SymPy — matematyka symboliczna w Pythonie

Mateusz Paprocki <mattpap@gmail.com>

Continuum Analytics, Inc.

30 listopada 2015

Co to jest matematyka symboliczna?

Python operuje na liczbach zmiennoprzecinkowych (IEEE-754):

```
In[1]: import math
In[2]: math.sqrt(3)
```

$$1.732050807568772$$

W matematyce symbolicznej operujemy na liczbach i symbolach:

```
In[3]: import sympy
In[4]: sympy.sqrt(3)
```

$$\sqrt{3}$$

```
In[5]: _4.evalf()
```

$$1.73205080756888$$

```
In[6]: _4.evalf(n=50)
```

$$1.7320508075688772935274463415058723669428052538104$$

Co to jest matematyka symboliczna?

Dokładne wyniki

```
In[7]: math.sqrt(3)**2
```

$$2.9999999999999996$$

```
In[8]: sympy.sqrt(3)**2
```

$$3$$

```
In[9]: 1 - math.sin(math.pi)
```

$$0.9999999999999999$$

```
In[10]: 1 - sympy.sin(sympy.pi)
```

$$1$$

Co to jest matematyka symboliczna?

Operacje na symbolach i wyrażeniach

```
In[11]: x**2 - 1
NameError: name 'x' is not defined

In[12]: f = lambda x: x**2 - 1
In[13]: f(10)
```

$$99$$

```
In[14]: sympy.var('x')
```

$$x$$

```
In[15]: x**2 - 1
```

$$x^2 - 1$$

```
In[16]: _.subs(x, 10)
```

$$99$$

```
In[17]: sympy.factor(_15)
```

$$(x-1)(x+1)$$

Co to jest matematyka symboliczna?

Wzory ogólne

```
In[18]: sum([ i for i in range(0, 10+1) ])
```

$$55$$

```
In[19]: n, k = sympy.symbols('n,k')
In[20]: sympy.summation(k, (k, 0, n))
```

$$\frac{n^2}{2} + \frac{n}{2}$$

```
In[21]: _.subs(n, 10)
```

$$55$$

Matematyka w Pythonie

- moduł `math`
  - wbudowany moduł matematyczny
  - podstawowe operacje na liczbach zmiennoprzecinkowych
- NumPy, SciPy
  - biblioteki numeryczne (macierze, optymalizacja)
- Swiginac, Pynac, ...
  - warstwa abstrakcji nad biblioteką GiNaC (C++)
  - duża wydajność, ale mała funkcjonalność
  - trudne w rozwijaniu
- Sage
  - kompletny system algebry komputerowej
  - podstawowy instalator to 0.8-1.4 GB (Linux 64)
  - zawiera SymPy
- SymPy

Dlaczego SymPy?

Istnieje wiele systemów matematycznych:

- systemy komercyjne:
  - Mathematica, Maple, Magma, ...
- systemy Open Source:
  - AXIOM, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- większość wymyśla swój własny język programowania
- trudna lub niemożliwa rozbudowa systemu i naprawa błędów
- systemy komercyjne są bardzo kosztowne

Dlaczego SymPy?

Istnieje wiele systemów matematycznych:

- systemy komercyjne:
  - Mathematica, Maple, Magma, ...
- systemy Open Source:
  - AXIOM, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- większość wymyśla swój własny język programowania
- trudna lub niemożliwa rozbudowa systemu i naprawa błędów
- systemy komercyjne są bardzo kosztowne

Co to jest SymPy?

- biblioteka pisana w Pythonie
  - `import sympy` i możemy całkować
  - bez nowego środowiska, języka, ...
  - bez rozszerzeń kompilowanych
  - brak obligatoryjnych zależności (poza mpmath)
  - działa dowolnej platformie (~~Jython~~)
- prostota architektury
  - kod w bibliotece oraz REPL nie powinny się (bardzo) różnić
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych gałęzi matematyki
  - wspieranie zaawansowanych metod i algorytmów
  - integracja z IPython/Jupiter, NumPy, ...
- liberalna licencja: BSD

SymPy: historia i trochę liczb

- 2006–teraz (Ondřej Čertík, od 2011 Aaron Meurer)
- 400 autorów
- 450+ tysięcy linii kodu, testów i dokumentacji
- wykonanie testów (na Travis CI) trwa średnio 9 godzin
- 48 projektów (45 studentów) w Google Summer of Code

Możliwości

- **podstawowe możlowści**
  - podstawowa arytmetyka: $+$, -, *, /, **
  - liczby dowolnej precyzji
  - rozwijanie wyrażeń
  - podstawianie wyrażeń
  - upraszczanie/przekształcanie wyrażeń
  - dopasowywanie do wzorców
  - wyrażenia nieprzemienne
  - stałe matematyczne (π, e, złoty podział)
- **funkcje**
  - elementarne (trygonometryczne, hiperboliczne, wykładnicza, logarytmy)
  - kombinatoryczne, całkowitoliczbowe ($n!$, liczby Stirlinga)
  - komponenty liczb zespolonych ($\Re x$, $\Im x$, $|x|$)
  - wielomiany specjalne (cyklotomiczne, Czebyszewa)
  - harmoniki sferyczne
  - inne funkcje specjalne ($\Gamma(x)$, $\zeta(x)$)
- **analiza matematyczna**
  - różniczkowanie
  - całkowanie
  - granice
  - szeregi (Taylor, Laurent, Puiseux)
- **algebra wielomianów**
  - artymetyka, największy wspólny dzielnik
  - rozkład na czynniki
  - rozkład bezkwadratowy
  - bazy Gröbnera
  - rozkład na ułamki proste
  - wyróżnik i rugownik
  - izolacja pierwiastków
- **rozwiązywanie równań**
  - równania wielomianowe
  - równania algebraiczne
  - równania transcendentalne
  - równania różniczkowe
  - równania różnicowe
  - równania diofantyczne
  - równania zadane przedziałami
  - układy równań
  - nierówności
- **kombinatoryka**
  - permutacje, kombinacje,partycje, podzbiory
  - grupy permutacji (wielościany, kostka Rubkika)
  - kody Prüfera, Graya

Więcej możliwości

- **matematyka dyskretna**
  - współczynniki dwumianowe
  - funkcje hipergeometryczne
  - sumy i produkty (skończone i nieskończone)
- **teoria liczb**
  - generowanie i rozpoznawanie liczb pierwszych
  - rozkład liczb całkowitych na czynniki
  - ułamki łańcuchowe
- **macierze**
  - arytmetyka, odwracanie macierzy
  - wartości i wektory własne
  - wielomian charakterystyczny
  - wyznacznik
- **Geometry**
  - punkty, linie, segmenty, elipsy, wielokąty, ...
  - przecinanie się, styczność, podobieństwo figur
- **fizyka**
  - jednostki miar (analiza wymiarowa)
  - stałe fizyczne
  - mechanika klasyczne (dynamika bryły sztywnej)
  - mechanika kwantowa (algebry Pauliego, algorytm Shora)
  - optyka
- **statystyka i rachunek prawdopodobieństwa**
  - rozkłady prawdopodobieństwa
- **wyświetlanie wyrażeń**
  - 2D ASCII/Unicode
  - LaTeX, MathML
  - integracja z IPython/Jupyter notebook
  - generacja kodu: C, Fortran, Python, JavaScript
- **rysowanie wykresów**
  - 2D i 3D
  - krzywe parametryczne
  - figury geometryczne

Przykłady

Podstawowa arytmetyka

SymPy automatycznie stosuje podstawowe transformacje:

```
In[1]: 1 + x - y - y - x**2/x, exp(1), sin(0)
```

$$(-2y + 1, \quad e, \quad 0)$$

Zaawansowane transformacje aplikowane są manualnie:

```
In[2]: sin(x)**2 + cos(x)**2
```

$$\sin^2(x) + \cos^2(x)$$

```
In[3]: simplify(_)
```

$$1$$

Przykłady

Całkowanie i różniczkowanie

```
In[1]: integrate(sin(x*y*z), x, y)
```

$$\begin{cases} 0 & \text{for } yz = 0 \\ -\frac{1}{z}\left(-\log\left(xyz\right) + \frac{1}{2}\log\left(x^2y^2z^2\right) + \mathrm{Ci}\left(xyz\right)\right) & \text{otherwise} \end{cases}$$

```
In[2]: diff(_, x, y)
```

$$\begin{cases} 0 & \text{for } yz = 0 \\ \sin\left(xyz\right) & \text{otherwise} \end{cases}$$

```
In[3]: _1.subs({y: 0, z: 0})
```

$$0$$

```
In[4]: _1.args[1].expr.subs({y: 0, z: 0})
```

$$\mathrm{NaN}$$

Przykłady

Granice i szeregi

```
In[1]: lim = Limit((1 + 1/n)**n, n, oo)
In[2]: Eq(lim, lim.doit())
```

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^{n} = e$$

```
In[3]: limit(log(2 + sqrt(atan(x))*sqrt(sin(1/x))), x, 0)
```

$$\log\left(2\right)$$

```
In[4]: series(sin(cos(x**2)), x, n=10)
```

$$\sin\left(1\right) - \frac{x^{4}}{2}\cos\left(1\right) + x^{8}\left(-\frac{1}{8}\sin\left(1\right) + \frac{1}{24}\cos\left(1\right)\right) + \mathcal{O}\left(x^{10}\right)$$

```
In[5]: summation(1/x**n, (n, 0, oo))
```

$$\begin{cases} \frac{1}{1-\frac{1}{x}} & \text{for } \left|\frac{1}{x}\right| < 1 \\ \sum_{n=0}^{\infty} x^{-n} & \text{otherwise} \end{cases}$$

Problemy i pułapki

liczby Pythona vs. liczby SymPy

```
In[1]: 1/2
```

$$0$$

```
In[2]: Rational(1, 2), S(1)/2, S("1/2"), S.Half
```

$$\left(\frac{1}{2}, \quad \frac{1}{2}, \quad \frac{1}{2}, \quad \frac{1}{2}\right)$$

```
In[3]: 1.2345678912345678 9
```

$$1.234567891234568$$

```
In[4]: Float(1.2345678912345678 9)
```

$$1.23456789123457$$

```
In[5]: Float("1.2345678912345678 9")
```

$$1.2345678912345678 9$$

Problemy i pułapki

liczby Pythona vs. liczby SymPy

```
In[1]: 1/2
```

$$0$$

```
In[2]: Rational(1, 2), S(1)/2, S("1/2"), S.Half
```

$$\left(\frac{1}{2}, \quad \frac{1}{2}, \quad \frac{1}{2}, \quad \frac{1}{2}\right)$$

```
In[3]: 1.23456789123456789
```

$$1.234567891234568$$

```
In[4]: Float(1.23456789123456789)
```

$$1.23456789123457$$

```
In[5]: Float("1.23456789123456789")
```

$$1.23456789123456789$$

Problemy i pułapki

porównywanie wyrażeń

```
In[1]: (x + 1)**2 == x**2 + 2*x + 1
```

$$\text{False}$$

```
In[2]: (x + 1)**2 == (x + 1)**2
```

$$\text{True}$$

```
In[3]: simplify((x + 1)**2 - x**2 + 2*x + 1) == 0
```

$$\text{True}$$

```
In[4]: Eq((x + 1)**2, x**2 + 2*x + 1)
```

$$(x+1)^2 = x^2 + 2x + 1$$

Problemy i pułapki

porównywanie wyrażeń

```
In[1]: (x + 1)**2 == x**2 + 2*x + 1
```

$$\text{False}$$

```
In[2]: (x + 1)**2 == (x + 1)**2
```

$$\text{True}$$

```
In[3]: simplify((x + 1)**2 - x**2 + 2*x + 1) == 0
```

$$\text{True}$$

```
In[4]: Eq((x + 1)**2, x**2 + 2*x + 1)
```

$$(x+1)^2 = x^2 + 2x + 1$$

Problemy i pułapki

limit rekurencji i pamięć podręczna

```
In[1]: from sympy.core.cache import clear_cache
In[2]: horner_poly = lambda n: horner(sum([ x**i for i in range(n) ]))

In[3]: clear_cache(); horner_poly(70).subs(x, 1)
70

In[4]: clear_cache(); horner_poly(71).subs(x, 1)
RuntimeError: maximum recursion depth exceeded

In[5]: import sys
In[6]: sys.setrecursionlimit(2*sys.getrecursionlimit())

In[7]: clear_cache(); horner_poly(142).subs(x, 1)
142

In[8]: clear_cache(); horner_poly(143).subs(x, 1)
RuntimeError: maximum recursion depth exceeded
```

Problemy i pułapki

limit rekurencji i pamięć podręczna

```
In[1]: from sympy.core.cache import clear_cache
In[2]: horner_poly = lambda n: horner(sum([ x**i for i in range(n) ]))

In[3]: clear_cache(); horner_poly(70).subs(x, 1)
70

In[4]: clear_cache(); horner_poly(71).subs(x, 1)
RuntimeError: maximum recursion depth exceeded

In[5]: import sys
In[6]: sys.setrecursionlimit(2*sys.getrecursionlimit())

In[7]: clear_cache(); horner_poly(142).subs(x, 1)
142

In[8]: clear_cache(); horner_poly(143).subs(x, 1)
RuntimeError: maximum recursion depth exceeded
```

Informacje kontaktowe

- Strona główna projektu:
  - `www.sympy.org`
- Dokumentacja:
  - `docs.sympy.org`
  - `wiki.sympy.org`
- Strony dodatkowe:
  - `live.sympy.org`, `gamma.sympy.org`
  - `try.jupyter.org`
- Lista mailingowa:
  - `sympy@googlegroups.com`
- Kanał IRC:
  - `#sympy` na FreeNode
- Repozytorium `git`:

```
git clone git://github.com/sympy/sympy.git
```

Dodatkowe materiały

- SymPy tutorial at SciPy 2011
- SymPy tutorial at SciPy 2013 (video)
- PyDy tutorial at SciPy 2015 (video)

Moja rola w projekcie

- pracuję nad SymPy od marca 2007 roku
- student w Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieoznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania
  - wielomiany, struktury algebraiczne
  - przekształcanie/upraszczanie wyrażeń
  - ...
- poza tym:
  - mentor w Google Summer of Code 2009, 2010, 2011, 2013
  - praca dyplomowa (Politechnika Wrocławska)
  - konferencje (SciPy, EuroScipy, SciPy.In, PyCon.Pl, py4science)

Dziękuję za uwagę!

Pytania, uwagi, dyskusja ...