SymPy

Es un módulo que permite trabajar con matemática simbólica en Python. Lo que sigue se ha tomado esencialmente de la sitguiente URL:

    http://scipy-lectures.org/packages/sympy.html

Primeros pasos

In [1]:

```
import sympy as sym

a = sym.Rational(1, 2)

# Simplificaciones algebraicas:
print("a) ", a, a**2)

# Números trascendentes:
print("b) ", sym.pi**2, sym.pi**2 ** sym.exp(1))
print("c) ", (sym.pi**2).evalf(), ((sym.pi**2) ** sym.exp(1)).evalf())
print("d) ", (sym.pi**2).evalf(25), ((sym.pi**2) ** sym.exp(1)).evalf(7))
print("e) ", sym.N(sym.pi**2, 25), sym.N((sym.pi**2) ** sym.exp(1), 7))

# Infinito:

print("f) ", sym.oo > 99999)
print("g) ", sym.oo + 1)
```

```
a)  1/2 1/4
b)  pi**2 pi**(2**E)
c)  9.86960440108936 504.413765418217
d)  9.8696044010893586188344910 504.4138
e)  9.8696044010893586188344910 504.4138
f)  True
g)  oo
```

Una calculadora simbólica

In [2]:

```
# Variables o símbolos y manipulaciones algebraicas:

x = sym.Symbol('x')
y = sym.Symbol('y')

# Simplificaciones algebraicas:
print("a) ", x + y + x - y)

# Desarrollos algebraicos:
print("b) ", (x + y) ** 2)
print("c) ", sym.expand((x + y) ** 2))

# Reorganización de términos:
print("d) ", 3 * x * y ** 2 + 3 * y * x ** 2 + x ** 3 + y ** 3)

# Trigonometría:
print("e) ", sym.expand(sym.cos(x + y), trig=True))

# Más simplificaciones algebraicas:
print("f) ", sym.simplify((x + x * y) / x))
```

```
a)  2*x
b)  (x + y)**2
c)  x**2 + 2*x*y + y**2
d)  x**3 + 3*x**2*y + 3*x*y**2 + y**3
e)  -sin(x)*sin(y) + cos(x)*cos(y)
f)  y + 1
```

Cálculo infinitesimal

In [3]:

```
# Límites:

print("a1) ", sym.limit(x, x, sym.oo))
print("a2) ", sym.limit(1 / x, x, sym.oo))
print("a3) ", sym.limit(x ** x, x, 0))

# Diferenciación:

print("b1) ", sym.diff(sym.sin(x), x))
print("b2) ", sym.diff(sym.sin(2 * x), x))
print("b3) ", sym.diff(sym.tan(x), x))

# Derivadas de orden superior:

print("c1) ", sym.diff(sym.sin(2 * x), x, 1))
print("c2) ", sym.diff(sym.sin(2 * x), x, 2))
print("c3) ", sym.diff(sym.sin(2 * x), x, 3))

# Desarrollo en serie:

print("d1) ", sym.series(sym.cos(x), x))
print("d2) ", sym.series(1/sym.cos(x), x))

# Integración indefinida:

print("e2) ", sym.series(1/sym.cos(x), x))
print("e3) ", sym.integrate(6 * x ** 5, x))
print("e4) ", sym.integrate(sym.sin(x), x))
print("e5) ", sym.integrate(sym.log(x), x))
print("e6) ", sym.integrate(2 * x + sym.sinh(x), x))

# Integración definida:

print("f6) ", sym.integrate(x**3, (x, -1, 1)))
print("f7) ", sym.integrate(sym.sin(x), (x, 0, sym.pi / 2)))
print("f8) ", sym.integrate(sym.cos(x), (x, -sym.pi / 2, sym.pi / 2)))
print("f9) ", sym.integrate(sym.exp(-x), (x, 0, sym.oo)))
print("f7) ", sym.integrate(sym.exp(-x ** 2), (x, -sym.oo, sym.oo)))
```

```
a1)  oo
a2)  0
a3)  1
b1)  cos(x)
b2)  2*cos(2*x)
b3)  tan(x)**2 + 1
c1)  2*cos(2*x)
c2)  -4*sin(2*x)
c3)  -8*cos(2*x)
d1)  1 - x**2/2 + x**4/24 + O(x**6)
d2)  1 + x**2/2 + 5*x**4/24 + O(x**6)
e2)  1 + x**2/2 + 5*x**4/24 + O(x**6)
e3)  x**6
e4)  -cos(x)
e5)  x*log(x) - x
e6)  x**2 + cosh(x)
f6)  0
f7)  1
f8)  2
```

```
f9)  1
f7)  sqrt(pi)
```

Resolución de ecuaciones

In [4]:

```
# Ecuaciones algebraicas:

print("a) ", sym.solveset(x ** 4 - 1, x))

# Ecuaciones algebraicas con números trascendentes:

print("b) ", sym.solveset(sym.exp(x) + 1, x))

# Sistemas de ecuaciones lineales:

solution = sym.solve((x + 5 * y - 2, -3 * x + 6 * y - 15), (x, y))
print("c) ", solution)

# Factorización:
f = x ** 4 - 3 * x ** 2 + 1
print("d) ", sym.factor(f))
print("e) ", sym.factor(f, modulus=5))
print("f) ", sym.satisfiable(x & y))
print("g) ", sym.satisfiable(x & ~x))
```

```
a)  {-1, 1, -I, I}
b)  ImageSet(Lambda(_n, I*(2*_n*pi + pi)), Integers())
c)  {x: -3, y: 1}
d)  (x**2 - x - 1)*(x**2 + x - 1)
e)  (x - 2)**2*(x + 2)**2
f)  {x: True, y: True}
g)  False
```

Álgebra lineal

In [5]:

```
# Matrices:
print("a) ", sym.Matrix([[1, 0], [0, 1]]))

x, y = sym.symbols('x, y')
A = sym.Matrix([[1, x], [y, 1]])
print("b) ", A, A**2)

# Ecuaciones diferenciales:

f, g = sym.symbols('f g', cls=sym.Function)

print("c) ", f(x))
print("d) ", f(x).diff(x, x) + f(x))
print("e) ", sym.dsolve(f(x).diff(x, x) + f(x), f(x)))

print("f) ", sym.dsolve(sym.sin(x) * sym.cos(f(x)) + sym.cos(x) * sym.sin(f(x)) * f(x).diff
```

```
a)  Matrix([[1, 0], [0, 1]])
b)  Matrix([[1, x], [y, 1]]) Matrix([[x*y + 1, 2*x], [2*y, x*y + 1]])
c)  f(x)
d)  f(x) + Derivative(f(x), x, x)
e)  Eq(f(x), C1*sin(x) + C2*cos(x))
f)  [Eq(f(x), -asin(sqrt(C1/(sin(x)**2 - 1) + 1)) + pi), Eq(f(x), asin(sqrt
(C1/(sin(x)**2 - 1) + 1)) + pi), Eq(f(x), -asin(sqrt(C1/(sin(x)**2 - 1) +
1))), Eq(f(x), asin(sqrt(C1/(sin(x)**2 - 1) + 1)))]
```

Referencias

La referencia principal, obligada, es la siguiente:

    https://www.scipy.org/