CPS 109 - Assignment 3

In this assignment, we are going to implement a custom list called CauchyList. A CauchyList is a standard Python list with some arithmetic operations associated to it. A description of the CauchyList class is in the following section. Your program is going to be one Python file, called assignment3.py, that contains the CauchyList class.

CauchyList class

Instance variables

- `p`: an integer that is used as the modulus. This means that all operations are done mod `p`.
- `content`: a list of integers.

Methods

- `__init__`: accepts an integer `p`. Initializes the instance variable `p` to `p` and the instance variable `content` to an empty list.
- `generate_random`: accepts an integer n. Populates `content` with n random integers in the range [0, `p` - 1].
- `length`: returns the length of this CauchyList.
- `get`: accepts an integer `i`. If `i` is smaller than the length of the list then returns the `i`-th element. Otherwise, returns 0.
- `__add__`: returns the sum of this CauchyList and the input CauchyList. The sum of two CauchyLists is defined component-wise. That means if a, b are two CauchyList then we define c = a + b by setting c.content[i] = a.content[i] + b.content[i] where the addition is done mod `p`. This is a special method that is called to evaluate expressions of the form a + b. We have seen other special methods such as `__str__` and `__lt__` in the lectures. For example, `__lt__` is called to evaluate expressions of the form a < b.
  This method should be able to handle CauchyLists of different lengths. When adding two lists of length m and n, where m > n, the resulting list should be of length m and the higher elements of shorter list is assumed to be zero. Hint: you can use the `get` method for this. If the two CauchyLists have different values of `p`, this method should raise a ValueError exception.
- `__sub__`: similar to `__add__` except that it is called to evaluate expressions of the form a - b.
- `__mul__`: returns the product of this CauchyList and the input CauchyList. Suppose that a, b are two CauchyLists. Then the product c = a ∗ b is defined as follows:

$$c[i] = a[0] * b[i] + a[1] * b[i - 1] + a[2] * b[i - 2] + \cdots + a[i] * b[0]$$

  where we denoted a.content[k] by a[k] for simplicity. Remember, c[i] must be reduced mod `p` at the end. Product of two CauchyLists of lengths m and n is a CauchyList of length m + n - 1. So the index i in c[i] can be larger than the lengths of a and b. You can use the `get` method to avoid IndexError.

If the two CauchyLists have different values of `p`, this method should raise a ValueError exception. If the second argument is an integer, not a list, Then this method performs a scalar product which is much simpler. That is, if b is an integer then c[i] = a[i] $*$ b. This allows us to be able to, for example, evaluate an expression of the form a $*$ 65.

- `__str__`: returns a string representation of the CauchyList in the following format:
  p: `p`
  length: `length`
  content: `content`

Testing

You can test your program with the test.py file. Open a terminal and run
`python test.py`

Submission

You only need to submit assignment3.py to D2L. DO NOT copy and paste the code into D2L, upload the file!

Plagiarism detection

You are to work alone when writing your code. You can discuss general ideas with your classmates, but you cannot copy code or develop code together nor take code from the web. We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism; see the following link for details: `http://theory.stanford.edu/~aiken/moss`. Note, MOSS can detect changing identifiers and rearranging code. The Department of Computer Science takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned. Please see Ryerson Universitys Policy 60 for possible penalties and consequences: `http://ryerson.ca/senate/policies/pol60_procedures.pdf`.