ISE 314X
Computer Programing for Engineers

SymPy for Symbolic Mathematics

Yong Wang
Assistant Professor
Systems Science & Industrial Engineering
Binghamton University

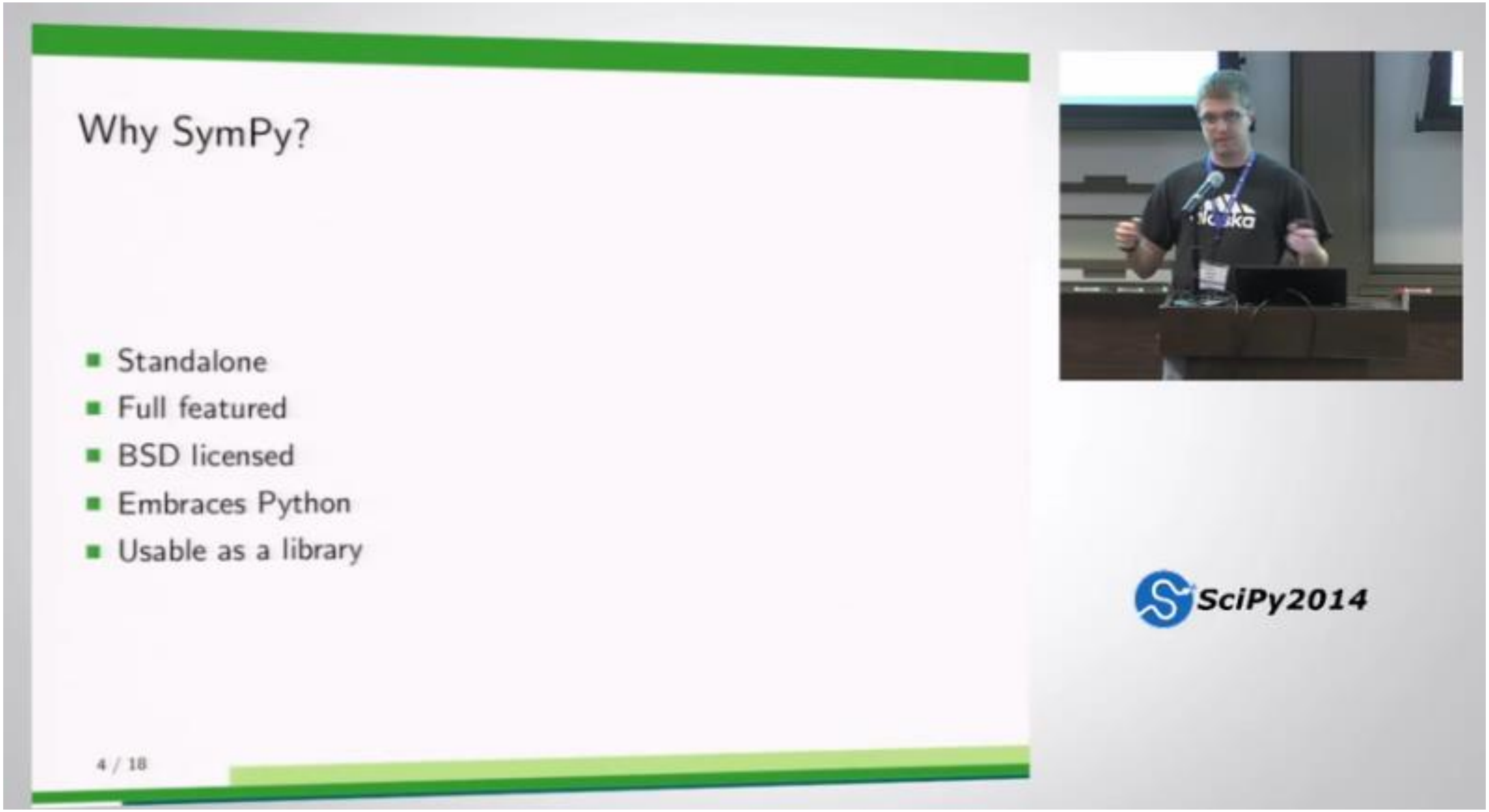
Objectives

- Perform algebraic and calculus computation with symbolic expressions

The Basics

- SymPy is a Python library for symbolic mathematics
- It is an alternative to Mathematica

The Basics



Why SymPy?
Standalone
Full featured
BSD licensed
Embraces Python
Usable as a library
4 / 18
SciPy2014

First Steps with SymPy

```
>>> from sympy import *
>>> init_session()
Python console for SymPy 0.7.6 (Python 3.4.3-
64-bit) (ground types: python)
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)
>>> init_printing()
Documentation can be found at
http://www.sympy.org
```

First Steps with SymPy

- The Rational class

```
>>> from sympy import *
>>> init_session()
>>> a = Rational(1, 3)
>>> a
1/3
>>> a * 2
2/3
>>> type(a)
<class 'sympy.core.numbers.Rational'>
>>> b = 1/3
>>> b
0.3333333333333333
>>> type(b)
<class 'float'>
```

First Steps with SymPy

- Some special constants: `E, pi, oo` (Infinity)

```
>>> pi ** 2
  2
pi
>>> pi.evalf(20) #the first 20 digits of π
3.1415926535897932385
>>> N(pi)  #15 digits by default
3.14159265358979
>>> N(pi,7)
3.141593
```

First Steps with SymPy

- Some special constants: `E, pi, oo` (Infinity)

```
>>> E.evalf(20)
2.7182818284590452354
>>> (pi + E).evalf()
5.85987448204884
>>> oo > 99999
True
>>> oo + 1
oo
```

First Steps with SymPy

- Declare symbolic variables

```
>>> from sympy import *
>>> x = Symbol('x')          #Captal 'S'
>>> y = Symbol('y')
>>> x, y = symbols('x y')  #lowercase 's'
>>> x + y + x - y
2*x
>>> (x + y)**2
       2
(x + y)
```


BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK

First Steps with SymPy

```
>>> from sympy import *
>>> x, y = symbols('y x')
>>> x
???
>>> y**2
???
```


Binghamton University
State University of New York

First Steps with SymPy

```
>>> from sympy import *
>>> x, y = symbols('var1 var2')
>>> x
???
>>> y**2
???
```

First Steps with SymPy

- Printing Mode

```
>>> from sympy import *
>>> x, y = symbols('x y')
>>> f = (x + y)**2
>>> init_printing(pretty_print=False)
>>> f
(x + y)**2
>>> init_printing(pretty_print=True)
>>> f
       2
(x + y)
```

Algebraic Manipulations

- Expansion and simplification

```
>>> expand((x + y)**3)
 3      2           2    3
x  + 3*x *y + 3*x*y  + y

>>> expand(cos(x + y), trig=True)
-sin(x)*sin(y) + cos(x)*cos(y)

>>> simplify((x + x*y) / x)
y + 1
```

Calculus

- Sum

```
>>> expr = Sum(1/(x**2 + 2*x), (x, 1, 10))
>>> expr
  10
____
\   `
 \         1
  \   --------
  /    2
 /    x  + 2*x
/___,
x = 1
>>> expr.doit()
175
---
264
```

$$\boxed{\frac{1}{1^2+2\cdot 1}+\frac{1}{2^2+2\cdot 2}+\ldots+\frac{1}{10^2+2\cdot 10}}$$

Calculus

- Product

```
>>> expr = Product(1/(x**2 + 2*x), (x, 1, 10))
>>> expr
     10
__________
|        |     1
|        | --------
|        |  2
|        | x  + 2*x
|        |
  x = 1
>>> expr.doit()
1/869100503040000
```

$$\boxed{\frac{1}{1^2+2\cdot 1} \times \frac{1}{2^2+2\cdot 2} \times \ldots \times \frac{1}{10^2+2\cdot 10}}$$

Calculus

- Limits
- $\lim_{x \to x0} f(x)$ is `limit(f(x), x, x0)`

```
>>> limit(sin(x)/x, x, 0)
1
>>> limit(x, x, oo)
oo
>>> limit(1/x, x, oo)
0
>>> limit(x**x, x, 0)
1
```

Calculus

- The limit from a certain direction

```
>>> limit(1/x, x, 0, dir='+')
oo
>>> limit(1/x, x, 0, dir='-')
-oo
```

$$\lim_{x\to 0^+} \frac{1}{x} = \infty.$$
$$\lim_{x\to 0^-} \frac{1}{x} = -\infty.$$

Calculus

- Differentiation (derivatives)
  - `diff(f(x), x)`

$$f'(x),\ \frac{d}{dx}f(x),\ \frac{df}{dx}$$

```
>>> diff(sin(x), x)
cos(x)

>>> diff(sin(2*x), x)
2*cos(2*x)
```

Calculus

- Higher derivatives
- `diff(f(x), x, n)`

```
>>> diff(sin(2*x), x, 1)
2*cos(2*x)

>>> diff(sin(2*x), x, 2)
-4*sin(2*x)

>>> diff(sin(2*x), x, 3)
-8*cos(2*x)
```

Calculus

- Indefinite Integrals

$$\int_{-\infty}^{\infty} f(x)dx$$

```
>>> integrate(6 * x**5, x)
 6
x
>>> integrate(sin(x), x)
-cos(x)
>>> integrate(log(x), x)
x*log(x) - x
>>> integrate(2*x + sinh(x), x)
 2
x + cosh(x)
```

Calculus

- Definite Integrals

$$\int_a^b f(x)dx$$

```
>>> integrate(x**3, (x, -1, 1))
0
>>> integrate(sin(x), (x, 0, pi/2))
1
>>> integrate(cos(x), (x, -pi/2, pi/2))
2
>>> integrate(exp(-x), (x, 0, oo))
1
>>> integrate(exp(-x**2), (x, -oo, oo))
  ____
\/ pi
```

Equation Solving

```
>>> solve(x**2-3*x+2, x)
[1, 2]

>>> solve([x+5*y-2, -3*x+6*y-15], [x, y])
{x: -3, y: 1}

>>> solve(exp(x) + 1, x)
[I*pi]
```

Equation Solving

- `Factor` returns the polynomial factorized into irreducible terms

```
>>> f = x**4 - 3*x**2 + 1
>>> f
 4      2
x  - 3*x  + 1
>>> factor(f)
/ 2        \ / 2        \
\x  - x - 1/*\x  + x - 1/
```

Equation Solving

- Solving (some) Ordinary Differential Equations

```
>>> f = symbols('f', cls=Function)
>>> f(x)
f(x)
>>> f(x).diff(x)
d
--(f(x))
dx
>>> f(x).diff(x, x)
  2
 d
---(f(x))
  2
dx
```

Equation Solving

```
>>> f(x).diff(x, x) + f(x)
         2
        d
f(x) + ---(f(x))
         2
       dx
>>> dsolve(f(x).diff(x, x) + f(x), f(x))
f(x) = C1*sin(x) + C2*cos(x)
```

Substitution

```
>>> expr = x**2 + 2*x + 1
>>> expr
 2
x  + 2*x + 1
>>> expr.subs(x, 2)
9
>>> expr.subs(x, y+1)
               2
2*y + (y + 1)  + 3
```

Substitution

```
>>> a = factorial(n)
>>> a
n!
>>> a.subs(n, 3)
6
>>> b = binomial(n, k)
>>> b
/n\
| |
\k/
>>> b.subs([(n,3), (k,2)])
3
```

Substitution

```
>>> x, y, z = symbols('x y z')
>>> expr = x**3 + 4*x*y - z
>>> expr.subs([(x, 2), (y, 4), (z, 0)])
40
```

Converting Strings to SymPy Expressions

```
>>> str_expr = "x**2 + 3*x - 1/2"
>>> str_expr
x**2 + 3*x - 1/2
>>> str_expr.subs(x, 2)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
AttributeError: 'str' object has no attribute 'subs'
```

Converting Strings to SymPy Expressions

- The `sympify` function (not to be confused with `simplify`)

```
>>> str_expr = "x**2 + 3*x - 1/2"
>>> expr = sympify(str_expr)
>>> expr = S(str_expr)          #equivalent
>>> expr
 2          1
x  + 3*x - -
            2
>>> expr.subs(x, 2)
19/2
```

SymPy Statistics Module

Expression	Meaning
`P`(condition)	Probability
`E`(expr)	Expected value
`variance`(expr)	Variance
`density`(expr)	Probability Density Function
`sample`(expr)	Produce a realization
`Die`(expr)	Create a die
`Coin`(expr)	Create a coin
`Normal`(expr)	Normal random variable
...	...

SymPy Statistics Module

```
>>> from sympy.stats import *
>>> X, Y = Die('X',6), Die('Y',6) #two 6-sided dice
>>> density(X).dict
{1: 1/6, 2: 1/6, 3: 1/6, 4: 1/6, 5: 1/6, 6: 1/6}
>>> sample(X)          # rolling die X
3
>>> sample(Y)          # rolling die Y
5
```

SymPy Statistics Module

```
>>> P(X > 3)     #Probability X > 3
1/2
>>> E(X+Y)       #Expectation of the sum of two dice
7
>>> variance(X+Y) #Variance of the sum of two dice
35/6
```

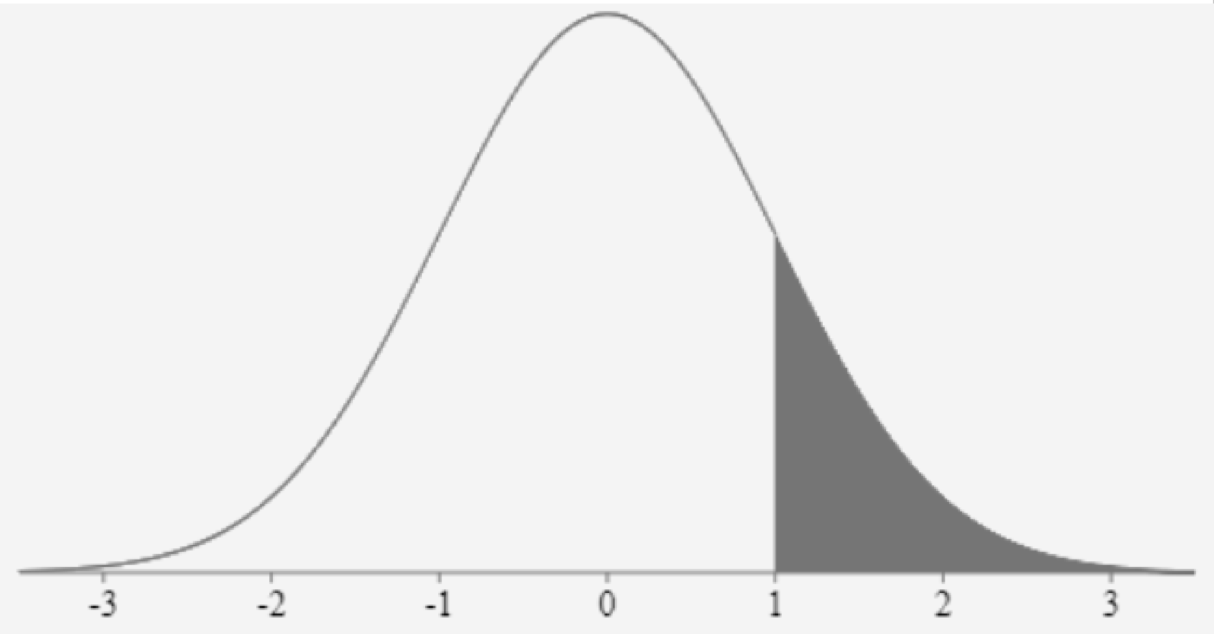
Fun with Dice

SymPy Statistics Module

```
>>> from sympy.stats import *
>>> W = Coin('W') # a coin
>>> density(W).dict
{H: 1/2, T: 1/2}
```

SymPy Statistics Module

```
>>> from sympy.stats import *
>>> Z = Normal('Z', 0, 1) #Normal random var
>>> P(Z>1).evalf()      # Probability of Z > 1
0.158655253931457
```



SymPy Statistics Module

```
>>> from sympy.stats import *
>>> V = Bernoulli('V', 0.9) #Bernoulli rv
>>> density(V).dict
{0: 0.1, 1: 0.9}
>>> sample(V)
1
>>> sample(V)
1
>>> sample(V)
1
>>> sample(V)
0
```