DS-GA-1007 Programming for Data Science

Assignment 7

Submission Instructions

You are free to use whichever development environment you wish to create and submit the assignment answers.

1. Create a directory using your *Net ID* as the directory name.
2. Place your Python code in this directory.
3. There should be at least one file called `assignment7.py` at the top level of this directory. This is the main program that will generate your answers.
4. Fork the `assignment7` repository from the ds-gs-1007 user on GitHub.
5. Clone this repository onto your local system.
6. Place your new directory (the Net ID) into the working directory of this repository either using PyDev or manually.
7. Add your directory to the staging area, commit, and push to the remote repository.
8. Submit a pull request to the repository owner (ds-ga-1007).

Questions

1. Define a class called `interval` that represents the range of integers between a lower bound and an upper bound. Either of the bounds of an interval can be "inclusive" or "exclusive" and can be positive or negative. The bounds must always meet the requirement that `lower <= upper` if both bounds are inclusive, `lower < upper` if one bound is exclusive and one inclusive, or `lower < upper-1` if both are exclusive. When the class is printed, intervals are displayed using square brackets "[ ]" for inclusive bounds or parenthesis "( )" for exclusive bounds. The class should have a constructor that takes a string representation of the interval and must ensure that the interval in this string conforms to these requirements.

   Examples: "`[1,4]`" represents the numbers 1 through 4
   "`(2,5]`" represents the numbers 3 through 5
   "`[4,8)`" represents the numbers 4 through 7
   "`(3,9)`" represents the numbers 4 through 8.

2. Define a function `mergeIntervals(int1, int2)` that takes two intervals. If the intervals overlap or are adjacent, returns a merged interval. If the intervals cannot be merged, an exception should be thrown.

3. Define the function `mergeOverlapping(intervals)` that takes a list of intervals and merges all overlapping or adjacent intervals.

   Example: Given the interval list `[1,5], [2,6), (8,10], [8,18]`, the function would return `[1,6), [8,18]`.

4. Define a function `insert(intervals, newint)` that takes two arguments: a list of non-overlapping intervals (i.e. any adjacent or overlapping intervals have been merged); and a single interval. The function should insert `newint` into `intervals`, merging the result if necessary. You may assume that the intervals in `intlist` were initially sorted according to their lower bounds. The resulting list should also be sorted according to their lower bounds.

   Example 1: If `intervals` has the value `[[1,3], [6,9]]`, the result of calling `insert(intervals, [2,5])` would be `[[1,9]]`.

   Example 2: If `intervals` has the value `[[1,2], (3,5), [6,7), (8,10], [12,16]]`, the result of calling `insert(intervals, [4,9])` would be `[[1,2], (3,10], [12,16]]`. This is because the new interval `[4,9]` overlaps with `(3,5), [6,7), [8,10]`.

5. Write a program using this class and these functions that prompts the user for a list of intervals, reads a string from the user, and creates a list containing these intervals. Once this string has been read, the program should continue prompting for intervals from the user, insert the interval into the list, and display the list at the end of each operation. *The program should be careful to correctly validate the input from the user.* The following shows a possible *example* of the input and output from such a program.

```
List of intervals? [-10,-7], (-4,1], [3,6), (8,12), [15,23]
Interval? [4,8]
[-10,-7], (-4,1], [3,12), [15,23]
Interval? [24,24]
[-10,-7], (-4,1], [3,12), [15,24]
Interval? [4,-1]
Invalid interval
Interval? [12,13)
[-10,-7], (-4,1], [3,13), [15,24]
Interval? (3,4)
Invalid interval
```

```
Interval? (2,12)
[-10,-7], (-4,1], [3,13), [15,24]
Interval? (-7,-2]
[-10,1], [3,13), [15,24]
Interval? foo
Invalid interval
Interval? [-2,5]
[-10,13), [15,24]
Interval? quit
```

6. Provide unit tests for the class `interval` and the functions `mergeIntervals()`, `mergeOverlapping()`, and `insert()`.

Grading

This assignment will be graded according to the following criteria:

- The program produces the correct output when run
- There is an `interval` class that takes a string argument and prints the interval correctly
- There are `mergeIntervals`, `mergeOverlapping`, and `insert` functions that generate correct results
- Possible exceptions are handled correctly, and specific exceptions are caught rather than using a catchall
- Invalid user input is handled correctly
- User defined exception(s) are employed for indicating error conditions
- Comments are used to describe the overall program behavior, explain intent, and/or warn of consequences
- Doc strings are used to describe each function and public methods
- The class is in separate module from main program
- The program is correctly structured as a Python package
- The code is easily understandable (i.e. divided into logical sections, well structured, etc.)
- The code uses meaningful names for variables, functions, and methods
- Adequate test cases are provided