A sympy modul, egyszerűsítés és zárójelfelbontás

2017. október 3.

1. Kifejezések egyszerűsítése és zárójelfelbontás

Ha nem megy Jupyter notebook, akkor a `http://live.sympy.org` oldalon lehet gyakorolni.

```
In [1]: import sympy
        from sympy import *
        init_printing()
        x, y, z, t = symbols('x y z t')
        k, m, n = symbols('k m n', integer=True)
        f, g, h = symbols('f g h', cls=Function)
```

```
In [2]: print(sympy.__doc__)
```

```
SymPy is a Python library for symbolic mathematics. It aims to become a
full-featured computer algebra system (CAS) while keeping the code as simple
as possible in order to be comprehensible and easily extensible.  SymPy is
written entirely in Python. It depends on mpmath, and other external libraries
may be optionally for things like plotting support.

See the webpage for more information and documentation:

    http://sympy.org
```

2. Egyszerűsítés

Egyszerüsíthetjük a kifejezéseket a `simplify` paranccsal.

Feladat: Egyszerűsítsük a $cos^2x + sin^2x$ kifejezést!

```
In [18]: g = cos(x)**2 + sin(x)**2
         print(g, "egyszerűsítve", simplify(g))
         g
```

```
sin(x)**2 + cos(x)**2 egyszerűsítve 1
```

Out[18]:

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

In [20]:
```
f = (x**2 + 2*x -15) / (2*x**3 + 10*x**2)
f
```

Out[20]:

$$\frac{x^2 + 2x - 15}{2x^3 + 10x^2}$$

In [21]:
```
simplify(f)
```

Out[21]:

$$\frac{x - 3}{2x^2}$$

Hivatkozások:

- Rational Expression Arithmetic
- Simplifying Rational Expressions (feladatok, végén eredményekkel)

2.1. Polinomok hatványai, a Pascal-háromszög

In [14]:
```
f = x + y
expand(f**2)
```

Out[14]:

$$x^2 + 2xy + y^2$$

In [3]:
```
f = x + y
for i in range(8):
    print("{}. hatvány:".format(i))
    pprint(expand(f**i))
    print()
```

```
0. hatvány:
1

1. hatvány:
x + y

2. hatvány:
 2           2
x  + 2·x·y + y

3. hatvány:
 3      2          2    3
```

```
x  + 3·x ·y + 3·x·y  + y

4. hatvány:
 4      3        2  2        3    4
x  + 4·x ·y + 6·x ·y  + 4·x·y  + y

5. hatvány:
 5      4         3  2       2  3        4    5
x  + 5·x ·y + 10·x ·y  + 10·x ·y  + 5·x·y  + y

6. hatvány:
 6      5         4  2       3  3       2  4        5    6
x  + 6·x ·y + 15·x ·y  + 20·x ·y  + 15·x ·y  + 6·x·y  + y

7. hatvány:
 7      6         5  2       4  3       3  4       2  5        6    7
x  + 7·x ·y + 21·x ·y  + 35·x ·y  + 35·x ·y  + 21·x ·y  + 7·x·y  + y
```

2.2. Az alábbi program kinyomtatja a Pascal-háromszöget

Nem kell érteni a programkódot, de az eredményt érdemes összehasonlítani a fenti polinomok együtthatóival.

```
In [14]: from math import factorial as f


         def C(n, k):
             return f(n) // (f(k) * f(n-k))


         def Pascal_row(n):
             return [C(n, k) for k in range(n+1)]


         def binom_eloszlas(n, p, k):
             return C(n, k) * p**k * (1 - p)**(n - k)


         def binom(n):
             print("(x+y)^{} = ".format(n), end="")
             x_exps = [e for e in range(n+1)]
             y_exps = [n-e for e in x_exps]
             tri_tuples =  [t for t in zip(Pascal_row(n), x_exps, y_exps)]
             terms = ["{}*x^{}*y^{}".format(*t) for t in tri_tuples]
             print(*terms, sep=" + ")
```

```
        def maxlength(n):
            return len(str(C(n, n//2)))


        def print_Pascal_triangle(n_max=16):
            ml = maxlength(n_max)
            rowlength = (ml+1)*(n_max+1)
            for n in range(n_max+1):
                Pascal_row_text = [str(c).ljust(ml) for c in Pascal_row(n)]
                print(" ".join(Pascal_row_text).center(rowlength))

        print_Pascal_triangle()
```

```
                                                1
                                             1     1
                                          1     2     1
                                       1     3     3     1
                                    1     4     6     4     1
                                 1     5     10    10    5     1
                              1     6     15    20    15    6     1
                           1     7     21    35    35    21    7     1
                        1     8     28    56    70    56    28    8     1
                     1     9     36    84    126   126   84    36    9     1
                  1     10    45    120   210   252   210   120   45    10    1
               1     11    55    165   330   462   462   330   165   55    11    1
            1     12    66    220   495   792   924   792   495   220   66    12    1
         1     13    78    286   715   1287  1716  1716  1287  715   286   78    13    1
      1     14    91    364   1001  2002  3003  3432  3003  2002  1001  364   91    14    1
   1     15    105   455   1365  3003  5005  6435  6435  5005  3003  1365  455   105   15    1
1     16    120   560   1820  4368  8008  11440 12870 11440 8008  4368  1820  560   120   16
```

2.3. Behelyettesítés

Behelyettesíthetünk a függvények `subs` metódusával. Ilyenkor meg kell adni azt, hogy melyik változó helyére, és azt is, hogy milyen értéket, vagy kifejezést.)

In [5]:
```
g = expand(f**5)
g
```

Out[5]:

$$x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5$$

In [6]:
```
g.subs(y, 2)      # y helyére 2-őt írunk.
```

Out[6]:

$$x^5 + 10x^4 + 40x^3 + 80x^2 + 80x + 32$$

```
In [7]: g.subs(y, 2).subs(x, 3)  # Az előző lépés után még x helyére 3-at helyettesítünk.
```

Out[7]:

$$3125$$

```
In [8]: (2 + 3)**5
```

Out[8]:

$$3125$$

Az eredeti g-ben x helyére -2y értéket írunk:

```
In [9]: g.subs(x, -2*y)
```

Out[9]:

$$-y^5$$

```
In [10]: (-2*y + y) ** 5
```

Out[10]:

$$-y^5$$

Az eredeti g-ben x helyére -2 értéket írunk. (Ilyenkor y kitevője szerint fogja csökkenő sorrendbe rendezni a sympy modul a tagokat.)

```
In [11]: g.subs(x, -2)
```

Out[11]:

$$y^5 - 10y^4 + 40y^3 - 80y^2 + 80y - 32$$

```
In [13]: expand((-2 + y) ** 5)
```

Out[13]:

$$y^5 - 10y^4 + 40y^3 - 80y^2 + 80y - 32$$