1. sympy基础

- sympy 的语法与python保持一致, 例如3x不能表达3与x相乘，二是写为3*x
- sympy 是开源免费的，体积小，不需要单独学习一门新的设计语言
- sympy 封装为python的一个库，引用十分方便
    - import sympy as sp
    - from sympy import *
    - from sympy import

symbols

In [1]:

```
from sympy import *
x + 1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-4f19b17923f5> in <module>()
      1 from sympy import *
----> 2 x + 1

NameError: name 'x' is not defined
```

这是因为python中如果不对一个变量进行定义是没有办法使用的。在sympy中也是一样，如果要使用一个符号x，必须要使用函数赋值

In [2]:

```
from sympy import *
x = symbols('x')
x+1
```

Out[2]:

```
x + 1
```

其中symbol()函数用于声明进行运算的符号，参数为string，并且使用空格隔开，比如

In [3]:

```
x, y, z = symbols('x y z')
print(x,y,z)
```

```
x y z
```

此处请注意 python 中的变量x和sympy的符号x是不同的，sympy中的符号可以赋予给python中不同的变量名，例如

In [4]:

```
x, y, z = symbols('x z z')
print(x,y,z)
```

```
x z z
```

基于sympy中的符号，我们可以构造数学中一般的数学表达式，例如

In [5]:

```
x = symbols('x')
expr = x + 1
print(expr)
```

```
x + 1
```

如果想把某个具体的值代入到某个数学表达式，应该如何做呢?

In [6]:

```
x = symbols('x')

expr = x + 1
x = 2

print(expr)
```

```
x + 1
```

在此处expr已经成为数学表达式，而x=2只是赋予变量x为2，并没有改写表达式的内容，因此无法实现带入功能。正确方式应当是使用subs函数进行符号带入：

In [7]:

```
x = symbols('x')
expr = x + 1
a = expr.subs(x,2)
print(a)
```

```
3
```

等于符号

python中的=号属于强内置操作符，因此sympy并没有尝试去改写，而是引入了==作为数学表达式中的等于符号，例如：

In [8]:

```
x = symbols('x')
print(x + 1 == 4)
a = (x+1)**2
b = x**2 + 2*x + 1
print(a == b)
print(a.equals(b))
```

```
False
False
True
```

2. 基础操作

2.1表达式带入（替换）

用于数学表达式中某些变量符号的替换

- 可以使用数字替换符号
- 可以使用另外一个表达式替换符号
- 可以同时对几个符号进行替换

In [9]:

```
from sympy import *
x, y, z = symbols("x y z")
expr = cos(x) + 1
expr.subs(x,0)
```

Out[9]:

```
2
```

In [10]:

```
expr = cos(x) + 1
expr.subs(x,y)
```

Out[10]:

```
cos(y) + 1
```

In [11]:

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

Out[11]:

```
40
```

使用subs()进行符号替换时，返回的是一个新的数学表达式，原符号、原表达式都不会产生变化，例如：

In [12]:

```
expr = cos(x) + 1
print(expr.subs(x,x**y))
print(expr)
print(x)
```

```
cos(x**y) + 1
cos(x) + 1
x
```

其缘由是sympy里的数学表达式expr是不可修改的，无法通过内置函数来修改他们。

2.2将字符串转化为数学表达式

使用sympify()函数可以方便的将字符串转为sympy数学表达式

In [13]:

```
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))
```

```
x**2 + 3*x - 1/2
19/2
```

In [14]:

```
expr=sqrt(8)
print(expr)
```

```
2*sqrt(2)
```

2.3 估计数字表达式的值

使用evalf()将数字表达式转换为浮点值，并且可以通过参数控制精度

In [15]:

```
expr = sqrt(8)
print(expr)
print(expr.evalf())
```

```
2*sqrt(2)
2.82842712474619
```

In [16]:

```
pi.evalf(42)
```

Out[16]:

```
3.14159265358979323846264338327950288419717
```

有时由于计算机精度处理不够导致浮点错误，可以使用chop=True参数来消除误差，例如

In [17]:

```
one = cos(1)**2 + sin(1)**2
print((one - 1).evalf())
print((one - 1).evalf(chop=True))
```

```
-0.e-124
0
```

3. 输出与打印

3.1 输出器/打印器

有下列几种常用的打印器:

- str
- srepr
- ASCII pretty printer
- Unicode pretty printer
- LaTeX
- MathML
- Dot

打印之前通常使用init_printing()函数来初始化打印器，其将自动选择环境下最优的打印器。在不同的交互界面将会产生不同的绘制过程。

In [18]:

```
from sympy import *
init_printing()
```

In [19]:

```
x,y,z=symbols('x y z')
Integral(sqrt(1/x),x)
```

Out[19]:

$$\int \sqrt{\frac{1}{x}}\, dx$$

3.2 输出/打印函数

使用不同的输出函数有不同的输出效果

In [20]:

```
from sympy import *
x, y, z = symbols('x y z')
print(str(Integral(sqrt(1/x), x)))
```

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

In [21]:

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

```
\int \sqrt{\frac{1}{x}}\, dx
```

In [22]:

```
pprint(Integral(sqrt(1/x), x), use_unicode=False)
```

```
  /
 |
 |     ___
 |    / 1
 |   /  -  dx
 | \/   x
 |
/
```

In [23]:

```
pprint(Integral(sqrt(1/x), x), use_unicode=True)
```

```
⌠
⎮     ___
⎮    ╱ 1
⎮   ╱  ─  dx
⎮ ╲╱   x
⌡
```

4. 化简表达式

4.1 通用化简式simplify()

sympy中有几十个用于化简数学表达式的函数，最常用的是simplify()函数，用于尝试将表达式参数化简为最简化的形式：

In [24]:

```
print(simplify(sin(x)**2 + cos(x)**2))
```

```
1
```

In [25]:

```
print(simplify((x**3 + x**2 - x - 1)/(x**2 + 2*x + 1)))
```

```
x - 1
```

In [26]:

```
print(simplify(gamma(x)/gamma(x - 2)))
```

```
(x - 2)*(x - 1)
```

simplify()函数的缺点

- 某些表达式无法转换为直观意义上的最简式
- 由于其属于遍历搜索，因此运行速度较慢

In [27]:

```
print(simplify(x**2 + 2*x + 1))
```

```
x**2 + 2*x + 1
```

上述表达式的最简式应为$(x+1)^2$，然而simplify()判定$x+1$作为底要比x作为底更加复杂。但是纵使由以上缺点，simplify()依然是常用的表达式化简函数。

4.2 多项式/有理式化简

- expand() 是sympy里最常用的表达式展开函数

In [28]:

```
expand((x + 1)**2)
```

Out[28]:

$$x^2 + 2x + 1$$

In [29]:

```
expand((x + 2)*(x - 3))
```

Out[29]:

$$x^2 - x - 6$$

- factor() 常用于多项式的化简，用于解决因式分解。通常和expand()函数相对立。

In [30]:

```
factor(x**3 - x**2 + x - 1)
```

Out[30]:

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

In [31]:

```
factor(x**2*z + 4*x*y*z + 4*y**2*z)
```

Out[31]:

$$z(x + 2y)^2$$

In [32]:

```
print(expand((cos(x) + sin(x))**2))
print(factor(cos(x)**2 + 2*cos(x)*sin(x) + sin(x)**2))
```

```
sin(x)**2 + 2*sin(x)*cos(x) + cos(x)**2
(sin(x) + cos(x))**2
```

- collect()函数用于合并同类项

In [33]:

```
expr = x*y + x - 3 + 2*x**2 - z*x**2 + x**3
collect(expr, x)
```

Out[33]:

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

- cancel()函数用于化简有理分式

In [34]:

```
cancel((x**2 + 2*x + 1)/(x**2 + x))
```

Out[34]:

$$\frac{1}{x}(x + 1)$$

In [ ]:

```

```

In [ ]:

```

```

5. 微积分

- sympy提供了非常多用于积分计算的操作符，使用他们我们可以很方便的计算定积分和不定积分
- 使用之前应当初始化或优化python环境输出

In [35]:

```
from sympy import *
x, y, z = symbols('x y z')
init_printing(use_unicode=True)
```

5.1 函数求导

- 使用diff()函数进行函数求导

In [36]:

```
diff(cos(x), x)
```

Out[36]:

$$-\sin(x)$$

In [37]:

```
diff(exp(x**2), x)
```

Out[37]:

$$2xe^{x^2}$$

当使用diff()求多阶导数时，可以

- 依次指定求导的符号
- 指定求导符号和求导次数

In [38]:

```
diff(x**4, x, x, x)
```

Out[38]:

$$24x$$

In [39]:

```
diff(x**4, x, 3)
```

Out[39]:

$$24x$$

In [40]:

```
(x**4).diff(x,3)
```

Out[40]:

$$24x$$

同样的方法可以拓展到高阶混合偏导，例如：

In [41]:

```
expr = exp(x*y*z)
diff(expr, x, y, y, z, z, z, z)
```

Out[41]:

$$x^3y^2\left(x^3y^3z^3+14x^2y^2z^2+52xyz+48\right)e^{xyz}$$

In [42]:

```
diff(expr, x, y, 2, z, 4)
```

Out[42]:

$$x^3y^2\left(x^3y^3z^3+14x^2y^2z^2+52xyz+48\right)e^{xyz}$$

In [43]:

```
diff(expr, x, y, y, z, 4)
```

Out[43]:

$$x^3y^2\left(x^3y^3z^3+14x^2y^2z^2+52xyz+48\right)e^{xyz}$$

5.2 求积分

通常使用integrate()来求某个积分表达式

- 对于定积分，需要指定其上限、下限参数
- 对于不定积分，则无需指定上下限参数

In [44]:

```
integrate(cos(x),x)
```

Out[44]:

$$\sin(x)$$

In [45]:

```
integrate(cos(x), (x, 0, 1))
```

Out[45]:

$\sin(1)$

In [46]:

```
integrate(cos(x), (x, 0, pi))
```

Out[46]:

0

- 对于∞符号，使用'oo'来代替

In [47]:

```
integrate(exp(-x), (x, 0, oo))
```

Out[47]:

1

对于二重积分$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-x^2-y^2}\, dxdy$也可以使用该函数进行计算

In [48]:

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

Out[48]:

π

In [49]:

```
Integral(x**y*exp(-x), (x, 0, oo))
```

Out[49]:

$$\int_{0}^{\infty} x^{y} e^{-x}\, dx$$

5.3 求极限

- 在sympy中使用limit()函数求下列极限 $\lim_{x\to+\infty} f(x)$

In [50]:

```
limit(sin(x)/x, x, 0)
```

Out[50]:

1

- 在求一般多项式值时候，subs()可以代替limit()，但limit()更加常用于简短、不可导、无穷的情况：

In [51]:

```
expr = x**2/exp(x)
print(expr.subs(x, oo))
print(limit(expr,x,oo))
```

```
nan
0
```

6. 求解

6.1 值的求解

- 在sympy中，使用Eq()来表示数学表达式中的'='符号

In [52]:

```
from sympy import *
x, y, z = symbols('x y z')
init_printing(use_unicode=True)
Eq(x, y)
```

Out[52]:

$x = y$

- 进而使用solveset()函数对普通多项式进行求解
- 在solveset()函数默认第一个参数表达式=0

In [53]:

```
print(solveset(Eq(x**2, 1), x))
print(solveset(x**2 - 1, x))
```

```
{-1, 1}
{-1, 1}
```

6.2 代数表达式求解

- 除了用于多项式值求解，solveset()函数还可以用于代数表达式的求解

In [54]:

```
solveset(x - x, x, domain=S.Reals)
```

Out[54]:

$$\mathbb{R}$$

In [55]:

```
solveset(sin(x) - 1, x, domain=S.Reals)
```

Out[55]:

$$\left\{2n\pi + \frac{\pi}{2} \mid n \in \mathbb{Z}\right\}$$

In [56]:

```
solveset(exp(x), x)
```

Out[56]:

$$\emptyset$$

- 对于线性多元方程求解，可以使用solveset系列的函数linsolve()

In [57]:

```
linsolve([x + y + z - 1, x + y + 2*z - 3 ], (x, y, z))
```

Out[57]:

$$\{(-y-1, \quad y, \quad 2)\}$$

In [58]:

```
linsolve(Matrix(([1, 1, 1, 1], [1, 1, 2, 3])), (x, y, z))
```

Out[58]:

$$\{(-y-1, \quad y, \quad 2)\}$$

- 对于非线性多元方程，可以使用nonlinsolve()函数

In [59]:

```
a, b, c, d = symbols('a, b, c, d', real=True)
nonlinsolve([a**2 + a, a - b], [a, b])
```

Out[59]:

$$\{(-1, \quad -1), (0, \quad 0)\}$$

In [60]:

```
nonlinsolve([x**2 + 1, y**2 + 1], [x, y])
```

Out[60]:

$$\{(-i, \quad -i), (-i, \quad i), (i, \quad -i), (i, \quad i)\}$$

In [61]:

```
nonlinsolve([exp(x) - sin(y), 1/y - 3], [x, y])
```

Out[61]:

$$\left\{\left(\log\left(\sin\left(\frac{1}{3}\right)\right), \frac{1}{3}\right), \left(\left\{2ni\pi + \left(\log\left(\sin\left(\frac{1}{3}\right)\right) \bmod 2i\pi\right) \mid n \in \mathbb{Z}\right\}, \frac{1}{3}\right)\right\}$$

6.3 微分方程求解

- sympy 使用dsolve()函数来求解微分方程
- 需要指定cls=Function参数来生成函数符号对象

In [62]:

```
from sympy import *
init_printing()
x, y, z = symbols('x y z')
f, g = symbols('f g', cls=Function)
f(x)
```

Out[62]:

$$f(x)$$

In [63]:

```
f(x).diff(x)
```

Out[63]:

$$\frac{d}{dx}f(x)$$

将常微分方程$f''(x) - 2f'(x) + f(x) = sin(x)$转化为sympy数学表达式:

In [64]:

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

求解以上常微分方程:

In [65]:

```
dsolve(diffeq, f(x))
```

Out[65]:

$$f(x) = (C_1 + C_2 x)\, e^x + \frac{1}{2}\cos(x)$$

本节作业

利用sympy包的函数求解下列数学问题

(1)

求解下列极限

$$\lim_{x \to 0} \frac{e - e^{\cos x}}{\sqrt[3]{1 + x^2} - 1}$$

(2)

求下列二重积分，其中D为$x = 1, x = 2, xy = 1, y = 2$所围成的平面区域

$$\iint_D y e^{xy}\, dxdy$$

(3)

求解下常微分方程，并利用其结果求在初始值$(0, \frac{1}{2})$及$(\frac{\pi}{2}, \frac{1}{2} + \frac{\pi}{2})$处的解

$$y'' + y = x$$

In [ ]:

```

```