SymPy

Ben Bolker

10:16 30 March 2015

SymPy is a Python library that implements a **computer algebra system**, like (parts of) Maple or Mathematica or MATLAB's symbolic math toolbox

- sympy cheat sheet (from sympy github)

Basics

- need to define **symbols** first

```
from sympy import *         ## not usually recommended but OK here
x, y = symbols(('x','y'))   ## names don't *have* to match but should usually
## or
x = Symbol('x')
print(x)
print(y)
print(x+x+x)  ## surprise!
```

```
## x
## y
## 3*x
```

- then we can do anything we like:
- `.factor()`: polynomial factoring

```
z = x**2-x*y-x
z2 = z.factor())
print(z2)
```

```
## x*(x - y - 1)
```

- `.expand()`: multiply out a polynomial

```
print(z2.expand())
```

```
## x**2 - x*y - x
```

- `.collect(x)`: collect terms in powers of `x`

```
print(z.collect(x))
```

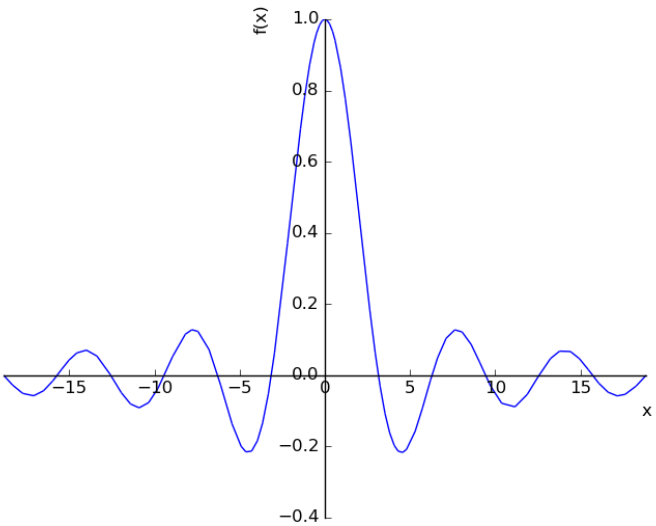
```
## x**2 + x*(-y - 1)
```

```
print(z.subs(y,0))
```

- `.subs(x,val)`: substitute values

```
## x**2 - x
```

Calculus

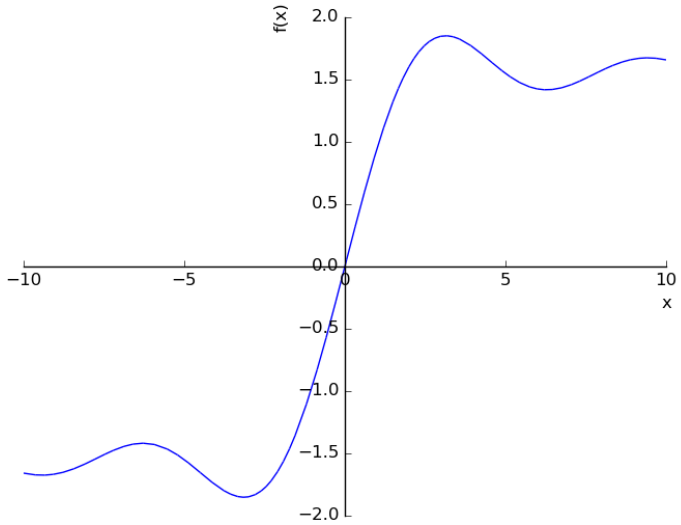


Limits:

```
f = sin(x)/x
print(f.subs(x,0))
print(limit(f,x,0))
```

Integrate and differentiate:

```
print(diff(f,x))
i1 = integrate(f,x)
print(i1)    ## what's this??
plot(i1)
```




```
## Si(x)
```

Series expansion:

```
print(i1.series(x,0,5))
```

```
## x - x**3/18 + O(x**5)
```

Solving equations

When solving equations you either need to adjust so the right-hand side of the equation is zero, or use `Eq()`:

```
from sympy import *
x = Symbol('x')
s1 = solve(exp(-x)-x,x)
s2 = solve(Eq(exp(-x),x),x)
print(s1)
print(s1==s2)
print(s1[0].n(10))
```

```
## [LambertW(1)]
## True
## 0.5671432904
```

This solution is expressed in terms of the Lambert W function, defined as the solution to the equation $z = W(z)e^{W(z)}$.

Constants

```
from sympy import *
oo ## infinity
print(E.n(10)) ## e
```

```
## 2.718281828
```

Convolutions

Arbitrary-precision arithmetic: mpmath