FIT2107
Week 7 Workshop

This week, please work in pairs.

Unit Testing with PyUnit.

Part 1

You are to implement a priority queue class, called PriorityQueue, in Python.

From Wikipedia:

In computer science, a **priority queue** is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.

Your priority queue class must have:

- a constructor that creates an empty priority queue.
- a method `add(self, priority, item)`, that adds *item* to the queue with priority.
- a method `pop(self)` that removes the highest priority *item* from the queue and returns that item. If there are multiple items with the same priority still on the queue, you can return any one of them. If there are no items on the queue, you should raise an EmptyQueue exception (you will need to define this exception class).
- a method `peek(self)` that returns the highest priority *item* still in the queue (or if there are multiple items with the same priority on the queue, any one of them)

Come up with some test cases for the methods in this class. You don't have to formally document the black-box testing methods we've discussed in the unit, but you should at least keep them in mind as you devise your test cases.

Part 2

Now, write those test cases as PyUnit tests!

Part 3

Implement your priority queue.

Don't worry about making your implementation fancy - **we don't care about performance** (and in any case, one of the benefits of encapsulating your implementation in a class is that you can change the implementation later). Just worry about functional correctness!
As you develop your code, run your PyUnit tests to test whether things are working correctly.