Documentation for the custom class My_Priority_Queue
used in Assignments 2 and 3.
CSE 415, University of Washington, Spring 2019

The implementation of this Python class is included in
the starter code for Assignment 2.

Here are the methods of this class and what they do.

 __init__(self)
 This is the initializer, which defines what happens
when a new instance of the class is constructed.
It basically sets the contents of the new priority
queue to be empty.

 __contains__(self, elt)
 This special method returns True if the argument elt
represents a state that is already in the queue (with
some priority value). It returns False, if there is
no such state in the priority queue. The existence
of this method enables the use of the <code>in</code>
operator of Python for this class.

 delete_min(self)
 This is the standard priority-queue dequeuing method.
If the queue is empty, it returns an empty list.
Otherwise it returns the pair (a tuple) of the form
(state, priority_value) where priority_value is no
greater than any other priority value in the queue.

 insert(self, state, priority)
 This is the method for entering a new state into the
priority queue, with the given priority value.

 __len__(self):
 This returns the number of elements in the priority queue.

 __getitem__(self, state):
 This method enables Python's right-bracket syntax
to work with this data structure. Here, something like
  priority_val = my_queue[state]
becomes possible. Note that the syntax is actually used
in the implementation of the insert method:
   self[state] != -1
 If state is not in the priority queue, then this
method returns -1.

 __delitem__(self, state)
 This method enables Python's del operator to delete
items from the queue. It is used in the implementation
of delete_min.

 __str__(self)
 This method helps when printing a human-readable
description of the priority queue.