一変数有理関数の評価と微分

問題

Newton の課題 4 では Expression クラスのサブクラスとして Add, Sub, Mul, Div, Number を定義し、それぞれの値を返すメソッド eval()を実装した。しかしそれでは $(1+2)\times 5$ のような簡単な計算しかできなかった。本ミニプロジェクトは $f(x)=x^3-2x$ や $g(x)=\frac{x+5}{x^2+1}$ のような有理関数の計算、さらにはその微分係数の計算まで行えるようにするのが目的であり、大きく分けて次の 3 つのタスクからなっている。

1. Expression クラスのサブクラスとして、変数 x に相当する X を定義する。X は Expression クラスのサブクラスなので、eval() および後述する diff() メソッドが実装されていなくてはならない。
2. 課題 4 では、Expression クラスの eval() メソッドは引数なしのメソッドであったが、本ミニプロジェクトでは eval(x_value) の形をとる。すなわち変数 x の値が x_value であったときの Expression オブジェクトの値を返すことになる。例えば Expression オブジェクト e = Add(Mul(Number(2), X()), Number(1))は式 $2x+1$ に対応するので、e.eval(1) の戻り値は 3、e.eval(5) の戻り値は 11 となる。
3. さらに Expression オブジェクトを変数 x で微分した数式(これもまた Expression オブジェクトである)を返すメソッド diff()を実装する。例えば上記の e に対して、e.diff()は $\frac{d}{dx}(2x+1)=2$ と等価な Expression オブジェクトを返すことが要求されるので、x_value の値とは無関係に e.diff().eval(1) の戻り値も e.diff().eval(100) の戻り値も 2 となる。

これらのコードは提供された expr.py に保存し提出することになるが、別途提供される expr_test.py にある単体テストをすべて通過させることを目標にしてプログラムを作成するとよい。[10 点]

単体テストの例

expr_test.py からの抜粋

```
    def test_xx_3x_2_eval(self):
        x = X()
        f = Add(Add(Mul(x, x), Mul(Number(3), x)), Number(2))
        self.assertEqual(f.eval(10), 132)
```

これは $f(x)=x^2+3x+2$ に対して、$f(10)$ の値が 132 になっているかのテスト。

```
    def test_xx_3x_2_diff(self):
        x = X()
        f = Add(Add(Mul(x, x), Mul(Number(3), x)), Number(2))
        self.assertEqual(f.diff().eval(10), 23)
```

同じ $f(x)=x^2+3x+2$ に対して、$f'(x)=\frac{d}{dx}f(x)=2x+3$ の$x=10$ における値が 23 になっているかのテスト。

微分についてのヒント

$f(x)g(x)$ や $\frac{f(x)}{g(x)}$ の微分に関する公式を忘れてしまった人は、KIT 数学ナビゲーション（http://w3e.kanazawa-it.ac.jp/math/）などを参照するとよい。

注意

- 2019 年 1 月 7 日（月）深夜までに e シラバスに `expr.py` をアップロードすること
- 採点にあたっては、`expr_test.py` には無いテストケースによるチェックも行われる
- 書かなければいけないコードの量は決して多くはないが、オブジェクト指向特有の抽象的な思考に慣れていない人は苦労するかもしれない。そういう人は、まず課題 4 の締め切り後に Newton にアップロードされる課題 4 の解答を理解した上で本ミニプロジェクトに取り組むこと。