Funkce jako objekt

Funkce jako objekt

David Bernhauer, 2014

Teoretické shrnutí

Funkce a její definiční obor

Zobrazení $f : \mathbb{R} \to \mathbb{R}$ nazýváme **reálnou funkcí reálné proměnné**. **Definičním oborem** D_f funkce f nazýváme množinu všech $x \in \mathbb{R}$ pro něž existuje $y \in \mathbb{R}$ splňující $f(x) = y$.

Funkce je často zadána pomocí explicitního vzorce pro výpočet funkční hodnoty $f(x)$. V tomto případě je D_f množina všech $x \in \mathbb{R}$ takových, že $f(x)$ má jednoznačný smysl jakožto reálné číslo.

Obor hodnot

Nechť f je funkce a D_f její definiční obor. Množinu všech $y \in \mathbb{R}$, k nimž existuje $x \in D_f$ tak, že $y = f(x)$, nazveme **obor hodnot** funkce f.

Operace s funkcemi

Nechť f a g jsou funkce. **Součtem** $f + g$ nazveme funkci definovanou vztahem $(f + g)(x) = f(x) + g(x)$ pro všechna $x \in D_f \cap D_g$

Nechť f a g jsou funkce. **Součinem** $f \cdot g$ nazveme funkci definovanou vztahem $(f \cdot g)(x) = f(x) \cdot g(x)$ pro všechna $x \in D_f \cap D_g$

Vlastnosti

Zobrazení $f : A \to B$ je **prosté** (injektivní), jestliže pro každou dvojici $x, y \in D_f, x \neq y$, platí $f(x) \neq f(y)$.

Zobrazení $f : A \to B$ je **na** (surjektivní), jestliže pro každé $y \in B$ existuje $x \in D_f$ splňující $f(x) = y$.

Zobrazení $f : A \to B$ je **vzájemně jednoznačné** (bijektivní), jestliže f je prosté, na a $D_f = A$.

Následující třída implementuje výše uvedené pojmy v některých zjednodušených případech (vizte níže).

```
class Funkce:
# Konstruktor
    def __init__ ( self, zapis, definicniObor = None ):
        # Je parametr typu "relace"
        if isinstance( zapis, dict ):
            if definicniObor == None:
                definicniObor = zapis.keys()

            self.relace = dict()
            for x in definicniObor:
                if x in zapis.keys():
                    self.relace[ x ] = zapis[ x ]

        # Je parametr typu "vyraz"
        elif ( isinstance( zapis,
sage.symbolic.expression.Expression ) ):
            if definicniObor == None:
                raise ValueError( 'Při zadávání výrazu je
nutné dodat i definiční obor.' )

            self.relace = dict()
            for x in definicniObor:
                self.relace[ x ] = zapis( x )

        else: # Ostatní vstupy nepřijmeme
            raise ValueError('Error')

    # Definiční obor
    def D ( self ):
        return set( self.relace.keys() )

    # Obor hodnot
    def H ( self ):
        return set( self.relace.values() )

    # Injekce ( funkce je prostá )
    def jeProsta ( self ):
        # Všechny páry, kde x != y
        for p in [(x, y) for x in self.D() for y in self.D()
if x != y]:
            if ( self( p[ 0 ] ) == self( p[ 1 ] ) ):
                return False
        return True

    # Surjekce ( funkce je na )
    def jeNa ( self, mnozina ):
```

```
        return mnozina == self.H()

    # Výpis funkce
    def __str__ ( self ):
        return str( self.relace )

    # Hodnota v bodě
    def __call__ ( self, x ):
        if ( x not in self.D() ):
            return None
        return self.relace[ x ]

    # Sčítání dvou funkcí
    def __add__ ( self, other ):
        relace = dict()
        # Průnik definičních oborů
        for x in ( self.D() & other.D() ):
            relace[ x ] = self( x ) + other( x )
        return Funkce( relace )

    # Násobení dvou funkcí
    def __mul__ ( self, other ):
        relace = dict()
        # Průnik definičních oborů
        for x in ( self.D() & other.D() ):
            relace[ x ] = self( x ) * other( x )
        return Funkce( relace )
```

Práce s třídou

Vytvořme zobrazení $f : \mathbb{N} \to \mathbb{N}$, které přiřazuje $f(1) = 5$, $f(3) = 8$ a $f(4) = 3$.

```
# Vytvoření funkce
f = Funkce({1:5,3:8,4:3})
# Projděme definiční obor a vypišme obrazy prvků
for i in f.D():
    print "f(" + str(i) + ") = " + str( f( i ) )
```

```
f(1) = 5
f(3) = 8
f(4) = 3
```

Jaké jsou vlastnosti takovéhoto zobrazení?

```
print "Funkce " + ( "je" if f.jeProsta() else "není" ) + "
injektivní."
```

```
M1 = {2,3,5,7,8}
print "Funkce " + ( "je" if f.jeNa( M1 ) else "není" ) + " surjektivní množinu " + str( M1 )

M2 = {3, 5, 8}
print "Funkce " + ( "je" if f.jeNa( M2 ) else "není" ) + " surjektivní množinu " + str( M2 )
```

```
Funkce je injektivní.
Funkce není surjektivní množinu set([8, 2, 3, 5, 7])
Funkce je surjektivní množinu set([8, 3, 5])
```

Vytvořme součet této funkce sama se sbou.

```
print f + f
```

```
{1: 10, 3: 16, 4: 6}
```

Vytvořme novou funkci zadáním definičního oboru a předpisu.

```
g = Funkce(x^2,{2,3,4,5})
for i in g.D():
    print "g(" + str(i) + ") = " + str( g( i ) )
```

```
__main__:3: DeprecationWarning: Substitution using function-c
syntax and unnamed arguments is deprecated and will be remove
a future release of Sage; you can use named arguments instead
EXPR(x=..., y=...)
See http://trac.sagemath.org/5930 for details.
g(2) = 4
g(3) = 9
g(4) = 16
g(5) = 25
```

Součet našich dvou funkcí.

```
h1 = f + g
for i in h1.D():
    print "h1(" + str(i) + ") = " + str( h1( i ) )
```

```
h1(3) = 17
h1(4) = 19
```

Součin našich dvou funkcí.

```
h2 = f * g
for i in h2.D():
    print "h2(" + str(i) + ") = " + str( h2( i ) )
```

```
h2(3) = 72
h2(4) = 48
```