PA3 OOP Part: Play with Polynomials

Brief Introduction

In this part, you are going to capsulate a class `Polynomial` .

You have to implement

- the instantiation of an `Polynomial` object
    - return a string with certain format that represents a `Polynomial` object
    - return the degree of a `Polynomial` object
    - return evaluation of an `Polynomial` object at a certain point
- unary and binary operations of `Polynomial` objects, including:
    - the negation of an `Polynomial` object
    - the addition/subtraction between two `Polynomial` objects
    - the multiplication between two `Polynomial` objects
- other advanced Math operations, including:
    - the derivative of an `Polynomial` object
    - the integral and definite integral of an `Polynomial` object

Template

Note that

- Your task is to implement the class `Polynomial` in this template.
- You should not change the names of the methods (member functions) in the template.
- You can choose to use numpy or not in this task.
- Since this homework was made in a hurry, if you find anything uncleared or wrong in the text, please ask TA ASAP.

```
class Polynomial:
    # Task1
    def __init__(self, coeffs):
        pass
    def __str__(self):
        pass
    def deg(self):
        pass
    def evaluate(self, x):
        pass
    # Task2
    def __neg__(self):
        pass
    def __add__(self, other):
        pass
    def __sub__(self, other):
        pass
    def __mul__(self, other):
        pass
    # Task3
    def derivate(self,m):
        pass
    def integral(self,m):
        pass
    def definite_integral(self,m,x1,x2):
        pass

if __name__ == "__main__":
    print(eval(input()))
```

Task 1

In Task 1, you're going to implement the basic properties of a Polynomial,

which include:

- Initialize the Polynomial class.
- return a string with certain format that represents a `Polynomial` object
- return the degree of a `Polynomial` object
- return evaluation of an `Polynomial` object at a certain point

The relevant functions are marked in the .py file.

Things about input and output

- When initialize the Polynomial class, we will provide a list, which contains the coefficient of each term. Be remembered that the degrees of the terms are in ascending order.

The input and output will be:

Input:

```
Polynomial([1,2,3])
```

Output:

```
1 + 2x + 3x^2
```

Input:

```
Polynomial([-1,-1,3])
```

Output:

```
-1 + -1x + 3x^2
```

Input:

```
Polynomial([1,2,3]).deg()
```

Output:

```
2
```

Input:

```
Polynomial([1,2,3]).evaluate(1)
```

Output:

```
6
```

Note that

- Each coefficent of degree lower than the highest degree should be represented.
- For example, if polyA = Polynomial([1,0,2]), print(polyA) should be "1 + 0x + 2x^2"
- The length of the list is not fixed, but the list will never be empty.
- To be continued...

Function used in Test Cases on OJ

```
+ 1-3: __str__()
+ 4-5: deg()
+ 6-8: evaluate()
```

Task 2

In task 2, you're asked to implement the unary and binary operations of `Polynomial` objects, including:

- the negation of an `Polynomial` object
- the addition/subtraction between two `Polynomial` objects
- the multiplication between two `Polynomial` objects

The input and output will be:

Input:

```
-Polynomial([1,2,3])
```

Output:

```
-1 + -2x + -3x^2
```

Input:

```
Polynomial([1,2,3]) + Polynomial([3,2,1])
```

Output:

```
4 + 4x + 4x^2
```

Input:

```
Polynomial([1,2,3]) - Polynomial([3,2,1])
```

Output:

```
-2 + 0x + 2x^2
```

Input:

```
Polynomial([1,2,3]) * Polynomial([3,2,1])
```

Output:

```
3 + 8x + 14x^2 + 8x^3 + 3x^4
```

Function used in Test Cases on OJ

```
1: __neg__() 2: __add__() 3: __sub__() 4: __mul__()
5-10: All functions you have written
```

Task 3 (Bonus)

In task 3, you need to implement some advanced Math operations, including:

- the derivative of an `Polynomial` object
- the integral and definite integral of an `Polynomial` object

The input and output will be:
Input:

```
Polynomial([1,2,3]).derivate(1)
```

Output:

```
2 + 6x
```

Input:

```
Polynomial([1,2,3]).integral(1)
```

Output:

```
0.0 + 1.0x + 1.0x^2 + 1.0x^3
```

Input:

```
Polynomial([1,2,3]).definite_integral(1,3,2)
```

Output:

```
25.0
```

Function used in Test Cases on OJ

```
+ 1-2: derivate()
+ 4-5: integral()
+ 7-8: definite_integral()
+ 3,6,9,10: All functions you have written
```

Note that

- In the functions of Task 3, there is a parameter 'm', which indicates the degree of your derivate and integral.
- In the function 'definite_integral', there are two parameters 'x1' and 'x2', which indicate the interval of the integral.
- If you are not quite understand the Math operations, feel free to ask TAs.
- Any result of Polynomial which include the integral operation or definite_integral operations shall always and only remain the first decimal and remove the tail.
  (**However, in the process of the operation, there is no ask to remove the tail.)
- For example:
- Polynomial([4,3,5]).integral(2).evaluate(1) = 2.9

Hint

- You can turn to numpy for more information about doing integral and derivate in python.