Výrazy, funkce a proměnné v programu Sage

Robert Mařík

27. listopadu 2010

1 Proměnné

Všechny proměnné se kterými pracujeme je nutno nejprve deklarovat jako proměnné. Výjimku tvoří předdeklarovaná proměnná x a automaticky deklarované proměnné při použití příkazu `automatic_names` (viz Základy ...).

Sage code

```
s  # vyvola chybu, s neni deklarovana promenna
```

```
Traceback (most recent call last):
...
NameError: name 's' is not defined
```

Sage code

```
s=var('s')  # deklarace promenne
```

Sage code

```
print type(s)  # s je deklarovana promenna (symbolicky vyraz)
```

```
<type 'sage.symbolic.expression.Expression'>
```

Sage code

```
s=5  # nyni do promenne dosadime cislo 5
s
```

```
5
```

Sage code

```
print type(s)
```

```
<type 'sage.rings.integer.Integer'>
```

Sage code

```
s=var('s')  # timto prirazenim se opet s deklaruje jako promenna
print type(s)
```

```
<type 'sage.symbolic.expression.Expression'>
```

2 Výrazy, funkce

V mnoha případech pracujeme s výrazy, které obsahují více proměnných, do některých proměnných potřebujeme dosazovat konkrétní hodnoty nebo z výrazů potřebujeme sestavovat rovnice a tyto rovnice řešit. Zde si uvedeme jak na to.

[0]Podporováno grantem FRVŠ 131/2010.

3

Sage code

```
x,y,a,b = var('x,y,a,b')  # deklarace promennych
```

Nadeklarujeme funkci. Aby to nebylo úplně triviální, budeme definovat funkci f na kterou budeme pohlížet jako na funkci jedné proměné x, která obsahuje dva parametry a a b.

Sage code

```
f(x)=b*exp(a*x)  # deklarace funkce
```

Sage code

```
f  # f je funkce
```

$x \mapsto be^{(ax)}$

Budeme počítat funkční hodnoty pro obecnou či pro konkrétní volbu parametrů. Existuje celá řada metod, jak kýženého cíle dosáhnout.

Sage code

```
f(9) # funkcni hodnota
```

$be^{(9\,a)}$

Sage code

```
f(9).subs({a:10, b:8})  # totez, ale parametry nabyvaji konkretnich hodnot
```

$8\,e^{90}$

Sage code

```
f(9).subs({a:10})  # totez, ale jenom parametr a ma konkretni hodnotu
```

be^{90}

Sage code

```
f(9).subs(a=10,b=8)  # jina forma zapisu tehoz
```

$8\,e^{90}$

Sage code

```
f(9,a=10,b=8)  # jina forma zapisu tehoz
```

$8\,e^{90}$

Nekdy nechceme f chápat jako funkci, ale jako označení pro výraz be^{ax}. V tomto případě nedefinujeme funkci f, ale použijeme stejný postup, jako když dosazujeme hodonu do proměnné, tj. použijeme jedno rovnítko a nedáváme argument za označení funkce.

Sage code

```
f=b*exp(a*x)  # deklarace f jako vyrazu, ne jako funkce
```

Sage code

```
f
```

$be^{(ax)}$

Sage code

```
f(x=9, a=8) # parametry x a a maji konkretni hodnotu
```

be^{72}

Sage code

```
f.subs({a:8, x:9}) # totez zapsano jinak
```

be^{72}

Představte si, že ve vzorci $f = be^{ax}$ známe f, b a x a máme vypočítat proměnnou a. V tom případě musíme i f předeklarovat jako proměnnou, sestavit rovnici, vyřešit ji vzhledem k a a dosadit konkrétní hodnoty.

Sage code

```
f=var('f')
f==b*exp(a*x)  # rovnice
```

$f = be^{(ax)}$

Sage code

```
solve(f==b*exp(a*x),a)  # vypocet a
```

$$\left[a = \frac{\log\left(\frac{f}{b}\right)}{x}\right]$$

Pokus o dosazení skončí chybovou hláškou, že do n-tice není možno dosazovat.

Sage code

```
solve(f==b*exp(a*x),a).subs({f:10, b:3, x:4})
```

```
Traceback (most recent call last):
...
AttributeError: 'Sequence' object has no attribute 'subs'
```

Proto z množiny řešení nejprve vybereme výraz se kterých chceme pracovat. Množina řešení je jednoprvková, přemýšlení je snadné: zajímá nás první prvek, první prvek má v Sage a Pythonu index 0.

Sage code

```
solve(f==b*exp(a*x),a)[0]  # vyraz se kterym budeme dale pracovat
```

$$a = \frac{\log\left(\frac{f}{b}\right)}{x}$$

Sage code

```
solve(f==b*exp(a*x),a)[0].subs({f:10, b:3, x:4})  # dosazeni hodnot
```

$$a = \frac{1}{4}\log\left(\frac{10}{3}\right)$$

V následujícím se pokusíme o numerickou aproximaci řešení. Ukážeme si nejprve, že snaha použít přímo příslušnou metodu (tj. `n`) skončí chybovou hláškou spočívající v tom, že pracujeme s rovnicí a na levé straně je výraz, do kterého jsme numerickou hodnotu nedosadili. Proto budeme pracovat jenom s pravou stranou. K tomu slouží metoda `rhs`, kterou předřadíme metodě `n`.

Sage code

```
solve(f==b*exp(a*x),a)[0].subs({f:10, b:3, x:4}).n()
```

```
Traceback (most recent call last):
...
TypeError: cannot evaluate symbolic expression numerically
```

Sage code

```
solve(f==b*exp(a*x),a)[0].subs({f:10, b:3, x:4}).rhs().n()
```

0.300993201081484