Sympy

Дербышева Т.Н.
googleTalk: tatyderb@gmail.com

9 апреля 2017 г.

Содержание

1 Где работать? 2
1.1 Jupiter Notebook . . . 2
1.2 Jupiter QTConsole . . . 2

2 Help 2

3 Import 2
3.1 Вывод красиво (в начале пишем) . . . 3

4 Числа в python 3

5 Математические константы 4

6 Символы 4
6.1 Характеристики символов . . . 5
6.2 Function . . . 6
6.3 Выражения . . . 6
6.4 Математические функции . . . 7

7 Подставление значений и вычисление выражений 7

8 Действия с выражениями 8
8.1 simpify - попробуем упросить как-нибудь . . . 8
8.2 Способы упрощения . . . 8
8.3 expand - раскрыть скобки . . . 8
8.4 factor, collect, combine - вынести за скобки . . . 9

9 Решение уравнений **9**
9.1 Проверка функции и найденных корней 10
9.2 Решение систем уравнений 10

10 Самостоятельная работа **10**

1 Где работать?

Удобнее работать так, чтобы математические символы, например, $\sqrt{2}$ показывались как математические символы, а не как `sqrt(2)`

1.1 Jupiter Notebook

Запустите Jupiter Notebook как приложение или запустите его из командной строки

```
ipython notebook
```

Как работать в тетради, написано в wiki

1.2 Jupiter QTConsole

Запустите Jupiter QTConsole как приложение или запустите его из командной строки

```
ipython qtconsole
```

2 Help

```
help(имя функции) # покажет справку по этой функции
```

3 Import

В примерах будем предполагать, что мы импортировали *все* функции из модуля sympy.

```
from sympy import *
```

Почему так лучше не писать? Библиотеки math, numpy, sympy (и дальше мы будем учить другие библиотеки) могут содержать одинаковые имена функций. Например, функция sin разная. Это math.sin, numpy.sin, sympy.sin. Если мы пишем в коде sympy.sin(0) то точно знаем какую функцию вызываем (из библиотеки sympy).

Если сделаем:

```
from sympy import *
from numpy import *

sin(0)    # какой sin использовали? sympy.sin или numpy.sin ?
```

Лучше перечислить все, что нужно из библиотеки:

```
from sympy import Symbol, symbols, sin, PI

sin(0)
```

3.1 Вывод красиво (в начале пишем)

init_printing - для красивой печати результатов математическими символами (LaTeX)

4 Числа в python

В Python есть два типа для хранения чисел int и float.

У int работает целочисленное деление (берется целая часть результата)

```
>>> type(3)
int
>>> type(3.14)
float
>>> 1//3
0
```

Встроенная функция sqrt может вычислить квадратный корень

```
>>> import math
>>> math.sqrt(9)
3.0
>>> math.sqrt(8)
2.82842712475
```

Иногда дроби плохо считать в float , например, 1/3 + 2/3 это 1. Но если посчитать в виде десятичных дробей, то потеряем точность и не получим 1. Хочется считать точно и не терять корни.

```
>>> import sympy
>>> sympy.sqrt(3)
√3          # sqrt(3)
>>> sympy.sqrt(8)
2√2         # 2*sqrt(2)
```

Как определить дробь 1/3 в sympy?

```
Rational(1,3)          # 1/3 как дробь
```

5 Математические константы

Не только функции в разных библиотеках имеют одинаковые имена, но разный смысл.

Константы в библиотке sympy свои. Нельзя путать π из библиотек math, numpy и sympy. numpy.pi - это число, приблизительно равное π, а в sympy.pi - это символ. О символах в sympy - дальше.

sympy	математика
sympy.pi	π
sympy.E	E, экспонента
sympy.I	$\sqrt{-1}$
sympy.oo	∞ бесконечность

6 Символы

Дальше считаем, что у нас был from sympy import *

то есть работают все функции из sympy

В python не нужно было заранее объявлять переменную. Можно было сразу ее использовать:

```
x = 3
```

В sympy так нельзя:

```
>>> x + 1
Traceback (most recent call last):
...
NameError: name 'x' is not defined
```

Нужно сначала сделать символы, а потом с ними работать.

Функции создания символов:

функция	пример	что делает
Symbol	x = Symbol('x')	один символ
symbols	x,y,z = symbols('x y z')	один или много символов
var	x,y,z = var('x y z')	один или много глобальных символов

Пользуйтесь symbols для создания символов

Не создавайте символы I, E, S, N, C, O, Q - они зарезервированы.

Имя переменной не обязательно должно совпадать:

```
>>> from sympy import var
>>> r, t, d = var('rate time short_life')
```

```
>>> d = r*t
>>> print(d)
rate*time
>>> r = 80
>>> t = 2
>>> print(d)          # мы не изменили d, только r и t
rate*time
>>> d = r*t
>>> print(d)          # сейчас d использует актуальное значение r и t
160
```

6.1 Характеристики символов

При создании можно указать дополнительную информацию о символе. От нее может изменяться результат выражения.

```
In [9]: x = sympy.Symbol("x")
In [10]: y = sympy.Symbol("y", positive=True) # назначаем свойство "положительно
In [11]: sympy.sqrt(x ** 2)
Out[11]: √x²
In [12]: sympy.sqrt(y ** 2)
Out[12]: y
In [12]: y.is_positive      # проверяем назначенные свойства
Out[12]: True
```

Еще пример:

```
In [13]: n1 = sympy.Symbol("n")
In [14]: n2 = sympy.Symbol("n", integer=True)
In [15]: n3 = sympy.Symbol("n", odd=True)
In [16]: sympy.cos(n1 * pi)
Out[16]: cos π * n
In [17]: sympy.cos(n2 * pi)
Out[17]: (-1)ⁿ
In [18]: sympy.cos(n3 * pi)
Out[18]: -1
```

Свойство	Проверяем	Описание
real, imaginary	is_real, is_imaginary	действительное число или нет
positive, negative	is_positive, is_negative	положительное или отрицательное
integer	is_integer	целое
odd, even	is_odd, is_even	нечетное или четное целое число
prime	is_prime	простое и целое
finite, infinite	is_finite, is_infinite	конечное или бесконечное

6.2 Function

Можно определить объекты - функции. Для этого задают имя функции и список символов в аргументах.

Функции нам будут нужны позже, в производных.

```
In [41]: x, y, z = symbols("x␣y␣z")
In [42]: f = Function("f") # недоопределили функцию, нет аргументов
In [43]: type(f)
Out[43]: sympy.core.function.UndefinedFunction
In [44]: f(x)
Out[44]: f (x)
In [45]: g = Function("g")(x, y, z) # полностью определили функцию
In [46]: g
Out[46]: g(x, y, z)
In [47]: g.free_symbols
Out[47]: {x, y, z}
```

6.3 Выражения

Можно записать несколько переменных в выражение. Можно придумать любое имя выражению, например, expr или aaa.

```
>>> expr = 2*x + 3*x - sin(x) - 3*x + 42
>>> expr
2*x + 3*x - sin(x) - 3*x + 42
>>> aaa = x**2 + 2*x +1
>>> aaa
x**2 + 2*x +1
```

Еще выражений

```
>>> expr = x + 2*y
>>> expr
x + 2*y
>>> expr + 1
x + 2*y + 1
>>> expr - x
2*y
>>> x*expr
x*(x + 2*y)
```

6.4 Математические функции

sympy	математика
sqrt(x)	$\sqrt{x}$
root(8,3)	$\sqrt[3]{8}$
root(x,3)	$\sqrt[3]{x}$
x**Rational(1,3)	$x^{\frac{1}{3}}$
factorial(n)	n!
sin(x) cos(x) tan(x) cot(x)	sin, cos, tangence, cotangence
log(x)	ln(x)
log(x,b)	$\log_b x$

7 Подставление значений и вычисление выражений

subs - подставить значение (получить выражение)
   xreplace - подставить значение (можно заменять выражение)
   evalf, n, N - вычислить (получить число)

```
>>> expr = sin(x) + cos(y)      # expr это выражение sin(x) + cos(y)
>>> expr
sin(x) + cos(y)
>>> expr.subs({x:1, y:2})       # подставим вместо x число 1, вместо y число 2
sin(1) + cos(2)                 # все равно у нас выражение, а не число
>>> expr.subs({x:1, y:2}).n()   # подставим и вычислим полученное выражение
0.425324148260754               # теперь число
```

Или запишем результат подстановки в отдельное выражение e1

```
>>> expr = sin(x) + cos(y)        # expr это выражение sin(x) + cos(y)
>>> expr
sin(x) + cos(y)
>>> e1 = expr.subs({x:1, y:2})    # e1 это выражение sin(1) + cos(2)
>>> e1
sin(1) + cos(2)
>>> e1.n()                        # вычислим выражение e1
0.425324148260754
```

Сделаем замену переменных:

```
>>> (x*y + z).xreplace({x*y: pi})
z + pi
>>> (x*y*z).xreplace({x*y: pi})
x*y*z
>>> (2*x).xreplace({2*x: y, x: z})
y
>>> (2*2*x).xreplace({2*x: y, x: z})
```

```
4*z
>>> (x + y + 2).xreplace({x + y: 2})
x + y + 2
>>> (x + 2 + exp(x + 2)).xreplace({x + 2: y})
x + exp(y) + 2
```

8 Действия с выражениями

8.1 simpify - попробуем упросить как-нибудь

simpify - пытается изменить выражение; перебирает разные методы

```
In [67]: expr = 2 * (x**2 - x) - x * (x + 1)
In [68]: expr
Out[68]: 2x^2 - x(x + 1) - 2x
In [69]: simplify(expr)        # функция(выражение)
Out[69]: x(x - 3)
In [70]: expr.simplify()       # выражение.функция()
Out[70]: x(x - 3)
In [71]: expr                  # исходное выражение не изменилось
Out[71]: 2x^2 - x(x + 1) - 2x
```

Можно использовать явные функции упрощения:

функция	что делает
simplify	перебирает разные методы
trigsimp	упростить, используя тригонометрические тождества
powsimp	упростить, используя правила работы со степенями
compsimp	упростить комбинаторное выражение
ratsimp	привести к общему знаменателю

8.2 Способы упрощения

sympy	e1	e2	как называется
e2 = e1.expand()	$(x+1)*(x+2)$	x^2+3x+2	раскрыть скобки
e2 = e1.factor()	x^2-1	$(x-1)(x+1)$	разложить на множители
e2 = e1.collect(x)	$x+y+x*y*z$	$x(yz+1)+y$	вынести x за скобку
e2 = apart(e1, x)	$\frac{1}{x^2+3x+2}$	$-\frac{1}{x+2}+\frac{1}{x+1}$	разложить на простейшие дроби
e2 = together(e1, x)	$\frac{1}{xy+y}+\frac{1}{x+1}$	$\frac{y+1}{y(x+1)}$	привести к общему знаменателю
e2 = cancel(e1, x)	$\frac{y}{xy+y}$	$\frac{1}{x+1}$	сократить дробь

8.3 expand - раскрыть скобки

```
In [78]: expr = (x + 1) * (x + 2)
In [79]: expand(expr)
Out[79]: x^2 + 3x + 2
```

Раскроем тригонометрические функции trig=True:

```
In [80]: sin(x + y).expand(trig=True)
Out[80]: sin(x)cos(y) + sin(y)cos(x)
```

и логарифмы log=True (заметьте, мы предполагаем, что a и b - положительные числа):

```
In [81]: a, b = symbols("a, b", positive=True)
In [82]: log(a * b).expand(log=True)
Out[82]: log(a) + log(b)
```

8.4 factor, collect, combine - вынести за скобки

```
In [86]: factor(x**2 - 1)
Out[86]: (x - 1)(x + 1)
In [87]: factor(x * cos(y) + sin(z) * x)
Out[87]: x(sin(x)+cos(y))
```

```
In [89]: expr = x + y + x * y * z
In [90]: expr.collect(x)
Out[90]: x (yz + 1) + y
In [91]: expr.collect(y)
Out[91]: x + y (xz + 1)
```

```
In [93]: expr = cos(x + y) + sin(x - y)
In [94]: expr.expand(trig=True).collect([cos(x),sin(x)]).collect(cos(y) - sin(y)
Out[95]: (sin(x)+ cos(x))(-sin(y)+ cos(y))
```

9 Решение уравнений

Для решения уравнения $f(x) = 0$ используйте функцию solve

```
In [170]: x = sympy.Symbol("x")
In [171]: sympy.solve(x**2 + 2*x - 3)
Out[171]: [-3,1]
```

9.1 Проверка функции и найденных корней

Надо проверять правильно ли мы записали функцию и верное ли решение уравнения:

```
In [170]: x = sympy.Symbol("x")
In [171]: f = x**2 + 2*x - 3
In [172]: f
Out[172]: x^2 + 2x - 3
In [173]: sols = sympy.solve(f)
In [174]: sols
Out[174]: [-3,1]
In [175]: for s in sols :
              print(f.subs({x:s}).n())
0
0
```

9.2 Решение систем уравнений

Система может быть линейной и нелинейной:

```
In [178]: eq1 = x + 2 * y - 1
...: eq2 = x - y + 1
In [179]: solve([eq1, eq2], [x, y], dict=True)
Out[179]: [{x: -1/3, y: 2/3}]
In [180]: eq1 = x**2 - y
...: eq2 = y**2 - x
In [181]: sols = sympy.solve([eq1, eq2], [x, y], dict=True)
In [182]: sols
Out[182]: [{x: 0, y: 0}, {x: 1, y: 1}, {x: (-1/2 - sqrt(3)*i/2)^2, y: -1/2 - sqrt(3)*i/2}, {x: (-1/2 + sqrt(3)*i/2)^2,
```

10 Самостоятельная работа

Задача С.1. Раскройте скобки $a(a+3)(a-3)(a+2)$. Вычислите значение при $a = 2$

Задача С.2. Разложите на множители $a^4 + 2a^3 - 9a^2 - 18a$

Задача С.3. Найдите функцию $f(g(h(x,y),z),t)$. Вычислите ее значение при $x = 1, y = 2, z = 5, t = 2.1$, если

$$f(a,b) = \sin(10 * a + b)$$

$$g(a,b) = b * \exp -a$$

$$h(a, b) = a + b$$

Ответ: $sin(t + 10z \exp(-x - y))$, -0.992440668403706

Задача С.4. Решить уравнение. Проверить найденные корни.

$$\sqrt{x + 1} = 3$$

Ответ: 8

Задача С.5. Решить уравнение. Проверить найденные корни.

$$\sqrt[3]{2x + 3} = 1$$

Ответ: -1

Задача С.6. Решить уравнение. Проверить найденные корни.

$$\sqrt{4x + 2\sqrt{2x^2 + 4}} = x + 2$$

Ответ: 0

Задача С.7. Решить уравнение относительно x. Проверить найденные корни.

$$\sqrt{x - 2}\sqrt{x + 1} = a$$

Ответ: $\left[-\frac{1}{2}\sqrt{4a^2 + 9} + \frac{1}{2}, \quad \frac{1}{2}\sqrt{4a^2 + 9} + \frac{1}{2}\right]$

Задача С.8. Решить систему уравнений. Систему взять у преподавателя.