Mastering Python

ألدرس #1_10

الرياضيات بالرموز والتفاضل والتكاملSympy

By:

Hussam Hourani

V1.0 - NOV 2019

Agenda

- What is SymPy?
- SymPy Sample Functions
- Plotting using Sympy
- 3D Plotting using Sympy

What is SymPy?

SymPy Library allows to perform all types of computations symbolically.

It provides the calculus functionality (limits, differentiation,…) Discrete mathematics and polynomial operations .

Solving various types of equations, The functionality of matrix representations and operations and Geometric functions

http://www.sympy.org/

SymPy modules

- **Assumptions**: The assumption engine
- **Concrete**: Symbolic products and summations
- **Core basic class structure**: Basic, Add, Mul, Pow, and so on
- **Functions**: Elementary and special functions
- **Galgebra**: Geometric algebra
- **Geometry**: Geometric entities
- **Integrals**: Symbolic integrator
- **Interactive**: Interactive sessions (for example, IPython)
- **Logic**: Boolean algebra and theorem proving
- **Matrices**: Linear algebra and matrices
- **mpmath**: Fast arbitrary precision numerical math
- **ntheory**: Number theoretical functions
- **Parsing**: Mathematica and maxima parsers
- **Physics**: Physical units and quantum stuff
- **Plotting**: 2D and 3D plots using Pyglet
- **Polys**: Polynomial algebra and factorization
- **Printing**: Pretty-printing and code generation
- **Series**: Symbolic limits and truncated series
- **Simplify**: Rewriting expressions in other forms
- **Solvers**: Algebraic, recurrence, and differential
- **Statistics**: Standard probability distributions
- **Utilities**: Test frameworks and compatibility-related content

SymPy Symbols

```
from sympy import *
x = symbols('x')

>>> x + 1
```

Output

```
x + 1
```

```
import sympy as sym
a, b = sym.symbols('b a')

>>> a + b + 1
```

Output

```
a + b + 1
```

```
import sympy as sym
x = sym.symbols('x')
expr = x + 1
print ( expr.subs(x, 2) )
```

Output

```
3
```

```
import sympy as sym
x = sym.symbols('x')
expr = x + x**2 + 1
print ( expr.subs(x, 2) )
```

Output

```
7
```

SymPy Equations

```
import sympy as sym
x, y, z = sym.symbols('x y z')
expr = x**3 + 4*x*y - z
print( expr.subs([(x, 2), (y, 4), (z, 0)]) )
```

Output

```
40
```

```
from sympy import *
x = symbols('x')
str_expr = "x**2 + 3*x - 1/2"
expr = sympify(str_expr)

print ( expr)
print ( expr.subs(x, 2))
```

Output

```
x**2 + 3*x - 1/2

19/2
```

```
from sympy import *
x = symbols('x')
init_printing()
expr = Integral(sqrt(1/x),x)
print (expr)
```

Output

```
Integral(sqrt(1/x), x)

In [8]: expr
Out[8]:

∫√(1/x) dx
```

SymPy Equations

```
from sympy import *
x = symbols('x')
init_printing()
expr = (4*x**3 + 21*x**2 + 10*x + 12)/(x**4 + 5*x**3 + 5*x**2 + 4*x)
print (expr)
```

Output

```
(4*x**3 + 21*x**2 + 10*x + 12)/(x**4 + 5*x**3 + 5*x**2 + 4*x)

In [14]: expr
Out[14]:
```

$$\frac{4x^3 + 21x^2 + 10x + 12}{x^4 + 5x^3 + 5x^2 + 4x}$$

SymPy Sample Functions

```
import sympy as sym
x = sym.Symbol('x')
y, i ,n, a, b = sym.symbols('y i n a b')
```

Use this code to run all the Code in this slide and the next 2 slides

```
f = x**2 + 1
print( f.subs(x, 2))
```

Output

```
5
```

```
print(  sym.expand( (x + y) ** 3 )   )
```

Output

```
x**3 + 3*x**2*y + 3*x*y**2 + y**3
```

```
print( sym.simplify((x + x * y) / x))
```

Output

```
y + 1
```

```
print(  sym.limit(sym.sin(x) / x, x, 0))
```

Output

```
1
```

$\lim_{x \to 0} \frac{\sin(x)}{x}$

```
print( sym.diff(x**4, x))
```

Output

```
4*x**3
```

```
print( sym.integrate(6 * x ** 5, x))
```

Output

```
x**6
```

```
print( sym.solveset(x - 1, x))
```

Output

```
{1}
```

```
solution = sym.solve((x + 5 * y - 2, -3 * x + 6
* y - 15), (x, y))
print(solution[x], solution[y])
```

Output

```
-3 1
```

SymPy Sample Functions

```
print(sym.integrate(exp(x)*sin(x) +
exp(x)*cos(x), x))
```

Output

```
exp(x)*sin(x)
```

Compute $\int (e^x \sin(x) + e^x \cos(x))\, dx$.

```
print(sym.summation(1/2**i, (i, 0, sym.oo)))
print(sym.summation(1/log(n)**n,(n,2, sym.oo)))
```

Output

```
2
Sum(log(n)**(-n), (n, 2, oo)
```

```
print( sym.limit(1 / x, x, sym.oo))
```

Output

```
0
```

```
print(sym.summation(2*i - 1, (i, 1, n)))
```

Output

```
n**2
```

```
a = (x + 1)**2 + 5*x
b = x**2 + 2*x + 4
print(sym.simplify(a - b))
```

Output

```
5*x - 3
```

```
func = x**2 + 2*x*y - 2* z
print( func.subs([(x, 1), (y, 6), (z, -1)]) )
```

Output

```
15
```

```
str_func = "2*x +1"
func = sympify(str_func)
print(func)
```

Output

```
2*x + 1
```

```
func = sin(x)
print( func.evalf(subs={x: 1.2}) )
```

Output

```
0.932039085967226
```

SymPy Sample Functions

```
print( factor(x**3 - x**2 + x - 1) )
print( factor(x**2*z + 4*x*y*z + 4*y**2*z) )
```

Output

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

```
print ( sym.integrate(exp(-x**2 - y**2), (x, -sym.oo, sym.oo), (y, -sym.oo, sym.oo)))
```

Output

```
pi
```

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-x^2-y^2}\,dx\,dy,$$

```
print (sym.solveset(x-2, x) )
```

Output

```
{2}
```

```
print (sym.solveset(Eq(x-2, 1), x) )
```

Output

```
{3}
```

```
m1 = sym.Matrix([[1, 2, 3], [3, 2, 1]])
m2 = sym.Matrix([0, 1, 1])
m3=  sym.Matrix([2, 3, 0])
print(m1)
print( m1*m2 )
print(m2+m3)
```

Output

```
Matrix([[1, 2, 3], [3, 2, 1]])
Matrix([[5], [3]])
Matrix([[2], [4], [1]])
```

```
m4= sym.zeros(2, 3)
m5= sym.eye(3)
print(m4)
print(m5)
print(sym.ones(3, 2))
print(sym.diag(1,2,3))
```

Output

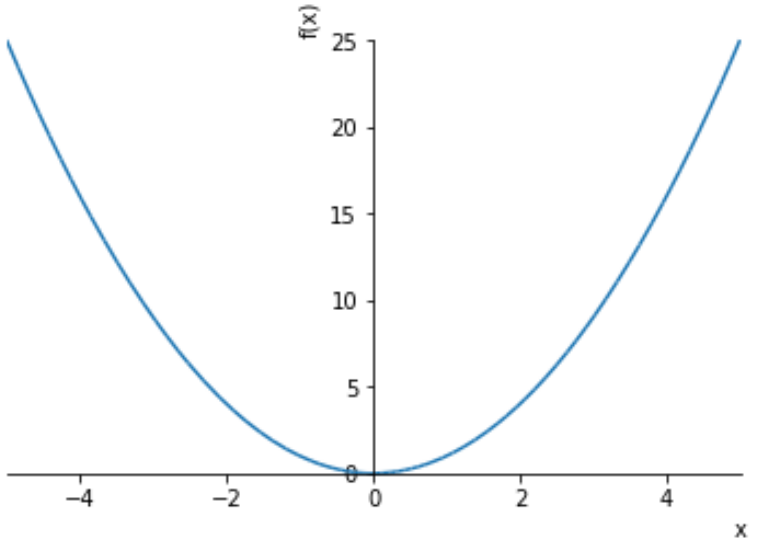
```
Matrix([[0, 0, 0], [0, 0, 0]])
Matrix([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
Matrix([[1, 1], [1, 1], [1, 1]])
Matrix([[1, 0, 0], [0, 2, 0], [0, 0, 3]])
```

Plotting using Sympy

```
from sympy import symbols
from sympy.plotting import plot
x = symbols('x')
```

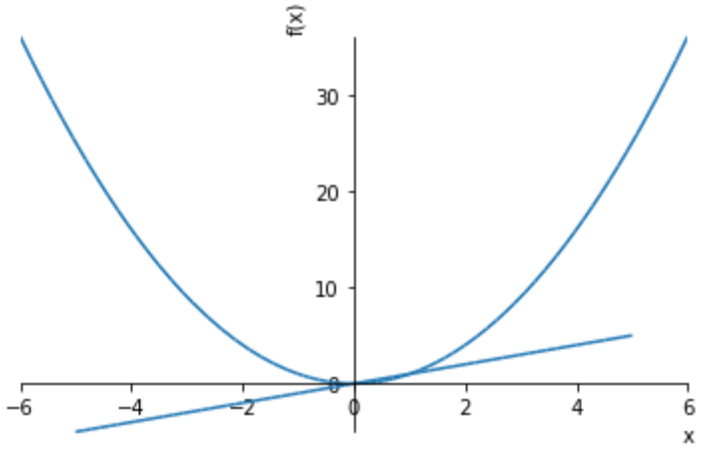
```
plot(x**2, (x, -5, 5))
```

Output




```
plot((x**2, (x, -6, 6)), (x, (x, -5, 5)))
```

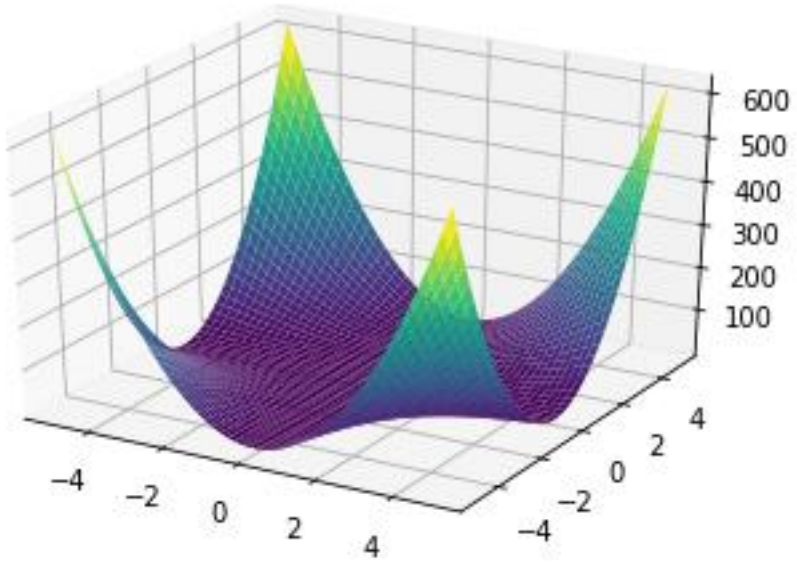
Output



3D Plotting using Sympy

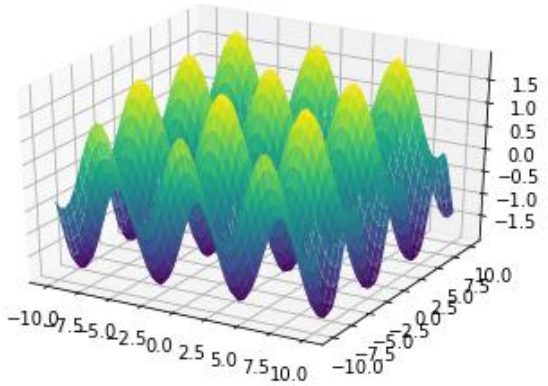
```
from sympy import symbols
from sympy.plotting import plot3d
x, y = symbols('x y')
f=x**2*y**2
plot3d(f, (x, -5, 5), (y, -5, 5))
```

Output




```
from sympy import symbols
from sympy.plotting import plot3d
x, y = symbols('x y')
f=cos(x)+sin(y)
plot3d(f, (x, -10, 10), (y, -10, 10))
```

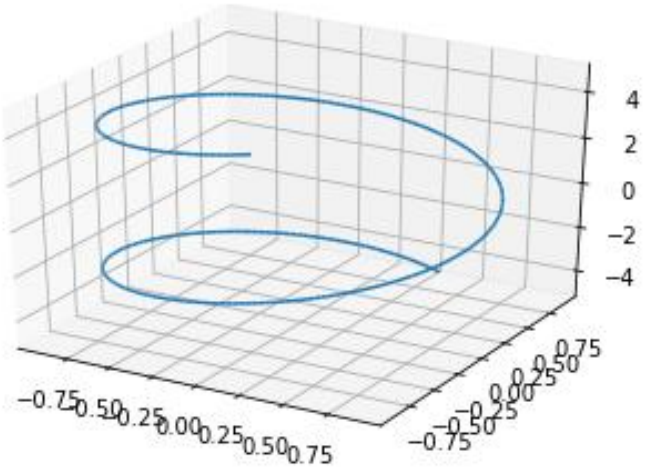
Output



3D Plotting using Sympy

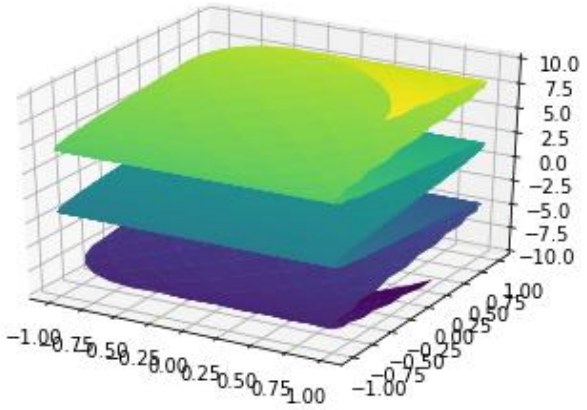
```
from sympy import symbols, cos, sin
from sympy.plotting import plot3d_parametric_line
u, v = symbols('u v')
plot3d_parametric_line(cos(u), sin(u), u, (u, -5, 5))
```

Output



Output

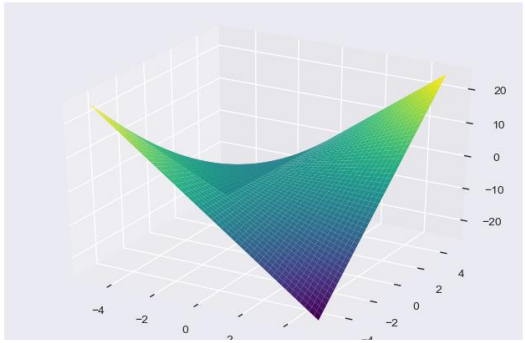
```
from sympy import symbols, cos, sin
from sympy.plotting import plot3d_parametric_surface
u, v = symbols('u v')
plot3d_parametric_surface(cos(u + v), sin(u - v), u - v, (u, -5, 5), (v, -5, 5))
```

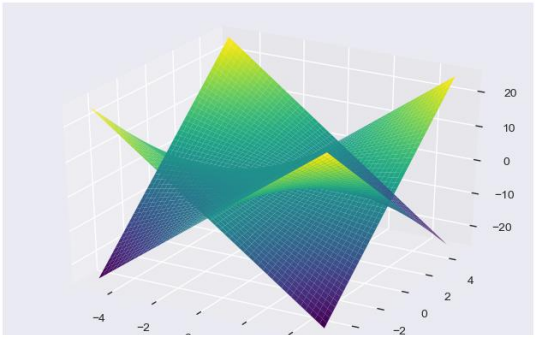


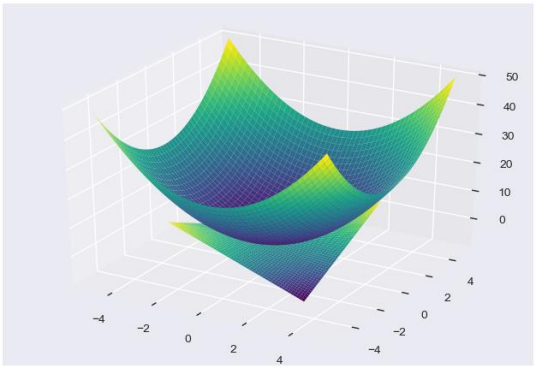
3D Plotting using Sympy

```
from sympy import symbols
from sympy.plotting import plot3d

x, y = symbols('x y')

plot3d(x*y, (x, -5, 5), (y, -5, 5))

plot3d(x*y, -x*y, (x, -5, 5), (y, -5, 5))

plot3d((x**2 + y**2, (x, -5, 5), (y, -5, 5)), (x*y, (x, -3, 3), (y, -3, 3)))
```

Output







Master in Software Engineering

Hussam Hourani has over 25 years of Organizations Transformation, VROs, PMO, Large Scale and Enterprise Programs Global Delivery, Leadership, Business Development and Management Consulting. His client experience is wide ranging across many sectors but focuses on Performance Enhancement, Transformation, Enterprise Program Management, Artificial Intelligence and Data Science.