Sympy_Cheet_Sheet

January 19, 2020

0.0.1 Import

```
from sympy import init_printing
# initializing LaTex printing
init_printing()
```

0.0.2 Creating (one or more than one) symbol

```
from sympy import symbols
x, y = symbols('x y')
z = symbols('z')
# also we can specify the type of values that a mathematical symbol can hold
x, y = symbols('x y', positive = True)
a, b = symbols('a b', real = True)
#  this function recognizes special characters such as alpha
```

0.0.3 Math

```
from sympy import Rational, sqrt, log, exp, sin, pi, I
# Rational: calculates an exact solution for fraction
# evalf() method still allows us to get a numerical approximation
(Rational(5, 3)).evalf(3)

# square root
sqrt(8)

# euler number
exp(1).evalf(40)

# others
sin(pi / 6)
log(exp(1))

# substitutions
<expression>.subs(x, 3)
<expression>.subs([(x, 2), (y, 5)])
<expression>.evalf(subs = {x:2, y:5})
```

```
# lambdify()
from numpy import arange
from sympy import lambdify
my_domain = arange(-3, 4, 1)
f = lambdify(x, <expression>, 'numpy')
f(my_domain)
```

0.0.4 Equation

```
# Simplify a expression ()
from sympy import simplify
simplify(<expression>)

# Factoring ()
(<expression>).factor()
#
(<expression>).expand()
# or
from sympy import factor, expand
factor(<expression>)
expand(<expression>)

# Collect: the results is an expression with descending powers of
from sympy import collect
collect(<expression>, x)
# for example
collect(x * y - 2 + 2 * x**2 - z * x**2 + x**3, x)
# we get
x**3+**2*(z+2)+x*y2

# Cancel: puts a rational function in canonical form
from sympy import cancel
cancel(<expression 1> / <expression 2>)
# the difference from
(<expression 1> / <expression 2>).factor
# is the later one also factoring the polynomials

# Apart: It calculates partial fraction decomposition ()
from sympy import apart
apart(<expression>)
```

0.0.5 Trigonometric

```
# Trigonometric simplification
from sympy import trigsimp, sin, cos
theta = symbols('theta')
trigsimp((sin(theta))**2 + (cos(theta))**2)
```