Programming Assignment: Priority Queue

Problem Statement:

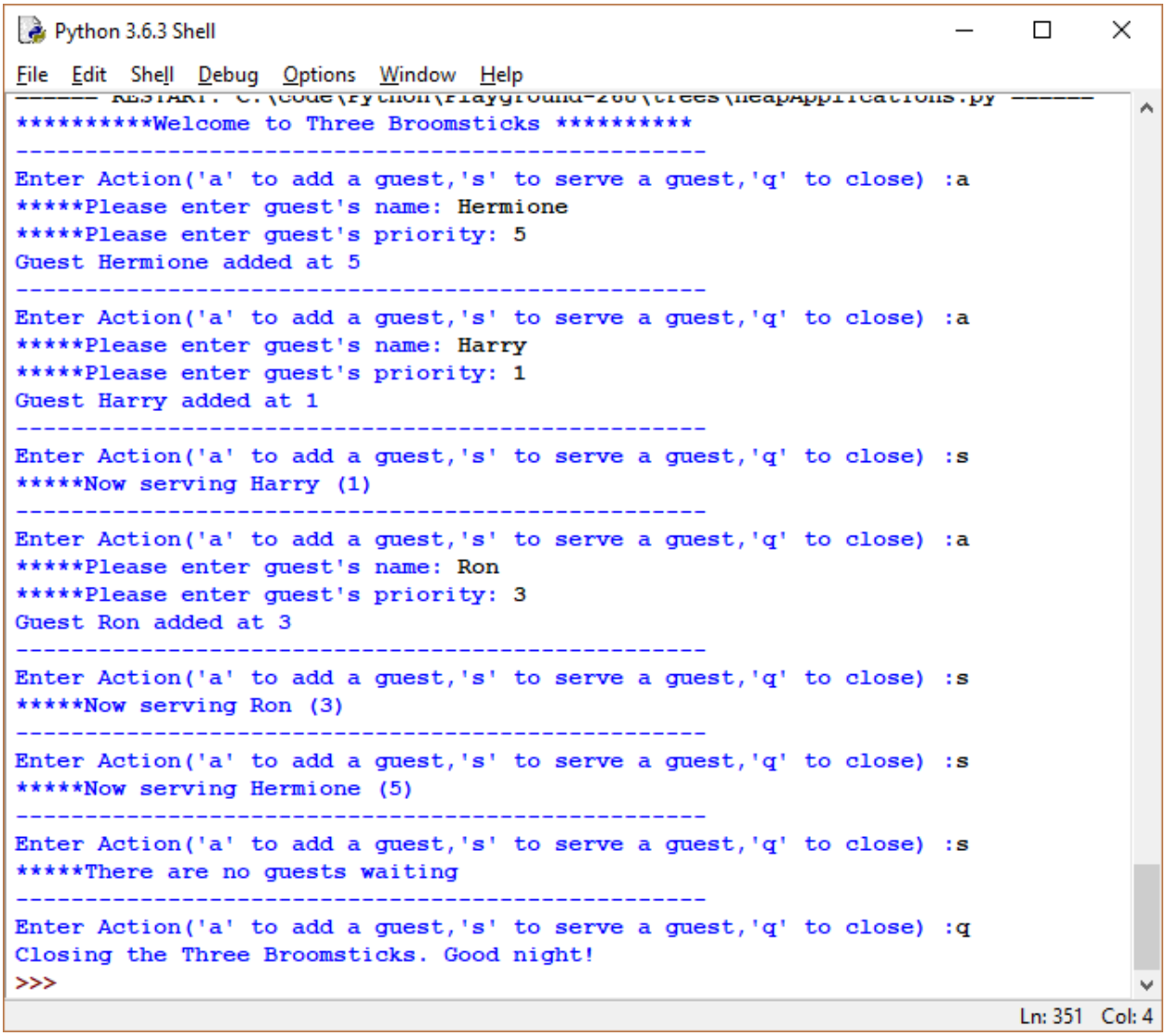
In this assignment, you will implement an application of a binary heap: a reservation system based on a priority queue.

Part A: Simulate a reservation system based on priority queue using a Binary Heap

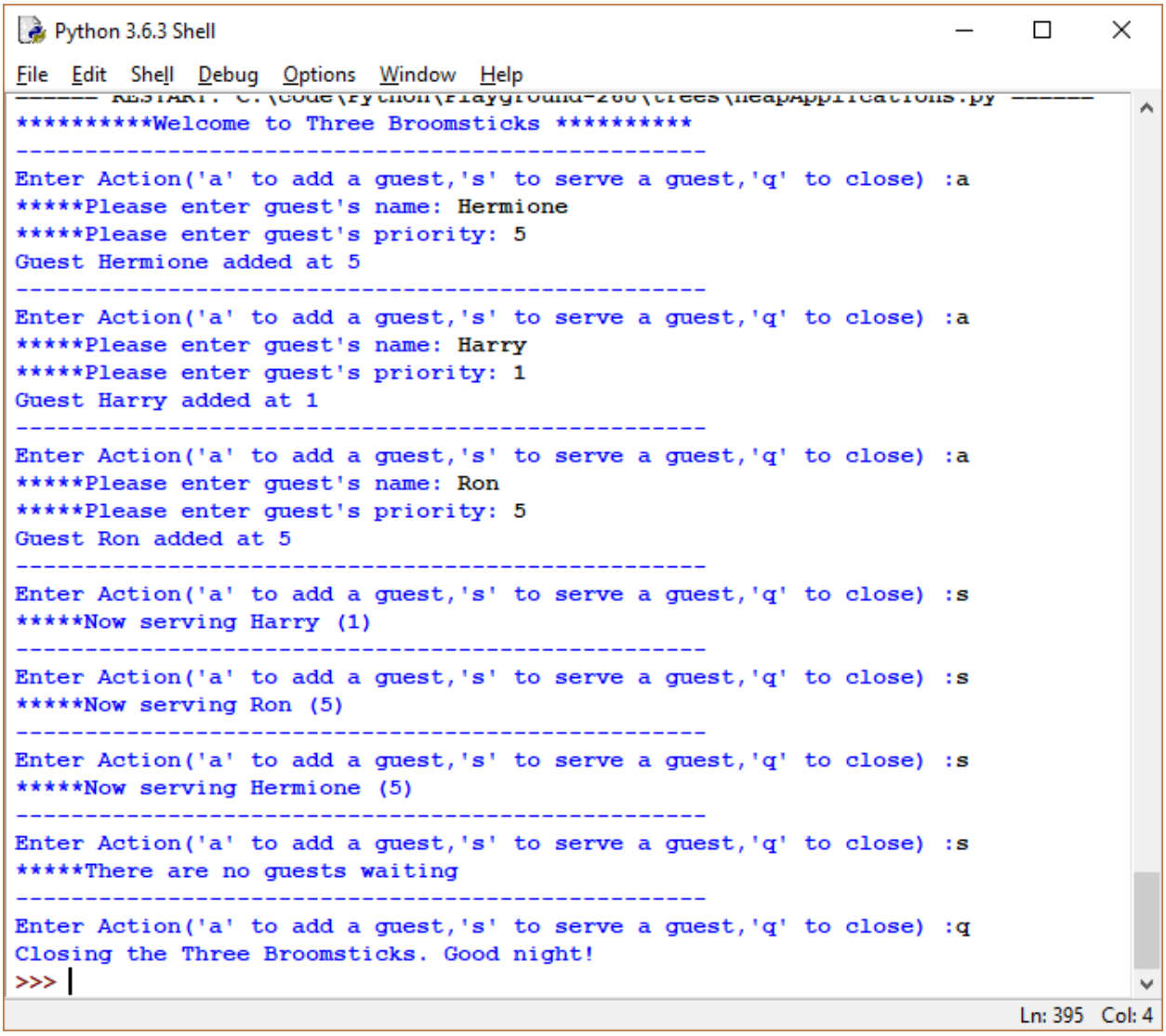
Download the **binheap.py** file with an implementation of a min heap. In a file called **priorityReservation.py**, you will simulate a reservation system of a restaurant where guests are added to the reservation list with a name and an associated priority. Lower the priority, sooner the guest is served. Use the BinHeap class as a priority queue. Specifically,

1. Import the **BinHeap** class from the **binheap.py** file.
2. Add a class called **Guest** that represents a guest at a restaurant reservation system.
    a. Add two public attributes to the class: **name** and **priority**.
    b. Make the class comparable: When two objects of Guest class are compared, the comparison should be based on the **priority**.
3. In the **main** function,
    a. Create a **BinHeap** class object and use it as a priority queue. Note that the implementation of the **BinHeap** class doesn't have to change even when we are using it to store objects of **Guest** class (and not just integers) as long as the **Guest** class is made comparable.
    b. Prompt the user for an action:
        i. If user enters 'a', prompt the user for the name and the reservation priority. Create an object of **Guest** class with the name and priority, and insert the guest to the priority queue.
        ii. If user enters 's', pop the next guest from the queue with lowest priority value and display the name of the guest served.
        iii. If user enters 'q', end the program.

The guests should be served according to their priority. Guests with lower priority should be served first. Here's a sample run:


Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
**********Welcome to Three Broomsticks **********
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :a
*****Please enter guest's name: Hermione
*****Please enter guest's priority: 5
Guest Hermione added at 5
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :a
*****Please enter guest's name: Harry
*****Please enter guest's priority: 1
Guest Harry added at 1
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :s
*****Now serving Harry (1)
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :a
*****Please enter guest's name: Ron
*****Please enter guest's priority: 3
Guest Ron added at 3
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :s
*****Now serving Ron (3)
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :s
*****Now serving Hermione (5)
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :s
*****There are no guests waiting
Enter Action('a' to add a guest,'s' to serve a guest,'q' to close) :q
Closing the Three Broomsticks. Good night!
>>>
Ln: 351 Col: 4

Now try adding two guests with same priority. Here's a sample run:



What happens? Among the guests with same priority (e.g. Hermione and Ron with priority 5 in above run), is the guest that is added first (Hermione in above run) served first (before Ron)? Probably not. You would like to make the reservation system first-come-first-served. Effectively you would like to make the priority queue stable. In the next section you will modify the **Guest** class and the code in the **main** function so that the priority queue is stable.

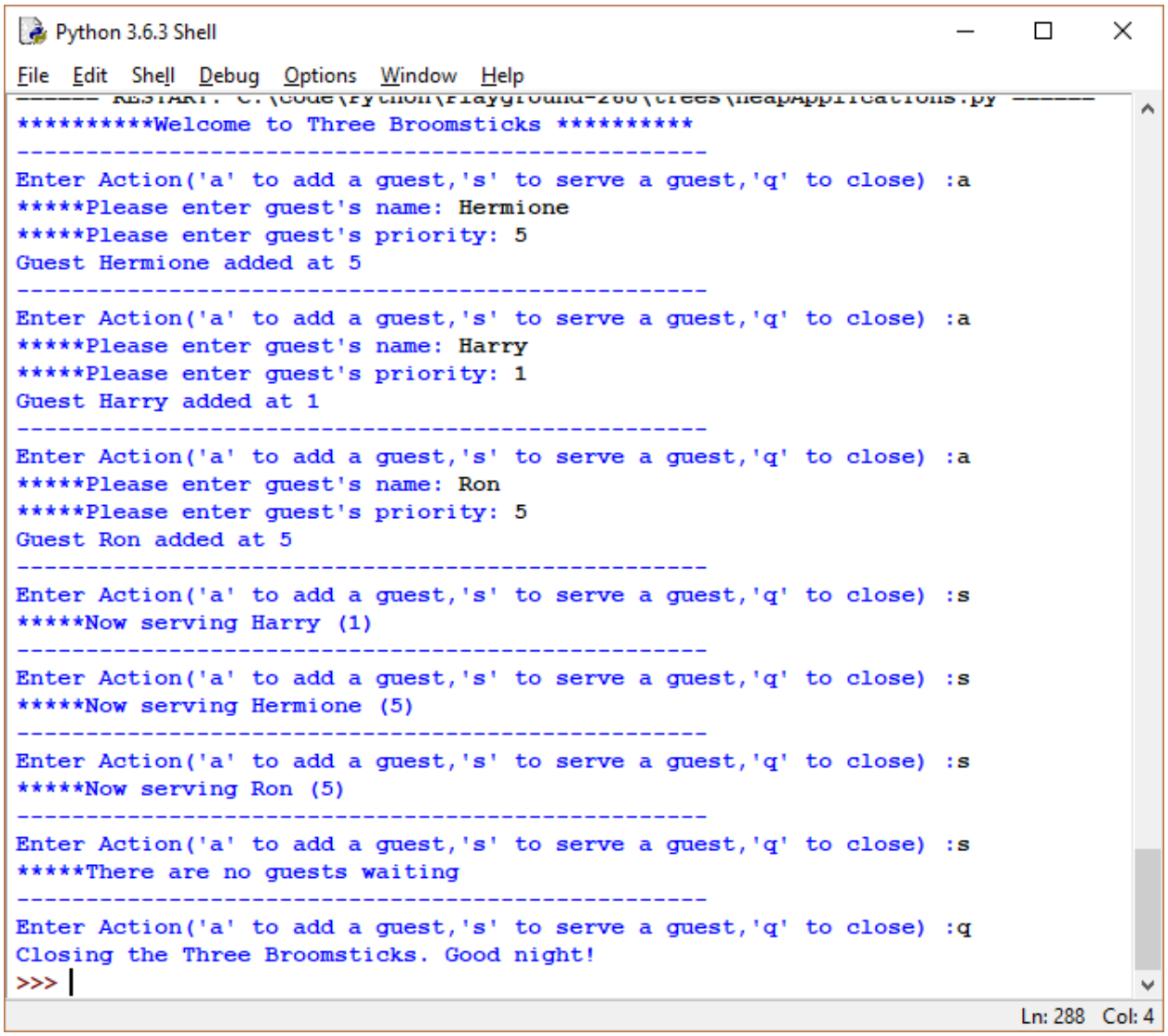
Part B: Make the priority queue stable

We want to make sure that guests with same priority are served in the order they were added to the priority queue. One way to do this is to add the order in which each guest is added to the queue as an attribute of the **Guest** object and make the **Guest** class use the priority along with the insertion order when comparing two objects: first compare the priorities, and if they are same compare the insertion order.

Specifically,

1. Add another public attribute to the **Guest** class called **insertionOrder**. Make the comparison of two **Guest** objects first compare the priority and if priority is same, compare the **insertionOrder**.
2. Make the code in **main** function, keep track of the insertion order as guests are added and set it to be the **insertionOrder** attribute of the guest.

Now try adding multiple guests with same priority and verify that the guests are still served in the priority order, but among the guests with same priority, they are served in the order they were added. Here's a sample run with a ***stable*** priority queue:



Submit your code in a file with a name of the form first_last_priorityReservation.py. Make sure to add the block comment at the top of all the files giving details including your name, date etc.