2. Polynomials

In this assignment, we will look at working with polynomials and representing them as lists of coefficients. The word "representation" here means approximately "a way to store in a computer". Concretely, we say that a polynomial like [1+3x+7x^2](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=1%2B3x%2B7x%5E2) is stored in a Python variable as the list [1,3,7]. Generally speaking, we store the coefficient for the [n](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=n)th degree term in element n of the list. The constant term has degree [0](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=0), so it is found in element 0 of the list.

| **Examples** | | |
| --- | --- | --- |
| **Polynomial** | **Python representation** | **Comment** |
| [x^4](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=x%5E4) | [0, 0, 0, 0, 1] | The coefficient of the 4th term is 1, the others are 0. |
| [4x^2+5x^3](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=4x%5E2%2B5x%5E3) | [0, 0, 4, 5] |  |
| [5+4x+3x^2+2x^3+x^4](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=5%2B4x%2B3x%5E2%2B2x%5E3%2Bx%5E4) | [5,4,3,2,1] |  |

**Note**: You don't need to worry about lists containing other elements than numbers.

The function in the linked Python file ([polynom.py](https://kurser.math.su.se/pluginfile.php/129274/mod_folder/content/0/Lab%202/polynom.py)) contains a function, poly\_to\_string, that renders (shows) polynomials given a list. To get started with the lab, create a file "lab2.py" and copy the poly\_to\_string function there. Complete the assignment by adding code as specified in the exercises and modifying the poly\_to\_string function we have given you (for Exercise 2).

Exercise 1

Suppose the polynomials [p](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p) and [q](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=q) are defined as below.  
  
[ p = 3 + x^3 ](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=%20p%20%3D%203%20%2B%20x%5E3%20)  
[ q = -1 + x^2 + x^4 ](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=%20q%20%3D%20-1%20%2B%20x%5E2%20%2B%20x%5E4%20)  
  
Store these polynomials in Python as lists assigned to variables p and q. That is, start with

p = ...

q = ...

and fill in the polynomials correctly. Check your attempts using poly\_to\_string: (">>>" is the prompt for the Python interpreter)

>>> poly\_to\_string(p)  
3 + 0x + 0x^2 + 1x^3  
  
>>> poly\_to\_string(q)  
-1 + 0x + 1x^2 + 0x^3 + 1x^4

Exercise 2

Rewrite poly\_to\_string such that:

* The empty list is printed as "0". The proper way to store the zero-polynomial ("0") is [0], but let us be a little bit more flexible.
* Terms with coefficient 1 is written *without* a coefficient. That is, "1x^2" should be printed as "x^2".
* Terms with coefficient 0 are not printed. That is, "0 + 0x + 2x^2" should be simplified to "2x^2". What happens with "0 + 0x + 0x^2"?
* A list containing only zeros, such as [0, 0, 0], is printed as "0".

Check the function! Here is how it should work:

>>> poly\_to\_string(p)  
3 + x^3  
  
>>> poly\_to\_string(q)  
-1 + x^2 + x^4  
  
>>> poly\_to\_string([])  
0  
  
>>> poly\_to\_string([0,0,0])  
0

**Obs:** Your function should still **return** a string, not just print a result.

Exercise 3

a) Write a function leading\_coefficient that takes a polynomial and returns the coefficient of the non-zero term of highest degree.

def leading\_coefficient(p\_list):  
 # here be code

Define some polynomials with trailing zeroes:

p0 = [2,0,3,0]      # 2 + 0x + 3x^2 + 0x^3  
q0 = [0,0,0]        # 0 + 0x + 0x2

and test the function:

>>> leading\_coefficient(p)  
1  
>>> leading\_coefficient(p0)  
3  
>>> leading\_coefficient(q0)  
0

**Obs:**we define the leading coefficient of the zero polynomial to be zero.

b) Write a function that returns the degree of a polynomial [p](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p), that is, the degree of the non-zero term in [p](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p) with highest degree.

def degree(p\_list):  
 # here be code

**Tests:**

>>> degree(p)  
3  
>>> degree(q)  
4  
>>> degree(p0)  
2  
>>> degree(q0)  
0

**Obs:**we define the degree of the zero polynomial to be zero.

Exercise 4

Write a function named eval\_poly that takes a polynomials and a value for *x* as parameters, and returns the polynomial´s value in *x*. Suggested algorithm:

* Iterate over the polynomial´s terms by iterating over the coefficients.
* Keep track of the current term degree and the sum of the terms you have processed so far. In every iteration you calculate the value of the term as "coeff \* x \*\* degree" (exponentiation in Python is written as "\*\*"). Add the term value to the sum.
* Return the sum when you have finished iterating.

Verify your solution:

>>> eval\_poly(p,0)

3  
>>> eval\_poly(p,1)  
4  
>>> eval\_poly(p,2)  
11  
>>> eval\_poly(q,2)  
19  
>>> eval\_poly(q,-2)  
19

Exercise 5

a) Define the negation of a polynomial (i.e., change sign of all coefficients).

def neg\_poly(p\_list):

# Your code here

b) Define polynomial addition.

def add\_poly(p\_list,q\_list):

# Your code here

c) Define polynomial subtraction. Hint: [p - q = p + -q](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p%20-%20q%20%3D%20p%20%2B%20-q).

def sub\_poly(p\_list, q\_list):

# Your code here

d) Define a function that tests if two polynomials are equal. Hint: use that [p = q](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p%20%3D%20q) iff [p - q = 0](https://kurser.math.su.se/filter/tex/displaytex.php?texexp=p%20-%20q%20%3D%200).

def eq\_poly(p\_list,q\_list)  
 # here be code

**Obs:** your function should return a boolean value.

Verify that your functions work:

>>> eq\_poly(p,p0)  
False  
  
>>> eq\_poly(q,p0)  
False  
  
>>> eq\_poly(q0,[])  
True  
  
>>> eq\_poly(add\_poly(p,q),add\_poly(q,p))  
True  
  
>>> eq\_poly(sub\_poly(p,p),[])  
True  
  
>>> eq\_poly(sub\_poly(p,neg\_poly(q)),add\_poly(p,q))  
True  
  
>>> eq\_poly(add\_poly(p,p),[])  
False  
  
>>> eval\_poly(add\_poly(p,q),12) == eval\_poly(p,12) + eval\_poly(q,12)  
True

Exercise 5

Read through, clean up, and document your code. So that grading can be objective, do not include your name anywhere in the file.

**Tip:** read through [Basic principles of programming](https://kurser.math.su.se/mod/resource/view.php?id=68437) for advice on how to write good code.

Handing in

Hand in the lab on [PeerGrade](https://www.peergrade.io/). When the hand-in deadline has passed, review three other solutions in PeerGrade.