can now use all kinds of math tricks we learn in school such as expanding, factoring and simplifying an equation to make our life easier.

Equations

Expand, Factor, and Simplify

We know that the expansion of the expression on the left is equal to the expression on the right.

$$x(3x + y) = 3x^2 + xy$$

Can this be done with SymPy? Yes! SymPy allows us to expand an equation using `expand` :

```
expansion = expand(x*expr)
expansion
```

executed in 4ms, finished 14:44:02 2021-07-01

$$3x^2 + xy$$

Cool! We can also factor our expression by using `factor` :

```
factor(expansion)
```

executed in 6ms, finished 14:44:02 2021-07-01

$$x\,(3x + y)$$

Another cool thing we can do with SymPy is to simplify an equation using `simplify` :

```
expr = (6*x**2 + 3*x)/(3*x)
expr
```

executed in 7ms, finished 14:44:02 2021-07-01

$$\frac{6x^2 + 3x}{3x}$$

```
simplify(expr)
```

executed in 11ms, finished 14:44:02 2021-07-01

$$2x + 1$$

Solve an Equation

One of the most common questions we see when dealing with mathematical symbols is to solve an equation. Luckily, this can also be done with SymPy.

To solve an equation, use `solve` :

```
eq = (2 * x + 1) * 3 * x
eq
```

executed in 29ms, finished 19:59:13 2021-07-01

$$x\,(6x + 3)$$

```
solve(eq, x)
```

executed in 10ms, finished 14:44:08 2021-07-01

```
[-1/2, 0]
```

Substitution

If we substitute the equation below by 2, what do we get?

$$x\,(6x + 3)$$

We can figure that out using `eq.subs(x, 2)` :

```
eq.subs(x, 2)
```

executed in 11ms, finished 18:38:55 2021-07-01

30

We can also substitute x with another variable to get an expression like below:

```
eq.subs(x**2, 2)
```

executed in 43ms, finished 18:39:15 2021-07-01

$$x\,(6x + 3)$$

Trigonometric

Remember all the fun trigonometric identities we learn in high school?

$$\cos x = \frac{1}{\sec x}$$

$$\tan x = \frac{\sin x}{\cos x}$$

$$\sin^2 x + \cos^2 x = 1$$

$$1 + \cot^2 x = \frac{1}{\sin^2 x}$$

Instead of looking up these identities, wouldn't it be nice if we can use SymPy to figure this out for us?

To simplify expressions using trigonometric identities, use `trigsimp()`

```
trigsimp(1/sec(x))
```

executed in 9ms, finished 14:53:51 2021-07-01

$$\cos(x)$$

```
trigsimp(sin(x)/cos(x))
```

executed in 16ms, finished 14:54:17 2021-07-01

$$\tan(x)$$

```
trigsimp(sin(x)**2 + cos(x)**2)
```

$$1$$

```
trigsimp(1 + cot(x) ** 2)
```

executed in 96ms, finished 20:09:08 2021-07-01

$$\frac{1}{\sin^2(x)}$$

Derivatives, Integrals, and Limit

With SymPy, you can also do calculus! What is the derivative of the expression below?

```
expr = sin(x) ** 2 + 2 * x
expr
```

executed in 24ms, finished 20:22:25 2021-07-01

$$2x + \sin^2(x)$$

If you cannot figure it out, don't worry. We can use SymPy to figure it out.

```
res = diff(expr)
res
```

executed in 37ms, finished 20:22:25 2021-07-01

$$2\sin(x)\cos(x) + 2$$

Now let's try to get back to the original expression by taking the integral of the derivate.

```
integrate(res)
```

executed in 35ms, finished 18:50:49 2021-07-01

$$2x + \sin^2(x)$$

We can also take the limit as x approaches infinity.

$$\lim_{x \to \infty} \frac{1}{x^2}$$

```
limit(
    1 / (x ** 2),
    x,
    oo,
)
```

executed in 26ms, finished 18:54:06 2021-07-01

$$0$$

or 2:

```
limit(
    1 / (x ** 2),
    x,
    2,
)
```

executed in 31ms, finished 18:54:12 2021-07-01

$$\frac{1}{4}$$

Special Functions

SymPy also provides special functions to:

- Find the factorial of a number

```
factorial(x)
```

executed in 18ms, finished 20:19:47 2021-07-01

$$x!$$

```
factorial(3)
```

executed in 10ms, finished 18:33:49 2021-07-01

$$6$$

- Rewrite the expression in terms of another

```
tan(x).rewrite(cos)
```

executed in 35ms, finished 18:35:13 2021-07-01

$$\frac{\cos\left(x - \frac{\pi}{2}\right)}{\cos\left(x\right)}$$

Print Latex

If you like the output and want to get the LaTex form of an expression, use `latex` :

```
print(latex(integrate(res)))
```

executed in 12ms, finished 20:29:51 2021-07-01

```
2 x + \sin^{2}{\left(x \right)}
```

You could paste this LaTex form to your notebook's markdown and get a nice mathematical expression like below!

```
$2 x + \sin^{2}{\left(x \right)}$
```

$2x + \sin^2(x)$

Conclusion

You have just learned how to compute mathematical objects symbolically in Python using SymPy. The next time you solve math, try using SymPy to make your life easier.

There are so many other useful methods SymPy provides that I could not cover here. I encourage you to check out this documentation for more inspiration.