sympy

代数計算ライブラリ

◆Python での四則演算

通常の四則演算などは Python 単体でも計算が可能。いくつかの演算を確認する。

・単純なかけ算

```
123*456
```

・累乗

累乗計算。累乗は通常の累乗記号の「＾」ではなく「**」で計算できる。

```
2**10
```

◆sympy での計算

sympy にはさまざまな代数計算ライブラリが用意されている。
上記の計算も当然 sympy で計算できる。そのためには sympy をインポートする必要がある。

```
from sympy import *
a = 123*456
print(a)
```

この計算の処理時間は、単純に四則演算をしたより遙かに長い時間がかかる。

■sympy での表記

・Integer　　：整数クラス
・Float　　：実数クラス
・Rational　：分数クラス

◆分数

```
文法：
  Rational（分子、分母）
```

通常の計算では小数に変換されてしまうけど Rational では分数のまま扱われる。

```
from sympy import *
a=1/3
b=Rational(1,3)
print(a)
print(b)
```

この結果は、a は「0.3333・・」になるのに対して、b は「1/3」と分数のまま結果が表記される。

・分数の実数変換

evalf か N 関数を使えば、Rational の値を実数に変換できる。

```
文法：
  N（元値、　桁数）
  インスタンス.evalf（桁数）
```

1/3 の小数部分を 20 桁表示する

```
a=Rational(1,3)
print(N(a,20))
```

・分数同士の計算

分数同士の計算が、そのままできる。結果も分数の状態で返される。

```
a=Rational(1,3)
b=Rational(5,6)
print(a+b)
```

・分数の累乗

```
a=Rational(1,3)
b=(a+3)**2-4
print(b)
```

数値は int 値になっている。

◆定数

・pi　：円周率（Pi クラス）

・E　：自然対数の底（Exp1 クラス）

・oo　：無限大（Infinity クラス）

```
a=Rational(1,3)
print(N(pi,30))
print(N(E,20))
print(N(oo, 30))
```

■平方根

平方根の計算には math パッケージの中にある sqrt を使うのが一般的。sympy の中での複合的な計算のために用意されている。

比較のために math での平方根と列挙する。

```
import math
from sympy import *

a1=sqrt(32)
b1=sqrt(24)
pprint(a1*b1)

a2=math.sqrt(32)
b2=math.sqrt(24)
print(a2*b2)
```

結果は、sympy が「16√3」と表示するのに対して、math の方は「27.712・・」と実数で返される。

■シンボル

シンボルとして変数を作成する。シンボルは数値ではなく、何らかの値の代わりとなるものとして扱われる。

```
文法：
  変数＝symbols（変数名）
    または
  変数＝Symbol（変数名）
```

実際の作成例では、

```
x=symbols(‘x’)
(x,y)=symbols(‘x y’)
```

もうひとつの例では、

```
x=Symbol(‘x’)
```

◆シンボルで式を作る

変数 res には色素の者が代入されている

```
import math
from sympy import *
x=symbols('x')
res=x**2+4*x-6
print(res)
```

■式の単純化

文法：
変数＝simplify（式オブジェクト）

まずは 2 つの式をシンボル化し、その後 simplify によって項ごとにまとめられる。

```
from sympy import *
x=Symbol('x')
re1=(x+1)**2
re2=2*x+7

print(re1)
print(re2)
print(re1+re2)
print(simplify(re1+re2))
```

◆式の展開

多項式の「展開」には expand 関数を使う

```
x=Symbol('x')
re=(x+2)**3
print(expand((re)))
```

答えは、(x+2)(x+2)(x+2)が展開されて「x**3+6*x**2+12*x+8」となる。乗数は「**」で表示される。

■多項式の因数分解

多項式の因数分解では「factor」を使う

```
x=Symbol('x')
re=x**3-x**2-x+1
print(factor((re)))
```

答えは、(x-1)**2*(x+1)と表示される。

◆問題

次の因数分解をしなさい

(1)　$4x^2-1$

(2)　$2x^2+10x+12$

(3)　x^3-7x^2-8x

(4)　$(x+1)^2-16$

■シンボルに値を代入する

subs を使えばシンボルに値を代入し、答えを得ることができる。

文法：

```
[Expr].subs(シンボル、値)
```

re に式を代入し、はじめにそのシンボルを表示してみる

そのシンボルに subs を使って値を代入し、計算結果を表示する。

```
x,y=symbols('x y')
re=x**2-2*y
print(re)
print(re.subs([(x,10),(y,20)]))
```

■方程式を解く

方程式は solve で解くことができる

2 次方程式 x^2+2x-8 を解くと、(-4、2) の結果が返される

```
x=Symbol('x')
re=x**2+2*x-8
print(re)
print(solve(re))
```

この場合、式は「式＝0」として扱われる。なので、そうでない場合は＝0 の形式に直す必要がある。

◆2 次方程式を解く

例として、$(x+y)^2-9$ を解く。

```
(x,y)=symbols('x y')
re=(x+y)**2-9
print(re)
print(solve(re))
print('x=%s' % solve(re,x))
print('y=%s' % solve(re,y))
```

※ %は「文字列フォーマット」演算子。「%ｓ」には文字列を表わしている。このほかにも「%ｄ」整数として展開、「%ｆ」小数点をして展開などがある。

◆連立方程式を解く

$$\begin{cases} 2x+6y+4=0 \\ (x+y)^2-9=0 \end{cases}$$

```
(x,y)=symbols('x y')
re1=2*x+6*y+4
re2=(x+y)**2-9
print(solve((re1,re2)))
```

・平方根を含む連立方程式

```
(x,y)=symbols('x y')
re1=sqrt(2)*x+sqrt(3)*y-1
re2=sqrt(3)*x+sqrt(2)*y-1
print(solve((re1,re2)))
```

◆y＝での書式

数学的には一般的に「ｙ＝」の形式で記述されている。y=x はプログラミング言語では x を y に代入すると解釈される。そのため y ＝をそのまま理解するための関数「Eq」がある。

次の 2 次連立方程式を解く。一般的な y＝の形式になっている

$$\begin{cases} y = x^2+6x+2 \\ y = (x+2)^2 \end{cases}$$

ｙ＝の記述の方法に注意

```
(x,y)=symbols('x y')
re1=Eq(y, x**2+6*x+2)
re2=Eq(y, (x+2)**2)
print(re1)
print(re2)
print(solve((re1,re2)))
```

■極限値

$$\lim_{x \to a} f(x)$$

は、sympy.limit(f(x), x, a)と記述する

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

を sympy で計算し、matplotlib でグラフを描いてみる。

```
sympy.var('x')
y = sympy.sin(x)/x
lim_y = sympy.limit(y, x, 0)
print(lim_y)
p = sympy.plot(y, (x, -12, 12), ylabel="y")
```

別な例

$$\lim_{x \to 0} x \sin \frac{1}{x}$$

```
y = x * sympy.sin(1/x)
lim_y = sympy.limit(y, x, 0)
print(lim_y)
p = sympy.plot(y, (x, -0.1, 0.1), ylabel = "y", line_color="red")
```

■微分

文法：
diff（式、シンボル、階数）

階数を省略した場合は 1 階微分と解釈される

```
(x,y)=symbols('x y')
re=2*x**3
print(diff(re,x))
print(limit((2*(x+y)**3-2*x**3)/y,y,0))
print(diff(re,x,2))
print(limit((6*(x+y)**2-6*x**2)/y,y,0))
```

■積分

文法：
integrate（式、シンボル）

閉区間の定積分については、シンボルの指定部分に（シンボル、下限、上限）の 3 つの要素からなるタプルを指定する。

```
x=Symbol('x')
re=2*x**3
print(integrate(re,x))
print(integrate(re, (x,0,1)))
```

この積分を Matplotlib を使ってグラフ化する

```
import matplotlib.pyplot as plt
import numpy as np
from sympy import Symbol

# シンボル x を定義
x = Symbol('x')
re = 2 * x**3

# SymPy の式を NumPy の関数に変換
re_np = lambdify(x, re, 'numpy')

# x の値を生成
x_values = np.linspace(-2, 2, 400)
y_values = re_np(x_values)

# グラフを描画
plt.plot(x_values, y_values, label='2x^3')
plt.title('Graph of 2x^3')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```