Test 01 – Implementing bigNum

Overview:

In a statically typed programming languages, such as C/C++, Java, and C# it is normal to have primitive data types be of restricted size based on allocated memory of the assigned type; for example, int values in Java are allocated 4 bytes (32 bits) and therefore can hold values in the range of -2,147,483,648 through 2,147,483,647.

The problem is even if the largest integer data type is used (i.e. long) you can still only hold a specific restricted set of values. The solution to overcome this is to use an reference type variable that is capable of expanding to an unrestricted size.

In Python version 3, all integers values are represented as bigNums allowing them to be of arbitrary precision. An oversimplified simplified Python uses a list to store the integer digits with the least significant digit first, for example the number 12345 would be stored in a python list as follows:

Interestingly the float in Python 3 is orestricted to the range of values supported by the C programming language's double type (i.e. double precision). To overcome this a programmer may use the Decimal .decimal class which supports floating-point values of arbitrary precision.

Test 01 – Deliverables

In this test I will have you implement:

- 1) Doubly-linked list (dblyLnkdLst.py)
- 2) A simplified version of bigNum using doubly-linked list (bigNum.py)
- 3) Driver program (bigNumCalc.py)
- 4) One Short Answer Question

Step 01 - Implement dblyLnkdLst (10 marks):

Each Node in a DoublyLinkedList has three properties:

next :: Nodeprev :: Nodedata :: Any Type

Implement a doubly-linked list with the following methods. Your doubly-linked list class should have the following properties:

head :: Nodetail :: Nodesize :: Integer

Construct an empty doubly-linked list object
Returns true if doubly-linked list is empty
Add element to front of doubly-linked list
Add element to end of doubly-linked list
Remove first item in doubly-linked list
Remove last item in doubly-linked list
Add element to specified index, if invalid
index display error message to user
Remove node at specified index
Forward traverse through the list and print all
the values in each node (one per line)
Return true if two doubly-linked lists have
the same content in the same order. Return
false otherwise.
Append <i>other</i> to the end of the doubly linked
list self.
Return length of doubly-linked list

Step 02 - Implement bigNum (12 marks):

As described in the overview section bigNum should be able to hold integers values of arbitrary length.

bigNum is an immutable object and should have the following private attribute:

- digits: Doubly-linked list of integers
- sign: String -> ['-' | '+']

RECALL: integers should be stored with the least significant digit at the start of the list; for example the number 12345 is stored as 5 -> 4 -> 3 -> 2 -> 1

bigNum should have aninit() method that has one optional parameter of type String called value. The parameter takes a integer an integer of arbitrary length as a string (i.e. '-12345') and constructs a doubly-linked list with those digits with the least significant digit first.
,
If no parameter is provided to init () use a default value '0'.
In addition to the above you should include the magic methodsadd(),sub(),div() andmul() that takes another bigNum parameter as input and performs
addition, subtraction, division, or multiplication between the two numbers. You should not
modify either the self object or the parameter object, but instead create a new bigNum object
to return from these methods.

- Hint 1: Think of numbers such as 103 as follows: $(3 * 10^0) + (0 * 10^1) + (3 * 10^2)$
- Hint 2: Consider the sign. If you are add a negative number to a positive one that is the same as subtraction and you can leverage __sub__()

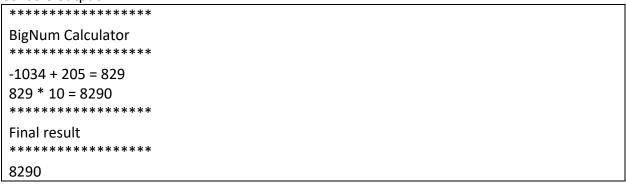
• IMPORTANT: Fordiv () you only need to return the integer portion as a new bigNum object
The class should also have aneq(),gt(), andlt() to compare if two bigNum objects are equal, greater than, or less than one another. All these functions should return boolean results.
Furthermore, you should be sure to include thestr() method which returns the a string representing the value of the bigNum. Be sure to include a leading negative sign in the returned string if sign is negative. You should omit the sign if it is positive. • Example: "-12345"
bigNum should also include alen() method to indicate the number of digits contained in the object.
Step 03 – Driver Program (5 marks):
INPUT: In this question you are tasked with implementing a bigNum calculator that takes a text file called bigNums.txt as input adhering to the below general format: bigNum1 operand bigNum2 operand bigNum3 You may assume that file has at least two bigNums and that input file data is valid
For example, adding the bigNums -1034 and 205 then multiplying by 10 would be represented as:
-1034 + 205 *
The possible energeds that can appear in big Numes that are: / *
The possible operands that can appear in bigNums.txt are: +, -, /, * OUTPUT:

Given the above input format read the 3 lines of text from *bigNums.txt* and perform the appropriate operation based on the operand (i.e. +) that appears.

bigNums.txt

-1034			
+			
205			
*			
10			

Console output



Step 04 – Short Answer (3 marks)

Question: Now that have you seen how to implement an integer of arbitrary precision via the bigNum class how would you implement a fractional number of arbitrary precision?

- 1) What attributes would your class have and of what type? Explain. (0.5 marks)
- 2) Assuming you take a string representing a fractional value as input to the __init__() method (i.e. '123.456') how would you determine the integer and fractional portions? (0.5 marks)
- 3) How would you store the fractional part of the numbers (i.e. 123.456)? Does this differ from how you store the integer portion? Explain your reasoning? (1 mark)
- 4) How would you modify your magic method __add__() from bigNum to apply to this fractional class? Explain. (1 mark)