00_symbols

December 30, 2019

1 Creating Symbols

[1]:
```
from sympy import *
```

[2]:
```
x, y, z = symbols('x, y, z')
```

[3]:
```
print(x)
print(type(x))
```

```
x
<class 'sympy.core.symbol.Symbol'>
```

[4]:
```
z = x + y
print(z)
print(type(z))
```

```
x + y
<class 'sympy.core.add.Add'>
```

01_equivalence

December 30, 2019

1 Different Types of Equivalence

[1]:
```
from sympy import *
x, y, z = symbols('x, y, z')
```

Assignment operator is a syntax error

[2]:
```
x + 1 = 4
```

```
  File "<ipython-input-2-612a9eeaa1e4>", line 1
    x + 1 = 4
          ^
SyntaxError: can't assign to operator
```

Python ==

[3]:
```
x == x, x == y, x + 1 == 4
```

[3]:
```
(True, False, False)
```

SymPy Equivalence

[4]:
```
Eq(x+1, 4)
```

[4]:
$$x + 1 = 4$$

02_basic_ops

December 30, 2019

1 Basic Operations

[1]:
```
import numpy as np
from sympy import *
x, y, z = symbols('x, y, z')
```

[2]:
```
c = cos(x) + 1
c
```

[2]: $\cos(x) + 1$

[3]:
```
a = c.subs(x, 5)
a
```

[3]: $\cos(5) + 1$

[4]:
```
a.evalf()
```

[4]: 1.28366218546323

[5]:
```
a.evalf(30)
```

[5]: 1.28366218546322626446663917151

[6]:
```
expr = x**2 + 2
expr
```

[6]: $x^2 + 2$

[7]:
```
func = lambdify(x, expr, "numpy")
func
```

[7]:
```
<function _lambdifygenerated(x)>
```

[8]:
```
func(5)
```

[8]: 27

[9]:
```
arr = np.arange(25)
arr
```

[9]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

[10]:
```
func(arr)
```

[10]:
```
array([  2,   3,   6,  11,  18,  27,  38,  51,  66,  83, 102, 123, 146,
       171, 198, 227, 258, 291, 326, 363, 402, 443, 486, 531, 578])
```

03_simplification

December 30, 2019

1 Simplification

[1]:
```
import numpy as np
from sympy import *
x, y, z = symbols('x, y, z')
```

[2]:
```
expr = sin(x)/cos(x)
expr
```

[2]: $\frac{\sin(x)}{\cos(x)}$

[3]:
```
simplify(expr)
```

[3]: $\tan(x)$

[4]:
```
expr = x**2 + 2*x + 1
expr
```

[4]: $x^2 + 2x + 1$

[5]:
```
expr = factor(expr)
expr
```

[5]: $(x + 1)^2$

[6]:
```
expr = expand(expr)
expr
```

[6]: $x^2 + 2x + 1$

[7]:
```
expr = x*y + x - 3*z + 2*x**2 - z*x**2 + x**3 - 3*y
expr
```

[7]: $x^3 - x^2 z + 2x^2 + xy + x - 3y - 3z$

[8]:
```
exprx = collect(expr, x)
exprx
```

[8]:

$$x^3 + x^2\left(2 - z\right) + x\left(y + 1\right) - 3y - 3z$$

[9]:
```
expry = collect(expr, y)
expry
```

[9]:
$$x^3 - x^2 z + 2x^2 + x + y\left(x - 3\right) - 3z$$

[10]:
```
exprz = collect(expr, z)
exprz
```

[10]:
$$x^3 + 2x^2 + xy + x - 3y + z\left(-x^2 - 3\right)$$

[11]:
```
expr = (x**2 + 2*x + 1)/(x**2 + x)
expr
```

[11]:
$$\frac{x^2 + 2x + 1}{x^2 + x}$$

[12]:
```
expr = cancel(expr)
expr
```

[12]:
$$\frac{x + 1}{x}$$

04_calculus

December 30, 2019

1 Calculus

[1]:
```
import numpy as np
from sympy import *
x, y, z = symbols('x, y, z')
```

1.1 Derivatives

[2]:
```
expr = x**2
expr
```

[2]: x^2

[3]:
```
diff(expr, x)
```

[3]: $2x$

[4]:
```
expr = cos(x)
expr
```

[4]: $\cos\left(x\right)$

[5]:
```
diff(expr, x)
```

[5]: $-\sin\left(x\right)$

[6]:
```
expr = exp(x*y*z)
expr
```

[6]: e^{xyz}

[7]:
```
diff(expr, x)
```

[7]: yze^{xyz}

[8]:
```
diff(expr, x, x)
```

[8]: $y^2z^2e^{xyz}$

[9]:
```
diff(expr, x, x, y)
```

[9]: $yz^2\left(xyz+2\right)e^{xyz}$

[10]:
```
expr = Derivative(expr, x, x)
expr
```

[10]: $\frac{\partial^2}{\partial x^2}e^{xyz}$

[11]:
```
expr.doit()
```

[11]: $y^2z^2e^{xyz}$

1.2 Integrals

[12]:
```
integrate(cos(x), x)
```

[12]: $\sin\left(x\right)$

[13]:
```
integrate(exp(-x), (x,0,oo))
```

[13]: 1

[14]:
```
integrate(exp(-x**2 - y**2), (x, -oo, oo), (y, -oo, oo))
```

[14]: π

[15]:
```
integrate(x*exp(x**2), x)
```

[15]: $\frac{e^{x^2}}{2}$

[16]:
```
expr = Integral(exp(-x**2 - y**2), (x, -oo, oo), (y, -oo, oo))
expr
```

[16]: $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}e^{-x^2-y^2}\,dx\,dy$

[17]:
```
expr.doit()
```

[17]: π

1.3 Limits

[18]:
```
expr = x**2/exp(x)
expr
```

[18]: $x^2 e^{-x}$

[19]:
```
expr.subs(x, oo)
```

[19]:
```
NaN
```

[20]:
```
limit(expr, x, oo)
```

[20]: 0

[21]:
```
expr = 1/x
expr
```

[21]: $\frac{1}{x}$

[22]:
```
limit(expr, x, 0, '+')
```

[22]: ∞

[23]:
```
limit(expr, x, 0, '-')
```

[23]: $-\infty$

1.4 Series Expansion

[24]:
```
expr = exp(x)
expr
```

[24]: e^x

[25]:
```
expr.series(x, 0, 4)
```

[25]: $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + O\left(x^4\right)$

[26]:
```
expr.series(x, 0, 4).removeO()
```

[26]: $\frac{x^3}{6} + \frac{x^2}{2} + x + 1$

[27]:
```
expr.series(x, 0, 7)
```

[27]: $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + O\left(x^7\right)$

[28]:
```
expr.series(x, 1, 9)
```

[28]:
$$e+e\left(x-1\right)+\frac{e\left(x-1\right)^{2}}{2}+\frac{e\left(x-1\right)^{3}}{6}+\frac{e\left(x-1\right)^{4}}{24}+\frac{e\left(x-1\right)^{5}}{120}+\frac{e\left(x-1\right)^{6}}{720}+\frac{e\left(x-1\right)^{7}}{5040}+\frac{e\left(x-1\right)^{8}}{40320}+O\left(\left(x-1\right)^{9};x\to 1\right)$$

1.5 Finite Differences

[29]:
```
f, g = symbols('f, g', cls=Function)
```

[30]:
```
differentiate_finite(f(x)*g(x))
```

[30]:
$$-f\left(x-\frac{1}{2}\right)g\left(x-\frac{1}{2}\right)+f\left(x+\frac{1}{2}\right)g\left(x+\frac{1}{2}\right)$$

[31]:
```
differentiate_finite(f(x)/g(x))
```

[31]:
$$-\frac{f\left(x-\frac{1}{2}\right)}{g\left(x-\frac{1}{2}\right)}+\frac{f\left(x+\frac{1}{2}\right)}{g\left(x+\frac{1}{2}\right)}$$

05_solvers

December 30, 2019

1 Algebraic and Differential Equation Solvers

[1]:
```
import numpy as np
from sympy import *
x, y, z = symbols('x, y, z')
f, g = symbols('f, g', cls=Function)
```

[2]:
```
expr = Eq(x**2 + 2*x +1, 0)
expr
```

[2]: $x^2 + 2x + 1 = 0$

[3]:
```
solveset(expr)
```

[3]: $\{-1\}$

[4]:
```
solveset(x - x, x, domain=S.Complexes)
```

[4]: $\mathbb{C}$

[5]:
```
solveset(x - x, x, domain=S.Reals)
```

[5]: $\mathbb{R}$

[6]:
```
solveset(x - x, x, domain=S.Rationals)
```

[6]: $\mathbb{Q}$

[7]:
```
solveset(x - x, x, domain=S.Integers)
```

[7]: $\mathbb{Z}$

[8]:
```
solveset(x - x, x, domain=S.Naturals)
```

[8]: $\mathbb{N}$

[9]:
```
solveset(y - x, x, domain=S.Naturals)
```

[9]: $\mathbb{N} \cap \{y\}$

[10]:
```
solveset(z**y - x, x, domain=S.Rationals)
```

[10]:
$$\mathbb{Q} \cap \{z^{y}\}$$

[11]:
```
f(x)
```

[11]:
$$f(x)$$

[12]:
```
f(x).diff(x)
```

[12]:
$$\frac{d}{dx} f(x)$$

[13]:
```
diffeq = Eq(f(x).diff(x,x) - 2*f(x).diff(x) + f(x),sin(x))
diffeq
```

[13]:
$$f(x) - 2\frac{d}{dx} f(x) + \frac{d^{2}}{dx^{2}} f(x) = \sin\left(x\right)$$

[14]:
```
dsolve(diffeq, f(x))
```

[14]:
$$f(x) = \left(C_{1} + C_{2} x\right) e^{x} + \frac{\cos\left(x\right)}{2}$$

06_matrices

December 30, 2019

1 Matrices

[1]:
```
import numpy as np
from sympy import *
x, y, z = symbols('x, y, z')
f, g = symbols('f, g', cls=Function)
```

[3]:
```
m = Matrix(np.arange(25).reshape(5,5))
m
```

[3]:
$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 & 24 \end{bmatrix}$$

[4]:
```
m.shape
```

[4]:
```
(5, 5)
```

[6]:
```
n = Matrix(np.eye(5).astype(int))
n
```

[6]:
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

[12]:
```
m.T
```

[12]:
$$\begin{bmatrix} 0 & 5 & 10 & 15 & 20 \\ 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \end{bmatrix}$$

[14]:
```
m.T * (n*2)
```

[14]:
$$\begin{bmatrix} 0 & 10 & 20 & 30 & 40 \\ 2 & 12 & 22 & 32 & 42 \\ 4 & 14 & 24 & 34 & 44 \\ 6 & 16 & 26 & 36 & 46 \\ 8 & 18 & 28 & 38 & 48 \end{bmatrix}$$

[15]:
```
diag(1,2,5)
```

[15]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

[16]:
```
n.det()
```

[16]:
$$1$$

[20]:
```
M = Matrix([[3, -2,  4, -2], [5,  3, -3, -2], [5, -2,  2, -2], [5, -2, -3,  3]])
```

[21]:
```
M.rref()
```

[21]:
```
(Matrix([
 [1, 0, 0, 0],
 [0, 1, 0, 0],
 [0, 0, 1, 0],
 [0, 0, 0, 1]]), (0, 1, 2, 3))
```

[22]:
```
M.eigenvals()
```

[22]:
```
{3: 1, -2: 1, 5: 2}
```

[23]:
```
M.eigenvects()
```

[23]:
```
[(-2, 1, [Matrix([
   [0],
   [1],
   [1],
   [1]])]), (3, 1, [Matrix([
   [1],
   [1],
   [1],
   [1]])]), (5, 2, [Matrix([
   [1],
   [1],
   [1],
   [0]]), Matrix([
   [ 0],
   [-1],
   [ 0],
```

```
        [ 1]])])]
```

[24]:
```
P, D = M.diagonalize()
```

[25]:
```
P
```

[25]:
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

[26]:
```
D
```

[26]:
$$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$