CSC148 - Inheritance Recap

1. What is an abstract class?

2. What is the difference between the following two methods? When would we use the first, and why?

```python
def my_method(self, x):
    raise NotImplementedError

def my_method(self, x):
    pass
```

3. Why should we include unimplemented methods in an abstract superclass? Isn’t it better to just implement them in each of the subclasses?
You are given the following documentation for the abstract class for `Container`:

class Container:
    """A container that holds objects.
    
    This is an abstract class. Only child classes should be instantiated.
    """

def add(self, item):
    """Add <item> to this Container.
    
    @type self: Container
    @type item: object
    @rtype: None
    """
    raise NotImplementedError

def remove(self):
    """Remove and return a single item from this Container.
    
    @type self: Container
    @type: object
    @rtype: None
    """
    raise NotImplementedError

def is_empty(self):
    """Return True iff this Container is empty.
    
    @type self: Container
    @rtype: bool
    """
    raise NotImplementedError

Write a function which takes a container and removes all of the items from that container, and returns a list of the results.

def empty_all(container):
    """Remove all items from <container> and return them in a list.
    
    @type container: Container
    @rtype: list
    """