Polynomials

In mathematics, a polynomial or multinomial in one variable x is an expression of the format: $$a_0+a_1 x+ a_2x^2+\dots+a_n x^n$$ where n is an integer and the numbers a_k ($k=0,\dots,n$) are named the coefficients of the polynomial. Polynomials form an important class of functions with many applications. They are used for complex functions, among other things.

Assignment

To simplify matters, we will confine ourselves to polynomials that only have whole coefficients. The assignments consists of making a class Polynomial. The class contains the following methods:

1. The initializing method __init__. The coefficients in the polynomial are given to the constructor as a list. Index i in the list then represents the coefficient of the x^i term in the polynomial. Any possible zero-coefficients at the end of the list must be left out and may not be saved in the attribute of the class.
2. The method __str__. Make sure that your polynomial is written out in the format that is given in the examples below. That is: terms with a zero as coefficient are left out, coefficients that are one are left out, etc. If all coefficients are zero, 0 is printed as a polynomial.
3. The method __repr__. This method prints a string-expression that would make an object with the same value as when it were to be evaluated.
4. The method __neg__ that implements the negation of a polynomial.

   ```
   >>> p = Polynomial([1, -1])
   >>> print(-p)
   - 1 + x
   ```

5. The method __add__ that allows to sum up two polynomials.
6. The method __sub__ that allows subtracting two polynomials.
7. The method __mul__ that multiplies two polynomials. If $$p(x)=\sum_{i=0}^m c_ix^i$$ and $$q(x)=\sum_{j=0}^n d_jx^j$$ are two polynomials, their product is $$\left(\sum_{i=0}^m c_ix^i\right)\left(\sum_{j=0}^n d_jx^j\right)=\sum_{i=0}^m\sum_{j=0}^nc_id_jx^{i+j}.$$
8. The method derivative that prints a new polynomial as a result, namely the derivative of the original polynomial. The derivative of a polynomial $$\sum_{i=0}^n c_ix^i$$ is given by $$\sum_{i=1}^nic_ix^{i-1}.$$

Example

```
>>> p = Polynomial([1, -1])
>>> print(p)
1 - x
>>> p
Polynomial([1, -1])
>>> q = Polynomial([0, 1, 0, 0, -6, -1])
>>> print(q)
x - 6 * x^4 - x^5
>>> q
Polynomial([0, 1, 0, 0, -6, -1])
>>> print(p + q)
1 - 6 * x^4 - x^5
>>> print(-p)
- 1 + x
```

```
>>> print(p - q)
1 - 2 * x + 6 * x^4 + x^5
>>> print(p * q)
x - x^2 - 6 * x^4 + 5 * x^5 + x^6
>>> print(q.derivative())
1 - 24 * x^3 - 5 * x^4
>>> Polynomial([1, -1, 0, 0, 0])
Polynomial([1, -1])
>>> w = Polynomial([0, 1, 0, 0, 5, -1])
>>> print(q - w)
- 11 * x^4
```

In de wiskunde is een veelterm of polynoom in één variabele x een uitdrukking van de vorm: $$a_0+a_1 x+ a_2x^2+\dots+a_n x^n$$ waarbij n een natuurlijk getal is en de getallen a_k ($k=0,\dots,n$) de coëfficienten van de veelterm genoemd worden. Veeltermen vormen een belangrijke klasse van functies met veel toepassingen. Zij worden onder meer gebruikt als benadering voor meer ingewikkelde functies.

Opgave

Om een aantal zaken te vereenvoudigen, zullen we ons beperken tot veeltermen met enkel gehele coëfficienten. De opdracht bestaat erin een klasse `Veelterm` aan te maken. De klasse bevat volgende methoden:

1. De initialisatiemethode `__init__`. De coëfficienten in de veelterm worden doorgegeven aan de constructor als een lijst. Index i in de lijst representeert dan de coëfficient van de x^i term in de veelterm. Eventuele nul-coëfficienten op het einde van de lijst moeten weggelaten worden en worden dus niet meeopgeslagen in het attribuut van de klasse.
2. De methode `__str__`. Zorg dat je veelterm uitgeschreven wordt in de vorm zoals aangegeven in de voorbeelden hieronder. Dat is: termen met een nul als coëfficient worden weggelaten, coëfficienten die één zijn worden weggelaten, etc. Indien alle coëfficienten nul zijn, wordt `0` als veelterm uitgeschreven.
3. De methode `__repr__`. Deze methode geeft een string-expressie terug die een object zou aanmaken met dezelfde waarde wanneer ze geëvalueerd zou worden.
4. De methode `__neg__` die de negatie van een veelterm implementeert.

   ```
   >>> p = Veelterm([1, -1])
   >>> print(-p)
   - 1 + x
   ```

5. De methode `__add__` die toelaat twee veeltermen op te tellen.
6. De methode `__sub__` die toelaat twee veeltermen van elkaar af te trekken.
7. De methode `__mul__` die twee veeltermen vermenigvuldigt. Als $$p(x)=\sum_{i=0}^m c_ix^i$$ en $$q(x)=\sum_{j=0}^n d_jx^j$$ twee veeltermen zijn, dan is hun product $$\left(\sum_{i=0}^m c_ix^i\right)\left(\sum_{j=0}^n d_jx^j\right)=\sum_{i=0}^m\sum_{j=0}^nc_id_jx^{i+j}.$$
8. De methode `afgeleide` die een nieuwe veelterm als resultaat teruggeeft, namelijk de afgeleide van de oorspronkelijke veelterm. De afgeleide van een veelterm $$\sum_{i=0}^n c_ix^i$$ wordt gegeven door $$\sum_{i=1}^nic_ix^{i-1}.$$

Voorbeeld

```
>>> p = Veelterm([1, -1])
>>> print(p)
1 - x
>>> p
Veelterm([1, -1])
>>> q = Veelterm([0, 1, 0, 0, -6, -1])
>>> print(q)
x - 6 * x^4 - x^5
>>> q
Veelterm([0, 1, 0, 0, -6, -1])
>>> print(p + q)
1 - 6 * x^4 - x^5
>>> print(-p)
- 1 + x
>>> print(p - q)
1 - 2 * x + 6 * x^4 + x^5
>>> print(p * q)
x - x^2 - 6 * x^4 + 5 * x^5 + x^6
>>> print(q.afgeleide())
1 - 24 * x^3 - 5 * x^4
>>> Veelterm([1, -1, 0, 0, 0])
Veelterm([1, -1])
>>> w = Veelterm([0, 1, 0, 0, 5, -1])
>>> print(q - w)
- 11 * x^4
```