Symbolic Python

CS*101* `lec19`

`Symbolic Algebra`

2019-11-25

Announcements

quiz: `quiz19` due on Tues 26/11

lab: `lab10` 29/11

hw: `hw10` due Wed 27/11

Missing Lab and Quiz

Roadmap

Week 1 (9/9): L0 Intro; L1 PY - Intro	Week 2 (16/9): L2 PY – Data& Bool; L3 PY – if & Loops	Week 3 (23/9): L4 PY - Lists; L5 PY - Funcs	30/9: L6 PY - Files
9/10: L7 PY - Muta; 12/10: Exam 01	Week 6 (14/10): L8 PY - Dict; L9 PY - IO	Week 7 (21/10): L10 PY - Numpy; L11 PY - Plot	Week 8 (28/10): L12 PY - State; L13 PY - Rand
Week 9 (4/11): L14 PY – De-bug; L15 PY - Eqn	Week 10 (11/11): L16 Brute; Exam 02	Week 11 (18/11): L17 PY - Opt II; L18 PY - Mod	Week 12 (25/11): L19 PY – Sym I; L20 PY – Sym II
Week 13 (2/12): L22 ML - Intro; L23 ML - IO	Week 14 (9/12): L24 ML - Poly; L25 ML - Stats	Week 15 (16/12): L26 ML - Fit; Re-view	Week 16 (23/12): Study Week
Week 17 (30/12): Finals			

Objectives

A. Use SymPy to establish symbolic variables.
B. Solve algebraic expressions analytically.
C. Factorize expressions.
D. Plot expressions using SymPy.

Review

Question

if `xx.py` runs as a main program, `'__name__'` == `'__main__'`

if `xx.py` is ran as `import xx` in another program, `'__name__'` for `xx.py` is `'__xx__'`

Symbolic Algebra

python

```
from math import pi
>>> pi
3.141592653589793
```

python

```
from math import pi
>>> pi
3.141592653589793
```

We have been using python as a simple calculator. Can we use python to represent an equation? For example,

$$ax^2 + bx + c = 0$$

and solve it to get

$$x = \left[\frac{1}{2a}\left(-b + \sqrt{-4ac + b^2}\right), \quad -\frac{1}{2a}\left(b + \sqrt{-4ac + b^2}\right)\right]$$

Symbolic Quantity

Yes!

```
import sympy
import sympy as sy  # rename it, it's easier
```

Symbolic Quantity

Yes!

```
import sympy
import sympy as sy  # rename it, it's easier
```

sympy provides symbolic and related mathematical functions.

Need to define variable

```
>>> x = sy.S( 'x' ) #or sy.Symbol('x')
>>> x*2
2*x

>>> a,b = sy.S( 'a,b' )
```

Question 1

```
>>> import sympy as sy
>>> x = sy.S( '2 * x + 3' )
>>> 3 * x
```

What is the value of x?

A error

B 6 * x + 9

C 3 * ' 2 * x + 9 '

Question 1

```
>>> import sympy as sy
>>> x = sy.S( '2 * x + 3' )
>>> 3 * x
```

What is the value of x?

A error

B 6 * x + 9 ★★★

C 3 * ' 2 * x + 9 '

Question 2

```
>>> import sympy as sy
>>> x = sy.S( 'y + 3' )
>>> y = sy.S( '2 * x' )
>>> z = 2 * x + 4 * y
```

What is the value of z?

A error

B 2 * y + 6 + 8 * x

C 2 * x + 4 * y

Question 2

```
>>> import sympy as sy
>>> x = sy.S( 'y + 3' )
>>> y = sy.S( '2 * x' )
>>> z = 2 * x + 4 * y
```

What is the value of z?

A error

B 2 * y + 6 + 8 * x ⋆⋆⋆

C 2 * x + 4 * y

sympy.init_printing

We can make the results from sympy look more mathematically familiar,

```
>>> sympy.init_printing()
>>> sympy.exp( -x ** 2 )
```

$$e^{-x^2}$$

math functions

Sympy also contains its own math library

```
>>> sympy.sqrt(8)
2*sqrt(2)
```

- `sympy.I` is sqrt(-1) is `j` in python's complex number
- `sympy.re`, `sympy.im`, `sympy.pi`
- `sympy.E` is e**1 = 2.718281828459...
- `sympy.exp`, `sympy.log`, `sympy.sin` and related,
- `sympy.sqrt` and others

Solving equation analytically

Steps:

1. Define symbolic quantities, e.g., a = sympy.S('a')
2. Define the equation to solve and set to 0, e.g., x+2 = 0
3. Use sympy.solve(your equation, variable to solve)

The answer is stored in a `list` data type

Solving equation analytically

To solve for:

$$ax^2 + bx + c = 0$$

1. Define sym quantities: `a,b,c,x = sympy.S('a,b,c,x')`
2. Define eqn: `eqn = a*x**2+b*x+c`

Solving equation analytically

To solve for:

$$ax^2 + bx + c = 0$$

1. Define sym quantities: `a,b,c,x = sympy.S('a,b,c,x')`
2. Define eqn: `eqn = a*x**2+b*x+c`
3. `x = sympy.solve(eqn, x)`

Solving equation analytically

To solve for:

$$ax^2 + bx + c = 0$$

1. Define sym quantities: `a,b,c,x = sympy.S('a,b,c,x')`
2. Define eqn: `eqn = a*x**2+b*x+c`
3. `x = sympy.solve(eqn, x)`

Your anwer:

$$x = \left[\frac{1}{2a}\left(-b + \sqrt{-4ac + b^2}\right), \quad -\frac{1}{2a}\left(b + \sqrt{-4ac + b^2}\right)\right]$$

Substitute value

$$x = \left[\frac{1}{2a}\left(-b + \sqrt{-4ac + b^2}\right), \quad -\frac{1}{2a}\left(b + \sqrt{-4ac + b^2}\right) \right]$$

To get a value of `x` for `a = 1, b = 2, c = 1`

```
>>> x[0].subs(a,1).subs(b,2).subs(c,1)
-1
```

Not all equation can be solved!

Naturally, there are limits to its ability as mathematical techniques don't render closed-form solutions to every equation.

```
>>> sympy.solve( a*x**5+b*x+c,x )
 [ ]
```

An empty list means cannot be solved by any technique known to sympy. You may also see a `NotImplementedError` if sympy cannot even identify a way to proceed.

Question 3

```
>>> import sympy as sy
>>> x, y = sy.S( 'x, y' )
>>> eq1 = x + y - 6
>>> eq2 = - y + x + 4
>>> z = sy.solve((eq1,eq2), (x, y))
```

What is the value of z?

A error

B x : 1

C { x: 1, y: 5 }

Question 3

```
>>> import sympy as sy
>>> x, y = sy.S( 'x, y' )
>>> eq1 = x + y - 6
>>> eq2 = - y + x + 4
>>> z = sy.solve((eq1,eq2), (x, y))
```

What is the value of *z*?

A error

B x : 1

C { x: 1, y: 5 } ⋆⋆⋆

Polynomials and Expressions

Expand and Factor

1. We can `sympy.expand` to create a polynomial
2. `sympy.factor` to factor a polynomial

Assume x is already a symbol,

```
>>> y = x**2+4*x+4
>>> sy.factor(y)
```

$$(x+2)^2$$

```
>>> sympy.expand( (x+1)*(x-1) )
```

$$x^2 - 1$$

Simplify

Another function that can help to make your complicated equation looks easier,

```
>>> symsy.simplify((x**3 + x**2 - x - 1)/
                                    (x**2 + 2*x + 1))
x-1
```

You can use any of these three functions (`.expand`, `.factor`, `.simplify`) to change your equation depending on your needs.

Rational Expressions

What is a rational number? What is an irrational number?

Rational Expressions

What is a rational number? What is an irrational number?

sympy preserves simple rational expressions automatically

```
>>> sympy.S( '1/60' )
1/60
```

sympy does NOT combines rational expressions by default

$$\frac{b}{c} + \frac{x}{a}$$

together and apart

sympy combines rational expressions using `together()`

```
>>> sympy.together( b/c+x/a )
```

$$\frac{1}{ac}(ab + cx)$$

sympy uses `apart()` to perform a partial fraction decomposition on a rational function

Trigonometric functions

`Sympy` supports `sin`, `cos`, `tan`, etc and their inverses. Do not use those from `math` implementation. (may still work but maybe not be what you want)

```
>>> sympy.sin( 0 )
0

>>> sympy.cos( sympy.pi )
-1
```

Question 4

```
>>> sympy.simplify(sympy.sin(x)**2 +
                                      sympy.cos(x)**2)
```

Question 4

```
>>> sympy.simplify(sympy.sin(x)**2 +
                                   sympy.cos(x)**2)
```

ans: 1

```
>>> sympy.simplify(math.sin(x) + math.cos(x))
```

Question 4

```
>>> sympy.simplify(sympy.sin(x)**2 +
                                       sympy.cos(x)**2)
```

ans: 1

```
>>> sympy.simplify(math.sin(x) + math.cos(x))
```

ans: error

```
>>> sympy.expand((sympy.cos(x) + sympy.sin(x))**2)
```

Question 4

```
>>> sympy.simplify(sympy.sin(x)**2 +
                                        sympy.cos(x)**2)
```

ans: 1

```
>>> sympy.simplify(math.sin(x) + math.cos(x))
```

ans: error

```
>>> sympy.expand((sympy.cos(x) + sympy.sin(x))**2)
```

ans: sin(x)**2 + 2*sin(x)*cos(x) + cos(x)**2

Plotting

.plot

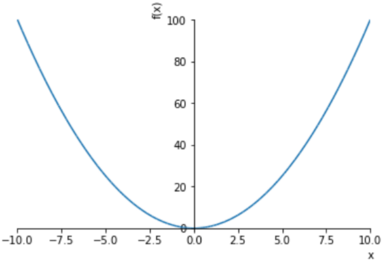
Sympy can also plot expressions

```
>>> sympy.plotting.plot( x**2 )
```

.plot

Sympy can also plot expressions

```
>>> sympy.plotting.plot( x**2 )
```

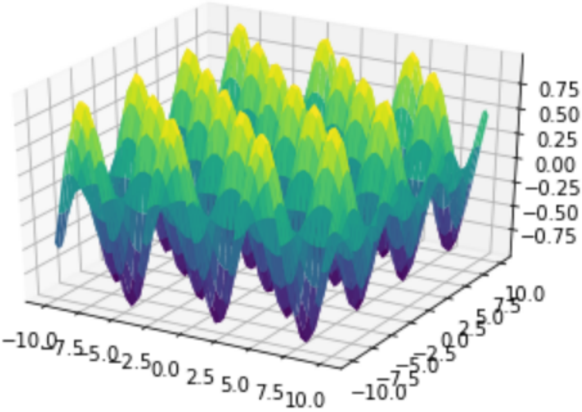



```
>>> sympy.plotting.plot( x**2,( x,-2,2 ) )
# this limits the -2<=x<=2
# Use a tuple to specify the range
```

.plot3d

Plotting 3d surfaces where `z = f(x,y)`

```
>>> sympy.plotting.plot3d( sympy.cos( x )
                                        *sympy.sin( y ) )
```

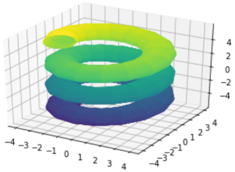


.plot3d_parametric_???

```
.plot3d_parametric_surface(x,y,z)
.plot3d_parametric_line(x,y,z)
```

Parametric surfaces are determined by functions for x, y, and z in two variables:

```
x(u,v),y(u,v),z(u,v)
>>> u,v = sympy.S( 'u,v' )
>>> x = ( 3 + sympy.cos( u ) ) * sympy.cos( v )
>>> y = ( 3 + sympy.cos( u ) ) * sympy.sin( v )
>>> z = sympy.sin( u ) + 0.5 * v
>>> sympy.plotting.plot3d_parametric_surface(x,y,z)
```



Summary

Summary

A. Sympy and its mathematics library
B. `.solve()`
C. Polynomials and Expressions: `.expand()`, `.factor()`, `.simplify()`
D. Rational numbers: `.together()`, `.apart()`
E. Trigonometry functions and other numbers
F. `.plot()` and related